ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μέθοδοι και Τεχνικές Αυτόματης Παραγωγής Παράλληλου Κώδικα για Αλγοριθμικές Περιγραφές Φωλιασμένων Βρόχων

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Νικόλαος Δροσινός

Αθήνα, Ιούλιος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μέθοδοι και Τεχνικές Αυτόματης Παραγωγής Παράλληλου Κώδικα για Αλγοριθμικές Περιγραφές Φωλιασμένων Βρόχων

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

του

Νικολάου Δροσινού

Διπλωματούχου Ηλεκτρολόγου Μηχανικού και Μηχανικού Υπολογιστών Ε.Μ.Π. (2001)

Συμβουλευτική Επιτροπή: Νεκτάριος Γ. Κοζύρης
Γεώργιος Κ. Παπακωνσταντίνου
Παναγιώτης Δ. Τσανάκας

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 21η Ιουλίου 2006.

.....	.....	.....
Ν. Κοζύρης	Γ. Παπακωνσταντίνου	Π. Τσανάκας
Επίκ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.
.....	.....	.....
Α. Σταφυλοπάτης	Κ. Χιτζανίδης	Τ. Σελλής
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

.....
Α. Μπουντουβής
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2006

.....

Νικόλαος Δροσινός
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

στη μητέρα μου Όλγα

και την αδερφή μου Λιλέα

Περίληψη

Η Παράλληλη Επεξεργασία αποτελεί κλάδο της επιστήμης των υπολογιστών, που τυγχάνει ολοένα και περισσότερης προσοχής σε διεπιστημονικό επίπεδο. Το γεγονός αυτό μπορεί να αποδοθεί πρωτίστως στο ότι η Παράλληλη Επεξεργασία παρέχει μια ελκυστική προοπτική για την αποτελεσματικότερη αντιμετώπιση προβλημάτων αυξημένης χωρικής ή/και υπολογιστικής πολυπλοκότητας υπό την παραδοχή των δεδομένων τεχνολογικών περιορισμών αναφορικά με τις δυνατότητες τόσο του υλικού όσο και του λογισμικού/μεσισμικού. Τα τελευταία χρόνια, οι συστοιχίες πολυεπεξεργαστικών στοιχείων έχουν αποδειχθεί εμπράκτως μια ιδιαίτερα ισχυρή, επεκτάσιμη και αξιόπιστη αρχιτεκτονική, στην οποία ο προγραμματιστής μπορεί να επιλύσει ένα σύνθετο αλγόριθμο σε υποπολλαπλάσιο του αρχικού χρόνου και με καλύτερη αξιοποίηση των πόρων του συστήματος για αντιμετώπιση ακόμα μεγαλύτερων προβλημάτων. Ταυτόχρονα, η προτυποποίηση και ευρύτατη αποδοχή εργαλείων παράλληλου προγραμματισμού σύγχρονων αρχιτεκτονικών υψηλών επιδόσεων, όπως η βιβλιοθήκη ανταλλαγής μηνυμάτων MPI και η διεπαφή πολυνηματικής επεξεργασίας OpenMP, αποτελούν ισχυρούς συμμάχους στην κατεύθυνση του αποδοτικού προγραμματισμού και της αξιοποίησης σε μεγάλη κλίμακα των αρχιτεκτονικών αυτών.

Ουσιαστικά, σε αρκετά μεγάλο βαθμό το επιστημονικό ενδιαφέρον έχει πλέον μετατοπιστεί στο χώρο των Εφαρμογών. Έτσι, σε διεπιστημονικό επίπεδο δίνεται ιδιαίτερη έμφαση στη γεφύρωση των τεχνολογικών και αρχιτεκτονικών εξελίξεων με τις ανάγκες και τις ιδιομορφίες των υπολογιστικά απαιτητικών εφαρμογών. Βασική κατηγορία τέτοιων εφαρμογών αποτελούν οι αλγόριθμοι φωλιασμένων βρόχων, που βρίσκονται στην καρδιά πληθώρας απαιτητικών αλγορίθμων, χαρακτηρίζονται δε συχνά από υψηλή πολυπλοκότητα και οδηγούν στον υπολογισμό μεγάλου όγκου δεδομένων. Απόρροια της κεντρικής θέσης που κατέχουν οι αλγόριθμοι φωλιασμένων βρόχων στην οικογένεια των εφαρμογών των υψηλών επιδόσεων αποτελεί η εκτεταμένη μελέτη και διερεύνησή τους στη διεθνή βιβλιογραφία. Έτσι, έχουν προταθεί διάφορες μέθοδοι τόσο για την παραλληλοποίηση τέτοιων αλγορίθμων όσο και για την απεικόνισή τους σε κατανεμημένα συστήματα μνήμης, όπως οι συστοιχίες υπολογιστών. Προς την κατεύθυνση αυτή, ξεχωριστή αναφορά θα πρέπει να γίνει στο μετασχηματισμό υπερκόμβων, ένα

δημοφιλή μη γραμμικό μετασχηματισμό που χρησιμοποιείται για τη διαμέριση των δεδομένων και των υπολογισμών του αρχικού προβλήματος και διευκολύνει την περαιτέρω κατανομή τους στις διαθέσιμες μονάδες εκτέλεσης του παράλληλου προγράμματος.

Η παραλληλοποίηση μιας γενικής φύσεως εφαρμογής φωλιασμένων βρόχων είναι σύνθετη και μη τετριμμένη διαδικασία. Το γεγονός αυτό αποδίδεται στην ποικιλομορφία των εφαρμογών αυτών, που ανάλογα με το φυσικό τους περιεχόμενο επιβάλλουν διαφορετικές σημασιολογικές εξαρτήσεις και καταπιάνονται με φυσικά χωρία ποικίλων γεωμετρικών συσχετισμών. Επιπλέον, η εξέλιξη των παράλληλων αρχιτεκτονικών οδηγεί συχνά στην εισαγωγή καινοτόμων προγραμματιστικών τεχνικών ή ακόμα και στην ανακύκλωση παλιότερων αρχών και προσεγγίσεων. Η τρέχουσα επικρατούσα αρχιτεκτονική τάση στην Παράλληλη Επεξεργασία, ήτοι οι αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης, χαρακτηρίζεται από μια διεπίπεδη αρχιτεκτονική ιεραρχία, που δεν διατηρείται στα συνήθη μονολιθικά μοντέλα παράλληλου προγραμματισμού μέσω ανταλλαγής μηνυμάτων.

Στο πλαίσιο της παρούσας διατριβής μας απασχόλησαν κυρίως δύο ζητήματα: κατά πρώτο λόγο, επιχειρήσαμε να αξιοποιήσουμε τη σημασιολογία ενός αλγορίθμου φωλιασμένων βρόχων, έστω σε επίπεδο του χώρου επαναλήψεων και των εξαρτήσεων δεδομένων, κατά την απεικόνιση του παράλληλου αλγορίθμου σε μια τοπολογία διεργασιών υπό δεδομένη επεξεργαστική υποδομή. Εξ όσων γνωρίζουμε, η ανάδειξη της φύσης του αλγορίθμου κατά την διαδικασία απεικόνισής του σε μια καρτεσιανή τοπολογία διεργασιών δεν έχει ληφθεί υπόψη στη σύγχρονη διεθνή βιβλιογραφία, παρότι πειραματικά οδηγεί σε σημαντικές βελτιώσεις του χρόνου εκτέλεσης του αλγορίθμου. Κατά δεύτερον, διερευνήθηκαν διάφορα υβριδικά μοντέλα παράλληλου προγραμματισμού για αλγορίθμους φωλιασμένων βρόχων, που χρησιμοποιούν τόσο ανταλλαγή μηνυμάτων μεταξύ διαφορετικών κόμβων της συστοιχίας όσο και πολυνηματική επεξεργασία στο εσωτερικό ενός κόμβου, αντανακλώντας έτσι προγραμματιστικά τη διεπίπεδη ιεραρχία τέτοιων αρχιτεκτονικών. Όπως έχει διαπιστωθεί κατά την απόπειρα υβριδικής παραλληλοποίησης εφαρμογών, η αξιοποίηση των πλεονεκτημάτων του υβριδικού προγραμματιστικού μοντέλου είναι δύσκολη διαδικασία, που απαιτεί μεταξύ άλλων τη χρήση τεχνικών για την άμβλυνση των περιορισμών που συχνά επιβάλλει η βιβλιοθήκη ανταλλαγής μηνυμάτων. Στην παρούσα εργασία, αναπτύξαμε τεχνικές εξισορρόπησης του φορτίου μεταξύ των νημάτων, επιτυγχάνοντας τη βελτίωση της επίδοσης του υβριδικού μοντέλου τουλάχιστον στα επίπεδα του μονολιθικού μοντέλου ανταλλαγής μηνυμάτων και σε αρκετές περιπτώσεις σε ακόμα ταχύτερους χρόνους εκτέλεσης.

Λέξεις Κλειδιά: Συστήματα Παράλληλης Επεξεργασίας, Τέλεια Φωλιασμένοι Βρόχοι, Μετασχηματισμός Υπερκόμβων, Χρονοδρομολόγηση Σωλήνωσης, Χώρος Επαναλήψεων, Εξαρτήσεις Δεδομένων, Συστοιχίες Πολυεπεξεργαστικών Στοιχείων, Ανταλλαγή Μηνυμάτων, Πολυνηματική Επεξεργασία, Υβριδική Παραλληλοποίηση, Τοπολογία Διεργασιών, Εξισορρόπηση Φορτίου Νημάτων, MPI, OpenMP.

Abstract

Parallel Processing is a major field of Computer Science, that attracts increasing attention on an interdisciplinary level. This is mainly due to the fact that Parallel Processing provides an alternative perspective for the efficient confrontation of scientific problems exhibiting high spatial and/or temporal complexity, assuming specific hardware and software/middleware technological restrictions. In the last years, clusters of multi-processing computing nodes have proved to be a particularly powerful, scalable and reliable architecture, where a user can address complex algorithms with the aid of parallel programming both in sub-multiple execution time and by exploiting system resources more efficiently. Moreover, the standardization and wide acceptance of parallel programming tools for high performance platforms, like the Message Passing Interface MPI and the Open Multi Processing interface OpenMP, constitute powerful allies as regards to the efficient programming and utilization of such architectures on a large scale.

Currently, the focus of related scientific research has shifted to the domain of Applications. Consequently, considerable emphasis has been placed on the interconnection of the technological and architectural advancements with the characteristics and requirements of the computationally intensive applications. A typical representative of such applications are nested loop algorithms, that lie within the core of many scientific kernels, often exhibit high programming complexity and lead to the calculation of large datasets. The importance of nested loop algorithms can be deduced from their extended study in related scientific literature. As a result, numerous methods have been proposed for the parallelization of nested loop algorithms, as well as for their mapping on distributed memory systems, like clusters. Special reference should be made to the tiling transformation, which is a popular non-linear transformation that can be used for the partitioning of data and computation of the initial problem, as it facilitates further distribution to the available execution entities of the parallel program.

The parallelization of a generic nested loop application is an intricate task. This is mainly due to the large variety of such applications, that impose specific semantic dependencies depending on their

physical meaning and concern the computation of different geometric spaces. Furthermore, the evolution of parallel programming platforms leads to the introduction of innovative programming techniques or even to the recycling of older principles and approaches. The current dominant trend in High Performance Computing, i.e., distributed shared memory platforms, exhibits a two-level hierarchical architecture, that is not preserved with the usual message passing parallel programming models.

In this dissertation, we were mainly concerned with two issues: firstly, when mapping a nested loop algorithm onto a process topology corresponding to a given processor infrastructure, we have attempted to exploit the particular semantics of the algorithm in an efficient and applicable manner, at least in terms of iteration space and data dependencies. To our knowledge, no other work in related literature has considered the nature of the specific algorithm during the mapping procedure of the latter on a cartesian process topology, although it experimentally leads to significant improvement of the algorithm execution time. Secondly, we have investigated different hybrid parallel programming models for nested loop algorithms, that employ both message passing for inter-node communication, as well as multi-threading processing for intra-node synchronization, thus reflecting the architectural hierarchy of distributed shared memory platforms on a programming level. According to both literature observations and practical programming experience, it has been spotted that exploiting the advantages of hybrid parallel programming is a complex task requiring among others the use of techniques for overcoming the restrictions often imposed by the message passing library. In this work, we have developed thread load balancing techniques and ensured performance for the hybrid model at least equal to the one attained by the pure message passing model and in many cases achieved improvement to even faster execution times.

Keywords: Parallel Processing Systems, Perfectly Nested Loops, Tiling Transformation, Pipeline Scheduling, Iteration Space, Data Dependencies, SMP Clusters, Message Passing, Multi-threading, Hybrid Parallelization, Process Topology, Thread Load Balancing, MPI, OpenMP.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

Πρόλογος

Η εργασία αυτή ξεκίνησε ουσιαστικά κατά την ενασχόλησή μου με την αυτόματη παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων στο πλαίσιο της διπλωματικής μου εργασίας στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π. Το τελικό περιεχόμενο διαμορφώθηκε αφενός μέσω της παρακολούθησης των σύγχρονων τάσεων στο χώρο της Παράλληλης Επεξεργασίας και αφετέρου μέσω μακράς διαδικασίας προτυποποίησης νέων τεχνικών βελτιστοποίησης και -κυρίως- πειραματικής επαλήθευσης της αποδοτικότητάς τους. Ιδιαίτερα ωφέλιμη υπήρξε η εκπόνηση σχετικών εργασιών σε συνεργασία με άλλα μέλη του Εργαστηρίου και η συμμετοχή μου σε πολλά διεθνή συνέδρια, τόσο για την παρουσίαση των πορισμάτων και προβολή του ερευνητικού έργου του Εργαστηρίου όσο και για τη συνεχή ενημέρωση σχετικά με τις σύγχρονες τεχνολογικές εξελίξεις από κορυφαίους επιστήμονες του χώρου.

Στο τέλος της μακράς αυτής διαδρομής θα ήθελα να ευχαριστήσω όλους τους καθηγητές που με πλαισίωσαν και με καθοδήγησαν στην έρευνά μου. Ευχαριστώ ιδιαίτερα τον επιβλέποντα καθηγητή μου, Επίκουρο Καθηγητή Ε.Μ.Π. κ. Νεκτάριο Κοζύρη, για την ουσιαστική καθοδήγηση και την πολύπλευρη συμβολή του τόσο στην παρούσα εργασία όσο και στη δημιουργία συνολικής αντίληψης και εποπτείας του χώρου των Συστημάτων Υψηλών Επιδόσεων. Ο ζήλος του για την επιστήμη των υπολογιστών αποτέλεσε άλλωστε το βασικότερο λόγο για τον οποίο επέλεξα να συνεχίσω τις σπουδές μου στο Εργαστήριο Υπολογιστικών Συστημάτων. Ευχαριστώ επίσης τον Καθηγητή Ε.Μ.Π. κ. Γεώργιο Παπακωνσταντίνου, που στάθηκε πάντοτε αρωγός στην ερευνητική μου δραστηριότητα, αλλά κυρίως μου μετέδωσε μέρος της ακεραιότητας του χαρακτήρα του και των αξιών του, ακόμα και σε θέματα που ξεπερνούν το στενό ακαδημαϊκό πλαίσιο. Τέλος, ευχαριστώ και τον Καθηγητή Ε.Μ.Π. κ. Παναγιώτη Τσανάκα τόσο για όλη τη βοήθεια που μου προσέφερε όσο και για την πάντα ευχάριστη συνεργασία μας.

Επίσης, νιώθω την ανάγκη να ευχαριστήσω ολόψυχα όλα τα μέλη του Εργαστηρίου Υπολογιστικών Συστημάτων, που μου έδωσαν την ευκαιρία να εργάζομαι καθημερινά σε ένα ευχάριστο και δημιουργι-

κό περιβάλλον για την εκπόνηση της παρούσας εργασίας. Ξεχωριστή αναφορά θα ήθελα να κάνω στο μεταδιδάκτορα ερευνητή κ. Γεώργιο Γκούμα, που σε όλα τα επίπεδα συνέβαλε τα μέγιστα στην πορεία αυτή, από την αρχική εξοικείωσή μου με το χώρο της Παράλληλης Επεξεργασίας μέχρι την τελική διαμόρφωση του περιεχομένου της διατριβής μου. Κυρίως όμως θα ήθελα να τον ευχαριστήσω γιατί πάνω απ' όλα υπήρξε πραγματικός φίλος και μου έδωσε κίνητρο για προσωπική βελτίωση σε πολλά θέματα, πέραν του ερευνητικού. Ιδιαίτερα θα ήθελα να ευχαριστήσω τη μεταδιδάκτορα ερευνήτρια κυρία Μαρία Αθανασάκη, με την οποία είχα την τύχη να συνεργαστώ στενά σε πολλά ερευνητικά ζητήματα και της οποίας η συμπαράσταση και βοήθεια υπήρξε πάντα αυθόρμητη, ανιδιοτελής και ουσιαστική. Τέλος, θα ήθελα να ευχαριστήσω όλα τα υπόλοιπα μέλη του Εργαστηρίου, και ιδιαίτερα το μεταδιδάκτορα ερευνητή κ. Αριστείδη Σωτηρόπουλο και τους υποψήφιους διδάκτορες κυρία Ευαγγελία Αθανασάκη και κ.κ. Ευάγγελο Κούκη, Γεώργιο Τσουκαλά, Κορνήλιο Κούρτη, Αντώνιο Χαζάπη, Αντώνιο Ζήσιμο, Νικόλαο Αναστόπουλο και Γεώργιο Βεριγάκη. Πρόκειται αναμφίβολα για χαρισματικούς ερευνητές εξαιρετικών επιστημονικών και τεχνικών δυνατοτήτων, κυρίως όμως ανθρώπους με σπάνια καλοσύνη και προθυμία.

Για τη στοιχειοθεσία του παρόντος κειμένου έλαβα μεγάλη βοήθεια από τον υποψήφιο διδάκτορα κ. Παναγιώτη Χείλαρη, τον οποίο ευχαριστώ βαθύτατα. Στο ίδιο ζήτημα, πολύτιμη υπήρξε η συνδρομή των κ.κ. Γεωργίου Τσουκαλά και Αριστείδη Σωτηρόπουλου, τους οποίους ευχαριστώ και για όλες τις γόνιμες συζητήσεις που μοιραστήκαμε κατά καιρούς σχετικά με ερευνητικά και μη ζητήματα, και με βοήθησαν να παραμείνω προσηλωμένος στην ολοκλήρωση της διατριβής μου. Για την επιμέλεια του κειμένου θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Νεκτάριο Κοζύρη και τους κ.κ. Γεώργιο Γκούμα και Κορνήλιο Κούρτη, που συνέβαλαν καθοριστικά στην τεχνική και συντακτική αρτιότητα του παρόντος. Αναφορικά με την επίλυση πολλών προβλημάτων τεχνικής φύσεως, θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Ευάγγελο Κούκη, που υπήρξε όχι μόνο σταθερά πρόθυμος και άκρως αποτελεσματικός συνεργάτης, αλλά και πραγματικός φίλος.

Τέλος, αναμφίβολα το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, τη μητέρα μου Όλγα και την αδερφή μου Λήδα, που με στήριξαν ουσιαστικά, υλικά και συναισθηματικά κατά τη διδακτορική μου διατριβή, όπως και σε όλα τα υπόλοιπα εγχειρήματα της ζωής μου.

Αθήνα, Ιούλιος 2006 *Νικόλαος Δροσινός*

Η παρούσα διδακτορική διατριβή αποτελεί υποέργο του προγράμματος: «Ηράκλειτος: Υποτροφίες έρευνας με προτεραιότητα στην βασική έρευνα». Το Πρόγραμμα «ΗΡΑΚΛΕΙΤΟΣ» συγχρηματοδοτείται από το Ευρωπαϊκό Κοινοτικό Ταμείο (75%) και από Εθνικούς Πόρους (25%).
The Project HRAKLEITOS is co-funded by the European Social Fund (75%) and National Resources (25%).

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Η παρούσα εργασία στοχεύει στην έρευνα και προτυποποίηση αποδοτικών τεχνικών παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων για σύγχρονες παράλληλες αρχιτεκτονικές. Εισαγωγικά, θα επιχειρήσουμε να παρουσιάσουμε συνοπτικά τα κυριότερα ερευνητικά ζητήματα, στα οποία εστιάζει η διατριβή, καθώς και να αποσαφηνίσουμε τη συμβολή αυτής στην ανάδειξη και τεκμηρίωση νέων μεθόδων και τεχνικών αξιοποίησης των σύγχρονων παράλληλων αρχιτεκτονικών. Σημαντικό τμήμα της ερευνητικής δραστηριότητας αναλώθηκε στην εις βάθος μελέτη της σχετικής σύγχρονης διεθνούς βιβλιογραφίας, αποσκοπώντας αφενός στη σχηματοποίηση ενός πλαισίου αξιοποίησης ώριμης συναφούς έρευνας στο χώρο του παράλληλου προγραμματισμού και αφετέρου στην επέκταση, επικαιροποίηση ή ακόμα και αναθεώρηση πορισμάτων της σχετικής έρευνας υπό την τρέχουσα οπτική των σύγχρονων τεχνολογικών εξελίξεων.

1.1 Αντικείμενο της Διατριβής

Αντικείμενο της εργασίας αποτελεί η αξιοποίηση σύγχρονων αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης μέσω της εύρεσης αποδοτικών μοντέλων παραλληλοποίησης αλγορίθμων τέλεια φωλιασμένων βρόχων. Οι αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης (Distributed Shared Memory - DSM) έχουν στην παρούσα φάση επικρατήσει στο χώρο του υπολογισμού υψηλών επιδόσεων. Τα κορυφαία υπολογιστικά συστήματα της γνωστής λίστας TOP500 [DMS99] είναι κατ' ουσίαν DSM αρχιτεκτονικές, αποδεικνύοντας ότι τέτοιες αρχιτεκτονικές μπορούν να επιτύχουν επιχειρησιακά υψηλή από-

δοση, κοντά στις θεωρητικά μέγιστες τιμές. Οι συστοιχίες συμμετρικών πολυεπεξεργαστικών στοιχείων (SMP clusters) αποτελούν τον πλέον δημοφιλή εκπρόσωπο DSM αρχιτεκτονικής και χρησιμοποιούνται ευρέως σε κέντρα υπερυπολογιστών τόσο για ερευνητικούς όσο και για εμπορικούς σκοπούς. Οι SMP συστοιχίες συνδυάζουν την επεκτασιμότητα των απλών συστοιχιών, ενώ παρέχουν επιπλέον τη δυνατότητα πολυεπεξεργασίας μέσω μοιραζόμενης μνήμης στο εσωτερικό κάθε κόμβου. Για αυτό το είδος των αρχιτεκτονικών, υπάρχει ζωηρό ερευνητικό ενδιαφέρον όσον αφορά στη διερεύνηση εναλλακτικών παράλληλων προγραμματιστικών μοντέλων.

Παραδοσιακά, το *μοντέλο ανταλλαγής μηνυμάτων* (*message passing model*) έχει επικρατήσει ως de facto προσέγγιση παραλληλοποίησης για πληθώρα αρχιτεκτονικών υψηλών επιδόσεων, καθώς επιτυγχάνει υψηλή απόδοση και επεκτασιμότητα (scalability) σε επιχειρησιακές συνθήκες. Ένας σημαντικός όγκος επιστημονικών εφαρμογών έχει ήδη παραλληλοποιηθεί με επιτυχία με τη βοήθεια βιβλιοθηκών ανταλλαγής μηνυμάτων, όπως κυρίως η διεπαφή Message Passing Interface (MPI) αλλά και λιγότερο η βιβλιοθήκη Parallel Virtual Machine (PVM), συχνά παρέχοντας γραμμικές ή ακόμα και υπερ-γραμμικές (super-linear) επιταχύνσεις. Μεγάλο ποσοστό του κώδικα που έχει παραλληλοποιηθεί με το μοντέλο ανταλλαγής μηνυμάτων αφορά αλγορίθμους επαναληπτικής φύσεως (iterative algorithms), δηλαδή κατά κανόνα φωλιασμένους βρόχους που διατρέχουν ένα χώρο επαναλήψεων εκτελώντας υπολογισμούς. Η παραλληλοποίηση των αλγορίθμων αυτών μέσω ανταλλαγής μηνυμάτων μπορεί να επιτευχθεί βάσει κάποιας στρατηγικής διαμέρισης του χώρου των επαναλήψεων και καταμερισμού της εργασίας. Προς την κατεύθυνση αυτή ιδιαίτερα χρήσιμος έχει αποδειχθεί ο *μετασχηματισμός υπερκόμβων* (*tiling transformation*), ο οποίος έχει ήδη αποτελέσει αντικείμενο εκτενούς μελέτης στη σύγχρονη διεθνή βιβλιογραφία.

Εντούτοις, σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης ολοένα και συχνότερα εξετάζεται η χρήση *υβριδικών μοντέλων προγραμματισμού* (*hybrid programming models*). Γενικά, τα υβριδικά μοντέλα διατηρούν την επικοινωνία μέσω ανταλλαγής μηνυμάτων μεταξύ των διαφορετικών πολυεπεξεργαστικών κόμβων, αλλά καταφεύγουν σε πολυνηματική επεξεργασία μέσω μοιραζόμενης μνήμης στο εσωτερικό κάθε κόμβου. Διαισθητικά, τα μοντέλα αυτά φαίνεται να ταιριάζουν καλύτερα στην υποδομή των αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης απ' ότι τα αντίστοιχα μοντέλα ανταλλαγής μηνυμάτων, γιατί απαλείφουν τη χρησιμοποίηση της χρονοβόρας επικοινωνίας μέσω μηνυμάτων στις περιπτώσεις που υπάρχει η δυνατότητα γρηγορότερης πρόσβασης σε μοιραζόμενο χώρο μνήμης. Όμως, στην πράξη η υβριδική παραλληλοποίηση αποτελεί ανοιχτό ερευνητικό ζήτημα, καθώς η αξιοποίηση της υφιστάμενης αρχιτεκτονικής μέσω υβριδικών μοντέλων είναι σύνθετη και μη τετριμμένη διαδικασία. Έτσι, κατά γενική ομολογία τα υβριδικά μοντέλα παραλληλισμού στην παρούσα φάση δεν είναι τόσο διαδεδομένα όσο το σύνηθες προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων, που επιπλέον υποστηρίζεται από υψηλά βελτιστοποιημένες βιβλιοθήκες επικοινωνίας.

Στο πλαίσιο της παρούσας εργασίας προτείνουμε αποδοτικά μοντέλα παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων, ήτοι ένα μοντέλο που χρησιμοποιεί ανταλλαγή μηνυμάτων, καθώς και εναλλακτικά υβριδικά μοντέλα λεπτού και χονδρού κόκκου. Προς τη διασφάλιση της υψηλής επίδοσης των παράλληλων μοντέλων προτείνουμε αφενός μεθόδους για τον καθορισμό κατάλληλης τοπολογίας απεικόνισης των διεργασιών και αφετέρου τεχνικές εξισορρόπησης του φορτίου μεταξύ των νημάτων για το υβριδικό μοντέλο χονδρού κόκκου. Πιο συγκεκριμένα, υποθέτουμε αλγόριθμο φωλιασμένων βρόχων διάστασης $N + 1$, που παραλληλοποιείται ως προς τις N εξωτερικές διαστάσεις και υλοποιεί σειριακή εκτέλεση με τη μέθοδο της σωλήνωσης (pipelining) κατά μήκος της πλέον εσωτερικής. Αποδεικνύουμε ότι η επιλογή μιας κατάλληλης τοπολογίας για το N-διάστατο πλέγμα διεργασιών μπορεί να μειώσει σημαντικά το συνολικό χρόνο εκτέλεσης και παρέχουμε μια αυτόματη μέθοδο για τον καθορισμό μιας τέτοιας τοπολογίας. Επιπλέον, συγκρίνουμε διάφορα υβριδικά μοντέλα λεπτού και χονδρού κόκκου και αναδεικνύουμε την αναγκαιότητα αποδοτικής εξισορρόπησης του φορτίου μεταξύ των νημάτων, ώστε να υπερκεραστούν οι ενδεχόμενοι περιορισμοί της βιβλιοθήκης ανταλλαγής μηνυμάτων. Αμφότερες οι βελτιστοποιήσεις μπορούν εύκολα να υλοποιηθούν είτε σε επίπεδο μεταγλωττιστή είτε ως συναρτήσεις βιβλιοθήκης. Τέλος, συγκρίνουμε όλα τα μοντέλα, δίνοντας έμφαση στη σύγκριση μεταξύ του μοντέλου ανταλλαγής μηνυμάτων και του υβριδικού μοντέλου χονδρού κόκκου, ώστε να εξαγάγουμε πιο γενικά συμπεράσματα για τον παράλληλο προγραμματισμό αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης.

1.2 Επισκόπηση Σχετικής Βιβλιογραφίας

Οι φωλιασμένοι βρόχοι κατέχουν ήδη από τα τέλη της δεκαετίας του '80 κομβικό ρόλο στην παραλληλοποίηση υπολογιστικά απαιτητικών εφαρμογών. Εμπειρικά έχει διαπιστωθεί πως πολλοί αλγόριθμοι καταναλώνουν το μεγαλύτερο τμήμα του χρόνου εκτέλεσής τους με τη διάσχιση δομών επαναληπτικής φύσεως, όπως είναι οι φωλιασμένοι βρόχοι, αναδεικνύοντας έτσι τους τελευταίους σε βασικούς στόχους της προγραμματιστικής βελτιστοποίησης. Είναι κοινός τόπος πως η κατάλληλη αναδιάταξη της ακολουθίας των βρόχων ή η αξιοποίηση της ιεραρχίας μνήμης κατά την προσπέλαση δεδομένων σε φωλιασμένους βρόχους μπορούν να επιταχύνουν σημαντικά το χρόνο εκτέλεσης ενός προγράμματος. Ως φυσικό επακόλουθο, η παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων αποτελεί ένα ιδιαίτερα δημοφιλές αντικείμενο της διεθνούς βιβλιογραφίας, απόρροια της πρακτικής αξίας του ζητήματος για την επιτάχυνση πολλών πραγματικών εφαρμογών. Μάλιστα, οι εξαρτήσεις που επιβάλλονται από τη σημασιολογία του αλγορίθμου κατά την προσπέλαση και τον υπολογισμό των δεδομένων έχουν μοντελοποιηθεί μαθηματικά [Ban88, WL91b, Zig94], διακρίνοντας τους βρόχους σε DOALL, όπου υπάρχει εγγενής ανεξαρτησία μεταξύ των διαφορετικών επαναλήψεων του βρόχου, και σε DOACROSS, όπου

η διασφάλιση της ορθότητας του αλγορίθμου απαιτεί την υλοποίηση μηχανισμού συγχρονισμού και επικοινωνίας μεταξύ των φορέων της παράλληλης εκτέλεσης του προγράμματος. Όπως είναι φανερό, η παραλληλοποίηση των DOALL βρόχων είναι σχετικά απλή διαδικασία, ενώ αντίθετα για τους περισσότερο σύνθετους DOACROSS βρόχους έχουν προταθεί πολυάριθμες τεχνικές αποδοτικής παραλληλοποίησης, κυρίως μέσω του μοντέλου ανταλλαγής μηνυμάτων και της διεπαφής MPI.

Ιδιαίτερη ώθηση στην αποδοτική παραλληλοποίηση των αλγορίθμων φωλιασμένων βρόχων προσέφερε η εισαγωγή του μετασχηματισμού υπερκόμβων (tiling transformation) από τους Irigoin και Triolet το 1988 [IT88]. Ο μετασχηματισμός αυτός έχει χρησιμοποιηθεί τόσο για την αποδοτικότερη αξιοποίηση της τοπικότητας αναφοράς δεδομένων σε φωλιασμένους βρόχους όσο και για τη διαμέριση του χώρου επαναλήψεων του αλγορίθμου σε όμοιες ατομικές μονάδες, που μπορούν να διευκολύνουν τη διαδικασία απεικόνισης του αρχικού προγράμματος σε μια παράλληλη αρχιτεκτονική. Μεγάλος όγκος εργασιών έχει ασχοληθεί με το πρόβλημα της επιλογής του βέλτιστου μετασχηματισμού υπερκόμβων για μια δεδομένη αλγοριθμική περιγραφή φωλιασμένων βρόχων [RS92, Zig94, TZ94, BDRR94, OSKO95, Xue97, HCF97, HS98, HS99, Hod99, HCF99, ACN+00, HS02, XW02, HCF03, Γκο03]. Συνηθέστερα, οι εργασίες αυτές στοχεύουν στον καθορισμό τόσο του σχήματος όσο και του μεγέθους του υπερκόμβου, βάσει ενός προβλήματος βελτιστοποίησης ή συγκεκριμένων κριτηρίων.

Για παράδειγμα, οι Ohta, Saito, Kainaga και Ono [OSKO95], οι Hodzic και Shang [HS98], καθώς και οι Andonov, Balev, Rajopadhye και Yanev [ABRY03] εστίασαν στην επιλογή του βέλτιστου μεγέθους υπερκόμβου βάσει των χαρακτηριστικών της εξεταζόμενης εφαρμογής και της υφιστάμενης αρχιτεκτονικής. Οι Ramanujam και Sadayappan [RS92], οι Boulet, Darte, Risset και Robert [BDRR94] και ο Xue [Xue97] ασχολήθηκαν με τον προσδιορισμό του σχήματος υπερκόμβου που ελαχιστοποιεί τον όγκο επικοινωνίας *ανά υπερκόμβο*, δηλαδή ουσιαστικά ελαχιστοποιεί το πλήθος των εξαρτήσεων που τέμνουν τις πλευρικές επιφάνειες του υπερκόμβου. Στην περίπτωση αυτή, το βέλτιστο σχήμα υπερκόμβου ορίζεται από επίπεδα παράλληλα προς τον κώνο εξαρτήσεων του αλγορίθμου. Ιδιαίτερα αξιοσημείωτη είναι η δουλειά των Hodzic και Shang [HS98, HS02] και των Högstedt, Carter και Ferrante [HCF99, HCF03] που καθόρισαν το σχήμα του υπερκόμβου που ελαχιστοποιεί το συνολικό αριθμό των βημάτων εκτέλεσης του παράλληλου αλγορίθμου. Εδώ ο μετασχηματισμός υπερκόμβου *ελάχιστης καθυστέρησης* προκύπτει καθορίζοντας αρχικά το βασικό σχήμα του υπερκόμβου (συνήθως παράλληλο προς τον κώνο εξαρτήσεων) και κλιμακώνοντας στη συνέχεια τις πλευρές του υπερκόμβου, ώστε να ελαχιστοποιηθεί το μέγιστο μονοπάτι εκτέλεσης μεταξύ του «πρώτου» και του «τελευταίου» υπερκόμβου.

Ένα λεπτό ζήτημα εντοπίζεται στη σύνδεση του εκάστοτε προτεινόμενου μετασχηματισμού υπερκόμβων με την απεικόνιση και δρομολόγηση του παράλληλου αλγορίθμου σε μια συγκεκριμένη επεξεργαστική υποδομή. Πολλές από τις προαναφερθείσες εργασίες προσδιορίζουν το σχήμα και το μέγεθος

του υπερκόμβου ανεξάρτητα π.χ. από το συνολικό πλήθος των διαθέσιμων επεξεργαστών. Έτσι, είναι συχνό το φαινόμενο είτε να προκύπτει δυσαναλογία μεταξύ του πλήθους των επεξεργαστών και εκείνου των υπερκόμβων, στους οποίους διαμερίζεται ο χώρος επαναλήψεων του αλγορίθμου φωλιασμένων βρόχων, είτε ακόμα και να καθορίζονται σχήματα υπερκόμβων που διαφέρουν από τη μορφολογία του χώρου επαναλήψεων ή/και του χώρου δεδομένων, δυσκολεύοντας σε κάθε περίπτωση την ανάθεση των υπερκόμβων στους διαθέσιμους επεξεργαστές. Στη διεθνή βιβλιογραφία [OSKO95, CDR97, CDR99, BDRV99] προτείνεται ως βέλτιστο σχήμα απεικόνισης των υπερκόμβων στους επεξεργαστές εκείνο της κυκλικής ανάθεσης κατά στήλες ή γραμμές υπερκόμβων (cyclic columnwise/rowwise tile allocation), καθώς επιτρέπει την προσομοίωση του φαινομένου της σωλήνωσης και την επικάλυψη των υπολογισμών με την απαιτούμενη επικοινωνία [AKPT99, GSK01]. Πάντως, για τη συνήθη απλή περίπτωση ορθογώνιων χώρων επαναλήψεων και ορθογώνιων μετασχηματισμών υπερκόμβων, το σχήμα της κυκλικής ανάθεσης παρουσιάζει σχετική προγραμματιστική πολυπλοκότητα, που πηγάζει κυρίως από το διαχωρισμό της επιλογής του μετασχηματισμού υπερκόμβων από την υφιστάμενη υπολογιστική υποδομή.

Μια σχετικά καινούρια εξέλιξη στον παράλληλο προγραμματισμό, που εμφανίστηκε προς τα τέλη της δεκαετίας του '90, είναι η ανάπτυξη τεχνικών υβριδικής παραλληλοποίησης για τις πολυεπίπεδες αρχιτεκτονικές υψηλών επιδόσεων. Θεωρητικά τουλάχιστον, η υβριδική παραλληλοποίηση αλγορίθμων παρουσιάζει συγκριτικά πλεονεκτήματα σε σχέση με τη συνήθη προσέγγιση της ανταλλαγής μηνυμάτων, στην περίπτωση που η υφιστάμενη αρχιτεκτονική ακολουθεί ιεραρχική δομή διαφοροποιώντας π.χ. το κόστος προσπέλασης των δεδομένων στα διάφορα επίπεδα της ιεραρχίας. Μια σημαντική ομάδα ερευνητών έχει ασχοληθεί με τη θεωρητική ανάλυση του υβριδικού μοντέλου παραλληλοποίησης, κυρίως με χρήση του προτύπου MPI για την ανταλλαγή μηνυμάτων και του προτύπου OpenMP για την επίτευξη πολυνηματικής επεξεργασίας. Στην εργασία [SB01] οι Smith και Bull πραγματοποιούν μια πολύ καλή θεωρητική επισκόπηση των ιδιαιτεροτήτων και των προτερημάτων της υβριδικής παραλληλοποίησης έναντι του μοντέλου ανταλλαγής μηνυμάτων. Οι Rabenseifner και Wellein [Rab02, Rab03, RW03] εστιάζουν κυρίως στην αξιοποίηση της δυνατότητας παροχής δεδομένων του δικτύου διασύνδεσης μέσω του υβριδικού μοντέλου. Οι Cappello, Etiemble, Richard και Krawezik [CRE00, CE00, KC03a, KC03b] μελετούν την επίδοση του υβριδικού μοντέλου με τη βοήθεια των δημοφιλών μετροπρογραμμάτων NAS Parallel Benchmarks (NPB) τόσο σε SMP συστοιχίες όσο και σε συστήματα αμιγώς μοιραζόμενης μνήμης. Αντίστοιχα, οι Ayguadé, González,Martorell και Jost [AGMJ04] και οι Wong, Jin και Becker [WJB04] θεωρούν τη multi-zone εκδοχή των μετροπρογραμμάτων NPB (NPB-MZ) που ευνοεί την πολυεπίπεδη παραλληλοποίηση, και μεταξύ άλλων επισημαίνουν την αναγκαιότητα εξισορρόπησης του φορτίου των νημάτων για την πλήρη αξιοποίηση του υβριδικού μοντέλου. Μάλιστα, στην εργασία [WM03] οι συγγραφείς περιγράφουν εργαλείο λογισμικού για την

ανάλυση της επίδοσης υβριδικών προγραμμάτων.

Η ερευνητική δραστηριότητα πάνω στην υβριδική παραλληλοποίηση αλγορίθμων δεν περιορίζεται στη θεωρητική μελέτη και ανάλυση της επίδοσης με χρήση μετροπρογραμμάτων, αλλά επεκτείνεται και στην εφαρμογή της τεχνικής σε πραγματικά προβλήματα. Πράγματι, παραδείγματα εφαρμογής της υβριδικής παραλληλοποίησης έχουν καταγραφεί στη ρευστοδυναμική [GKKS00, Taf01, DK02, DK04, BDH$^+$02], την οπτική [Maj00, SEKBL04], τη γραμμική άλγεβρα [HD02, WHBF02], τη μετεωρολογία [LTD01], τη μοριακή δυναμική [Hen00, BDH$^+$02], τις αριθμητικές μεθόδους επίλυσης [Nak03, DGR05] κ.ά. Τα συμπεράσματα για την αποδοτικότητα της υβριδικής προσέγγισης είναι σχετικά αμφιλεγόμενα, καθώς σε κάποιες περιπτώσεις έχουν καταγραφεί σημαντικές βελτιώσεις, ενώ στις υπόλοιπες η σύνθετη υβριδική παραλληλοποίηση είτε αδυνατεί να υπερκεράσει σε απόδοση το μοντέλο ανταλλαγής μηνυμάτων ή ακόμα και υστερεί έναντι των ιδιαίτερα βελτιστοποιημένων μονολιθικών MPI υλοποιήσεων.

Σύμφωνα με τη βιβλιογραφία, τα κυριότερα προτερήματα του υβριδικού μοντέλου εντοπίζονται αφενός στην αξιοποίηση της μοιραζόμενης μνήμης για τη *μείωση του χρόνου επικοινωνίας* και αφετέρου στη δυνατότητα εύκολης και αποδοτικής *εξισορρόπησης του φορτίου μεταξύ των νημάτων*. Στην πράξη, έχει διαπιστωθεί ότι η πολυνηματική επεξεργασία ευνοεί τις περιπτώσεις εφαρμογών που επιζητούν λεπτομερή παραλληλοποίηση, καθώς έτσι μπορούν να αξιοποιηθούν αποδοτικότερα τόσο η ιεραρχία μνήμης μέσω της τοπικότητας αναφοράς δεδομένων όσο και οι δυνατότητες εξισορρόπησης του φορτίου των νημάτων. Αντίθετα, η επικοινωνία μέσω ανταλλαγής μηνυμάτων ευνοείται κυρίως από χονδροειδή παραλληλισμό, καθώς έτσι αμβλύνεται η παράλληλη επιβάρυνση της διαχείρισης των διεργασιών και μετριάζεται το κόστος αρχικοποίησης της επικοινωνίας λόγω της αρχικής καθυστέρησης του δικτύου. Το υβριδικό μοντέλο παρουσιάζει συχνά προβλήματα στην επίδοση είτε λόγω συμφόρησης κατά την πρόσβαση στο κοινό υποσύστημα μνήμης, είτε εξαιτίας των λιγότερο βελτιστοποιημένων βιβλιοθηκών πολυνηματικής επεξεργασίας σε σχέση με τις αντίστοιχες βιβλιοθήκες ανταλλαγής μηνυμάτων, είτε τέλος λόγω μη αποδοτικής εξισορρόπησης του φορτίου των νημάτων κατά την απλή υβριδική προσέγγιση. Η σχετικά αυξημένη προγραμματιστική δυσκολία του υβριδικού μοντέλου έχει οδηγήσει στη σχεδόν αποκλειστική εμφάνιση υβριδικών παραλληλοποιήσεων λεπτού κόκκου (fine-grain), όπου εφαρμόζεται επαυξητική παραλληλοποίηση επιλεκτικά σε συγκεκριμένα τμήματα του κώδικα ανταλλαγής μηνυμάτων. Αντίθετα, οι υβριδικές υλοποιήσεις χονδρού κόκκου (coarse-grain), όπου υιοθετείται ένα SPMD προγραμματιστικό μοντέλο εκτός των διεργασιών και ως προς τα νήματα, είναι σχετικά περιορισμένες, αφού άλλωστε δεν υποστηρίζονται επαρκώς από τις υπάρχουσες MPI υλοποιήσεις.

Συνοψίζοντας, ο προγραμματισμός αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης γενικότερα, καθώς και το υβριδικό μοντέλο παραλληλοποίησης ειδικότερα, θέτουν νέες προκλήσεις και αναθεωρούν σε κάποιο βαθμό τον τρόπο προσέγγισης των τεχνικών παραλληλοποίησης. Παρότι θεωρητικά τα υβριδικά μοντέλα προγραμματισμού φαίνονται πιο κατάλληλα για τις πολυεπίπεδες ιεραρχι-

κές αρχιτεκτονικές υψηλών επιδόσεων, η πειραματική αξιολόγηση αυτών καταδεικνύει πως η επίτευξη υψηλής απόδοσης με το υβριδικό μοντέλο είναι πολύπλοκη διαδικασία, που απαιτεί αποδοτικές τεχνικές χρονοδρομολόγησης και εξισορρόπησης φορτίου. Επίσης, ο βαθμός αποδοτικής πολυνηματικής υποστήριξης από τη μεριά της βιβλιοθήκης ανταλλαγής μηνυμάτων αποτελεί πολύ βασικό ζήτημα: συχνά, η επίτευξη υψηλής απόδοσης μέσω υβριδικής παραλληλοποίησης είναι άρρηκτα συνδεδεμένη με την αντιμετώπιση των περιορισμών που επιβάλλει η βιβλιοθήκη ανταλλαγής μηνυμάτων. Η παρούσα εργασία αποτελεί συνέχιση μακρόχρονης ερευνητικής δραστηριότητας στο Εργαστήριο Υπολογιστικών Συστημάτων του Ε.Μ.Π. σχετικά με την αυτόματη παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων [AKPT99, GSK01, Γκο03, Σωτ04, Αθα05] και επιχειρεί να συνδέσει αποδοτικά την απεικόνιση αλγορίθμων φωλιασμένων βρόχων σε σύγχρονες παράλληλες αρχιτεκτονικές με τις τεχνικές υβριδικού προγραμματισμού.

1.3 Συμβολή της Διατριβής

Η παρούσα διατριβή αποτελεί προϊόν εκτενούς έρευνας σχετικά με το ζήτημα της αποδοτικής παραλληλοποίησης αλγοριθμικών περιγραφών φωλιασμένων βρόχων σε συστοιχίες πολυεπεξεργαστικών στοιχείων. Η πορεία και εξέλιξη της διατριβής επηρεάστηκε σε μεγάλο βαθμό αφενός από την επικράτηση των ευέλικτων αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης και αφετέρου από τη διευρυμένη χρήση δημοφιλών προγραμματιστικών διεπαφών, όπως το MPI, για την παραλληλοποίηση υπολογιστικά ή/και χωρικά απαιτητικών αλγορίθμων. Εστιάζοντας λοιπόν στην πολυπληθή κατηγορία των πλήρως αντιμεταθέσιμων βρόχων, η διατριβή προτείνει ολοκληρωμένες μεθόδους αποδοτικής παραλληλοποίησης για τις αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Πιο συγκεκριμένα, η συμβολή της διατριβής εντοπίζεται στους εξής άξονες:

- Περιγράφεται μια ολοκληρωμένη μέθοδος για τον καθορισμό μιας κατάλληλης εικονικής **τοπολογίας διεργασιών**, που μπορεί να χρησιμοποιηθεί για την παράλληλη απεικόνιση ενός πολυδιάστατου αλγορίθμου φωλιασμένων βρόχων. Η μέθοδος αποσκοπεί στην ελαχιστοποίηση του όγκου των δεδομένων επικοινωνίας που ανταλλάσσει μια τυχούσα διεργασία με τις γειτονικές της. Έτσι, επιχειρεί να αμβλύνει τις καθυστερήσεις που σχετίζονται με το σημαντικό κόστος επικοινωνίας τέτοιων παράλληλων αλγορίθμων σε κατανεμημένα περιβάλλοντα, ιδίως σε συμβατικά δίκτυα διασύνδεσης. Θεωρώντας ως βασική παραδοχή το ρεαλιστικό περιορισμό της ύπαρξης ενός συγκεκριμένου, πεπερασμένου πλήθους διεργασιών P για την παράλληλη εκτέλεση του αλγορίθμου, η προτεινόμενη μέθοδος λαμβάνει υπόψη το χώρο επαναλήψεων και τις εξαρτήσεις δεδομένων ενός αλγορίθμου φωλιασμένων βρόχων διάστασης $N+1$ και προσδιορίζει κατάλληλη N-διάστατη εικονική τοπολογία διεργασιών $P = P_1 \times \cdots \times P_N$ για την παράλληλη απεικόνιση

του αλγορίθμου αυτού.

- Ισοδύναμα, περιγράφεται αυτόματη τεχνική για τον υπολογισμό κατάλληλου **μετασχηματισμού υπερκόμβων**, που ελαχιστοποιεί την επικοινωνία μεταξύ διεργασιών κατά την παράλληλη εκτέλεση του προγράμματος σε πεπερασμένο πλήθος διεργασιών. Αναδεικνύεται ο ισομορφισμός μεταξύ των δύο προβλημάτων, δηλαδή αυτού της επιλογής κατάλληλης καρτεσιανής τοπολογίας διεργασιών και εκείνου του προσδιορισμού κατάλληλου μετασχηματισμού υπερκόμβων. Έτσι, συσχετίζεται η ιδιαίτερα δημοφιλής παράλληλη προγραμματιστική πρακτική διανυσματοποίησης του χώρου των διεργασιών, που συνταιριάζει αρμονικά τις μονάδες εκτέλεσης του παράλληλου προγράμματος με το φυσικό πεδίο εφαρμογής (application domain) του προβλήματος, με έναν ιδιαίτερα διαδεδομένο μη γραμμικό μετασχηματισμό της διεθνούς βιβλιογραφίας, το μετασχηματισμό υπερκόμβων. Ο προτεινόμενος μετασχηματισμός συγκρίνεται πειραματικά με τη συνήθη πρακτική, που αποσκοπεί στην ελαχιστοποίηση του αριθμού των παράλληλων βημάτων εκτέλεσης, χωρίς όμως να λαμβάνει υπόψη τη φύση του συγκεκριμένου υπό παραλληλοποίηση αλγορίθμου (χώρος επαναλήψεων, εξαρτήσεις δεδομένων). Στο πλαίσιο της πειραματικής αξιολόγησης, καταγράφονται σημαντικές βελτιώσεις της επίδοσης του προγράμματος κατά την υιοθέτηση της προτεινόμενης τοπολογίας διεργασιών.

- Γίνεται εκτενής προτυποποίηση, διερεύνηση και αξιολόγηση διαφορετικών **τεχνικών υβριδικής παραλληλοποίησης** για τους αλγορίθμους φωλιασμένων βρόχων. Συγκεκριμένα, προδιαγράφονται υλοποιήσεις σε MPI-OpenMP λεπτού κόκκου, όπου εφαρμόζεται επιλεκτικά επαυξητική παραλληλοποίηση υπολογιστικά απαιτητικών τμημάτων του κώδικα ανταλλαγής μηνυμάτων με χρήση οδηγιών πολυνηματικής επεξεργασίας. Επιπλέον, προτείνονται και προδιαγράφονται υβριδικές υλοποιήσεις χονδρού κόκκου, στις οποίες η πολυνηματική ροή εκτέλεσης υιοθετείται σχεδόν στο σύνολο του προγράμματος με κάθε νήμα να διαχωρίζει το υποσύνολο δεδομένων και λειτουργιών του βάσει του μοναδικού αναγνωριστικού του κατά το SPMD πρότυπο, παρόμοια με τις διεργασίες στο μοντέλο ανταλλαγής μηνυμάτων. Εξετάζεται η περίπτωση τόσο της πλήρους όσο και της μερικής πολυνηματικής υποστήριξης από τη μεριά της βιβλιοθήκης ανταλλαγής μηνυμάτων, ανάλογα με το κατά πόσο είναι επιτρεπτό να γίνονται κλήσεις σε λειτουργίες επικοινωνίας εντός της εμβέλειας πολυνηματικών περιοχών. Παράλληλα, εντοπίζονται και επισημαίνονται τα εγγενή μειονεκτήματα της πιο συχνά χρησιμοποιούμενης προσέγγισης υβριδικού προγραμματισμού, που σε αρκετές περιπτώσεις ακυρώνουν πολλά από τα πλεονεκτήματα αυτού, καταλήγοντας ακόμα και σε χειρότερη επίδοση σε σχέση με τη μονολιθική παραλληλοποίηση μέσω ανταλλαγής μηνυμάτων.

- Προτείνονται τεχνικές αποδοτικής **εξισορρόπησης του φορτίου των νημάτων** στο υβριδικό μο-

ντέλο. Η υιοθέτηση τέτοιων τεχνικών κρίνεται επιτακτική για την περίπτωση της μερικής πολυνηματικής υποστήριξης, που στην παρούσα φάση προσφέρεται σχεδόν από το σύνολο των διαθέσιμων MPI υλοποιήσεων, ενώ μπορεί να αμβλύνει την επιβάρυνση στην περίπτωση της πλήρους πολυνηματικής υποστήριξης, που προσφέρεται σε λίγες περιπτώσεις υλοποιήσεων MPI εμπορικού τύπου. Οι προτεινόμενες τεχνικές εξισορρόπησης φορτίου δίνουν έμφαση στην απλότητα και την εφαρμοσιμότητα και επιτρέπουν στις υβριδικές υλοποιήσεις να επιτυγχάνουν χαμηλότερους χρόνους εκτέλεσης σε σχέση με τη μονολιθική παραλληλοποίηση ανταλλαγής μηνυμάτων. Η δυνατότητα για ένα ιδιαίτερα αποδοτικό υβριδικό προγραμματιστικό μοντέλο είναι πολύ επιθυμητή, καθώς μεταξύ άλλων συμβάλλει στην ευελιξία επιλογής τρόπου παραλληλοποίησης ανά περίπτωση αλγορίθμου, παρέχοντας μεγαλύτερη ελευθερία επιλογής τόσο της τοπολογίας των διεργασιών όσο και εκείνης των νημάτων. Τα δύο επίπεδα βελτιστοποίησης, δηλαδή αυτό της παράλληλης απεικόνισης σε τοπολογία διεργασιών, καθώς και εκείνο της εξισορρόπησης του φορτίου των νημάτων, είναι ορθογώνια και έτσι τα επιμέρους οφέλη που αποδίδουν σε ό,τι αφορά στη μείωση των χρόνων εκτέλεσης λειτουργούν αθροιστικά.

- Όλες οι προτεινόμενες τεχνικές και βελτιστοποιήσεις εφαρμόζονται και αξιολογούνται κατά την παραλληλοποίηση πραγματικών εφαρμογών, που ως επί το πλείστον προέρχονται από το χώρο των *Μερικών Διαφορικών Εξισώσεων*. Η σύνδεση οποιασδήποτε προγραμματιστικής μεθοδολογίας με υπαρκτά προβλήματα από άλλα επιστημονικά πεδία αποτελεί άλλωστε και τον απώτερο στόχο στο χώρο της Παράλληλης Επεξεργασίας.

1.4 Οργάνωση της Διατριβής

Στο παρόν κεφάλαιο παρουσιάζεται συνοπτικά το αντικείμενο της διατριβής. Επιχειρείται βιβλιογραφική επισκόπηση της θεματικής περιοχής και επισημαίνεται η συμβολή της διατριβής στο ζήτημα της αποδοτικής παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Επίσης, καταγράφονται οι δημοσιεύσεις που προέκυψαν κατά την εκπόνηση της διατριβής.

Στο Κεφάλαιο 2 γίνεται εισαγωγή στο χώρο της Παράλληλης Επεξεργασίας, τόσο από τη σκοπιά των αρχιτεκτονικών υψηλών επιδόσεων όσο και αναφορικά με τα βασικότερα μοντέλα παράλληλου προγραμματισμού.

Στο Κεφάλαιο 3 προσδιορίζεται θεωρητικά το εξεταζόμενο αλγοριθμικό μοντέλο. Ορίζονται θεμελιώδεις έννοιες για την παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων, όπως ο χώρος επαναλήψεων και οι εξαρτήσεις δεδομένων. Επιπλέον, γίνεται αναφορά στη διακριτοποίηση Μερικών Διαφορικών Εξισώσεων, που συνιστούν μια τυπική κατηγορία αλγοριθμικών περιγραφών στις οποίες μπορούν

να εφαρμοστούν οι προτεινόμενες τεχνικές παραλληλοποίησης.

Στο Κεφάλαιο 4 παρουσιάζεται ο μετασχηματισμός υπερκόμβων, που χρησιμοποιούμε για την επίτευξη χονδροειδούς παραλληλισμού σε αρχιτεκτονικές κατανεμημένης μνήμης, και περιγράφεται μέθοδος παραλληλοποίησης αλγορίθμων πλήρως αντιμεταθέσιμων βρόχων μέσω ανταλλαγής μηνυμάτων με δυνατότητα επικάλυψης υπολογισμού και επικοινωνίας. Προτείνεται μια αυτόματη τεχνική για τον καθορισμό εικονικής τοπολογίας απεικόνισης διεργασιών ελάχιστης επικοινωνίας συναρτήσει του χώρου επαναλήψεων και των εξαρτήσεων δεδομένων του αλγορίθμου. Επισημαίνεται η ισοδυναμία της προτεινόμενης τοπολογίας με τον καθορισμό μετασχηματισμού υπερκόμβων που ελαχιστοποιεί το συνολικό όγκο επικοινωνίας μεταξύ των διεργασιών.

Στο Κεφάλαιο 5 προτείνονται τεχνικές υβριδικής παραλληλοποίησης του υπό εξέταση αλγοριθμικού μοντέλου. Αναλυτικότερα, περιγράφονται τα επίπεδα πολυνηματικής υποστήριξης που παρέχουν οι βιβλιοθήκες ανταλλαγής μηνυμάτων και εφαρμόζεται η χρονοδρομολόγηση υπερεπιπέδων για την υλοποίηση εναλλακτικών υβριδικών μοντέλων τόσο λεπτού όσο και χονδρού κόκκου (εκδοχές funneled και multiple). Προτείνονται τεχνικές στατικής και δυναμικής εξισορρόπησης του υπολογιστικού φορτίου μεταξύ των νημάτων για τη συνήθη περίπτωση της περιορισμένης πολυνηματικής υποστήριξης. Επιχειρείται θεωρητική σύγκριση των χρόνων εκτέλεσης του μοντέλου ανταλλαγής μηνυμάτων με τον αντίστοιχο χρόνο εκτέλεσης κατά την υβριδική παραλληλοποίηση, ώστε να διερευνηθούν θεωρητικά οι δυνατότητες για επίτευξη υψηλής επίδοσης μέσω της υλοποίησης βελτιστοποιημένων υβριδικών μοντέλων.

Στο Κεφάλαιο 6 πραγματοποιείται πειραματική αξιολόγηση των εναλλακτικών μοντέλων παραλληλοποίησης και των προτεινόμενων τεχνικών βελτιστοποίησης. Συγκεκριμένα, διερευνούμε το συνολικό χρόνο παράλληλης εκτέλεσης του μοντέλου ανταλλαγής μηνυμάτων κατά την επιλογή αφενός της τοπολογίας διεργασιών ελάχιστης καθυστέρησης και αφετέρου της προτεινόμενης τοπολογίας ελάχιστης επικοινωνίας. Στη συνέχεια, επιχειρείται σύγκριση της επίδοσης των υβριδικών μοντέλων παραλληλοποίησης και αξιολόγηση της αποτελεσματικότητας των τεχνικών εξισορρόπησης φορτίου. Τέλος, παρουσιάζεται συνολική σύγκριση της επίδοσης όλων των παράλληλων προγραμματιστικών μοντέλων και αξιολογούνται οι βελτιώσεις που μπορούν να επιτευχθούν με χρήση των προτεινόμενων μεθόδων και τεχνικών.

Στο Κεφάλαιο 7 ολοκληρώνεται η διατριβή με σύνοψη και παρουσίαση όλων των συμπερασμάτων που προέκυψαν, ενώ παράλληλα προτείνονται μελλοντικές κατευθύνσεις συνέχισης της ερευνητικής δραστηριότητας.

1.5 Δημοσιευμένες Εργασίες

Η ερευνητική δραστηριότητα αποτυπώθηκε σε μια σειρά εργασιών, που δημοσιεύτηκαν ή έχουν γίνει αποδεκτές για δημοσίευση σε διεθνή συνέδρια και περιοδικά που ακολουθούν τη μέθοδο των κριτών. Ακολούθως παρατίθενται επιλεγμένες ερευνητικές εργασίες, που εκπονήθηκαν στο πλαίσιο της διδακτορικής διατριβής. Για πληρότητα, όπου υπάρχουν, παρέχονται επιπλέον και οι σχετικοί υπερσύνδεσμοι στα ψηφιακά αναγνωριστικά αντικειμένου (digital object identifiers - DOI), ώστε ο ενδιαφερόμενος αναγνώστης να μπορεί να αναζητήσει τις σχετικές εργασίες στο Διαδίκτυο.

- Δημοσιεύσεις σε περιοδικά:

    1. N. Drosinos and N. Koziris. *Efficient Hybrid Parallelization of Tiled Algorithms on SMP Clusters.* Accepted for publication on the *International Journal of Computational Science and Engineering*, 2006.
    2. N. Drosinos and N. Koziris. *The Effect of Process Topology and Load Balancing on Parallel Programming Models for SMP Clusters and Iterative Algorithms.* In *Journal of Supercomputing*, 35(1):65–91, January 2006.
       DOI: `http://dx.doi.org/10.1007/s11227-006-1156-z`
    3. G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. *Message-Passing Code Generation for Non-rectangular Tiling Transformations.* Accepted for publication on the *Journal of Parallel Computing*, 2006.

- Δημοσιεύσεις σε συνέδρια:

    1. N. Drosinos, G. Goumas and N. Koziris. *Selecting the Tile Shape to Reduce the Total Communication Volume.* In *Proceedings of the 20th International Parallel and Distributed Processing Symposium 2006 (IPDPS 2006)*, Rhodes, Greece, April 2006.
    2. N. Drosinos and N. Koziris. *Load Balancing Hybrid Programming Models for SMP Clusters and Fully P ermutable Loops.* In *Proceedings of the 7th International Workshop on High Performance Scientific and Engineering Computing (ICPP-HPSEC 2005)*, pages 113–120, Oslo, Norway, June 2005.
       DOI: `http://dx.doi.org/10.1109/ICPPW.2005.46`
    3. N. Drosinos and N. Koziris. *Performance Comparison of Pure MPI vs Hybrid MPI-OpenMP Parallelization Models on SMP Clusters.* In *Proceedings of the 18th International Parallel and Distributed Processing Symposium 2004 (IPDPS 2004)*, page 15, Santa Fe, New Mexico, April

2004.
DOI: http://doi.ieeecomputersociety.org/10.1109/IPDPS.2004.1302919

4. G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. *Automatic Parallel Code Generation for Tiled Nested Loops.* In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*, pages 1412–1419, Nicosia, Cyprus, March 2004.
DOI: http://doi.acm.org/10.1145/967900.968184

5. N. Drosinos and N. Koziris. *Advanced Hybrid MPI/OpenMP Parallelization Paradigms for Nested Loop Algorithms onto Clusters of SMPs.* In *Proceedings of the 10th EuroPVM/MPI Conference 2003 (EuroPVM/MPI 2003)*, pages 203–213, Venice, Italy, September 2003.
DOI: http://dx.doi.org/10.1007/b14070

6. N. Drosinos, G. Goumas, M. Athanasaki, and N. Koziris. *Delivering High Performance to Parallel Applications Using Advanced Scheduling.* In *Proceedings of the Parallel Computing 2003 (ParCo 2003)*, pages 233–240, Dresden, Germany, September 2003.

7. G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. *Compiling Tiled Iteration Spaces for Clusters.* In *Proceedings of the IEEE International Conference on Cluster Computing*, pages 360–369, Chicago, Illinois, September 2002.
DOI: http://doi.ieeecomputersociety.org/10.1109/CLUSTR.2002.1137768

8. G. Goumas, N. Drosinos, M. Athanasaki, and N. Koziris. *Data Parallel Code Generation for Arbitrarily Tiled Nested Loops.* In *Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 610–616, Las Vegas, Nevada, USA, June 2002.

ΚΕΦΑΛΑΙΟ 2

Παράλληλη Επεξεργασία

Ο χώρος της παράλληλης επεξεργασίας έχει ιδιαίτερη χρησιμότητα για τη σύγχρονη επιστημονική κοινότητα, καθώς στοχεύει στη γεφύρωση των προσφερόμενων υπολογιστικών *δυνατοτήτων*, που καθορίζονται από τις σύγχρονες τεχνολογικές εξελίξεις, με τις υπολογιστικές *απαιτήσεις*, που επιβάλλουν οι ποικιλόμορφες επιστημονικές εφαρμογές. Σχηματικά, θα μπορούσαμε να κάνουμε ένα διαχωρισμό μεταξύ του αρχιτεκτονικού σκέλους της παράλληλης επεξεργασίας, που αναφέρεται στις εναλλακτικές υπολογιστικές υποδομές, και των μοντέλων του παράλληλου προγραμματισμού, που υιοθετούνται κατά την παραλληλοποίηση αλγορίθμων. Μολονότι τα δύο αυτά ζητήματα είναι άρρηκτα συνδεδεμένα μεταξύ τους, καθώς ουσιαστικά συνδέονται από την αναγκαιότητα συγκερασμού του υλοποιούμενου λογισμικού με το υφιστάμενο υλικό, θα ακολουθήσουμε τον παραπάνω διαχωρισμό προς απλούστευση της ανάλυσής μας.

2.1 Αρχιτεκτονικές Παράλληλης Επεξεργασίας

Ως παράλληλη αρχιτεκτονική νοείται ένα πεπερασμένο σύνολο μονάδων επεξεργασίας (processing units), που επικοινωνούν μεταξύ τους για την επίλυση ενός προβλήματος αυξημένης χωρικής ή/και χρονικής πολυπλοκότητας. Η εξέλιξη των αρχιτεκτονικών παράλληλης επεξεργασίας έχει να παρουσιάσει μια μεγάλη ποικιλομορφία αρχιτεκτονικών προσεγγίσεων, από ογκώδεις και ακριβούς υπερυπολογιστές ειδικής χρήσεως (special-purpose supercomputers), μέχρι συστοιχίες χιλιάδων κόμβων, διασυνδεδεμένων με ένα ή περισσότερα δίκτυα σε μία ή περισσότερες δικτυακές τοπολογίες (π.χ. δενδρική,

υπερκύβου, πλέγματος κ.ά.). Προς την κατεύθυνση μιας έγκριτης καταγραφής των διαφόρων υπολογιστικών συστημάτων υψηλών επιδόσεων, η λίστα TOP500 [DMS99] ενημερώνεται από το 1993 δύο φορές το χρόνο και διατηρεί μια κατάταξη των 500 ισχυρότερων υπολογιστικών συστημάτων παγκοσμίως, με κριτήριο την επίδοση των συστημάτων αυτών στο μετροπρόγραμμα Linpack.

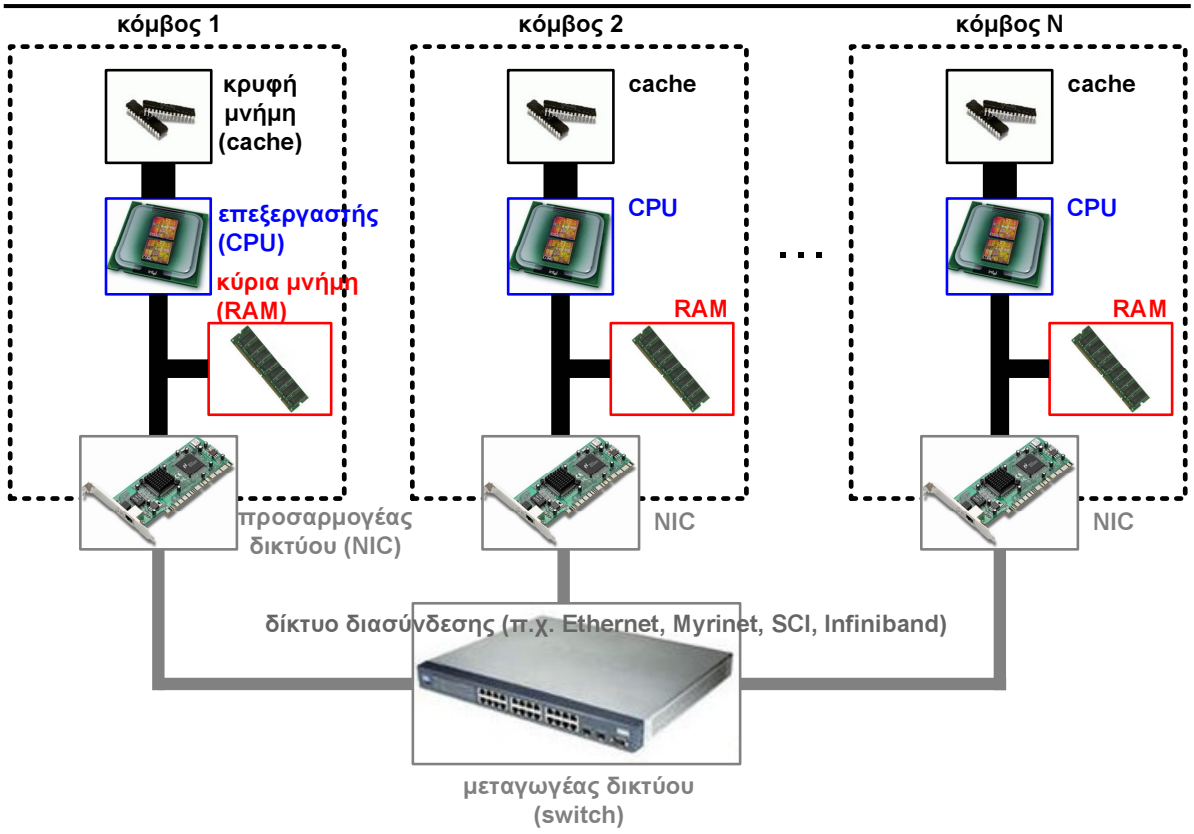
Παρά την πλούσια ποικιλομορφία των υπερυπολογιστικών συστημάτων, έγινε εξαρχής εμφανής η δυνατότητα ομαδοποίησης αυτών σε δύο θεμελιώδεις κατηγορίες, με γνώμονα το αν παρέχεται από το υλικό ένας κοινός, καθολικός χώρος διευθύνσεων μνήμης ή όχι. Θα πρέπει δε να τονιστεί πως η διάκριση αυτή είναι ιδιαίτερα ουσιαστική σε ό,τι αφορά τον προγραμματισμό τέτοιων συστημάτων και έτσι αναφερόμαστε συχνά σε αρχιτεκτονικές μοιραζόμενης μνήμης και αρχιτεκτονικές κατανεμημένης μνήμης. Τα τελευταία χρόνια έχει επικρατήσει μια τρίτη, υβριδική αρχιτεκτονική, που υιοθετεί την προσέγγιση της μοιραζόμενης μνήμης σε χαμηλότερο επίπεδο, ενώ προσομοιάζει στην αρχιτεκτονική κατανεμημένης μνήμης σε ένα δεύτερο, υψηλότερο επίπεδο. Η προσέγγιση αυτή χαρακτηρίζεται ως αρχιτεκτονική κατανεμημένης μοιραζόμενης μνήμης.

2.1.1 Αρχιτεκτονική Κατανεμημένης Μνήμης

Στην αρχιτεκτονική κατανεμημένης μνήμης κάθε υπολογιστικός κόμβος του συστήματος διαθέτει τη δική του *τοπική μνήμη* (*local memory*), στην οποία δεν παρέχεται άμεση πρόσβαση μέσω του υλικού στους άλλους κόμβους. Έτσι, προκειμένου να ανταλλάξουν δεδομένα δύο κόμβοι, συνήθως προτείνονται οι εξής δύο εναλλακτικές προσεγγίσεις:

- Η υλοποίηση με χρήση λογισμικού/μεσισμικού ενός εικονικού καθολικού χώρου διευθύνσεων (Distributed Shared Virtual Memory - DSVM, συχνά χαρακτηρίζεται και Software Distributed Shared Memory - SDSM). Οι εφαρμογές που εκτελούνται σε διαφορετικούς κόμβους μπορούν να επικοινωνούν με συνήθεις λειτουργίες ανάγνωσης και εγγραφής, αξιοποιώντας το DSVM υπόστρωμα. Ουσιαστικά, το DSVM υπόστρωμα αναλαμβάνει να υλοποιήσει σε επίπεδο λογισμικού παρόμοια λειτουργικότητα με εκείνη που παρέχεται στα συστήματα μοιραζόμενης μνήμης σε επίπεδο υλικού.

- Η ρητή (explicit) κλήση ρουτινών επικοινωνίας μέσω δικτύου, είτε για την ανταλλαγή μηνυμάτων (message passing), είτε για την υλοποίηση απομακρυσμένης προσπέλασης σε μνήμη (remote memory access - RMA). Στην πρώτη περίπτωση αναφερόμαστε σε αμφίδρομη επικοινωνία (two-sided communication), ενώ στη δεύτερη σε μονόδρομη επικοινωνία (one-sided communication). Η προσέγγιση αυτή έχει άμεσο αντίκτυπο στη σχεδίαση του λογισμικού τέτοιων αρχιτεκτονικών, καθώς πρέπει να γίνεται σαφής διαχωρισμός μεταξύ των φάσεων τοπικής επεξεργασίας και απομακρυσμένης δικτυακής προσπέλασης δεδομένων.



Σχήμα 2.1: *Αρχιτεκτονική κατανεμημένης μνήμης*

Σχηματικά, η αρχιτεκτονική κατανεμημένης μνήμης προσομοιάζει εκείνη του σχήματος 2.1. Δημοφιλέστερη εκδοχή της αρχιτεκτονικής κατανεμημένης μνήμης είναι η *συστοιχία υπολογιστών* (*cluster of computers*), όπου κάθε επεξεργαστικός κόμβος διαθέτει τη δική του τοπική μνήμη RAM. Οι κόμβοι διασυνδέονται μέσω κάποιου *δικτύου διασύνδεσης* (*interconnection network*), το οποίο επιτρέπει την υλοποίηση μεθόδων επικοινωνίας μεταξύ των διαφορετικών κόμβων.

Στη συγκεκριμένη αρχιτεκτονική είναι σαφές πως το δίκτυο διασύνδεσης έχει κομβικό ρόλο, γι' αυτό και θα αναφερθούμε περαιτέρω στα βασικά στοιχεία που το χαρακτηρίζουν, ήτοι την *αρχική καθυστέρηση* (*latency*) και τη *δυνατότητα παροχής* (*bandwidth*). Η αρχική καθυστέρηση ενός δικτύου διασύνδεσης $t_{startup}$ σχετίζεται με την επιβάρυνση που επιφέρει το δίκτυο κατά τη μετάδοση της ελάχιστης ποσότητας δεδομένων μεταξύ δύο κόμβων και αποτελεί συνισταμένη της επιβάρυνσης λόγω ενδιάμεσων αντιγραφών και επεξεργασίας των δεδομένων επικοινωνίας, σε επίπεδο υλικού και λογισμικού. Επειδή είναι πρακτικά δύσκολο να εκτιμήσουμε επακριβώς την αρχική καθυστέρηση ενός δικτύου διασύνδεσης, συνήθως αυτή προσεγγιστικά εξισώνεται με το χρόνο round-trip (round-trip time). Έτσι, θεωρού-

με την αρχική καθυστέρηση πρακτικά σταθερή και ίση με το χρόνο που απαιτείται για την ανταλλαγή του μηνύματος με το ελάχιστο μέγεθος του ενός byte μεταξύ δύο κόμβων για ένα μεγάλο πλήθος επαναλήψεων, διαιρεμένο δια του δύο (μονόδρομη καθυστέρηση) και δια του πλήθους των επαναλήψεων.

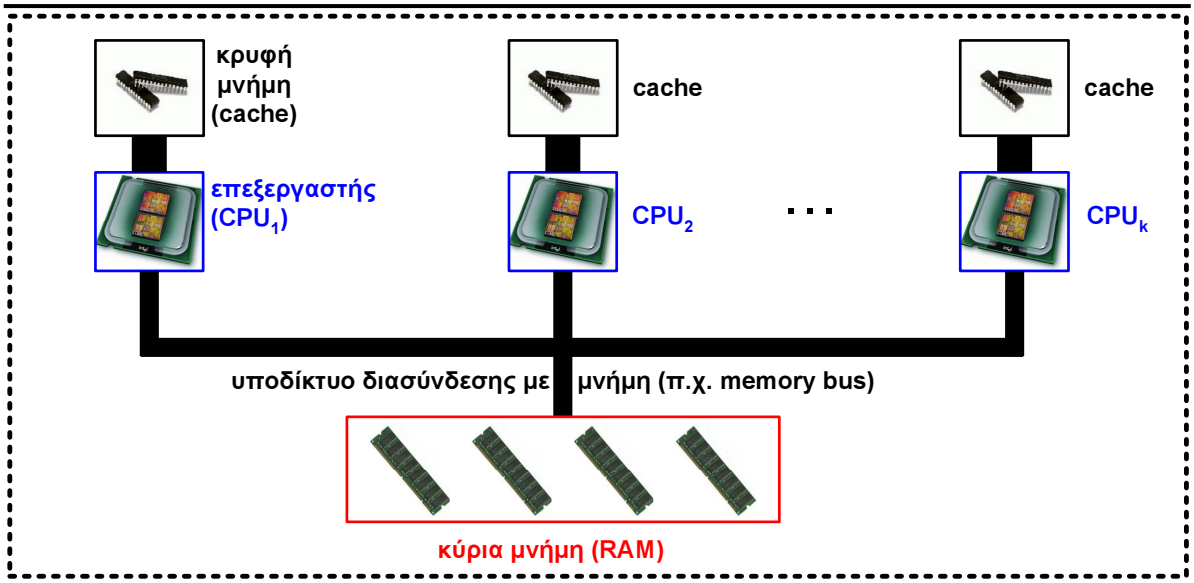
Η δυνατότητα παροχής B παρέχει ένα μέτρο της μέγιστης δυνατότητας μετάδοσης δεδομένων από το δίκτυο διασύνδεσης. Προκειμένου η δυνατότητα παροχής να διαχωριστεί από την αρχική καθυστέρηση του δικτύου διασύνδεσης, θεωρούμε μια μεγάλη ποσότητα δεδομένων (θεωρητικά άπειρη), την οποία επιθυμούμε να στείλουμε από κάποιο κόμβο σε κάποιο άλλο απομακρυσμένο κόμβο μέσω του δικτύου διασύνδεσης. Ο λόγος του πλήθους των δεδομένων προς το χρόνο που απαιτείται για τη μετάδοσή τους μέσω του δικτύου διασύνδεσης ορίζει τη δυνατότητα παροχής του δικτύου. Π.χ., στο δημοφιλές 100 Mbps FastEthernet, η δυνατότητα παροχής ορίζεται ως $B = 100$ Mbps ή $B = 12.5$ MB/s. Θεωρητικά συνεπώς, αν αποστέλλαμε πολύ μεγάλη ποσότητα δεδομένων D μέσω του δικτύου FastEthernet θα χρειαζόταν χρόνος ίσος με D/B για να φτάσουν τα δεδομένα στον προορισμό τους.

Στην πράξη, η δυνατότητα παροχής παρέχει ένα εξιδανικευμένο μέτρο της ταχύτητας του δικτύου διασύνδεσης, καθώς σε επίπεδο εφαρμογής η ταχύτητα μετάδοσης των δεδομένων εξαρτάται και από την αρχική καθυστέρηση του δικτύου. Για να πάρουμε ένα πιο ρεαλιστικό μέτρο της επίδοσης του δικτύου, ορίζουμε το *ρυθμό παροχής* (*throughput*) ως το πηλίκο μιας ποσότητας δεδομένων που αποστέλλονται προς τον πραγματικό χρόνο που καταγράφεται σε επίπεδο εφαρμογής. Ο ρυθμός παροχής $B_{sustained}$ είναι συνάρτηση του πλήθους των δεδομένων επικοινωνίας, καθώς σε λιγότερα δεδομένα γίνεται περισσότερο εμφανής η αρχική καθυστέρηση του δικτύου διασύνδεσης. Σε όλες τις περιπτώσεις πάντως, θα ισχύει προφανώς $B_{sustained} \leq B$. Συνήθως, ο ρυθμός παροχής δικτύου για κάποιο συγκεκριμένο μέγεθος μηνύματος προσδιορίζεται από κατάλληλο μετροπρόγραμμα ping-pong, όπου δύο διεργασίες εναλλάσσουν τους ρόλους του αποστολέα και του παραλήπτη και ανταλλάσσουν μηνύματα του συγκεκριμένου μεγέθους για ένα μεγάλο πλήθος επαναλήψεων. Ο ρυθμός παροχής υπολογίζεται ως ο συνολικός όγκος των δεδομένων που εστάλησαν διαιρεμένος τόσο με το συνολικό χρόνο όσο και με το συνολικό πλήθος των επαναλήψεων.

2.1.2 Αρχιτεκτονική Μοιραζόμενης Μνήμης

Η αρχιτεκτονική μοιραζόμενης μνήμης είναι η δυαδική του συστήματος κατανεμημένης μνήμης. Στην περίπτωση αυτή, το υλικό παρέχει σε όλους τους επεξεργαστικούς κόμβους τη δυνατότητα πρόσβασης στον ίδιο, ενιαίο χώρο διευθύνσεων μνήμης. Έτσι, η μεταξύ των κόμβων επικοινωνία σε κάποια εφαρμογή μπορεί να υλοποιηθεί μέσω απλών λειτουργιών ανάγνωσης/εγγραφής (read/write) στον κοινό χώρο μνήμης.

Συνοπτικά, η αρχιτεκτονική μοιραζόμενης μνήμης μπορεί να παρασταθεί όπως στο σχήμα 2.2. Στη συνήθη περίπτωση, πρόκειται για κάποιον πολυεπεξεργαστικό κόμβο (multi-processing node), ο οποί-



Σχήμα 2.2: *Αρχιτεκτονική μοιραζόμενης μνήμης*

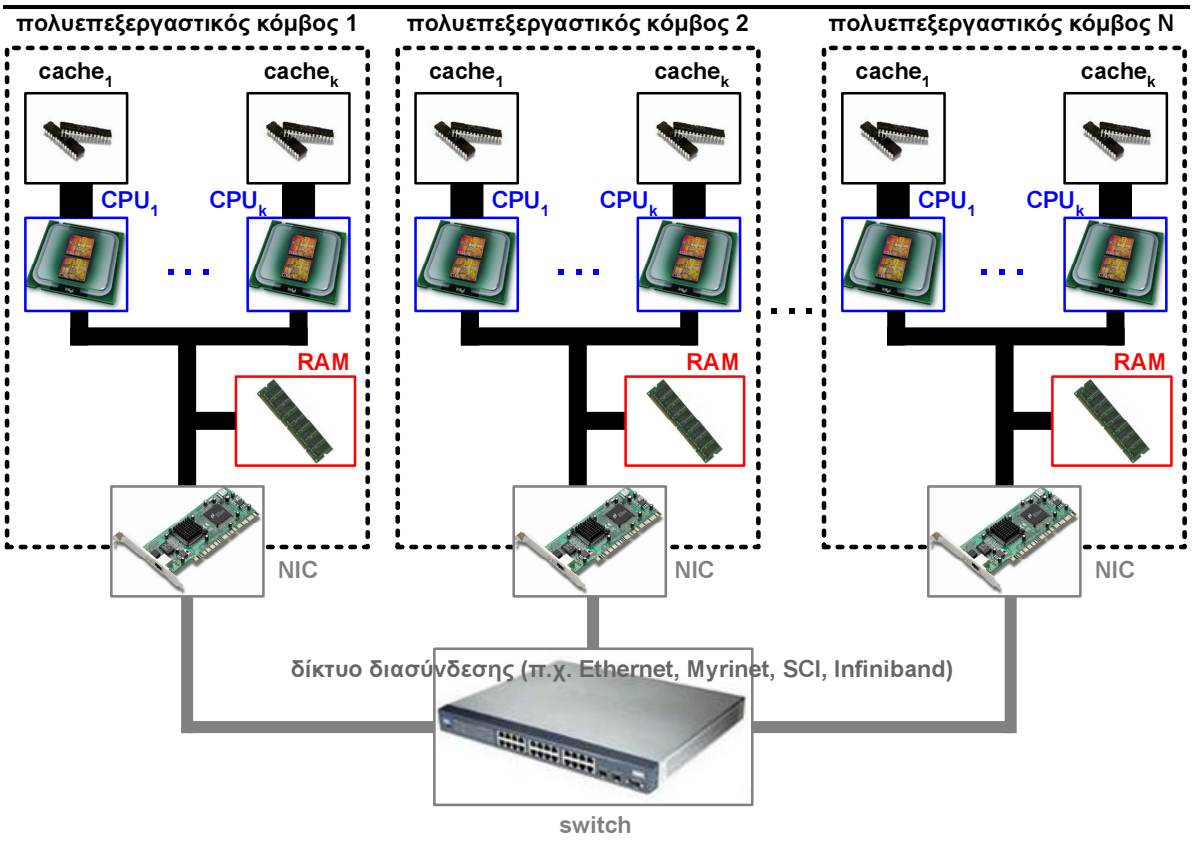
ος διαθέτει περισσότερους επεξεργαστές. Κάθε επεξεργαστής έχει τη δική του ιδιωτική κρυφή μνήμη (cache memory), αλλά μοιράζεται μια κοινή κύρια μνήμη (main memory) με τους υπόλοιπους επεξεργαστές. Ο ενιαίος χώρος διευθύνσεων προσπελάζεται με ομοιόμορφο τρόπο από όλους τους επεξεργαστές, αλλά κάθε επεξεργαστής διαθέτει τη δική του κρυφή μνήμη για διατήρηση των δεδομένων στα οποία αναφέρεται συχνά. Η συνέπεια (consistency) μεταξύ της κύριας μνήμης και των κρυφών μνημών των επεξεργαστών διασφαλίζεται από το υλικό, με τη χρήση κατάλληλου πρωτοκόλλου (π.χ. MSI, MESI, Dragon κ.ά.).

2.1.3 Αρχιτεκτονική Κατανεμημένης Μοιραζόμενης Μνήμης

Οι δυο βασικές αρχιτεκτονικές προσεγγίσεις, δηλαδή τόσο της κατανεμημένης όσο και της μοιραζόμενης μνήμης, παρουσιάζουν σημαντικές διαφορές στα προτερήματα και τα μειονεκτήματα. Πιο συγκεκριμένα, η αρχιτεκτονική μοιραζόμενης μνήμης παρέχει προγραμματιστική ευκολία, καθώς η επικοινωνία μεταξύ των κόμβων μπορεί να επιτευχθεί με τη χρήση συνήθων λειτουργιών ανάγνωσης και εγγραφής. Παρέχει μια ενιαία, καθολική εικόνα του συστήματος μνήμης, καθιστώντας έτσι την αρχιτεκτονική αυτή πιο ευέλικτη στη διαχείριση των δεδομένων, καθώς δεν απαιτούνται πολλαπλά αντίγραφα για το ίδιο δεδομένο, αφού όλοι οι κόμβοι μπορούν να το προσπελάσουν από την κοινή μνήμη. Η αρχιτεκτονική μοιραζόμενης μνήμης είναι ιδιαίτερα δημοφιλής σε περιβάλλοντα εξυπηρετητών (servers).

Στον αντίποδα, η ύπαρξη κοινού υποσυστήματος μνήμης περιορίζει δραστικά την επεκτασιμότητα

(scalability) των συστημάτων αυτών: δεδομένου ότι ο συνολικός διατιθέμενος ρυθμός παροχής δεδομένων του υποδικτύου μνήμης καθορίζεται από τις τεχνολογικές εξελίξεις, ο αριθμός των επεξεργαστών που μπορούν να υποστηριχθούν σε ένα τέτοιο σύστημα είναι περιορισμένος. Για παράδειγμα, ένας κόμβος συμμετρικής πολυεπεξεργασίας (Symmetric Multi-Processing node - SMP node) μπορεί να υποστηρίξει με τη σημερινή τεχνολογία το πολύ λίγες δεκάδες επεξεργαστών, στην πράξη δε σπανίως συναντάμε συστήματα μοιραζόμενης μνήμης με διψήφιο πλήθος επεξεργαστών. Το μειονέκτημα αυτό επιτείνεται ιδιαίτερα σε αρχιτεκτονικές που παρέχουν μοναδικό μονοπάτι εισόδου/εξόδου (I/O path), π.χ. σε περίπτωση αρχιτεκτονικής διαύλου μνήμης (memory bus). Στον πολυνηματικό προγραμματισμό, το μοναδικό αυτό μονοπάτι διαμοιράζεται κατ' ανάγκη μεταξύ του συνόλου των νημάτων που επιθυμούν να προσπελάσουν ταυτόχρονα το μοιραζόμενο χώρο διευθύνσεων, δημιουργώντας έτσι προβλήματα συμφόρησης (congestion) και περιορίζοντας τη συνολική μέγιστη δυνατή επίδοση του προγράμματος.



Σχήμα 2.3: *Αρχιτεκτονική κατανεμημένης μοιραζόμενης μνήμης*

Η αρχιτεκτονική κατανεμημένης μνήμης αντιμετωπίζει επιτυχώς τα παραπάνω προβλήματα, καθώς

η προσθήκη περαιτέρω κόμβων διασφαλίζει την επεκτασιμότητα της συνολικής επίδοσης του συστήματος. Πράγματι, στην περίπτωση αυτή κάθε επεξεργαστής διαθέτει το δικό του υποδίκτυο προσπέλασης της μνήμης, συνεπώς η επέκταση του συστήματος σε μεγαλύτερο αριθμό κόμβων συνεπάγεται και αύξηση του συνολικού διατιθέμενου ρυθμού παροχής δεδομένων του υποσυστήματος μνήμης. Τα συστήματα κατανεμημένης μνήμης μπορούν βάσει της σημερινής τεχνολογίας να επεκταθούν επιτυχώς σε χιλιάδες κόμβους, ενώ και σε επίπεδο εφαρμογής έχουν αναπτυχθεί παράλληλοι αλγόριθμοι που μπορούν να κλιμακώσουν την επίδοσή τους σχεδόν γραμμικά σε μεγάλα συστήματα χιλιάδων επεξεργαστών. Παράλληλα όμως, τα συστήματα κατανεμημένης μνήμης επιβάλλουν σημαντικές δυσκολίες στον προγραμματισμό και την αποσφαλμάτωσή τους σε σχέση με τα συστήματα μοιραζόμενης μνήμης, ενώ συχνά απαιτούν και πολλαπλές εικόνες του συστήματος, οδηγώντας έτσι σε κατασπατάληση των διαθέσιμων πόρων.

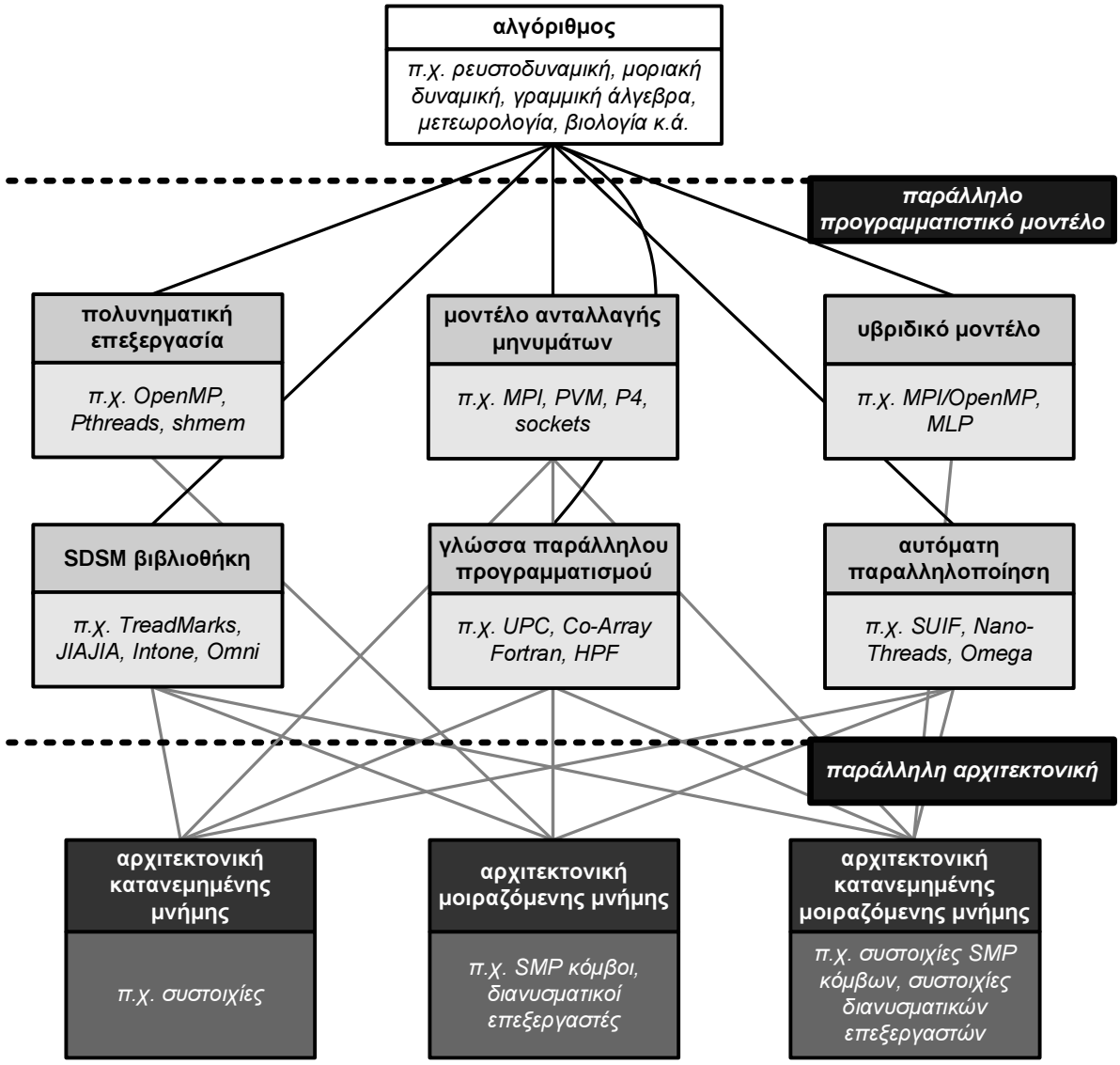
Απόρροια των παραπάνω προβληματισμών αποτελεί η επικράτηση στο χώρο των υπολογιστικών συστημάτων υψηλών επιδόσεων μιας τρίτης υβριδικής αρχιτεκτονικής, εκείνης της κατανεμημένης μοιραζόμενης μνήμης, που συνδυάζει επιτυχώς χαρακτηριστικά από τις δύο πρώτες (σχήμα 2.3). Στα συστήματα κατανεμημένης μοιραζόμενης μνήμης έχουμε τη διασύνδεση περισσότερων πρωτογενών συστημάτων μοιραζόμενης μνήμης, ώστε να συνθέσουν ένα σύστημα κατανεμημένης μνήμης σε ένα δεύτερο επίπεδο. Το βασικό τους πλεονέκτημα είναι ότι συνδυάζουν την επεκτασιμότητα των συστημάτων κατανεμημένης μνήμης με την προγραμματιστική ευελιξία των συστημάτων μοιραζόμενης μνήμης. Έτσι, μπορούμε αφενός να φτάσουμε μέχρι χιλιάδες κόμβους, εφόσον κάτι τέτοιο απαιτούν οι υπολογιστικές ανάγκες, και αφετέρου να πραγματοποιήσουμε εύκολα αποδοτική κατανομή του φορτίου στο εσωτερικό κάθε κόμβου μεταξύ των διαθέσιμων επεξεργαστών με αξιοποίηση της μοιραζόμενης μνήμης.

Στο πλαίσιο της εργασίας αυτής θα βασιστούμε κατά την πειραματική αξιολόγηση σε συστοιχίες κόμβων συμμετρικής πολυεπεξεργασίας (SMP clusters), που αποτελούν άλλωστε και βασικό εκπρόσωπο των συστημάτων κατανεμημένης μοιραζόμενης μνήμης.

2.2 Μοντέλα Παράλληλου Προγραμματισμού

Ο όρος παράλληλος προγραμματισμός αναφέρεται στη σχεδίαση και υλοποίηση ενός προγράμματος, που αξιοποιεί μια δεδομένη παράλληλη αρχιτεκτονική υποδομή με στόχο την επίλυση ενός συγκεκριμένου προβλήματος ή αλγορίθμου. Επιμερίζοντας τα δεδομένα ή/και τους υπολογισμούς του αλγορίθμου μεταξύ των επεξεργαστικών κόμβων της παράλληλης αρχιτεκτονικής, μπορούμε τόσο να *επιταχύνουμε* την επίλυση του προβλήματος για δεδομένη είσοδο όσο και να επιτύχουμε την επίλυση *χωρικά μεγαλύτερης εισόδου* για δεδομένο αλγόριθμο. Επίσης, μέσω του παράλληλου προγραμματισμού μπορούμε

να *βελτιώσουμε* την ποιότητα και την αξιοπιστία της υπολογιζόμενης λύσης ενός προβλήματος υπό δεδομένους χρονικούς περιορισμούς, π.χ. κατά την επιχειρησιακή λειτουργία ενός υπολογιστικού συστήματος πραγματικού χρόνου (real-time computing system - RTC system). Για τους λόγους αυτούς, η παράλληλη επεξεργασία βρίσκει πρωτίστως εφαρμογή σε πληθώρα απαιτητικών επιστημονικών αλγορίθμων, που επιβάλλουν αυξημένη υπολογιστική ή/και χωρική πολυπλοκότητα.



Σχήμα 2.4: *Αφαιρετικός διαχωρισμός μεταξύ της επιλογής του προγραμματιστικού μοντέλου κατά την παραλληλοποίηση μιας εφαρμογής και της επιλογής παράλληλης αρχιτεκτονικής για την εκτέλεση του παράλληλου προγράμματος*

Η επιλογή συγκεκριμένου προγραμματιστικού μοντέλου για την παραλληλοποίηση μιας δεδομένης εφαρμογής μπορεί σε πολλές περιπτώσεις να γίνει ανεξάρτητα της υφιστάμενης αρχιτεκτονικής υποδομής (βλ. και σχήμα 2.4). Πράγματι, η πληθώρα των σχετικών εργαλείων λογισμικού (π.χ. βιβλιοθήκες ή μεταγλωττιστές) διασφαλίζει τη μεταφερσιμότητα ενός προγράμματος σε μια ποικιλομορφία αρχιτεκτονικών υποδομών, από στενά συζευγμένων (π.χ. υπερυπολογιστικά συστήματα μοιραζόμενης μνήμης) έως πιο χαλαρά διασυνδεδεμένων (π.χ. συστοιχίες υπολογιστικών κόμβων). Ωστόσο, είναι προφανές ότι η *απόδοση* που επιτυγχάνεται με μια συγκεκριμένη προγραμματιστική προσέγγιση είναι άρρηκτα συνδεδεμένη τόσο με την επιβάρυνση που επιφέρουν οι χρησιμοποιούμενες βιβλιοθήκες λογισμικού όσο και κυρίως με το βαθμό υποστήριξης του συγκεκριμένου προγραμματιστικού μοντέλου από το υφιστάμενο υλικό. Ορισμένα προγραμματιστικά μοντέλα συνδέονται άμεσα με μια συγκεκριμένη υποδομή, όπως π.χ. η πολυνηματική επεξεργασία με την αρχιτεκτονική μοιραζόμενης μνήμης, συνεπώς η μεταφορά των μοντέλων αυτών σε άλλες αρχιτεκτονικές μπορεί να γίνει μόνο με τη βοήθεια ενός ενδιάμεσου στρώματος λογισμικού, που κατά κανόνα συντελεί σε μείωση της απόδοσης του παράλληλου προγράμματος.

Σε αντιστοιχία με τις τρεις βασικές αρχιτεκτονικές που αναλύσαμε προηγουμένως, θα εξετάσουμε ισάριθμα προγραμματιστικά μοντέλα, που χρησιμοποιούνται στην πράξη στους συγκεκριμένους τύπους παράλληλων συστημάτων. Τα μοντέλα αυτά θα αποτελέσουν το αντικείμενο των επόμενων ενοτήτων.

2.2.1 Μοντέλο Ανταλλαγής Μηνυμάτων

Το προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων (στη διεθνή βιβλιογραφία Message Passing ή συντομογραφικά MP) αποτελεί την κυρίαρχη τάση στο χώρο της Παράλληλης Επεξεργασίας. Φορείς εκτέλεσης του προγράμματος είναι συνήθως μία ή περισσότερες διεργασίες, που επικοινωνούν μεταξύ τους για την ανταλλαγή δεδομένων μέσω αποστολής/λήψης μηνυμάτων. Το μοντέλο ανταλλαγής μηνυμάτων υιοθετεί τη λογική του *Μοναδικού Προγράμματος Πολλαπλών Δεδομένων* (*Single Program Multiple Data* - SPMD). Συγκεκριμένα, όλες οι διεργασίες εκτελούν τον ίδιο κώδικα και σε καθεμία αποδίδεται ένα μοναδικό αναγνωριστικό (**p**rocess **id**entifier - **pid**), το οποίο ταυτοποιεί τη συγκεκριμένη διεργασία μεταξύ όλων των υπολοίπων. Κάθε διεργασία χρησιμοποιεί το αναγνωριστικό της για να καθορίσει το υποσύνολο των δεδομένων και των υπολογισμών που της αντιστοιχούν, διαφοροποιώντας έτσι τη ροή εκτέλεσής της. Σε οποιαδήποτε περίπτωση μία διεργασία χρειαστεί την τιμή κάποιου δεδομένου που έχει αποθηκευτεί στον εικονικό χώρο διευθύνσεων κάποιας άλλης διεργασίας, θα πρέπει το δεδομένο αυτό να αποσταλεί μέσω μηνύματος από τη διεργασία-ιδιοκτήτη, καθώς και να παραληφθεί με κλήση κατάλληλης ρουτίνας λήψης από τη διεργασία που το ζήτησε. Ανάλογα με το πλήθος των εμπλεκόμενων διεργασιών, οι λειτουργίες επικοινωνίας χαρακτηρίζονται είτε *από σημείο προς σημείο*

(*point-to-point communication primitives*) εφόσον αφορούν μόνο δύο διεργασίες (μια διεργασία αποστολέα και μια διεργασία παραλήπτη), είτε ως *συλλογικές* (*collective communication primitives*) στην περίπτωση που στην επικοινωνία εμπλέκεται ένα σύνολο διεργασιών.

Το μοντέλο ανταλλαγής μηνυμάτων παρουσιάζει τις εξής πολύ ενδιαφέρουσες ιδιαιτερότητες:

- Η επικοινωνία μεταξύ δύο η περισσότερων διεργασιών για ανταλλαγή δεδομένων είναι πάντα ρητή (explicit). Ο προγραμματιστής θα πρέπει να λάβει ειδική μέριμνα για την επίτευξη της λειτουργίας επικοινωνίας, εφοδιάζοντας το πρόγραμμά του με κατάλληλες ρουτίνες αποστολής/λήψης μηνυμάτων. Ενδεχομένως ο προγραμματιστής να πρέπει να φροντίσει και για την αντιγραφή των δεδομένων επικοινωνίας από την αλγοριθμική δομή σε δομή κατάλληλη για επικοινωνία, όπως αυτή καθορίζεται από τη χρησιμοποιούμενη βιβλιοθήκη ανταλλαγής μηνυμάτων (π.χ. μονοδιάστατος πίνακας στοιχείων).

- Κατά την επικοινωνία πραγματοποιείται έμμεσα (implicitly) και ο συγχρονισμός μεταξύ των διεργασιών, καθώς ουσιαστικά ενυπάρχει εγγενώς στη διαδικασία ανταλλαγής μηνυμάτων. Πιο συγκεκριμένα, η ανταλλαγή δεδομένων προϋποθέτει τη συμμετοχή όλων των εμπλεκομένων διεργασιών, καθώς όλες θα χρειαστεί να καλέσουν κάποια ρουτίνα επικοινωνίας, είτε ως αποστολείς είτε ως παραλήπτες. Έτσι, μια διεργασία αποστολέας γνωρίζει ότι έχει αποστείλει ένα μήνυμα εφόσον ολοκληρωθεί η ρουτίνα αποστολής, ενώ αντίστοιχα μια διεργασία παραλήπτης μπορεί να υποθέσει πως έχει λάβει τα δεδομένα που ζήτησε μετά την επιτυχή περάτωση της ρουτίνας λήψης που η ίδια κάλεσε. Με απλά λόγια, στο μοντέλο ανταλλαγής μηνυμάτων η επικοινωνία και ο συγχρονισμός είναι σε μεγάλο βαθμό αδιαίρετες έννοιες.

Η γενικότητα του μοντέλου ανταλλαγής μηνυμάτων του επιτρέπει να εφαρμοστεί σε πληθώρα αρχιτεκτονικών, από μοιραζόμενης μέχρι κατανεμημένης μνήμης. Στην πράξη άλλωστε, πολλές πραγματικές εφαρμογές έχουν παραλληλοποιηθεί επιτυχώς με το μοντέλο αυτό, επιτυγχάνοντας υψηλή απόδοση και διασφαλίζοντας την επεκτασιμότητα του προγράμματος. Το πλέον δημοφιλές πρότυπο για ανταλλαγή μηνυμάτων είναι το Message Passing Interface (MPI), για το οποίο θα κάνουμε ξεχωριστή αναφορά στο παράρτημα Α. Άλλες δημοφιλείς βιβλιοθήκες για προγραμματισμό μέσω ανταλλαγής μηνυμάτων είναι η πλατφόρμα PVM [GBD+94], η διεπαφή sockets [Ste90] ή ακόμα και η παλιότερη βιβλιοθήκη P4 [BL94], στην οποία έχει βασιστεί η γνωστή υλοποίηση MPICH του MPI.

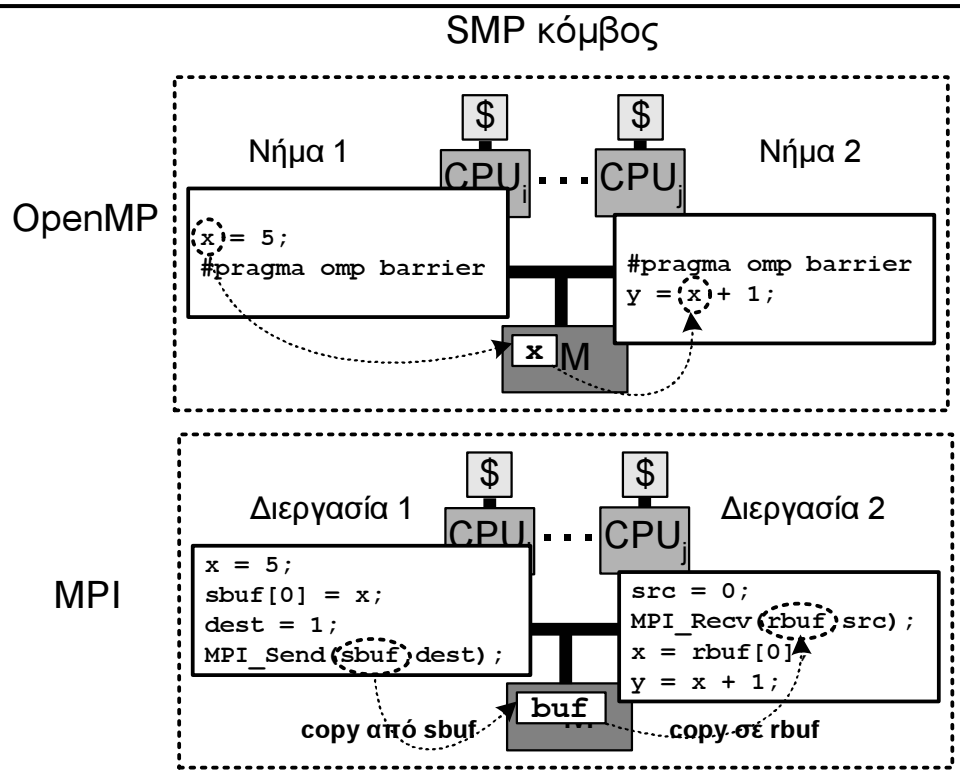
2.2.2 Μοντέλο Πολυνηματικής Επεξεργασίας

Η πολυνηματική επεξεργασία (multi-threaded programming) αποτελεί την επικρατέστερη υλοποίηση του προγραμματιστικού μοντέλου μοιραζόμενου χώρου μνήμης (Shared Address Space - SAS). Η

πολυνηματική επεξεργασία θεωρεί ως μονάδες εκτέλεσης του παράλληλου προγράμματος τα *νήματα* (*threads*). Τα νήματα μοιράζονται τον κοινό εικονικό χώρο διευθύνσεων της διεργασίας-γονέα και μπορούν να επικοινωνήσουν με χρήση μοιραζόμενων μεταβλητών (shared variables), τις οποίες προσπελάζουν με συνήθεις λειτουργίες ανάγνωσης και εγγραφής (read - write). Επιπρόσθετα, στην περίπτωση αυτή απαιτείται περαιτέρω συγχρονισμός μεταξύ των νημάτων, για να διασφαλιστεί η έγκυρη ακολουθία πρόσβασης στα μοιραζόμενα δεδομένα βάσει της σημασιολογίας του αλγορίθμου. Κάθε νήμα μπορεί να καθορίζει μεταβλητές και ως ιδιωτικές (private variables), σε περίπτωση που δεν επιθυμεί να τις μοιραστεί με τα υπόλοιπα νήματα της ομάδας.

Συγκριτικά με το μοντέλο ανταλλαγής μηνυμάτων, η πολυνηματική παραλληλοποίηση παρέχει προγραμματιστική ευκολία, καθώς η υλοποίηση πολυνηματικής επεξεργασίας μπορεί να επιτευχθεί με τροποποίηση μικρού μόνο τμήματος του αρχικού σειριακού προγράμματος με χρήση κατάλληλης διεπαφής. Επιπλέον, η επικοινωνία υλοποιείται με χρήση απλών αναγνώσεων και εγγραφών στη μοιραζόμενη μνήμη, σε αντιδιαστολή με τις εξειδικευμένες ρουτίνες ανταλλαγής μηνυμάτων. Όμως, το μοντέλο αυτό παρουσιάζει ένα σημαντικό μειονέκτημα έναντι του εναλλακτικού της ανταλλαγής μηνυμάτων, καθώς προϋποθέτει την ύπαρξη ενός καθολικού, μοιραζόμενου χώρου μνήμης. Ένας τέτοιος χώρος θα πρέπει είτε να παρέχεται εγγενώς από την υφιστάμενη αρχιτεκτονική (π.χ. αρχιτεκτονική μοιραζόμενης μνήμης), είτε να υλοποιείται έμμεσα με χρήση κατάλληλου λογισμικού, στην περίπτωση αρχιτεκτονικής κατανεμημένης μνήμης. Καθώς ένα τέτοιο ενδιάμεσο στρώμα λογισμικού επιφέρει στην πράξη σημαντική επιβάρυνση στην επίδοση του παράλληλου προγράμματος, ενώ και οι αρχιτεκτονικές αμιγώς μοιραζόμενης μνήμης έχουν αποδειχθεί μη επεκτάσιμες πέρα από μικρό αριθμό επεξεργαστών, το μοντέλο της πολυνηματικής επεξεργασίας έχει σημαντικά μικρότερη απήχηση σε σχέση με εκείνο της ανταλλαγής μηνυμάτων.

Κατά την υλοποίηση ενός παράλληλου προγράμματος σε αρχιτεκτονική μοιραζόμενης μνήμης μπορεί κανείς να χρησιμοποιήσει τόσο το μοντέλο ανταλλαγής μηνυμάτων όσο και εκείνο της πολυνηματικής επεξεργασίας. Το σχήμα 2.5 αποσκοπεί στο να αποσαφηνίσει εν μέρει τις διαφορές μεταξύ των δύο μοντέλων για την υποθετική περίπτωση που η πρώτη οντότητα (διεργασία ή νήμα) θέλει να αναθέσει μια τιμή στη μεταβλητή `x`, ενώ η δεύτερη οντότητα θέλει ακολούθως να χρησιμοποιήσει την ενημερωμένη τιμή της `x` για τον υπολογισμό μιας άλλης μεταβλητής `y`. Παρατηρούμε πως το μοντέλο πολυνηματικής επεξεργασίας απαιτεί μόνο μια λειτουργία συγχρονισμού μεταξύ των δύο νημάτων (`barrier`), που διασφαλίζει ότι η ανάγνωση της τιμής της μεταβλητής `x` από το νήμα 2 θα ακολουθήσει χρονικά την εγγραφή της `x` από το νήμα 1. Αντίθετα, το μοντέλο ανταλλαγής μηνυμάτων καταφεύγει στην κλήση κατάλληλων ρουτινών επικοινωνίας (`MPI_Send`, `MPI_Recv`), ενώ ενδεχομένως απαιτεί επιπλέον μια φάση ομαδοποίησης των δεδομένων αποστολής (`sbuf`) και ταξινόμησης των δεδομένων λήψης (`rbuf`). Το παράδειγμα αυτό καταδεικνύει εύγλωττα πως παρότι το μοντέλο ανταλλαγής μηνυμάτων



Σχήμα 2.5: *Σύγκριση προγραμματιστικού μοντέλου ανταλλαγής μηνυμάτων με το ισοδύναμο μοιραζόμενης μνήμης για αρχιτεκτονική μοιραζόμενης μνήμης*

είναι αρκετά γενικό, ώστε να μπορεί να εφαρμοστεί σε αρχιτεκτονικές μοιραζόμενης μνήμης, μια τέτοια υλοποίηση είναι σημαντικά πιο πολύπλοκη από την πολυνηματική εκδοχή. Επιπλέον, όπως θα δούμε και στην επόμενη ενότητα, η γενικότητα του μοντέλου ανταλλαγής μηνυμάτων συνεπάγεται πρόσθετη επιβάρυνση για αρχιτεκτονικές μοιραζόμενης μνήμης σε σχέση με το πολυνηματικό μοντέλο.

Στο πλαίσιο της παρούσας εργασίας θα χρησιμοποιήσουμε το πρότυπο πολυνηματικής επεξεργασίας OpenMP (βλ. παράρτημα Β), που αποτελεί μια εύχρηστη και ιδιαίτερα δημοφιλή διεπαφή για την υλοποίηση προγραμματισμού μοιραζόμενης μνήμης. Άλλες δημοφιλείς προσεγγίσεις είναι η συμβατή με το POSIX πρότυπο βιβλιοθήκη Pthreads [NBF96] καθώς και η χρήση SYS V μοιραζόμενης μνήμης μέσω της διεπαφής shmem.

2.2.3 Υβριδικό Προγραμματιστικό Μοντέλο

Θεωρώντας την περίπτωση αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης, σε μια πρώτη ανάλυση θα μπορούσε κανείς να καταγράψει αφενός την αδυναμία του πολυνηματικού μοντέλου λόγω μη ύπαρξης καθολικά μοιραζόμενης μνήμης στο υλικό, και αφετέρου την καθολική δυνατότητα εφαρμογής

του μοντέλου ανταλλαγής μηνυμάτων. Η διεθνής βιβλιογραφία όμως εξετάζει και μια τρίτη, υβριδική προσέγγιση, η διερεύνηση της οποίας αποτελεί και το αντικείμενο της παρούσας ενότητας.

Πέρα από τον προβληματισμό που αφορά στην προγραμματιστική πολυπλοκότητα του μοντέλου ανταλλαγής μηνυμάτων, όπως αυτός διατυπώθηκε στην προηγούμενη ενότητα, υπάρχει ένας βασικότερος λόγος για τον οποίο το συγκεκριμένο μοντέλο ενδέχεται να μην αξιοποιεί πλήρως μια αρχιτεκτονική μοιραζόμενης μνήμης. Ακόμα κι αν η χρησιμοποιούμενη βιβλιοθήκη ανταλλαγής μηνυμάτων αξιοποιεί την υφιστάμενη υποδομή της μοιραζόμενης μνήμης, αντιστοιχίζοντας ιδεατά την αποστολή και λήψη δεδομένων σε απλή εγγραφή και ανάγνωση στη μοιραζόμενη μνήμη, η γενικότητα του μοντέλου ανταλλαγής μηνυμάτων συνεπάγεται συχνά την ύπαρξη επιπλέον *λειτουργιών ομαδοποίησης και ταξινόμησης-επανατοποθέτησης* των δεδομένων επικοινωνίας. Το γεγονός αυτό αποδίδεται στους περιορισμούς που επιβάλλει η βιβλιοθήκη ανταλλαγής μηνυμάτων, που μπορεί να υποστηρίξει μόνο συγκεκριμένες δομές μηνυμάτων, οι οποίες ενδεχομένως διαφέρουν από τους πρωτογενείς τύπους δεδομένων της εφαρμογής. Επιπλέον, η αξιοποίηση της μοιραζόμενης μνήμης σε πολλές βιβλιοθήκες ανταλλαγής μηνυμάτων επιτυγχάνεται μόνο μέσω *ενδιάμεσων αντιγραφών* των δεδομένων επικοινωνίας από και προς συγκεκριμένες προκαθορισμένες περιοχές μοιραζόμενης μνήμης. Οι αντιγραφές αυτές έχουν επίπτωση στην επίδοση του προγράμματος, ειδικά στην περίπτωση που το πλήθος των δεδομένων επικοινωνίας είναι σχετικά υψηλό.

Εξίσου σημαντικό είναι το γεγονός πως το *υψηλό κόστος αρχικοποίησης*, που συνήθως σχετίζεται με την αποστολή μηνυμάτων μέσω ενός δικτύου διασύνδεσης, προκρίνει την αποστολή λίγων μεγάλων μηνυμάτων έναντι πολλών μικρών, για τη μετάδοση συγκεκριμένου όγκου δεδομένων. Συνεπώς, έχει νόημα τα δεδομένα επικοινωνίας που προορίζονται για την ίδια διεργασία να ομαδοποιούνται σε ένα μοναδικό μήνυμα, ώστε να ελαχιστοποιηθεί το συνολικό πλήθος των ανταλλασσόμενων μηνυμάτων. Απόρροια του γεγονότος αυτού είναι η αναγκαιότητα επιπλέον αντιγραφών, που προφανώς επιδρά ανασταλτικά στη συνολική επίδοση του παράλληλου προγράμματος.

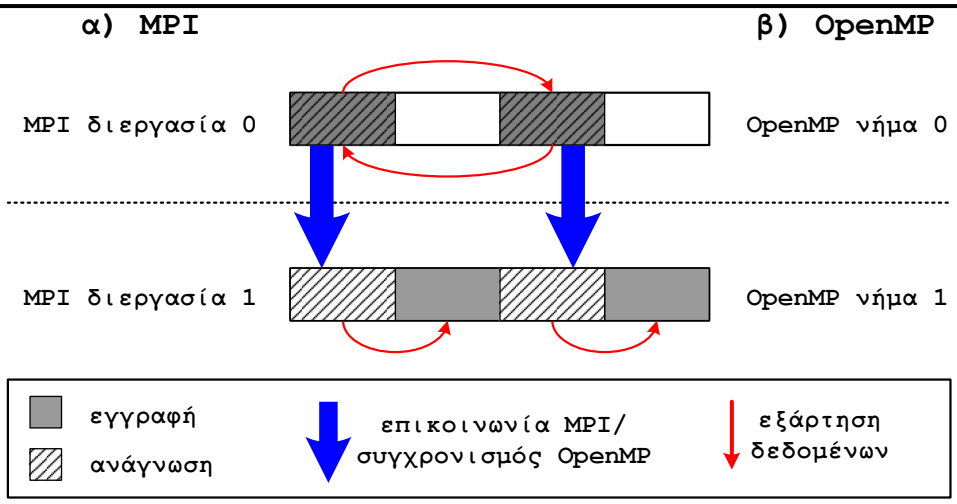
Για να επαληθεύσουμε πειραματικά τα παραπάνω, υλοποιήσαμε ένα απλό σενάριο επικοινωνίας, παρόμοιο με εκείνο του σχήματος 2.5, ακολουθώντας τέσσερις διαφορετικές προσεγγίσεις:

1. MPI+TCP/IP μεταξύ δύο διαφορετικών SMP κόμβων (δύο διεργασίες MPI)

2. MPI+διεπαφή ανάδρασης (loopback interface) μεταξύ δύο επεξεργαστών του ίδιου SMP κόμβου (δύο διεργασίες MPI)

3. MPI+διεπαφή shmem μοιραζόμενης μνήμης μεταξύ δύο επεξεργαστών του ίδιου SMP κόμβου (δύο διεργασίες MPI)

4. OpenMP μεταξύ δύο επεξεργαστών του ίδιου SMP κόμβου (δύο νήματα OpenMP)

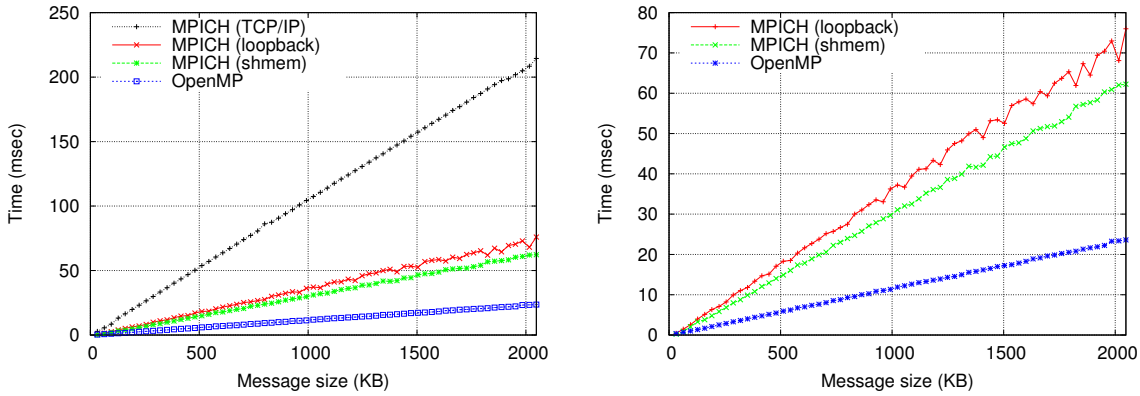


Σχήμα 2.6: *Επεξεργασία μονοδιάστατου πίνακα με χρήση τόσο του μοντέλου ανταλλαγής μηνυμάτων όσο και του πολυνηματικού μοντέλου*

Στις τρεις πρώτες περιπτώσεις εκκινήσαμε δύο διεργασίες MPI. Η διεργασία 0 υπολογίζει το πρώτο και τρίτο τεταρτημόριο ενός μονοδιάστατου πίνακα. Στη συνέχεια αντιγράφει το κατάλληλο τεταρτημόριο σε απομονωτή επικοινωνίας και το αποστέλλει μέσω του MPI στη διεργασία 1. Από την άλλη μεριά, η διεργασία 1 λαμβάνει τα δεδομένα επικοινωνίας, τα τοποθετεί στο κατάλληλο τεταρτημόριο (πρώτο ή τρίτο) του δικού της τοπικού αντίγραφου του πίνακα και τα χρησιμοποιεί για τον υπολογισμό του γειτονικού τεταρτημορίου (δεύτερο ή τέταρτο, αντίστοιχα). Στην περίπτωση 4 (OpenMP), την ίδια επεξεργασία αναλαμβάνουν δύο νήματα OpenMP, που επενεργούν στα τεταρτημόρια ενός μοιραζόμενου πίνακα και συγχρονίζονται με χρήση λειτουργίας `barrier`. Ο παραπάνω πυρήνας, που απεικονίζεται στο σχήμα 2.6 για λόγους οπτικοποίησης, αφενός αντιπροσωπεύει ένα αρκετά ρεαλιστικό σενάριο παράλληλης επεξεργασίας, αφετέρου αποφεύγει ή έστω μετριάζει διάφορες επιβαρύνσεις ή επιπλοκές του υλικού (διατήρηση συνέπειας μεταξύ νημάτων, φαινόμενα κρυφής μνήμης κλπ.), επιτρέποντας έτσι την άμεση σύγκριση των εναλλακτικών σχημάτων επικοινωνίας.

Τα αποτελέσματα της σύγκρισης αποτυπώνονται στο σχήμα 2.7. Παρατηρούμε πως παρότι η συγκεκριμένη υλοποίηση του MPI (MPICH) είναι αρκετά βελτιστοποιημένη, ώστε να βελτιώνει σημαντικά το χρόνο επικοινωνίας στο εσωτερικό του SMP κόμβου, δεν μπορεί να αποδώσει καλύτερα από το πολυνηματικό μοντέλο. Το γεγονός αυτό είναι αναμενόμενο και οφείλεται κυρίως στις επιπρόσθετες αντιγραφές που περιλαμβάνονται στο μοντέλο ανταλλαγής μηνυμάτων.

Ο παραπάνω προβληματισμός αποτέλεσε έναυσμα για τη διερεύνηση της επίδοσης υβριδικών προγραμματιστικών μοντέλων σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Σε τέτοιες αρχιτεκτονικές, εκτός του γενικού μοντέλου ανταλλαγής μηνυμάτων θα μπορούσε κανείς να υλοποιήσει



(α) Γενικά (β) Με αξιοποίηση της ιεραρχίας μνήμης

Σχήμα 2.7: *Σύγκριση της επίδοσης εναλλακτικών μεθόδων επικοινωνίας στο εσωτερικό SMP κόμβου. Χάριν πληρότητας, το σχήμα 2.7(α) απεικονίζει επιπλέον και τη χρονική επίδοση στην επικοινωνία μέσω ανταλλαγής μηνυμάτων μεταξύ διεργασιών σε διαφορετικούς SMP κόμβους (περίπτωση TCP/IP). Παρατηρούμε από τις υπόλοιπες τρεις ευθείες, που αντιστοιχούν σε επικοινωνία ή συγχρονισμό στο εσωτερικό του SMP κόμβου και αναπαράγονται για ευκολία στο σχήμα 2.7(β), ότι η συγχρονισμένη πρόσβαση στην κοινή μνήμη με χρήση OpenMP υπερτερεί της MPI επικοινωνίας γενικής μορφής, ακόμα κι όταν αυτή είναι ιδιαίτερα βελτιστοποιημένη κάνοντας χρήση SYS V μοιραζόμενης μνήμης (περίπτωση shmem).*

ένα διεπίπεδο σχήμα επικοινωνίας, θεωρώντας πολυνηματική επεξεργασία στο χαμηλότερο επίπεδο (εσωτερικό ενός πολυεπεξεργαστικού κόμβου) και ανταλλαγή μηνυμάτων σε ένα δεύτερο υψηλότερο επίπεδο (μεταξύ των διαφορετικών κόμβων). Το υβριδικό μοντέλο παράλληλης επεξεργασίας αποτελεί το αντικείμενο του κεφαλαίου 5.

2.2.4 Άλλα Μοντέλα Παράλληλου Προγραμματισμού

Το μοντέλο ανταλλαγής μηνυμάτων, η πολυνηματική επεξεργασία και ο υβριδικός συνδυασμός αυτών συνιστούν τις κυριότερες προσεγγίσεις στον παράλληλο προγραμματισμό, όχι όμως και τις μοναδικές. Σε ό,τι αφορά στις αρχιτεκτονικές μοιραζόμενης μνήμης, το πολυνηματικό μοντέλο με χρήση κατάλληλης προγραμματιστικής διεπαφής, π.χ. του OpenMP, αποτελεί μια καλή λύση, κυρίως γιατί δεν επιφέρει σημαντικές αλλαγές στον αρχικό σειριακό κώδικα. Όμως, όσον αφορά στις αρχιτεκτονικές κατανεμημένης ή κατανεμημένης μοιραζόμενης μνήμης, το μοντέλο ανταλλαγής μηνυμάτων μπορεί να έχει αποδειχθεί ιδιαίτερα αποδοτικό στην πράξη, σχετίζεται όμως κατά γενική ομολογία με υψηλή προγραμματιστική πολυπλοκότητα, απαιτώντας από τον προγραμματιστή να μεριμνήσει επιμελώς για τις ανάγκες επικοινωνίας της εφαρμογής του.

Δεδομένου ότι οι προς παραλληλοποίηση αλγόριθμοι συχνά προέρχονται από άλλους κλάδους της

επιστημονικής κοινότητας εκτός της επιστήμης των υπολογιστών, η προγραμματιστική πολυπλοκότητα μιας διεπαφής σαν το MPI κρίνεται ενίοτε σχετικά υψηλή. Έτσι, κατά καιρούς έχουν προταθεί εναλλακτικές προσεγγίσεις για τον προγραμματισμό συστημάτων κατανεμημένης μνήμης, υποσύνολο των οποίων θα παρουσιάσουμε για λόγους πληρότητας στην παρούσα ενότητα. Θα πρέπει να τονιστεί πως κοινή συνισταμένη των εναλλακτικών παράλληλων προγραμματιστικών μοντέλων αποτελεί η διαφανής μετατόπιση των πολύπλοκων λεπτομερειών της υλοποίησης από τον προγραμματιστή προς χαμηλότερα στρώματα λογισμικού, που χρησιμοποιούνται ακριβώς για το σκοπό αυτό. Παράλληλα όμως, η διασφάλιση της επιθυμητής προγραμματιστικής απλότητας συνεπάγεται συχνά σημαντικές επιπτώσεις στην επίδοση των μοντέλων αυτών, που στην πράξη έχει αποδειχθεί να υπολείπονται σχετικά του μοντέλου ανταλλαγής μηνυμάτων.

Τρεις επιπλέον βασικές προσεγγίσεις έχουν παρουσιαστεί στη διεθνή βιβλιογραφία για τον παράλληλο προγραμματισμό αρχιτεκτονικών υψηλών επιδόσεων:

1. Η χρησιμοποίηση ενός *ενδιάμεσου στρώματος λογισμικού* για την διασφάλιση ενός ενιαίου μοιραζόμενου χώρου μνήμης ως προς τον προγραμματιστή της εφαρμογής. Η προσέγγιση αυτή στη διεθνή βιβλιογραφία χαρακτηρίζεται ως Κατανεμημένη Μοιραζόμενη Μνήμη μέσω Λογισμικού (Software Distributed Shared Memory - SDSM) ή ακόμα και Κατανεμημένη Μοιραζόμενη Εικονική Μνήμη (Distributed Shared Virtual Memory - DSVM). Πρακτικά, συνήθως συνδυάζονται ένας μεταφραστής (translator) από κάποιο γνωστό προγραμματιστικό μοντέλο τύπου SAS (π.χ. OpenMP, Pthreads) με κάποια SDSM βιβλιοθήκη χρόνου εκτέλεσης για αρχιτεκτονικές κατανεμημένης και κατανεμημένης μοιραζόμενης μνήμης. Για παράδειγμα, τέτοια περιβάλλοντα είναι η διεπαφή JIAJIA [HST99], ο μεταγλωττιστής Omni για OpenMP [SSKT99], το περιβάλλον Intone [KLB02], το περιβάλλον Treadmarks [KCDZ94], η βιβλιοθήκη Cashmere-2L [SDH$^+$97] κ.ά. Σε γενικές γραμμές, η SDSM προσέγγιση επιδιώκει να κάνει διάκριση μεταξύ του μοντέλου προγραμματισμού (π.χ. SAS) και του μοντέλου εκτέλεσης (π.χ. στρώμα SDSM και βιβλιοθήκη χρόνου εκτέλεσης για συνέπεια μνήμης και λειτουργίες επικοινωνίας/συγχρονισμού). Το ζητούμενο στα SDSM συστήματα είναι η αποδοτική υλοποίηση της απαιτούμενης συνέπειας τόσο της κύριας όσο και της κρυφής μνήμης, χωρίς να προκαλείται σημαντική επιβάρυνση στην επίδοση της εφαρμογής.

   Η χρήση του προτύπου OpenMP είναι ιδιαίτερα δημοφιλής στη συγκεκριμένη κατηγορία, καθώς, αντίθετα με άλλα περιβάλλοντα πολυνηματικής επεξεργασίας, η αυστηρή και καλώς ορισμένη δομή των πολυνηματικών περιοχών σε OpenMP επιτρέπει στον SDSM μεταγλωττιστή την ανάλυση και βελτιστοποίηση των παράλληλων περιοχών, ιδιαίτερα σε θέματα συγχρονισμού και επικοινωνίας. Στις εργασίες [KKH03] (μεταγλωττιστής ParADE για OpenMP) και [HLCZ00] (πολυνηματική υλοποίηση TreadMarks) παρουσιάζονται μάλιστα πολυνηματικές SDSM υλοποιήσεις

για SMP συστοιχίες, που προσομοιώνουν σε επίπεδο μεταγλωττιστή τη λογική του υβριδικού παράλληλου προγραμματιστικού μοντέλου για αποδοτικότερη αξιοποίηση μιας ιεραρχικής αρχιτεκτονικής κατανεμημένης μοιραζόμενης μνήμης. Πάντως, συνολικά η SDSM προσέγγιση τυγχάνει περιορισμένης αποδοχής από την επιστημονική κοινότητα, ίσως γιατί βρίσκεται ακόμα σε αρκετά πρώιμο στάδιο βελτιστοποίησης, επιτυγχάνοντας αισθητά χαμηλότερη επίδοση σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων [SSOB03].

2. Η εισαγωγή *νέων γλωσσών παράλληλου προγραμματισμού*, που προβλέπουν συντακτικά την προσπέλαση ενός κατανεμημένου συστήματος μνήμης κυρίως μέσω επέκτασης του τύπου και του τρόπου πρόσβασης των δεδομένων. Τέτοιες περιπτώσεις είναι η Unified Parallel C [EGCD02, Con05], η Co-Array Fortran - CAF [NR98] και η High Performance Fortran - HPF [MH95, For97]. Η διαφορά με την προηγούμενη κατηγορία έγκειται στο ότι εδώ ο χρήστης δεν επαφίεται στη διάφανη υλοποίηση μοιραζόμενου χώρου μνήμης κατά τον προγραμματισμό, αλλά αντίθετα ενσωματώνει ο ίδιος τη σχετική σημασιολογία, π.χ. με έναν επιπλέον δείκτη σε δομή τύπου πίνακα, που υποδεικνύει το νήμα ιδιοκτήτη. Για παράδειγμα, η Unified Parallel C (UPC) επεκτείνει το ANSI C πρότυπο με δυνατότητες πολυνηματικού παράλληλου προγραμματισμού. Τα νήματα μοιράζονται μέρος του χώρου διευθύνσεων μνήμης τους και διατηρούν τόσο μοιραζόμενα όσο και ιδιωτικά δεδομένα. Ο προγραμματιστής μπορεί να φροντίσει να κατανεμηθούν φυσικά τα μοιραζόμενα δεδομένα μεταξύ των νημάτων κατά τέτοιο τρόπο, ώστε να τοποθετηθούν στην τοπική μνήμη του νήματος που αναμένεται να τα προσπελάζει συχνότερα. Το UPC πρότυπο επιτρέπει καθορισμό χαλαρής ή αυστηρής συνέπειας μνήμης, δυναμική διαχείριση μοιραζόμενης μνήμης και ασύγχρονες λειτουργίες συγχρονισμού (barrier), που επιτρέπουν την επικάλυψη της φάσης συγχρονισμού των νημάτων με τοπικούς υπολογισμούς. Το πρότυπο βρίσκεται ακόμα σε πρώιμα στάδια ανάπτυξης, όπως καταδεικνύουν και οι σχετικοί προβληματισμοί για τις ασάφειες στη σημασιολογία της συνέπειας μνήμης [KW04] αλλά και η μέτρια απόδοση που έχει καταγραφεί σε σχέση με άλλα παράλληλα προγραμματιστικά μοντέλα [CDMC$^+$05]. Εν πολλοίς, η προσπάθεια αυτή θα κριθεί στην πράξη από την ανάπτυξη βελτιστοποιημένων μεταγλωττιστών για τις δημοφιλέστερες παράλληλες αρχιτεκτονικές, δεδομένης της κατά τ' άλλα ευρύτατης αποδοχής των γλωσσών C και Fortran στην επιστημονική κοινότητα για τον προγραμματισμό υψηλών επιδόσεων.

3. Η απόπειρα προτυποποίησης *εργαλείων αυτόματης και ημιαυτόματης παραλληλοποίησης* αλγορίθμων. Ιδιαίτερα αξιόλογες εργασίες στο συγκεκριμένο πεδίο συναντάμε στην περίπτωση των Nano-Threads [MLNA96, PMN$^+$98], του μεταγλωττιστή SUIF αυτόματης παραλληλοποίησης [HAA$^+$96] καθώς και του περιβάλλοντος Omega [KMP$^+$96]. Εν γένει, η διαδικασία της

αυτόματης παραλληλοποίησης θεωρεί ως είσοδο ένα σειριακό πρόγραμμα και μέσω σύνθετων τεχνικών ανάλυσης εξαρτήσεων και εξαγωγής παραλληλισμού επιχειρεί να μετασχηματίσει τον αρχικό κώδικα σε ισοδύναμη παράλληλη εκδοχή, κατά το δυνατό χωρίς την παρέμβαση του χρήστη (αυτόματη παραλληλοποίηση) ή έστω με σχετικά μικρή καθοδήγηση από αυτόν (ημιαυτόματη παραλληλοποίηση). Πρόκειται προφανώς για το πλέον φιλόδοξο εγχείρημα στο χώρο της Παράλληλης Επεξεργασίας, καθώς η ποικιλομορφία των αλγορίθμων θέτει σε αμφισβήτηση ακόμα και το κατά πόσο η αμιγώς αυτόματη παραλληλοποίηση αποτελεί εφικτό και δόκιμο στόχο.

Τέλος, σύντομα έγινε αντιληπτό πως το μοντέλο ανταλλαγής μηνυμάτων σε μεγάλο βαθμό δυσχεραίνει το διαχωρισμό της επικοινωνίας με το συγχρονισμό. Στη φιλοσοφία όμως του παράλληλου προγραμματισμού ο συγχρονισμός και ο παραλληλισμός είναι έννοιες αντίρροπες, συνεπώς συγχρονισμός θα πρέπει να επιβάλλεται μόνο ρητά εφόσον κρίνεται σημασιολογικά απαραίτητος, και όχι έμμεσα ή ακούσια. Απόρροια της παραπάνω διαπίστωσης αποτελεί η προδιαγραφή της δυνατότητας για Απομακρυσμένη Προσπέλαση Μνήμης (Remote Memory Access - RMA) στο πρότυπο MPI-2 [GLT99]: ο προγραμματιστής θα πρέπει ρητά να καλεί λειτουργίες συγχρονισμού για να διαχωρίζει φάσεις τοπικής και απομακρυσμένης πρόσβασης, ώστε να μην απασχολούνται περισσότερες των απολύτως απαραίτητων διεργασιών σε κάποια προσπέλαση δεδομένων. Καθώς η υποστήριξη του MPI-2 προτύπου από τις υπάρχουσες MPI υλοποιήσεις βρίσκεται ακόμα σε πρώιμο στάδιο, η RMA προσέγγιση για την επικοινωνία μεταξύ διεργασιών αναμένεται να διερευνηθεί και να αξιολογηθεί πολύ πιο διεξοδικά στο άμεσο μέλλον.

ΚΕΦΑΛΑΙΟ 3

Αλγοριθμικό Μοντέλο Φωλιασμένων Βρόχων

Οι αλγοριθμικές περιγραφές φωλιασμένων βρόχων αποτελούν μια από τις δημοφιλέστερες κατηγορίες εφαρμογών που υπόκεινται σε παραλληλοποίηση με στόχο τόσο την επιτάχυνση του χρόνου εκτέλεσης όσο και την αντιμετώπιση προβλημάτων αυξημένης χωρικής πολυπλοκότητας. Η σημασία των φωλιασμένων βρόχων καταδεικνύεται από την πληθώρα των εργασιών στη διεθνή βιβλιογραφία, που εδώ και δύο τουλάχιστον δεκαετίες καταπιάνονται τόσο με τη θεωρητική μελέτη και μοντελοποίησή τους όσο και με τεχνικές απεικόνισης σε παράλληλες αρχιτεκτονικές και βελτιστοποίησης της εκτέλεσης. Το μοντέλο των φωλιασμένων βρόχων είναι ιδιαίτερα σύνθετο και παρέχει μια ποικιλομορφία αλγοριθμικών περιγραφών. Για οριοθέτηση της θεωρητικής μελέτης και συστηματοποίηση της πειραματικής διερεύνησης τεχνικών βελτιστοποίησης, συχνά υιοθετούνται διάφορες λιγότερο ή περισσότερο περιοριστικές παραδοχές στο μοντέλο των φωλιασμένων βρόχων, όπως η θεώρηση τέλεια φωλιασμένων βρόχων και η παραδοχή για ομοιόμορφες μη αρνητικές εξαρτήσεις δεδομένων, που δεχόμαστε και στην παρούσα εργασία. Εντούτοις, η διευρυμένη χρήση αλγοριθμικών περιγραφών φωλιασμένων βρόχων στον πυρήνα πολλών επιστημονικών εφαρμογών καθιστά τους φωλιασμένους βρόχους εκ των πραγμάτων σημαντικούς, ιδιαίτερα δε στο χώρο του υπολογισμού υψηλών επιδόσεων και της παράλληλης επεξεργασίας.

3.1 Αλγοριθμικό Μοντέλο

Το αλγοριθμικό μοντέλο των υπό παραλληλοποίηση προγραμμάτων αφορά *τέλεια φωλιασμένους βρόχους* (*perfectly nested loops*) με *ομοιόμορφες μη αρνητικές εξαρτήσεις δεδομένων* (*uniform nonnegative*

data dependencies). Οι βρόχοι αυτοί χαρακτηρίζονται ισοδύναμα και ως *πλήρως αντιμεταθέσιμοι* (*fully permutable*), καθώς λόγω της φύσης των εξαρτήσεων δεδομένων επιτρέπεται η εναλλαγή δύο οποιωνδήποτε επαναληπτικών εντολών `for` του φωλιασμένου βρόχου. Όλα τα παράλληλα προγραμματιστικά μοντέλα που εξετάζονται στο πλαίσιο της παρούσας εργασίας μπορούν να εφαρμοστούν άμεσα σε οποιονδήποτε αλγόριθμο εμπίπτει στην κατηγορία αυτή.

Είναι σαφές ότι το εν λόγω αλγοριθμικό μοντέλο είναι σχετικά περιορισμένο, κυρίως λόγω του ότι εξετάζει τέλεια φωλιασμένους βρόχους (σε αντιδιαστολή με γενικότερες αλγοριθμικές περιγραφές, π.χ. μη τέλεια φωλιασμένους βρόχους) και προϋποθέτει ομοιόμορφες (σταθερές) μη αρνητικές εξαρτήσεις δεδομένων. Παρά τους περιορισμούς αυτούς, οι φωλιασμένοι βρόχοι αποτελούν τη συχνότερα απαντώμενη κατηγορία υπολογιστικά απαιτητικών αλγοριθμικών περιγραφών, που εμφανίζεται σε πληθώρα επιστημονικών εφαρμογών. Σημαντικός όγκος από σχετικές εφαρμογές, που εντάσσονται στο υπό εξέταση αλγοριθμικό μοντέλο, πηγάζει από τη διακριτοποίηση **Μερικών Διαφορικών Εξισώσεων** - ΜΔΕ (**Partial Differential Equations** - PDEs). Οι ΜΔΕ μπορούν να διακριτοποιηθούν στη μορφή αλγοριθμικών περιγραφών τριπλά ή τετραπλά φωλιασμένων βρόχων, που αντιστοιχούν σε δύο ή τρεις χωρικές και ενδεχομένως μία χρονική μεταβλητή. Αλγόριθμοι ΜΔΕ βρίσκουν εφαρμογή σε ποικίλα επιστημονικά πεδία, όπως η ρευστοδυναμική, η αεροναυπηγική, η μετεωρολογία, η βιολογία, η γραμμική άλγεβρα κ.ά.

Επιπλέον, θα πρέπει να τονιστεί ότι η επιλογή του συγκεκριμένου αλγοριθμικού μοντέλου στο πλαίσιο της παρούσας εργασίας έγινε αποκλειστικά για λόγους απλοποίησης της πειραματικής διαδικασίας, καθώς οι προτεινόμενες προγραμματιστικές βελτιστοποιήσεις βρίσκουν χωρίς σημαντικές αλλαγές εφαρμογή σε γενικότερα παράλληλα προγραμματιστικά μοντέλα για αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Άλλωστε, όπως θα φανεί και στη συνέχεια κατά την αναλυτική παρουσίαση των προτεινόμενων τεχνικών και βελτιστοποιήσεων, τα στοιχεία της απλότητας και της εφαρμοσιμότητας αποτέλεσαν βασικούς σχεδιαστικούς άξονες, στους οποίους δόθηκε ξεχωριστή προτεραιότητα. Συνεπώς, σε πολλές περιπτώσεις το εν λόγω αλγοριθμικό μοντέλο δεν συνιστά περιορισμό του ερευνητικού πεδίου, αλλά απλώς αφαιρετική σχηματοποίηση τυπικών επιστημονικών εφαρμογών που παραλληλοποιούνται σε αρχιτεκτονικές υψηλών επιδόσεων.

3.1.1 Φωλιασμένοι Βρόχοι

Το γενικό μοντέλο των αλγορίθμων, που εξετάζονται στο πλαίσιο της παρούσας εργασίας, είναι εκείνο που αποδίδεται σχηματικά στον αλγόριθμο 3.1. Ο αλγόριθμος αυτός αποτελεί τη γενική σχηματική περιγραφή εφαρμογής πλήρως αντιμεταθέσιμων βρόχων, όπου ένα διάνυσμα επαναλήψεων $\vec{j} = (j_1, \ldots, j_{N+1})$ σαρώνει ένα $N+1$-διάστατο χώρο $[1 \ldots X_1] \times \cdots \times [1 \ldots X_N] \times [1 \ldots Z]$. Ακόμα κι αν ο χώρος επαναλήψεων είναι της μορφής $[L_1 \ldots U_1] \times \cdots \times [L_N \ldots U_N] \times [L_{N+1} \ldots U_{N+1}]$ με

$L_i \neq 1$ για κάποια $i, 1 \leq i \leq N+1$, ο αλγόριθμος μπορεί παρόλα αυτά να γραφεί εύκολα στην πρώτη μορφή με την απλή αντικατάσταση $j_i \rightarrow j_i - L_i + 1$, οπότε προκύπτει και ότι $X_i = U_i - L_i + 1$.

Αλγόριθμος 3.1: Αλγοριθμικό μοντέλο πλήρως αντιμεταθέσιμων βρόχων

```
1 for j_1 ← 1 to X_1 do
2   ...
3   for j_N ← 1 to X_N do
4     for j_{N+1} ← 1 to Z do
5       Compute(A, j⃗, D);
6     endfor
7   endfor
8   ...
9 endfor
```

Στο σώμα των βρόχων (loop body) πραγματοποιούνται συνήθως υπολογισμοί, που αφορούν στοιχεία ενός ή περισσοτέρων πολυδιάστατων πινάκων. Ο τρόπος με τον οποίο τα στοιχεία των πινάκων δεικτοδοτούνται στο σώμα των βρόχων από το διάνυσμα επανάληψης $\vec{j}$ ενδέχεται να επιβάλλει περιορισμούς στην ακολουθία υπολογισμών, υπό την έννοια ότι κάποια στοιχεία χρησιμοποιούνται για τον υπολογισμό της τιμής άλλων στοιχείων. Οι περιορισμοί αυτοί μοντελοποιούνται αυστηρά από τον πίνακα εξαρτήσεων δεδομένων $D = [\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}]$ του αλγορίθμου, όπου $\vec{d}^{(i)} = (d_1^{(i)}, \ldots, d_{N+1}^{(i)})$ το i-οστό από τα m συνολικά διανύσματα εξάρτησης δεδομένων.

3.1.2 Χώρος Επαναλήψεων

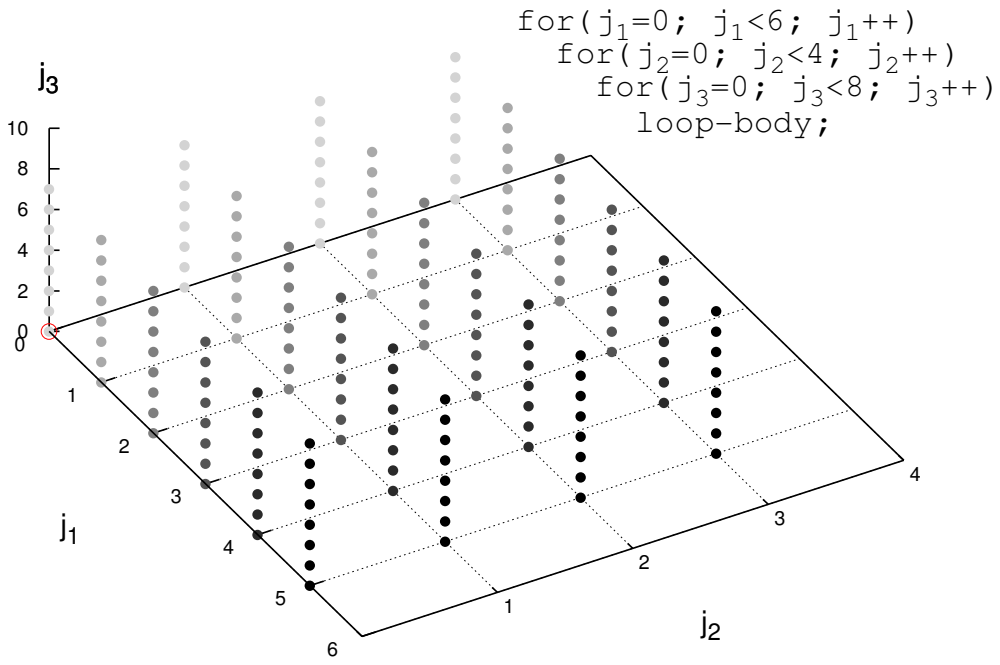
Για τον τυπικό ορισμό του υπό εξέταση αλγοριθμικού μοντέλου θα χρειαστεί να αποσαφηνίσουμε κάποιες θεμελιώδεις έννοιες που σχετίζονται με τη θεωρητική περιγραφή των φωλιασμένων βρόχων, όπως ο χώρος επαναλήψεων και οι εξαρτήσεις δεδομένων. Στις έννοιες αυτές αναφερθήκαμε ήδη στην προηγούμενη ενότητα, χωρίς όμως να επεξηγήσουμε ιδιαίτερα το περιεχόμενό τους. Χάριν μαθηματικής πληρότητας, στην παρούσα ενότητα θα αναφερθούμε εν συντομία στους αυστηρούς ορισμούς των εννοιών αυτών.

Ορισμός 3.1 (Χώρος Επαναλήψεων)**.** *Ορίζουμε ως χώρο επαναλήψεων (iteration space) ενός αλγορίθμου φωλιασμένων βρόχων διάστασης $N+1$ που σαρώνεται από τις μεταβλητές ελέγχου $j_1, \ldots, j_{N+1}$ τον υποχώρο $J = \{\vec{j} = (j_1, \ldots, j_{N+1}) \,|L_i \leq j_i \leq U_i, 1 \leq i \leq N+1\}$ του $\mathbb{N}^{N+1}$, όπου L_i, U_i τα κάτω και άνω όρια της μεταβλητής ελέγχου j_i στον i-στό βρόχο, αντίστοιχα.*

Ο χώρος επαναλήψεων ενός αλγορίθμου φωλιασμένων βρόχων αποτελείται ουσιαστικά από όλα τα σημεία που ορίζονται από τις τιμές που λαμβάνουν οι μεταβλητές ελέγχου των βρόχων καθώς δια-

Αλγόριθμος 3.2: Παράδειγμα αλγορίθμου φωλιασμένων βρόχων

```
1 for j_1 ← 0 to 5 do
2   for j_2 ← 0 to 3 do
3     for j_3 ← 0 to 7 do
4       A[j_1, j_2, j_3] = (A[j_1 − 1, j_2, j_3] + A[j_1, j_2 − 1, j_3] + A[j_1, j_2, j_3 − 2])/3;
5     endfor
6   endfor
7 endfor
```



Σχήμα 3.1: *Χώρος επαναλήψεων τριπλά φωλιασμένου βρόχου. Το σύνολο των σημείων που σαρώνονται κατά την εκτέλεση του τρισδιάστατου βρόχου αντιστοιχεί σε ένα υποσύνολο σημείων του $\mathbb{N}^3$ στον τρισδιάστατο χώρο.*

τρέχουν τις διαδοχικές επαναλήψεις. Για παράδειγμα, ας υποθέσουμε ότι δίνεται ο αλγόριθμος 3.2 φωλιασμένων βρόχων. Ο χώρος επαναλήψεων του αλγορίθμου αυτού ορίζεται ως

$$J = \{\vec{j} = (j_1, j_2, j_3) \in \mathbb{N}^3 | (0 \leq j_1 \leq 5) \wedge (0 \leq j_2 \leq 3) \wedge (0 \leq j_3 \leq 7)\}$$

Γεωμετρικά, ο χώρος επαναλήψεων του αλγορίθμου αυτού αντιστοιχεί στο σύνολο διακριτών σημείων που απεικονίζονται στο σχήμα 3.1.

Στο πλαίσιο της παρούσας διατριβής θα αναφερθούμε σε ορθογώνιους χώρους επαναλήψεων της μορφής $[1 \dots X_1] \times \cdots \times [1 \dots X_N] \times [1 \dots Z]$. Η επιλογή αυτή δεν αποτελεί τεχνητή σύμβαση, αλλά αντίθετα αντανακλά σε προγραμματιστικό επίπεδο τη συνήθη πρακτική τεχνικών αριθμητικής επίλυσης. Έτσι, ακόμα και αν ο φυσικός χώρος εφαρμογής του προβλήματος είναι μορφολογικά περισσότερο σύνθετος από ορθογώνιος, για απλότητα συνήθως θεωρείται ένας περιβάλλοντας ορθογώνιος χώρος, που π.χ. περικλείει τον αρχικό ή στον οποίο μπορεί εύκολα να αντιστοιχιστεί ο αρχικός με χρήση κατάλληλου μετασχηματισμού. Η διακριτοποίηση του νέου ορθογώνιου φυσικού πεδίου θα οδηγήσει στον υπολογισμό διακριτών τιμών σε ένα επίσης ορθογώνιο χώρο επαναλήψεων, της μορφής που θεωρήσαμε παραπάνω. Χάριν συντομίας, θα αναφερόμαστε στο χώρο επαναλήψεων $[1 \dots X_1] \times \cdots \times [1 \dots X_N] \times [1 \dots Z]$ ως $X_1 \times \cdots \times X_N \times Z$.

3.1.3 Εξαρτήσεις Δεδομένων

Οι εξαρτήσεις δεδομένων είναι ιδιαίτερα βασική έννοια στους φωλιασμένους βρόχους. Μάλιστα, η σημασία της θα αναδειχθεί ιδίως κατά τη διαδικασία παραλληλοποίησης, καθώς οι εξαρτήσεις συνδέονται άρρηκτα με την αναγκαιότητα επικοινωνίας και συγχρονισμού μεταξύ των φορέων εκτέλεσης του παράλληλου προγράμματος, στοιχείο που αποτελεί και το βασικότερο παράγοντα περιορισμού της επίδοσης της παραλληλίας.

Ορισμός 3.2 (Εξάρτηση Δεδομένων)**.** *Ορίζουμε ότι υπάρχει εξάρτηση δεδομένων (data dependence) από μια επανάληψη $\vec{j}$ προς μια επανάληψη $\vec{j'}$ ενός φωλιασμένου βρόχου διάστασης $N+1$ αν και μόνο αν*

- *η $\vec{j}$ προηγείται* ***λεξικογραφικά*** *της $\vec{j'}$ (γράφουμε $\vec{j} \prec \vec{j'} \Leftrightarrow \{\exists k \in \mathbb{N}, 1 \leq k \leq N+1 \mid (j_k < j'_k) \wedge (j_i \leq j'_i, \forall i, 1 \leq i < k)\}$)*
- *κατά τις επαναλήψεις $\vec{j}$ και $\vec{j'}$ προσπελάζεται μία τουλάχιστον κοινή θέση μνήμης*
- *μία τουλάχιστον από τις προσπελάσεις αυτές είναι εγγραφή*

Ορισμός 3.3 (Διάνυσμα Εξάρτησης Δεδομένων)**.** *Έστω ένας αλγόριθμος φωλιασμένων βρόχων διάστασης $N+1$. Αν υπάρχει εξάρτηση δεδομένων από την επανάληψη $\vec{j}$ προς την επανάληψη $\vec{j'}$, τότε ορίζουμε ως διάνυσμα εξάρτησης δεδομένων (data dependence vector) το διάνυσμα $\vec{d} = \vec{j'} - \vec{j}$.*

Οι εξαρτήσεις δεδομένων συσχετίζουν τους υπολογισμούς που πραγματοποιούνται σε μια συγκεκριμένη επανάληψη του αλγορίθμου με τους αντίστοιχους υπολογισμούς σε άλλες προγενέστερες ή μεταγενέστερες επαναλήψεις. Θα πρέπει να σημειωθεί ότι αυτού του είδους το διάνυσμα εξάρτησης

δεδομένων ορίζεται ακριβέστερα στη διεθνή βιβλιογραφία ως *διάνυσμα απόστασης* (*distance vector*), σε αντιδιαστολή με το *διάνυσμα κατεύθυνσης* (*direction vector*) που είναι γενικότερο και μπορεί να περιγράψει τη φύση της εξάρτησης σε κάποια διάσταση [WL91b]. Τα διανύσματα απόστασης επαρκούν για την περιγραφή των εξαρτήσεων του υπό εξέταση αλγοριθμικού μοντέλου, συνεπώς θα είναι αυτά που θα χρησιμοποιηθούν στη συνέχεια.

Ιδιαίτερα χρήσιμος για τη μαθηματική μοντελοποίηση των εξαρτήσεων σε περιγραφές φωλιασμένων βρόχων είναι ο *πίνακας εξάρτησης δεδομένων*:

Ορισμός 3.4 (Πίνακας Εξάρτησης Δεδομένων)**.** *Ορίζουμε ως πίνακα εξάρτησης δεδομένων D για αλγόριθμο $N+1$ φωλιασμένων βρόχων με m διανύσματα εξάρτησης δεδομένων $\vec{d}^{(l)} = \left(d_1^{(l)}, \ldots, d_{N+1}^{(l)}\right), 1 \leq l \leq m$ τον πίνακα που έχει ως στήλες τα $\vec{d}^{(l)}$, δηλαδή τον $(N+1) \times m$ πίνακα*

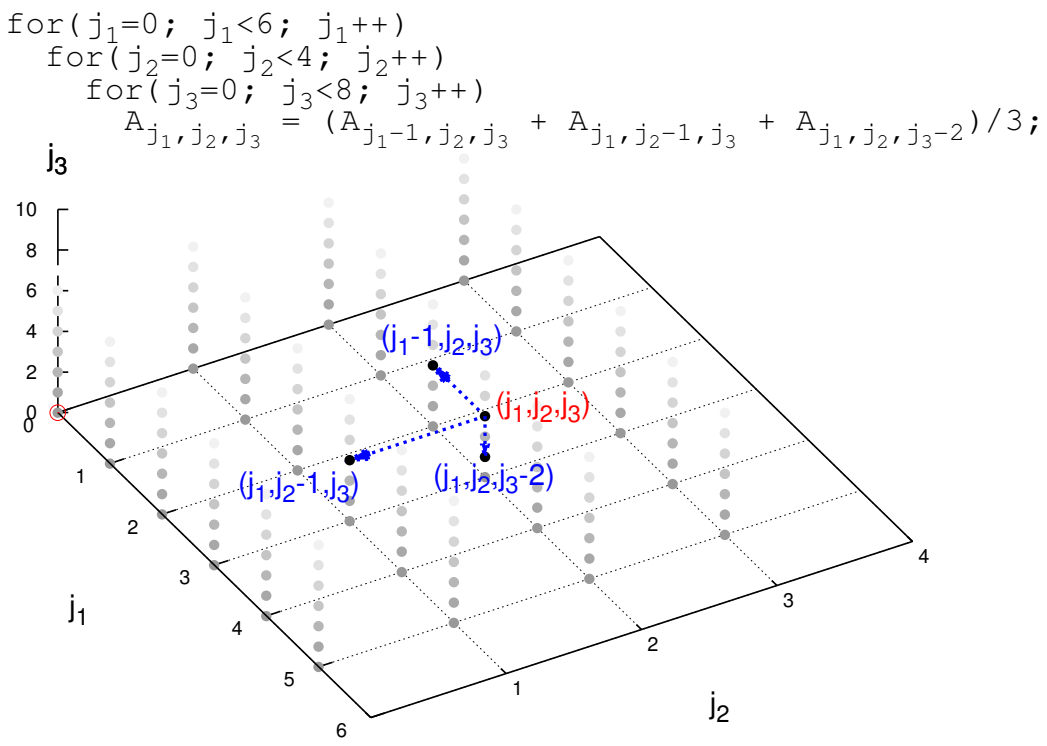
$$D = \begin{bmatrix} d_1^{(1)} & d_1^{(2)} & \cdots & d_1^{(m)} \\ d_2^{(1)} & d_2^{(2)} & \cdots & d_2^{(m)} \\ \vdots & \vdots & \cdots & \vdots \\ d_{N+1}^{(1)} & d_{N+1}^{(2)} & \cdots & d_{N+1}^{(m)} \end{bmatrix} \tag{3.1}$$

Για παράδειγμα, ο πίνακας εξάρτησης δεδομένων του αλγορίθμου 3.2 με τα αντίστοιχα διανύσματα εξάρτησης ως στήλες δίνεται από τη σχέση

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

ενώ γεωμετρικά τα διανύσματα εξάρτησης έχουν σημειωθεί στο χώρο επαναλήψεων του σχήματος 3.2.

Στο πλαίσιο της παρούσας εργασίας θα περιοριστούμε στην εξέταση *σταθερών* εξαρτήσεων δεδομένων, δηλαδή δεχόμαστε ότι για κάθε διάνυσμα εξάρτησης δεδομένων $\vec{d}^{(l)} = (d_1^{(l)}, \ldots, d_{N+1}^{(l)})$, $1 \leq l \leq m$, του αλγορίθμου 3.1 θα ισχύει $d_i^{(l)} =$ σταθερό, $1 \leq i \leq N+1$. Επιπλέον, θα εξετάσουμε *εξαρτήσεις δεδομένων τύπου ροής* (*flow dependencies*), σε αντιδιαστολή με τις *εξαρτήσεις εξόδου* (*output dependencies*) ή τις *αντιεξαρτήσεις* (*anti-dependencies*). Πιο συγκεκριμένα, οι εξαρτήσεις ροής αφορούν περιπτώσεις στις οποίες δεδομένα που υπολογίζονται σε κάποια επανάληψη θα χρησιμοποιηθούν για τον υπολογισμό άλλων δεδομένων σε μεταγενέστερη επανάληψη (*η εγγραφή προηγείται της ανάγνωσης*). Η περίπτωση αυτή είναι η πλέον συνήθης που συναντάται σε επιστημονικούς αλγορίθμους επαναληπτικής φύσεως και εξετάζεται σχεδόν κατ' αποκλειστικότητα στη διεθνή βιβλιογραφία. Στις εξαρτήσεις εξόδου έχουμε διαδοχικές τροποποιήσεις της τιμής του ίδιου δεδομένου (*εγγραφή ακολουθείται από εγγραφή*), συνεπώς μπορούν εύκολα να απαλειφθούν, καθώς σημασιολογικά μόνο η τελευταία εγγραφή πρέπει να διατηρηθεί. Ομοίως, οι αντιεξαρτήσεις (*ανάγνωση προηγείται της εγγραφής*,



Σχήμα 3.2: *Εξαρτήσεις δεδομένων τριπλά φωλιασμένου βρόχου. Ο υπολογισμός του στοιχείου A_{j_1,j_2,j_3} κατά την επανάληψη (j_1, j_2, j_3) απαιτεί τη χρήση της τιμής των στοιχείων A_{j_1-1,j_2,j_3}, A_{j_1,j_2-1,j_3} και A_{j_1,j_2,j_3-2}, που υπολογίστηκαν στις επαναλήψεις $(j_1 - 1, j_2, j_3)$, $(j_1, j_2 - 1, j_3)$ και $(j_1, j_2, j_3 - 2)$, αντίστοιχα.*

δηλαδή η ίδια θέση μνήμης αρχικά διαβάζεται και στη συνέχεια γράφεται) μπορούν να εξαλειφθούν με τη χρήση επιπλέον μεταβλητών. Στην παρούσα εργασία θα περιοριστούμε συνεπώς στις εξαρτήσεις δεδομένων τύπου ροής, ενώ εκτενέστερη αναφορά στο θέμα των εξαρτήσεων δεδομένων γίνεται στα [Ban88, Zig94].

3.2 Βασικές Παραδοχές

Στο πλαίσιο της παρούσας εργασίας θα υιοθετήσουμε ένα σύνολο από παραδοχές για τον περιορισμό του αλγοριθμικού μοντέλου σε συγκεκριμένες προδιαγραφές. Οι περισσότερες από αυτές τις παραδοχές δεν συνιστούν περιορισμό των υπό εξέταση εφαρμογών, τουλάχιστον όχι σε μη ρεαλιστικό βαθμό. Συνοψίζοντας, μπορούμε να πούμε ότι οι παραδοχές αφορούν σε τέσσερις άξονες, τη *μορφή των φω-*

λιασμένων βρόχων, το *είδος του υπολογισμού* στο σώμα των φωλιασμένων βρόχων, τη *μορφή του χώρου επαναλήψεων*, καθώς και κυριότερα τις *εξαρτήσεις δεδομένων* του αλγορίθμου. Με βάση τη διάκριση αυτή, μπορούμε να επισημάνουμε τις ακόλουθες παραδοχές στο αλγοριθμικό μας μοντέλο:

1. **Μορφή φωλιασμένων βρόχων**

   Στην παρούσα εργασία θα καταπιαστούμε με την εξέταση *τέλεια φωλιασμένων βρόχων*, σε αντιδιαστολή με άλλες, γενικότερες περιγραφές φωλιασμένων βρόχων. Για παράδειγμα, σε αλγορίθμους μη τέλεια φωλιασμένων βρόχων παρεμβάλλονται επιπλέον κι άλλες εντολές μεταξύ των εντολών διάσχισης `for` των επαναλήψεων των βρόχων. Στους τέλεια φωλιασμένους βρόχους που θα εξετάσουμε, κάθε επανάληψη συσχετίζεται με την ίδια ακολουθία υπολογισμών και συνεπώς προσεγγιστικά με το ίδιο υπολογιστικό φορτίο. Κάτι τέτοιο δεν ισχύει στη γενική περίπτωση φωλιασμένων βρόχων, γεγονός που καθιστά την ομοιόμορφη κατανομή των επαναλήψεων μεταξύ των φορέων παράλληλης εκτέλεσης μη βέλτιστη επιλογή από πλευράς απόδοσης για την κατηγορία αυτή. Εντούτοις, όλες οι τεχνικές που αναφέρονται στην εργασία αυτή μπορούν να εφαρμοστούν και σε περιγραφές μη τέλεια φωλιασμένων βρόχων, εφόσον οι τελευταίες μετασχηματιστούν σε ισοδύναμες περιγραφές τέλεια φωλιασμένων βρόχων με χρήση εντολών `if`.

2. **Είδος υπολογισμού**

   Για απλότητα, και χωρίς βλάβη της γενικότητας, θεωρούμε ότι οι υπολογισμοί στο σώμα των βρόχων σχετίζονται με τις τιμές των στοιχείων ενός πίνακα A. Η χρήση μοναδικού πίνακα γίνεται καθαρά για λόγους ευκολίας και δεν συνιστά περιορισμό, αφού οι προτεινόμενες τεχνικές και μεθοδολογίες μπορούν να εφαρμοστούν αυτούσιες σε περιπτώσεις πιο σύνθετων σωμάτων των βρόχων. Στους αλγορίθμους θα υιοθετήσουμε μια αφαιρετική αναπαράσταση του υπολογισμού μέσω μιας γενικής συνάρτησης `Compute` σε αντιδιαστολή με αναφορά σε συγκεκριμένο είδος υπολογισμού (π.χ. απλή εντολή ανάθεσης παράστασης, περισσότερο σύνθετη δομή ελέγχου ή ακόμα και κλήση συνάρτησης βιβλιοθήκης). Άλλωστε, το εκάστοτε σώμα υπολογισμού θα χαρακτηρίζεται μακροσκοπικά κυρίως από το χώρο επαναλήψεων και τις εξαρτήσεις δεδομένων, ενώ στην περίπτωση της εξισορρόπησης φορτίου θα κάνουμε χρήση και του μέσου χρόνου υπολογισμού ανά επανάληψη.

3. **Μορφή χώρου επαναλήψεων**

   Για τους λόγους που αναφέραμε στην ενότητα 3.1.2 θα εξετάσουμε μόνο ορθογώνιους χώρους επαναλήψεων. Θα θεωρήσουμε συνεπώς ότι οι χώροι επαναλήψεων είναι της μορφής $[1 \ldots X_1] \times \cdots \times [1 \ldots X_N] \times [1 \ldots Z]$ ή σε συμπτυγμένη μορφή $X_1 \times \cdots \times X_N \times Z$. Όπως αναλύθηκε, η θεώρηση κατάλληλων ορθογώνιων πεδίων εφαρμογής συνάδει με τις συνήθεις πρακτικές αριθ-

μητικής επίλυσης, ακόμα και στην περίπτωση πιο σύνθετων φυσικών πεδίων. Κατά συνέπεια, το πλήθος των αλγορίθμων που θεωρούν ορθογώνιους χώρους επαναλήψεων είναι ιδιαίτερα ευρύ, ώστε η συγκεκριμένη παραδοχή να μη συνιστά ιδιαίτερο περιορισμό.

4. **Εξαρτήσεις δεδομένων**

Οι περισσότερες παραδοχές που θα κάνουμε αφορούν στη φύση των εξαρτήσεων δεδομένων, που σχετίζονται με τους υπό παραλληλοποίηση αλγορίθμους. Συγκεκριμένα, δεχόμαστε τα εξής:

(α) Οι εξαρτήσεις δεδομένων είναι *ομοιόμορφες και μη αρνητικές*, δηλαδή σταθερές και με όλες τις συνιστώσες τους μη αρνητικές. Η παραδοχή αυτή συνιστά ενδεχομένως και το βασικότερο περιορισμό του αλγοριθμικού μοντέλου, καθώς περιορίζει *φαινομενικά* το εύρος των υπό παραλληλοποίηση εφαρμογών σε εκείνες που σχετίζονται με μονόδρομη ροή πληροφορίας. Ωστόσο, είναι σημαντικό να επισημάνουμε ότι ο περιορισμός για ομοιόμορφες μη αρνητικές εξαρτήσεις δεδομένων οφείλεται αποκλειστικά στη χρήση του μετασχηματισμού υπερκόμβων για την επίτευξη χονδροειδή παραλληλισμού. Όπως θα δούμε, οι βασικές προγραμματιστικές βελτιστοποιήσεις που προτείνουμε επεκτείνονται άμεσα στο σύνολο των αλγορίθμων τέλεια φωλιασμένων βρόχων με εξαρτήσεις δεδομένων τύπου ροής.

(β) Στον αλγόριθμο 3.1 θα θεωρήσουμε ότι ο βαθμός $rank(D)$ του πίνακα D ισούται με $N+1$, δηλαδή ότι υπάρχουν $N+1$ γραμμικά ανεξάρτητα διανύσματα εξάρτησης δεδομένων. Αν υπάρχουν λιγότερα από $N+1$ γραμμικά ανεξάρτητα διανύσματα εξάρτησης ($rank(D) < N+1$) το πρόβλημα απλοποιείται σημαντικά, καθώς ο χώρος επαναλήψεων μπορεί να διαμεριστεί σε ανεξάρτητα υποσύνολα χωρίς τη χρήση του μετασχηματισμού υπερκόμβων. Έτσι, η περίπτωση $rank(D) < N+1$ επιτρέπει άμεση παραλληλοποίηση των ανεξάρτητων υποχώρων που προκύπτουν και δεν θα μας απασχολήσει στο παρόν σύγγραμμα, καθώς έχει μελετηθεί επιτυχώς στη διεθνή βιβλιογραφία [WL91b, D'H92]. Άλλωστε, η δυσκολία στην παραλληλοποίηση των υπό μελέτη αλγορίθμων έγκειται ακριβώς στην ύπαρξη εξαρτήσεων δεδομένων σε όλες τις διαστάσεις του χώρου επαναλήψεων, που αφενός μεν συνεπάγονται αυξημένες ανάγκες επικοινωνίας, αφετέρου δε καθιστούν την ίδια την παραλληλοποίηση του αλγορίθμου μη τετριμμένη διαδικασία. Συμπερασματικά, καταγράφουμε ως βασική παραδοχή για τον πίνακα εξαρτήσεων του υπό παραλληλοποίηση αλγορίθμου την ακόλουθη

συνθήκη:

$$rank(D) = rank\left(\begin{bmatrix} d_1^{(1)} & d_1^{(2)} & \cdots & d_1^{(m)} \\ d_2^{(1)} & d_2^{(2)} & \cdots & d_2^{(m)} \\ \vdots & \vdots & \cdots & \vdots \\ d_{N+1}^{(1)} & d_{N+1}^{(2)} & \cdots & d_{N+1}^{(m)} \end{bmatrix}\right) = N+1 \tag{3.2}$$

(γ) Επιπλέον, θα θεωρήσουμε ότι ο πίνακας εξαρτήσεων D είναι *διαγώνιος*, δηλαδή θα υποθέσουμε ότι ο αλγόριθμος περιέχει μόνο εξαρτήσεις παράλληλες με τις μοναδιαίες διευθύνσεις του $N+1$-διάστατου χώρου. Ακόμα κι αν κάτι τέτοιο δεν ισχύει, μπορούμε να εφαρμόσουμε όλες τις προτεινόμενες μεθοδολογίες της παρούσας εργασίας ορίζοντας αρχικά ένα νέο πίνακα εξαρτήσεων $D' = diag(d'_i)$, όπου $d'_i = max\{d_i^{(j)}, 1 \leq j \leq m\}$, και εφαρμόζοντας τις τεχνικές *έμμεσης ανταλλαγής μηνυμάτων* (*indirect message passing*) που περιγράφονται στην εργασία [TZ94]. Έτσι, η παραδοχή ορθογωνίων εξαρτήσεων δεδομένων σε καμία περίπτωση δεν συνιστά πρόσθετο περιορισμό του αλγοριθμικού μοντέλου. Θα θεωρήσουμε συνεπώς ότι

$$D = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & d_{N+1} \end{bmatrix} \tag{3.3}$$

(δ) Το μέτρο των διανυσμάτων εξάρτησης είναι μικρότερο από τις διαστάσεις του υπερκόμβου, όπως αυτός ορίζεται από τον αντίστοιχο μετασχηματισμό H που υιοθετείται κατά τη διαδικασία παραλληλοποίησης του προγράμματος. Τυπικά, δοθέντων m το πλήθος διανυσμάτων εξάρτησης $\vec{d}^{(l)}, 1 \leq l \leq m$, ενός χώρου επαναλήψεων $X_1 \times \cdots \times X_N \times Z$ διάστασης $N+1$ και ενός μετασχηματισμού υπερκόμβων $H = diag\{h_{11}, \ldots, h_{NN}, z\}$, αρκεί να πληρείται η ακόλουθη σχέση:

$$|d_i^{(l)}| < \left\lceil \frac{X_i}{h_{ii}} \right\rceil, \forall i, 1 \leq i \leq N \tag{3.4}$$

Η παραδοχή αυτή ισοδυναμεί με τη διαπίστωση ότι στην παράλληλη διαμερισμένη εκδοχή του προγράμματος θα απαιτείται επικοινωνία μόνο μεταξύ γειτονικών διεργασιών, δηλαδή μεταξύ διεργασιών που αναλαμβάνουν την εκτέλεση γειτονικών ακολουθιών από υπερκόμβους. Γενικά, η συγκεκριμένη παραδοχή κρίνεται ρεαλιστική, δεδομένου ότι μια τυπική υπολογιστικά απαιτητική εφαρμογή χαρακτηρίζεται από υψηλή χωρική πολυπλοκότητα,

ενώ οι εξαρτήσεις στη συντριπτική πλειοψηφία των περιπτώσεων συσχετίζουν γειτονικές ή έστω αρκούντως κοντινές επαναλήψεις. Έτσι, για κάθε δυνατή διαμέριση του αρχικού αλγορίθμου υπό την εφαρμογή ενός μετασχηματισμού υπερκόμβων, οι εξαρτήσεις αφορούν μόνο γειτονικούς υπερκόμβους και δεν οδηγούν σε αναγκαιότητα επικοινωνίας μεταξύ απομακρυσμένων διεργασιών.

(ε) Τέλος, όπως προαναφέραμε θα περιοριστούμε σε εξαρτήσεις ροής, έναντι των αντιεξαρτήσεων και των εξαρτήσεων εξόδου, καθώς οι δύο τελευταίες κατηγορίες μπορούν εύκολα να απαλειφθούν με τεχνικές που αναφέραμε παραπάνω.

Επιπλέον, θα πρέπει να επισημανθεί πως στη συνήθη περίπτωση η παράμετρος N δεν υπερβαίνει την τιμή 2 ή το πολύ 3, καθώς οι αλγόριθμοι αυτοί ως επί το πλείστον μοντελοποιούν φυσικά χωροχρονικά φαινόμενα. Η παρατήρηση αυτή δεν συνιστά παραδοχή, ενώ και η μεθοδολογία που θα αναπτυχθεί στη συνέχεια είναι αρκετά γενική, ώστε να καλύπτει τη γενική $N+1$-διάστατη περίπτωση. Ωστόσο, το πρακτικά περιορισμένο εύρος τιμών της διάστασης N θα πρέπει να λαμβάνεται υπόψη κατά τη θεωρητική ανάλυση εκφράσεων πολυπλοκότητας που εμπλέκουν την παράμετρο αυτή, ώστε να αποδίδεται μια ρεαλιστική εικόνα της αλγοριθμικής επίδοσης.

3.3 Παράδειγμα: Μερικές Διαφορικές Εξισώσεις

Βασική κατηγορία εφαρμογών που επιλύονται επαναληπτικά μέσω διακριτοποίησης αποτελούν οι Μερικές Διαφορικές Εξισώσεις. Μάλιστα, σε κάποιες περιπτώσεις η διακριτοποίηση τέτοιων εξισώσεων οδηγεί σε αλγορίθμους που μπορούν να υπαχθούν άμεσα στο υπό εξέταση αλγοριθμικό μοντέλο και κατά συνέπεια να παραλληλοποιηθούν με χρήση όλων των προτεινόμενων τεχνικών και βελτιστοποιήσεων. Στην ενότητα αυτή θα κάνουμε μια πολύ σύντομη αναφορά στις ΜΔΕ, τις τεχνικές διακριτοποίησης αυτών, καθώς και σε συγκεκριμένες διακριτοποιήσεις ορισμένων τυπικών διαφορικών εξισώσεων. Η κεντρική θέση που κατέχουν οι ΜΔΕ σε πλείστες εφαρμογές καθιστά σκόπιμη μια τέτοια ξεχωριστή αναφορά, έστω και με σχετικά σύντομο και περιληπτικό τρόπο.

3.3.1 Γενικά

Από μαθηματικής πλευράς, οι ΜΔΕ διαφοροποιούνται βάσει των χαρακτηριστικών καμπυλών διάδοσης πληροφορίας στις ακόλουθες τρεις κατηγορίες:

- **υπερβολικές** π.χ. η μονοδιάστατη κυματική εξίσωση:

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2} \tag{3.5}$$

όπου v η ταχύτητα διάδοσης του κύματος.

- **παραβολικές** π.χ. η εξίσωση διάχυσης:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) \tag{3.6}$$

όπου D ο συντελεστής διάχυσης.

- **ελλειπτικές** π.χ. η εξίσωση Poisson:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x, y) \tag{3.7}$$

όπου ρ ο όρος της πηγής (αν $\rho = 0$, τότε έχουμε την εξίσωση Laplace).

Από υπολογιστικής πλευράς, η παραπάνω κατηγοριοποίηση δεν έχει ιδιαίτερη πρακτική αξία. Αντίθετα, περισσότερο χρήσιμη από την συγκεκριμένη οπτική γωνία θα ήταν μια ταξινόμηση στις ακόλουθες κατηγορίες, ανάλογα με τη φύση των προβλημάτων που σχετίζονται με τις προαναφερθείσες ομάδες ΜΔΕ:

- **Προβλήματα Αρχικών Τιμών (Initial Value Problems - IVP)** Στην κατηγορία αυτή εμπίπτουν οι εξισώσεις (3.5) και (3.6). Ουσιαστικά, στην περίπτωση αυτή παρέχονται πληροφορίες για την τιμή της u και ενδεχομένως χρονικών παραγώγων αυτής σε κάποια αρχική χρονική στιγμή t_0 για κάθε σημείο ενός χώρου ενδιαφέροντος (domain) και ζητείται η χρονική εξέλιξη της u σε κάθε σημείο του χώρου αυτού για $t > t_0$. Τα προβλήματα αυτά αφορούν συνεπώς χρονική εξέλιξη, και το βασικό ζητούμενο είναι η *ευστάθεια* (*stability*) του αλγορίθμου επίλυσης, δηλαδή η σύγκλιση της αριθμητικής λύσης στην πραγματική αναφορικά με κάποια συγκεκριμένη ακρίβεια.

- **Προβλήματα Συνοριακών Τιμών (Boundary Value Problems - BVP)** Στην κατηγορία αυτή εντάσσεται η εξίσωση (3.7). Εδώ διαθέτουμε την πληροφορία για την τιμή της συνάρτησης u στο σύνορο της περιοχής ενδιαφέροντος, και προσπαθούμε να εξαγάγουμε την τιμή της u σε κάθε εσωτερικό σημείο του χώρου. Στα προβλήματα αυτά η διασφάλιση της ευστάθειας στον αλγόριθμο υπολογισμού είναι σχετικά απλό ζήτημα, και η έμφαση μετατοπίζεται στην επίτευξη *υψηλής επίδοσης* (*efficiency*).

Για την αριθμητική επίλυση ΜΔΕ έχουν αναπτυχθεί πλείστες τεχνικές, κάθε μία από τις οποίες προσφέρεται για συγκεκριμένο υποσύνολο των παραπάνω περιπτώσεων και επιτυγχάνει διαφορετικούς συμβιβασμούς στα ζητήματα ευστάθειας και επίδοσης, ανάλογα και με τη φύση του συγκεκριμένου προβλήματος. Ως βασικότερες μεθόδους θα μπορούσε κανείς να αναφέρει συνοπτικά τις εξής:

- Μέθοδος πεπερασμένων διαφορών (finite differencing method).
- Μέθοδος πεπερασμένων στοιχείων (finite element method).
- Μέθοδος Monte Carlo.
- Φασματική μέθοδος (spectral method).
- Μέθοδος των μεταβολών (variational method).

Στη συγκεκριμένη εργασία, θα εστιάσουμε στη μέθοδο των πεπερασμένων διαφορών, που αποτελεί άλλωστε μια ιδιαίτερα διαδεδομένη προσέγγιση για την αριθμητική επίλυση ΜΔΕ. Η μέθοδος των πεπερασμένων διαφορών βασίζεται στη διακριτοποίηση των χωρικών και χρονικών παραγώγων της ΜΔΕ, ώστε να διατυπωθεί μια ισοδύναμη *συνεπής* (*consistent*) και *ευσταθής* (*stable*) **εξίσωση διαφορών** (**differences equation**). Η εξίσωση αυτή αποτελεί τη βάση για τη διαδικασία αριθμητικής επίλυσης της ΜΔΕ με την περαιτέρω υιοθέτηση κάποιας μεθόδου επίλυσης, όπως η επαναληπτική μέθοδος ή η επίλυση αραιού (π.χ. τριδιαγώνιου) γραμμικού συστήματος, ανάλογα με το αν η εν λόγω εξίσωση διαφορών χαρακτηρίζεται αντίστοιχα ως *ρητή* (*explicit*) ή *πεπλεγμένη* (*implicit*).

Στην πειραματική αξιολόγηση των προτεινόμενων τεχνικών παραλληλοποίησης και βελτιστοποίησης θα αναφερθούμε σε επαναληπτική επίλυση ρητών εξισώσεων διαφορών, που έχουν προκύψει από τη διακριτοποίηση της αρχικής ΜΔΕ. Ουσιαστικά, στις περιπτώσεις αυτές υπολογίζεται επαναληπτικά η τιμή της πληροφορίας σε κάθε θέση και χρονική στιγμή χρησιμοποιώντας την τιμή της πληροφορίας σε γειτονικά χωρικά και προγενέστερα χρονικά σημεία. Πριν προχωρήσουμε λοιπόν στην υλοποίηση επαναληπτικών αλγορίθμων υπολογισμού της λύσης των ΜΔΕ, θα επιχειρήσουμε μια σύντομη επισκόπηση των (ρητών) διακριτοποιήσεων χώρου και χρόνου, που βρίσκονται στον πυρήνα της μεθόδου των πεπερασμένων διαφορών.

3.3.2 Διακριτοποιήσεις Χώρου

Τα φυσικά μεγέθη χαρακτηρίζονται συνήθως από την έννοια της χωρικής και χρονικής συνέχειας, ενώ αντίθετα οι επαναληπτικοί αλγόριθμοι εκ φύσεως υπολογίζουν ένα πεπερασμένο σύνολο διακριτών τιμών για το υπό εξέταση μέγεθος. Συνεπώς, απαιτείται η *διακριτοποίηση* (*discretization*) του φυσικού μεγέθους (π.χ. ροή, πίεση, θερμοκρασία) μέσω της διαμέρισης του χωροχρονικού πεδίου εφαρμογής σε διακριτά σημεία με χρήση κατάλληλου *πλέγματος* (*grid*).

Για παράδειγμα, μια συνάρτηση u της συνεχούς μεταβλητής x, όπου $x \in [a, b]$, διακριτοποιείται μέσω της επιλογής κάποιας πυκνότητας διακριτοποίησης Δx για το διάστημα $[a, b]$ στις τιμές u_j, με

$$u_j \equiv u(j) \equiv u(a + j\Delta x) \tag{3.8}$$

όπου $j \in \mathbb{N}, 0 \leq j \leq \lfloor \frac{b-a}{\Delta x} \rfloor$. Από υπολογιστικής πλευράς, η ποσότητα $\lfloor \frac{b-a}{\Delta x} \rfloor + 1$ παρέχει το πλήθος των τιμών της u που θα υπολογίσουμε στο διάστημα $[a, b]$, καθορίζοντας έτσι τις αντίστοιχες απαιτήσεις μνήμης.

Η διακριτοποίηση παραγώγων της συνάρτησης βασίζεται στο ανάπτυγμα κατά Taylor μιας συνεχούς συνάρτησης. Συγκεκριμένα, μια συνάρτηση $u(x)$ με $u \in \mathbb{C}^n$, όπου $\mathbb{C}^n$ το σύνολο όλων των συναρτήσεων που έχουν συνεχή n-στή παράγωγο, μπορεί να προσεγγιστεί ως πολυώνυμο n-στής τάξης ως εξής:

$$u_{n,x_0}(x) = \sum_{k=0}^{n} \frac{u^{(k)}(x_0)}{k!}(x - x_0)^k \tag{3.9}$$

Η σχέση (3.9) αποτελεί το ανάπτυγμα Taylor n-στής τάξης της συνάρτησης u περί το σημείο x_0, ενώ με $u^{(k)}(x_0)$ υποδηλώνεται η παράγωγος τάξης k της u στο x_0. Για διαδοχικές θέσεις του πλέγματος, έστω j και $j \pm \Delta x$, αντικαθιστούμε στην (3.9) το x_0 με j και το x με $j \pm \Delta x$, άρα και $x - x_0 = \pm\Delta x$, οπότε το ανάπτυγμα Taylor μπορεί να γραφεί ως εξής:

$$u(j \pm \Delta x) = u(j) \pm \Delta x u_x(j) + \frac{(\Delta x)^2 u_{xx}(j)}{2!} \pm \frac{(\Delta x)^3 u_{xxx}(j)}{3!} + \dots \tag{3.10}$$

όπου $u_x(j)$ η πρώτη παράγωγος της u στο σημείο j, $u_{xx}(j)$ η δεύτερη παράγωγος στο j κ.ο.κ.

Η βασική ιδέα της διακριτοποίησης των παραγώγων έγκειται στη χρήση αναπτυγμάτων Taylor σε υποσύνολο διαδοχικών σημείων του πλέγματος. Για παράδειγμα, προς την κατεύθυνση της διακριτοποίησης της πρώτης παραγώγου, θα μπορούσε κανείς να προσεγγίσει τη σχέση (3.10) ως εξής:

$$u(j \pm \Delta x) = u(j) \pm \Delta x u_x(j) + O\left((\Delta x)^2\right) \tag{3.11}$$

Η (3.11) αποτελεί την προσέγγιση της τιμής $u(j \pm \Delta x)$ με ακρίβεια δεύτερης τάξης $O\left((\Delta x)^2\right)$. Θεωρώντας το θετικό πρόσημο (περίπτωση $+\Delta x$) και επιλύοντας ως προς $u_x(j)$, προκύπτει η εξής σχέση για την πρώτη παράγωγο της u:

$$u_x(j) = \frac{u(j + \Delta x) - u(j)}{\Delta x} + O(\Delta x) \tag{3.12}$$

όπου $O(\Delta x) = -\frac{\Delta x}{2} u_{xx}(j)$ η ακρίβεια της προσέγγισης της πρώτης παραγώγου. Η (3.12) στη διακριτοποιημένη της εκδοχή μας παρέχει την **forward διακριτοποίηση** πρώτης τάξης της πρώτης παραγώγου:

$$\frac{du_j}{dx} = \frac{u_{j+1} - u_j}{\Delta x} + O(\Delta x) \tag{3.13}$$

Ακολουθώντας παρόμοια συλλογιστική και θεωρώντας το αρνητικό πρόσημο για το Δx στη σχέση (3.11), μπορούμε να εξαγάγουμε την **backward ή upstream διακριτοποίηση** πρώτης τάξης της πρώτης

παραγώγου:

$$\frac{du_j}{dx} = \frac{u_j - u_{j-1}}{\Delta x} + O(\Delta x) \tag{3.14}$$

όπου $O(\Delta x) = \frac{\Delta x}{2} u_{xx}(j)$ η ακρίβεια της προσέγγισης.

Τέλος, ο συνδυασμός των δύο παραλλαγών της (3.11) επιτρέπει την εξαγωγή **central διακριτοποίησης** της πρώτης παραγώγου, που επιτυγχάνει μάλιστα ακρίβεια δεύτερης τάξης ($-\frac{(\Delta x)^2}{6} u_{xxx}(j)$):

$$\frac{du_j}{dx} = \frac{u_{j+1} - u_{j-1}}{2\Delta x} + O\left((\Delta x)^2\right) \tag{3.15}$$

Παράγωγος	***Μέθοδος Διακριτοποίησης***		
	forward	**backward**	**central**
$\frac{du_j}{dx}$	$\frac{u_{j+1}-u_j}{\Delta x}+O(\Delta x)$	$\frac{u_j-u_{j-1}}{\Delta x}+O(\Delta x)$	$\frac{u_{j+1}-u_{j-1}}{2\Delta x}+O\left((\Delta x)^2\right)$
$\frac{d^2u_j}{dx^2}$	$\frac{u_j-2u_{j+1}+u_{j+2}}{(\Delta x)^2}+O(\Delta x)$	$\frac{u_j-2u_{j-1}+u_{j-2}}{(\Delta x)^2}+O(\Delta x)$	$\frac{u_{j+1}-2u_j+u_{j-1}}{(\Delta x)^2}+O\left((\Delta x)^2\right)$

Πίνακας 3.1: *Διακριτοποιήσεις πρώτης και δεύτερης παραγώγου χώρου*

Με παραπλήσιο τρόπο μπορούμε να επιτύχουμε την διακριτοποίηση των μερικών παραγώγων ανώτερης τάξης. Χάριν εποπτικότητας, ο πίνακας 3.1 συνοψίζει διαφορετικές δυνατότητες διακριτοποίησης της πρώτης και της δεύτερης παραγώγου της συνάρτησης u μίας πραγματικής μεταβλητής. Χρησιμοποιώντας τα παραπάνω, μπορούμε να αντικαταστήσουμε συνεχείς παραγώγους χώρου σε μια ΜΔΕ με τις αντίστοιχες διακριτοποιημένες εκδοχές στις ισοδύναμες εξισώσεις διαφορών.

3.3.3 Διακριτοποιήσεις Χρόνου

Ας υποθέσουμε ότι έχουμε το ακόλουθο πρόβλημα αρχικών συνθηκών:

$$\frac{du}{dt} = F(t, u) \tag{3.16}$$

όπου $u = u(t)$ συνάρτηση με γνωστές αρχικές τιμές $u(t = 0) = u_0$. Επιθυμούμε να διακριτοποιήσουμε την συνάρτηση u. Δεδομένου ότι στόχος είναι ο υπολογισμός της χρονικής εξέλιξης της $u(t)$ για $t > 0$, η διακριτοποίηση ισοδυναμεί με τον καθορισμό ενός κβάντου χρόνου Δt, βάσει του οποίου θα υπολογίσουμε μια πεπερασμένη ακολουθία διακριτών τιμών $u^n = u(n\Delta t)$.

Στη βιβλιογραφία έχουν επισημανθεί διάφορες μέθοδοι για τη διακριτοποίηση χρόνου. Δειγματοληπτικά μπορούμε να αναφέρουμε τις εξής μεθόδους, καθώς και την εφαρμογή τους στην (3.16):

- **Adams-Bashforth διακριτοποίηση**

  - πρώτης τάξης (Euler-forward), μονόπλευρη

$$\frac{u^{n+1}-u^n}{\Delta t} = F(t,u^n)$$

  - δεύτερης τάξης

$$\frac{u^{n+1}-u^n}{\Delta t} = \frac{3}{2}F(t,u^n) - \frac{1}{2}F(t,u^{n-1})$$

  - τρίτης τάξης

$$\frac{u^{n+1}-u^n}{\Delta t} = \frac{23}{12}F(t,u^n) - \frac{16}{12}F(t,u^{n-1}) + \frac{5}{12}F(t,u^{n-2})$$

- **leap-frog διακριτοποίηση** (κεντρική, δεύτερης τάξης)

$$\frac{u^{n+1}-u^{n-1}}{2\Delta t} = F(t,u^n)$$

3.3.4 Διακριτοποίηση της Εξίσωσης Μεταφοράς

Χρησιμοποιώντας τις διακριτοποιήσεις χώρου και χρόνου, μπορούμε να εξαγάγουμε έναν επαναληπτικό αλγόριθμο για τον αριθμητικό υπολογισμό της λύσης της **εξίσωσης μεταφοράς** (**advective equation**). Συγκεκριμένα, έστω ότι έχουμε μια διαταραχή $u(t,x)$ που διαδίδεται προς την κατεύθυνση $+x$ με ταχύτητα διάδοσης v σύμφωνα με την εξίσωση

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} \tag{3.17}$$

Η (3.17) έχει γενική λύση της μορφής $u = f(x-vt)$, όπου f τυχούσα συνάρτηση και v η ταχύτητα διάδοσης. Για τη διακριτοποίηση της εξίσωσης μεταφοράς, και κατ' επέκταση κάθε IVP προβλήματος, είναι ιδιαίτερα σημαντικό να διασφαλίζεται η ευστάθεια της αριθμητικής μεθόδου επίλυσης μέσω της εξίσωσης διαφορών. Μια σχετικά χαλαρή, πλην όμως αρκετά απλή και ακριβής μέθοδος για την εξακρίβωση της ευστάθειας της εξίσωσης διαφορών είναι η **ανάλυση ευστάθειας Von Neumann**. Πιο συγκεκριμένα, δεχόμαστε ότι οι συντελεστές της εξίσωσης διαφορών μεταβάλλονται αρκετά αργά σε σχέση με το χρόνο, ώστε να μπορούν να θεωρηθούν πρακτικά σταθεροί. Στην περίπτωση αυτή, οι ιδιοτιμές της εξίσωσης διαφορών μπορεί να θεωρηθούν ίσες με

$$u_j^n = \xi^n e^{ikj\Delta x} \tag{3.18}$$

όπου $\xi = \xi(k)$ μιγαδικός αριθμός, που εξαρτάται από τον κυματικό αριθμό k της διαταραχής. Για να είναι ευσταθής η εξίσωση διαφορών πρέπει για κάθε κυματικό αριθμό k να ισχύει η ακόλουθη συνθήκη:

$$|\xi(k)| \leq 1 \tag{3.19}$$

Σε περίπτωση που κάτι τέτοιο δεν ισχύει, αν δηλαδή $|\xi(k)| > 1$ για κάποιους κυματαριθμούς k, τότε παρατηρούνται ανεπιθύμητα φαινόμενα ενίσχυσης πλάτους (amplitude amplification) και η εξίσωση διαφορών είναι ασταθής.

Η διακριτοποίηση της εξίσωσης μεταφοράς μπορεί να επιτευχθεί μεταξύ άλλων με τις εξής τεχνικές:

- **FTCS - Forward Time Centered Space** σχήμα:

$$\begin{aligned} \frac{u_j^{n+1} - u_j^n}{\Delta t} &= -v\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \\ u_j^{n+1} &= u_j^n - v\frac{\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) \end{aligned}$$

  Η συγκεκριμένη διακριτοποίηση επιτυγχάνει ακρίβεια πρώτης τάξης ως προς το χρόνο ($O(\Delta t)$) και δεύτερης τάξης ως προς το χώρο ($O((\Delta x)^2)$). Δυστυχώς, είναι σε κάθε περίπτωση ασταθής, και κατά συνέπεια παρουσιάζει μικρή πρακτική αξία.

- **Lax** σχήμα:

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - v\frac{\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n)$$

  Η διακριτοποίηση Lax προκύπτει άμεσα από την FTCS με την προσέγγιση του u_j^n με την έκφραση $\frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$. Επιτυγχάνει ακρίβεια πρώτης τάξης ως προς το χρόνο και δεύτερης τάξης ως προς το χώρο. Είναι ευσταθής εφόσον πληρείται η **συνθήκη Courant**, που προκύπτει από την ανάλυση Von Neumann:

$$\frac{|v|\Delta t}{\Delta x} \leq 1 \tag{3.20}$$

  Στον αντίποδα, η διακριτοποίηση Lax σχετίζεται με σφάλμα εκ μεταφοράς: μια διαταραχή της υπό εξέταση ποσότητας u στο σημείο j του πλέγματος διαδίδεται τόσο στο σημείο $j+1$ όσο και στο $j-1$ κατά την επόμενη χρονική στιγμή. Όμως, εφόσον η ταχύτητα διάδοσης v είναι θετική, σημασιολογικά η διαταραχή στο j θα έπρεπε να επηρεάσει μόνο τη θέση $j+1$.

- **upwind** σχήμα:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = -v\begin{cases}\frac{u_j^n-u_{j-1}^n}{\Delta x} & , \quad v>0\\ \frac{u_{j+1}^n-u_j^n}{\Delta x} & , \quad v<0\end{cases}$$

Αντίθετα με τις προαναφερθείσες, η διακριτοποίηση αυτή επιτυγχάνει ακρίβεια πρώτης τάξης τόσο ως προς το χρόνο όσο και ως προς το χώρο. Χαρακτηρίζεται ως ευσταθής εφόσον πληρείται η συνθήκη Courant, ενώ το βασικό της πλεονέκτημα είναι ότι προσθέτει αξιοπιστία σε προβλήματα όπου οι διαδιδόμενες μεταβλητές υπόκεινται σε ξαφνικές αλλαγές κατάστασης (π.χ. ξαφνικές διαταραχές ή άλλες ασυνέχειες). Στη συγκεκριμένη διακριτοποίηση, ύλη που βρίσκεται αρχικά σε απόσταση $v\Delta t$ καταφτάνει σε ένα συγκεκριμένο σημείο μετά την παρέλευση χρονικού διαστήματος Δt. Σε διακριτοποιήσεις με ακρίβεια πρώτης τάξης, η ύλη προέρχεται πάντα από απόσταση Δx. Σε περίπτωση που επιλέξουμε $v\Delta t \ll \Delta x$ για αύξηση της ακρίβειας, αποδεικνύεται πως ενδέχεται να προκληθεί σημαντικό σφάλμα.

- **staggered leapfrog** σχήμα:

$$\begin{aligned}\frac{u_j^{n+1}-u_j^{n-1}}{2\Delta t} &= -v\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}\\ u_j^{n+1} &= u_j^{n-1}-v\frac{\Delta t}{\Delta x}(u_{j+1}^n-u_{j-1}^n)\end{aligned}$$

Στην περίπτωση αυτή επιτυγχάνεται ακρίβεια δεύτερης τάξης τόσο ως προς το χρόνο όσο και ως προς το χώρο. Η τήρηση της συνθήκης Courant διασφαλίζει την ευστάθεια της μεθόδου, ενώ επιπλέον αποφεύγεται η διασπορά πλάτους, καθώς $|\xi| = 1$ για $v\Delta t \leq \Delta x$. Ενδέχεται όμως να εμφανιστούν αστάθειες λόγω της αποσύζευξης των άρτιων και των περιττών σημείων του πλέγματος.

- **two-step Lax-Wendroff** σχήμα:

$$\left.\begin{aligned}u_{j+1/2}^{n+1/2} &= \tfrac{1}{2}(u_{j+1}^n+u_j^n)-\tfrac{v\Delta t}{2\Delta x}(u_{j+1}^n-u_j^n)\\ u_j^{n+1} &= u_j^n-\tfrac{v\Delta t}{\Delta x}(u_{j+1/2}^{n+1/2}-u_{j-1/2}^{n+1/2})\end{aligned}\right\}\Rightarrow$$

$$\begin{aligned}u_j^{n+1} = u_j^n-\frac{v\Delta t}{\Delta x}\Big[&\frac{1}{2}(u_{j+1}^n+u_j^n)-\frac{v\Delta t}{2\Delta x}(u_{j+1}^n-u_j^n)\\ &-\frac{1}{2}(u_j^n+u_{j-1}^n)+\frac{v\Delta t}{2\Delta x}(u_j^n-u_{j-1}^n)\Big]\end{aligned}$$

Η διακριτοποίηση αυτή επιτυγχάνει ακρίβεια δεύτερης τάξης σε χώρο και χρόνο, όπως και η προηγούμενη. Η συνθήκη Courant πρέπει να πληρείται για διασφάλιση της ευστάθειας. Γενικά,

η συγκεκριμένη διακριτοποίηση αποτελεί την πλέον ενδεδειγμένη, εκτός αν μελετάμε περιπτώσεις φυσικών προβλημάτων με ασυνέχειες, όπου η μονόπλευρη upwind διακριτοποίηση συνήθως πλεονεκτεί.

Με ανάλογο τρόπο μπορεί να διακριτοποιηθεί η εξίσωση μεταφοράς σε περισσότερες διαστάσεις. Χάριν απλότητας, συνήθως θεωρείται ομοιόμορφο πλέγμα με χωρική διακριτοποίηση πάχους Δ προς όλες τις διαστάσεις του πλέγματος. Στις N διαστάσεις και για ομοιόμορφο πλέγμα, η συνθήκη Courant διαμορφώνεται ως εξής:

$$\Delta t \leq \frac{\Delta}{\sqrt{N}|v|} \tag{3.21}$$

όπου $|v|$ η μέγιστη ταχύτητα διάδοσης της διαταραχής.

Με χρήση των παραπάνω διακριτοποιήσεων είναι δυνατό να προκύψει αλγόριθμος φωλιασμένων βρόχων για την αριθμητική επίλυση φαινομένου που διέπεται από την εξίσωση μεταφοράς. Ιδιαίτερα η upwind διακριτοποίηση αντιστοιχεί σε αλγόριθμο φωλιασμένων βρόχων, που παρουσιάζει τις επιθυμητές ομοιόμορφες μη αρνητικές εξαρτήσεις δεδομένων, και μπορεί να παραλληλοποιηθεί άμεσα με όλες τις προτεινόμενες τεχνικές. Για να αποσαφηνιστεί η ακριβής εξαγωγή του επαναληπτικού αλγορίθμου φωλιασμένων βρόχων από την εξίσωση μεταφοράς με χρήση χωρικών και χρονικών διακριτοποιήσεων παραθέτουμε το ακόλουθο παράδειγμα.

Παράδειγμα 3.1. *Έστω ότι θέλουμε να μελετήσουμε την εφαρμογή της εξίσωσης μεταφοράς σε ένα δισδιάστατο χώρο Ω. Αν $u = u(t, x, y)$ η συνάρτηση μεταφοράς, η ΜΔΕ με αρχικές και συνοριακές συνθήκες δίνεται από την ακόλουθη σχέση:*

$$\left.\begin{aligned} \frac{\partial u}{\partial t} &= -v_x \frac{\partial u}{\partial x} - v_y \frac{\partial u}{\partial y} \\ u(t=0,x,y) &= f(x,y) \\ u(t,x,y) &= g(t,x,y) \text{ στο σύνορο } \partial\Omega \end{aligned}\right\} \tag{3.22}$$

όπου η συνάρτηση f αναπαριστά τις αρχικές συνθήκες κατά τη χρονική στιγμή $t = 0$, ενώ η g παρέχει τις συνοριακές συνθήκες στις επιφάνειες $x = 0$ και $y = 0$.

Αξιοποιώντας τις τεχνικές διακριτοποίησης που αναφέρθηκαν παραπάνω, διακριτοποιούμε με Euler forward σχήμα ακρίβειας πρώτης τάξης τη χρονική παράγωγο

$$\frac{\partial u}{\partial t} = \frac{u_{x,y}^{n+1} - u_{x,y}^{n}}{\Delta t} \tag{3.23}$$

και με backward σχήμα ακρίβειας πρώτης τάξης τις χωρικές παραγώγους

$$\frac{\partial u}{\partial x} = \frac{u^n_{x,y} - u^n_{x-1,y}}{\Delta x} \tag{3.24}$$

$$\frac{\partial u}{\partial y} = \frac{u^n_{x,y} - u^n_{x,y-1}}{\Delta y} \tag{3.25}$$

Από τις (3.22), (3.23), (3.24) *και* (3.25) *προκύπτει η ακόλουθη εξίσωση διαφορών:*

$$\begin{aligned} \frac{u^{n+1}_{x,y} - u^n_{x,y}}{\Delta t} &= -v_x \frac{u^n_{x,y} - u^n_{x-1,y}}{\Delta x} - v_y \frac{u^n_{x,y} - u^n_{x,y-1}}{\Delta y} \\ u^{n+1}_{x,y} &= \left(1 - v_x \frac{\Delta t}{\Delta x} - v_y \frac{\Delta t}{\Delta y}\right) u^n_{x,y} + v_x \frac{\Delta t}{\Delta x} u^n_{x-1,y} + v_y \frac{\Delta t}{\Delta y} u^n_{x,y-1} \end{aligned} \tag{3.26}$$

Η συνθήκη Courant για την ευστάθεια της επαναληπτικής μεθόδου υπαγορεύει την επιλογή κατάλληλου κβάντου χρόνου Δt, που να πληρεί την ακόλουθη σχέση:

$$\Delta t \leq \frac{1}{\sqrt{2\left(\left(\frac{v_x}{\Delta x}\right)^2 + \left(\frac{v_y}{\Delta y}\right)^2\right)}} \tag{3.27}$$

ή για ομοιόμορφο πλέγμα με $\Delta x = \Delta y = \Delta$

$$\Delta t \leq \frac{\Delta}{\sqrt{2\left(v_x^2 + v_y^2\right)}} \tag{3.28}$$

Τέλος, βασιζόμενοι στη σχέση (3.26)*, μπορούμε εύκολα να προδιαγράψουμε υπό μορφή κώδικα το σχετικό αλγόριθμο επαναληπτικού υπολογισμού της u: Αν X, Y τα μήκη των πλευρών του ορθογωνίου, που περικλείει τον Ω, και T το χρονικό παράθυρο, κατά τη διάρκεια του οποίου επιθυμούμε να παρατηρήσουμε το φαινόμενο της μεταφοράς, τότε ο τριπλά φωλιασμένος βρόχος θα έχει τη μορφή του αλγορίθμου 3.3. Ο αλγόριθμος αποτελείται από τρεις φωλιασμένους βρόχους, που περικλείουν τον υπολογισμό των τιμών της u στο διακριτοποιημένο πλέγμα. Αν η συνθήκη $t-1 == 0$ αληθεύει, τότε οι τιμές της u που αναφέρονται στην προηγούμενη χρονική στιγμή πρέπει να ληφθούν από το σύνολο των αρχικών τιμών, οπότε και χρησιμοποιείται η συνάρτηση f. Αν η συνθήκη $x - 1 == 0$ επιστρέφει αληθή τιμή, τότε για τη γειτονική τιμή της u κατά x θα πρέπει να χρησιμοποιηθεί κατάλληλη τιμή από τη συνοριακή επιφάνεια $x = 0$, η οποία λαμβάνεται μέσω κλήσης της συνάρτησης g. Ομοίως, αν $y - 1 == 0$ η γειτονική τιμή της u στην αμέσως προηγούμενη θέση κατά y υπολογίζεται μέσω της συνάρτησης g. Κατά τ' άλλα, ο αλγόριθμος 3.3 υλοποιεί άμεσα τον υπολογισμό που περιγράφεται από τη σχέση* (3.26).

Αλγόριθμος 3.3: Αλγοριθμική περιγραφή φωλιασμένων βρόχων για δισδιάστατη εξίσωση μεταφοράς

Δεδομένα: Πεδίο εφαρμογής ($T \times X \times Y$), παράμετροι διακριτοποίησης (Δt,Δx,Δy), ταχύτητες ροής (v_x,v_y), αρχικές τιμές (f), οριακές τιμές (g)
Ζητούμενα: Συνάρτηση μεταφοράς $u(t, x, y)$

```
1  for t ← 1 to T/Δt do
2    for x ← 1 to X/Δx do
3      for y ← 1 to Y/Δy do
4        if t − 1 == 0 then
5          u[t, x, y] = (1 − v_x Δt/Δx − v_y Δt/Δy) f(x, y) + v_x Δt/Δx f(x − 1, y) + v_y Δt/Δy f(x, y − 1) ;
6        else
7          u[t, x, y] = (1 − v_x Δt/Δx − v_y Δt/Δy) u[t − 1, x, y];
8          if x − 1 == 0 then
9            u[t, x, y] += v_x Δt/Δx g(t − 1, x − 1, y);
10         else
11           u[t, x, y] += v_x Δt/Δx u[t − 1, x − 1, y];
12         endif
13         if y − 1 == 0 then
14           u[t, x, y] += v_y Δt/Δy g(t − 1, x, y − 1);
15         else
16           u[t, x, y] += v_y Δt/Δy u[t − 1, x, y − 1];
17         endif
18       endif
19     endfor
20   endfor
21 endfor
```

ΚΕΦΑΛΑΙΟ 4

Παράλληλο Προγραμματιστικό Μοντέλο Ανταλλαγής Μηνυμάτων

Το προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων αποτελεί την καθιερωμένη προσέγγιση στο χώρο της Παράλληλης Επεξεργασίας, καθώς έχει υιοθετηθεί από τη μεγάλη πλειοψηφία της επιστημονικής και ακαδημαϊκής κοινότητας. Το πρότυπο MPI (Message Passing Interface, [For94]) έχει συντελέσει καθοριστικά προς την κατεύθυνση αυτή, παρέχοντας ένα ισχυρό προγραμματιστικό περιβάλλον, που όπως έχει αποδειχθεί στην πράξη μπορεί να εφαρμοστεί αποδοτικά σε πληθώρα παράλληλων αρχιτεκτονικών. Η επιτυχία της διεπαφής MPI έγκειται αφενός στην υψηλή επιτάχυνση του χρόνου εκτέλεσης που επιτυγχάνεται με τη συγκεκριμένη βιβλιοθήκη σε πραγματικές παράλληλες εφαρμογές μεγάλης κλίμακας (χιλιάδες διεργασίες) και αφετέρου στη γενικότητα του SPMD προγραμματιστικού μοντέλου που υποστηρίζει, καθώς ο ίδιος κώδικας MPI μπορεί να εκτελεστεί επιτυχώς τόσο σε αρχιτεκτονικές μοιραζόμενης μνήμης όσο και σε αρχιτεκτονικές κατανεμημένης μνήμης. Το βασικότερο μειονέκτημα της βιβλιοθήκης MPI εντοπίζεται στη σχετική προγραμματιστική πολυπλοκότητα του μοντέλου ανταλλαγής μηνυμάτων, που απαιτεί από τον προγραμματιστή τη ρητή υλοποίηση συγκεκριμένων σχημάτων κατανομής τόσο του υπολογισμού όσο και των δεδομένων του αλγορίθμου μεταξύ των διεργασιών. Επίσης, η γενικότητα που παρέχει ο προγραμματισμός με ανταλλαγή μηνυμάτων αντισταθμίζεται εν μέρει από τον ενιαίο, μονολιθικό τρόπο αντιμετώπισης και διαχείρισης των διεργασιών, που στη θεωρία τουλάχιστον αδυνατεί να αξιοποιήσει τα επιμέρους χαρακτηριστικά μιας πολυεπίπεδης, ιεραρχικής υποδομής, όπως είναι μια SMP συστοιχία.

Στο κεφάλαιο αυτό θα παρουσιαστεί μια αποδοτική μέθοδος παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων με ομοιόμορφες εξαρτήσεις δεδομένων με τη βοήθεια του μονολιθικού μοντέλου

παράλληλου προγραμματισμού.

4.1 Μετασχηματισμός Υπερκόμβων

Κομβικό ρόλο στην παραλληλοποίηση των αλγορίθμων φωλιασμένων βρόχων κατέχει ο *μετασχηματισμός υπερκόμβων* (*tiling transformation*). Ο μετασχηματισμός υπερκόμβων προτάθηκε το 1988 από τους Irigoin και Triolet [IT88] και έχει έκτοτε καθιερωθεί ως ιδιαίτερα δημοφιλής μετασχηματισμός για περιγραφές φωλιασμένων βρόχων. Ανήκει στην κατηγορία των *μη γραμμικών* (*non-linear*) μετασχηματισμών, σε αντιδιαστολή με τους γραμμικούς μετασχηματισμούς. Πιο συγκεκριμένα, έστω το διάνυσμα επαναλήψεων $\vec{j} = (j_1, \ldots, j_{N+1})$ ενός χώρου επαναλήψεων J διάστασης $N+1$. Ένας γραμμικός μετασχηματισμός απεικονίζει τον αρχικό χώρο J σε ένα νέο χώρο

$$J' = \{\vec{j'} \mid \vec{j'} = H\vec{j}, \vec{j} \in J\}$$

όπου H ο ισοδύναμος πίνακας περιγραφής του μετασχηματισμού. Αντίθετα, ο μη γραμμικός μετασχηματισμός υπερκόμβων απεικονίζει τον J σε ένα νέο χώρο

$$J' = \left\{ \vec{j'} \mid \vec{j'} = \begin{bmatrix} \lfloor H\vec{j} \rfloor \\ \vec{j} - H^{-1}\lfloor H\vec{j} \rfloor \end{bmatrix}, \vec{j} \in J \right\} \tag{4.1}$$

Ο μετασχηματισμός υπερκόμβων χρησιμοποιείται τόσο για την υλοποίηση παραλληλισμού χονδρού κόκκου σε συστήματα κατανεμημένης μνήμης όσο και για την αξιοποίηση της ιεραρχίας των συστημάτων μνήμης με ανάδειξη της *τοπικότητας αναφοράς δεδομένων* (*data locality of reference*). Στην παρούσα εργασία θα εξεταστεί η χρήση του μετασχηματισμού υπερκόμβων για την υλοποίηση παραλληλισμού χονδρού κόκκου, τόσο μέσω του απλού μοντέλου ανταλλαγής μηνυμάτων όσο και μέσω υβριδικών παράλληλων προγραμματιστικών μοντέλων. Περισσότερα για την εφαρμογή του μετασχηματισμού υπερκόμβων στην αξιοποίηση της τοπικότητας αναφοράς των δεδομένων μπορούν να αναζητηθούν στις εργασίες [LRW91, WL91a, KM92, LP92, CMT94, CM95, CL95, MCT96, KCS+98, KCRB99, KRC99, CM99, SL01].

Ο μετασχηματισμός υπερκόμβων ουσιαστικά διαμερίζει τον αρχικό χώρο επαναλήψεων σε ατομικές μονάδες εκτέλεσης, τους υπερκόμβους. Έτσι, η συνιστώσα $\lfloor H\vec{j} \rfloor$ του διανύσματος $\vec{j'}$ της (4.1) καθορίζει τον υπερκόμβο που περικλείει την αρχική επανάληψη $\vec{j}$, ενώ η έκφραση $\vec{j} - H^{-1}\lfloor H\vec{j} \rfloor$ ισοδυναμεί με τη σχετική θέση της επανάληψης $\vec{j}$ στο συγκεκριμένο υπερκόμβο. Σε ένα σύστημα κατανεμημένης μνήμης, η διαμέριση αυτή μπορεί να διευκολύνει σημαντικά την παραλληλοποίηση του αλγορίθμου, καθώς απλοποιεί τόσο τη διαδικασία κατανομής των υπολογισμών και των δεδομένων του αλγορίθμου όσο

και τη διαδικασία απεικόνισης του αλγορίθμου στο υφιστάμενο υπολογιστικό σύστημα. Συγκεκριμένα, κάθε επεξεργαστής αναλαμβάνει την εκτέλεση των υπολογισμών που σχετίζονται με μια ακολουθία υπερκόμβων, ενώ τα δεδομένα επικοινωνίας ομαδοποιούνται σε επίπεδο υπερκόμβου, ώστε να μειώνεται ο αριθμός των ανταλλασσόμενων μηνυμάτων και η σχετική καθυστέρηση που αυτά επιφέρουν. Ουσιαστικά, ο υπερκόμβος θεωρείται στο μετασχηματισμένο πρόγραμμα ως *ατομική* μονάδα εκτέλεσης, υπό την έννοια ότι οι υπολογισμοί που σχετίζονται με τις επαναλήψεις του υπερκόμβου εκτελούνται αδιαίρετα σε ένα βήμα, ενώ η επικοινωνία που απαιτείται για την αποστολή/λήψη δεδομένων αφορά στο σύνολο των δεδομένων που παράγονται/απαιτούνται από έναν υπερκόμβο. Το μοντέλο εκτέλεσης περιλαμβάνει διαδοχικές εναλλαγές μεταξύ υπολογισμών εντός του υπερκόμβου και επικοινωνίας για την ανταλλαγή συνοριακών δεδομένων.

Αλγόριθμος 4.1: Παράδειγμα τρισδιάστατου αλγορίθμου φωλιασμένων βρόχων

```
1 for j1 ← 0 to 5 do
2   for j2 ← 0 to 3 do
3     for j3 ← 0 to 7 do
4       loop-body ;
5     endfor
6   endfor
7 endfor
```

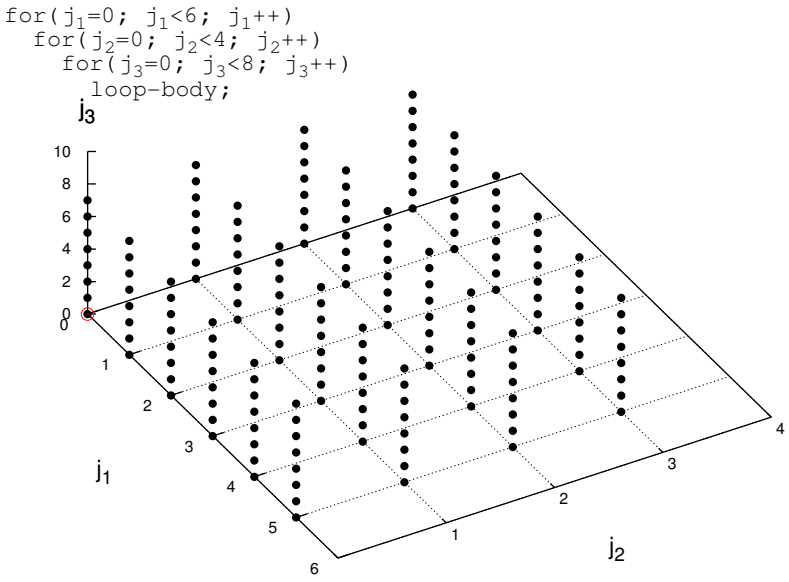
Παράδειγμα 4.1. *Έστω ο αλγόριθμος 4.1 τριπλά φωλιασμένων βρόχων, ο χώρος επαναλήψεων του οποίου απεικονίζεται στο σχήμα 4.1(α). Ας υποθέσουμε ότι στον αλγόριθμο εφαρμόζουμε μετασχηματισμό υπερκόμβων στις δύο εξωτερικές διαστάσεις j_1 και j_2 για την ομαδοποίηση των επαναλήψεων σε ορθογώνιους υπερκόμβους βάσης 3×2. Ο αντίστροφος του πίνακα μετασχηματισμού H^{-1} προκύπτει αν θεωρήσουμε ως στήλες τα διανύσματα των πλευρών του υπερκόμβου, δηλαδή θα είναι*

$$H^{-1} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$

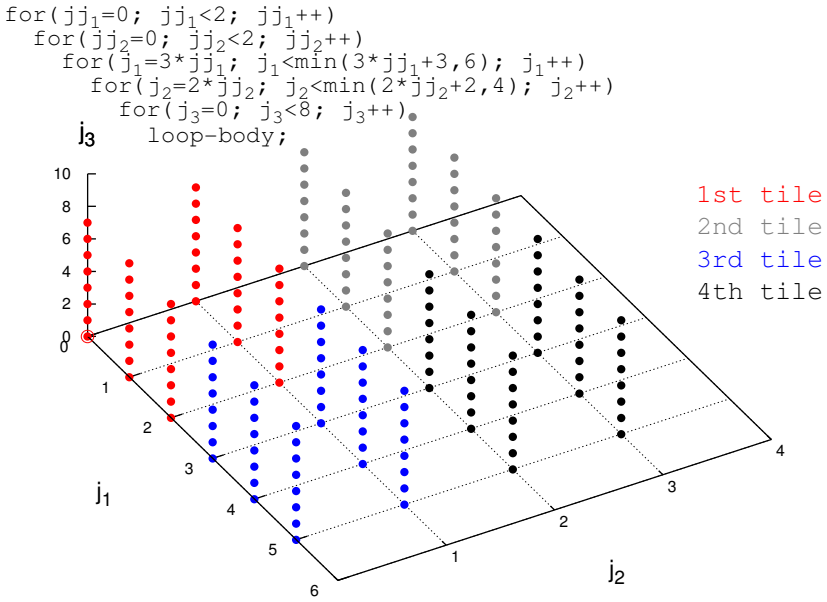
και συνεπώς ο πίνακας μετασχηματισμού παρέχεται από τη σχέση

$$H = \frac{1}{6} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

Ο πίνακας H μπορεί να εφαρμοστεί στη σχέση (4.1) *για να περιγραφούν τυπικά τα διανύσματα επανάληψης $\vec{j'}$, που σαρώνουν το μετασχηματισμένο χώρο στις δύο εξωτερικές διαστάσεις. Επειδή στην προκειμένη περίπτωση θεωρούμε ορθογώνιους υπερκόμβους, η κατάσταση απλοποιείται και ο μετασχημα-*



(α) αρχικός αλγόριθμος



(β) μετασχηματισμένος αλγόριθμος

Σχήμα 4.1: *Εφαρμογή μετασχηματισμού υπερκόμβων σε αλγόριθμο φωλιασμένων βρόχων. Ο αρχικός αλγόριθμος μετασχηματίζεται με χρήση του μετασχηματισμού υπερκόμβων σε ισοδύναμο, που διευκολύνει τη διαμέριση των δεδομένων και την κατανομή των υπολογισμών.*

τισμένος αλγόριθμος δεν χρειάζεται να περιγραφεί περισσότερο σύνθετα από τον 4.2. Στο σχήμα 4.1(β) ομαδοποιούνται με διαφορετικό χρώμα οι επαναλήψεις καθενός από τους τέσσερις υπερκόμβους, που ορίζει ο H.

Αλγόριθμος 4.2: Μετασχηματισμένος αλγόριθμος φωλιασμένων βρόχων με χρήση μετασχηματισμού υπερκόμβων

```
1  for jj1 ← 0 to 1 do
2    for jj2 ← 0 to 1 do
3      for j1 ← 3 * jj1 to min(3 * jj1 + 2;5) do
4        for j2 ← 2 * jj2 to min(2 * jj2 + 1;3) do
5          for j3 ← 0 to 7 do
6            loop-body ;
7          endfor
8        endfor
9      endfor
10   endfor
11 endfor
```

Στο πλαίσιο της παρούσας εργασίας θα ασχοληθούμε με ορθογώνιους μετασχηματισμούς υπερκόμβων (rectangular tiling transformations), σε αντιδιαστολή με γενικότερους, μη ορθογώνιους μετασχηματισμούς υπερκόμβων (non-rectangular tiling transformations). Άλλωστε, ορθογώνιοι μετασχηματισμοί εξετάζονται στη μεγάλη πλειοψηφία των σχετικών εργασιών, καθώς και εφαρμόζονται σχεδόν στο σύνολο του παραλληλοποιημένου κώδικα, όπου γίνεται χρήση του μετασχηματισμού υπερκόμβων. Ο λόγος είναι ότι η εφαρμογή ορθογώνιου μετασχηματισμού υπερκόμβων υλοποιείται προγραμματιστικά με την απλή χρήση τελεστών διαίρεσης και ακεραίου υπολοίπου (division-modulo operators), χωρίς να απαιτούνται μεγάλες αλλαγές στον υπάρχοντα κώδικα. Αντίθετα, η εφαρμογή γενικότερων μη ορθογώνιων μετασχηματισμών συνήθως ισοδυναμεί με την επίλυση συστήματος ανισοτήτων για τον υπολογισμό των ορίων των βρόχων, ενώ επίσης η απεικόνιση των εν γένει μη ορθογώνιων χώρων που προκύπτουν σε συγκεκριμένη τοπολογία διεργασιών είναι ιδιαίτερα σύνθετη διαδικασία. Άλλωστε, για την κατηγορία των πλήρως αντιμεταθέσιμων βρόχων που εξετάζουμε, οι ορθογώνιοι μετασχηματισμοί είναι πάντα έγκυροι για τη διαμέριση του χώρου επαναλήψεων. Ακόμα και στη γενικότερη κατηγορία των φωλιασμένων βρόχων με *λεξικογραφικά θετικά* διανύσματα εξάρτησης, δηλαδή εξαρτήσεις για τις οποίες διασφαλίζεται ότι μόνο η πρώτη μη μηδενική συνιστώσα είναι θετική, μπορεί έμμεσα να εφαρμοστεί ορθογώνιος μετασχηματισμός υπερκόμβων, αφού πρώτα εφαρμοστεί κατάλληλος *μετασχηματισμός αλλαγής κλίσης* (*skewing transformation*). Περισσότερα για το θέμα αυτό, που εκφεύγει του αντικειμένου της παρούσας διατριβής, μπορούν να αναζητηθούν στην εργασία [Γκο03], όπου προ-

τείνονται γρήγοροι και προγραμματιστικά αποδοτικοί τρόποι εφαρμογής μη ορθογώνιων μετασχηματισμών υπερκόμβων.

Θεωρώντας λοιπόν μόνο ορθογώνιους μετασχηματισμούς υπερκόμβων, η περαιτέρω ανάλυση απλοποιείται σημαντικά. Έτσι, αντί για τη γενική περιγραφή της (4.1), ο πίνακας μετασχηματισμού H είναι διαγώνιος πίνακας της μορφής

$$H = \begin{bmatrix} \frac{1}{h_{11}} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{h_{22}} & \cdots & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{h_{NN}} & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{z} \end{bmatrix} \tag{4.2}$$

ή σε συμπτυγμένη μορφή $H = diag\left\{\frac{1}{h_{11}}, \frac{1}{h_{22}}, \cdots, \frac{1}{h_{NN}}, \frac{1}{z}\right\}$. Ο λόγος για τον οποίο επιλέγουμε τη συγκεκριμένη κλασματική αναπαράσταση για τα μη μηδενικά στοιχεία του H έγκειται στο ότι ένας μετασχηματισμός υπερκόμβων που ορίζεται από τη σχέση (4.2) αντιστοιχεί σε ορθογώνιους υπερκόμβους διαστάσεων $h_{11} \times h_{22} \times \cdots \times h_{NN} \times z$. Επιπλέον, χρησιμοποιούμε διαφορετική αναπαράσταση για τα N πρώτα στοιχεία της διαγωνίου του H ($\frac{1}{h_{ii}}, 1 \leq i \leq N$), σε σχέση με το τελευταίο $(N+1)$ στοιχείο $\frac{1}{z}$, καθώς ο καθορισμός των πρώτων δεσμεύεται εν μέρει από το συνολικό διαθέσιμο πλήθος των διεργασιών, αφού οι N εξωτερικές διαστάσεις θα χρησιμοποιηθούν ουσιαστικά για την απεικόνιση του διαμερισμένου αλγορίθμου στις διεργασίες αυτές. Αντίθετα, η τιμή της μεταβλητής z αντιστοιχεί στο ύψος του κάθε υπερκόμβου, σχετίζεται αποκλειστικά με την ακολουθιακή (σειριακή) εκτέλεση υπερκόμβων εσωτερικά σε κάθε διεργασία, και μπορεί εύκολα να θεωρηθεί ως ελεύθερη παράμετρος, καθοριζόμενη από το χρήστη. Εφαρμόζοντας λοιπόν ορθογώνιο μετασχηματισμό υπερκόμβων H στον αλγόριθμο 3.1, προκύπτει η ισοδύναμη *διαμερισμένη* (*tiled*) εκδοχή του αλγορίθμου 4.3.

Στο αλγόριθμο 4.3 παρατηρούμε ότι η αρχική αλγοριθμική περιγραφή των $N+1$ φωλιασμένων βρόχων έχει μετατραπεί σε μια ισοδύναμη εκδοχή διπλάσιου πλήθους φωλιασμένων βρόχων $(2N+2)$, όπου οι $N+1$ εξωτερικοί (μεταβλητές ελέγχου $tile_i$) σαρώνουν το χώρο των υπερκόμβων (tile space), ενώ οι $N+1$ εσωτερικοί βρόχοι (μεταβλητές ελέγχου j_i) απαριθμούν το σύνολο των επαναλήψεων που αντιστοιχούν σε ένα συγκεκριμένο υπερκόμβο. Η χρήση των εντολών επανάληψης **foracross** αποσκοπεί στο να υποδείξει τους N εξωτερικούς βρόχους, στους οποίους θα βασιστεί η απεικόνιση των δεδομένων και των υπολογισμών στις διαθέσιμες διεργασίες κατά την παραλληλοποίηση του προγράμματος. Ουσιαστικά, οι εντολές **foracross** υποδεικνύουν τη δυνατότητα παράλληλης εκτέλεσης διαφορετικών υπερκόμβων με χρήση συγχρονισμού για τη διατήρηση της σημασιολογικής ορθότητας του αλγορίθμου.

Αλγόριθμος 4.3: Ισοδύναμο αλγοριθμικό μοντέλο υπό την εφαρμογή μετασχηματισμού υπερκόμβων $H = diag\left\{\frac{1}{h_{11}}, \frac{1}{h_{22}}, \cdots, \frac{1}{h_{NN}}, \frac{1}{z}\right\}$

```
1  foracross tile_1 ← 0 to ⌈X_1/h_11⌉ − 1 do
2    ...
3    foracross tile_N ← 0 to ⌈X_N/h_NN⌉ − 1 do
4      for tile_{N+1} ← 0 to ⌈Z/z⌉ − 1 do
5        for j_1 ← h_11 · tile_1 + 1 to min(h_11(tile_1 + 1), X_1) do
6          ...
7          for j_N ← h_NN · tile_N + 1 to min(h_NN(tile_N + 1), X_N) do
8            for j_{N+1} ← z · tile_{N+1} + 1 to min(z(tile_{N+1} + 1), Z) do
9              Compute(A, j⃗, D);
10           endfor
11         endfor
12         ...
13       endfor
14     endfor
15   endforacross
16   ...
17 endforacross
```

Συγκεκριμένα, μια γενική αποδοτική προσέγγιση, που επιλέγεται για την παραλληλοποίηση αλγοριθμικών περιγραφών φωλιασμένων βρόχων βάσει του *μοντέλου ανταλλαγής μηνυμάτων*, είναι η εξής: κάθε διεργασία του παράλληλου προγράμματος αναλαμβάνει την εκτέλεση μιας ακολουθίας υπερκόμβων, που είναι διαδοχικοί κατά μήκος της μεγαλύτερης διάστασης του μετασχηματισμένου χώρου επαναλήψεων ($\lceil \frac{Z}{z} \rceil$). Ουσιαστικά, οι N βρόχοι **foracross** αναπαριστούν την απεικόνιση του παράλληλου προγράμματος στις διαθέσιμες διεργασίες, ενώ οι $N+2$ βρόχοι **for** διατηρούνται αυτούσιοι στον SPMD κώδικα της κάθε διεργασίας, με τον πρώτο από αυτούς να απαριθμεί τους υπερκόμβους που έχουν ανατεθεί στη συγκεκριμένη διεργασία και τους υπόλοιπους $N+1$ να διατρέχουν τις επαναλήψεις στο εσωτερικό κάθε υπερκόμβου. Η μέθοδος αυτή επιτυγχάνει την αξιοποίηση του φαινομένου της *σωλήνωσης* (*pipelining*), που αποτελεί θεμελιώδη αρχή στην αρχιτεκτονική υπολογιστών προς την κατεύθυνση της υψηλής επίδοσης.

Το παραλληλοποιημένο ισοδύναμο πρόγραμμα του αλγορίθμου 4.3 βάσει του μοντέλου ανταλλαγής μηνυμάτων παρέχεται στον αλγόριθμο 4.4. Επιπλέον, ένα εποπτικό παράδειγμα για την καλύτερη κατανόησή του απεικονίζεται στο σχήμα 4.2.

Ο αλγόριθμος 4.4 μπορεί να περιγραφεί ως εξής: αρχικά γίνεται η κατανομή των υπερκόμβων στις διαθέσιμες διεργασίες, όπου κάθε διεργασία αναλαμβάνει την ακολουθία υπερκόμβων που ταυτοποιεί-

Αλγόριθμος 4.4: Προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων

Δεδομένα: Αλγόριθμος (`Compute`), χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, διεργασία $\vec{p}$

for $i \leftarrow 1$ **to** N **do**
  $tile_i = p_i$;
endfor
for $tile_{N+1} \leftarrow 0$ **to** $\lceil \frac{Z}{z} \rceil - 1$ **do**
  foreach $\overrightarrow{dir} \in \mathbb{S}_{\vec{p}}$ **do**
    `Pack(snd_buf` $[\overrightarrow{dir}]$,$tile_{N+1} - 1$,$\vec{p}$`);`
    `MPI_Isend(snd_buf` $[\overrightarrow{dir}]$,`dest(`$\vec{p} + \overrightarrow{dir}$`));`
  endforeach
  foreach $\overrightarrow{dir} \in \mathbb{R}_{\vec{p}}$ **do**
    `MPI_Irecv(recv_buf` $[\overrightarrow{dir}]$,`src(`$\vec{p} - \overrightarrow{dir}$`));`
  endforeach
  `Compute(`$\overrightarrow{tile}$`);`
  `MPI_Waitall;`
  foreach $\overrightarrow{dir} \in \mathbb{R}_{\vec{p}}$ **do**
    `Unpack(recv_buf` $[\overrightarrow{dir}]$,$tile_{N+1} + 1$,$\vec{p}$`);`
  endforeach
endfor

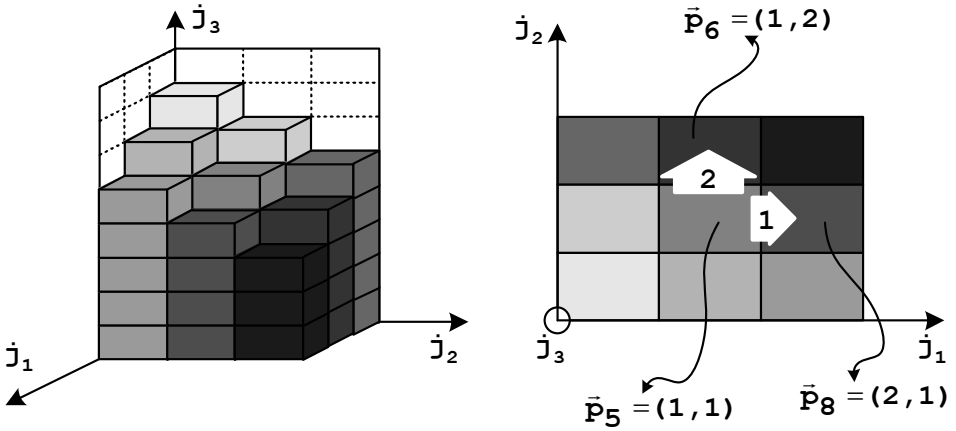
ται από το αναγνωριστικό που της έχει αποδοθεί από τη βιβλιοθήκη ανταλλαγής μηνυμάτων. Εν συνεχεία, στο κυρίως τμήμα του αλγορίθμου, κάθε διεργασία απαριθμεί τους υπερκόμβους που της έχουν ανατεθεί, και για κάθε έναν από αυτούς επιτελεί διαδοχικά τις εξής λειτουργίες:

1. Ομαδοποιεί τα συνοριακά δεδομένα που υπολόγισε κατά την επεξεργασία του *προηγούμενου* υπερκόμβου και εκκινεί μια ασύγχρονη λειτουργία αποστολής των δεδομένων αυτών προς τις γειτονικές διεργασίες που χρειάζεται να τα λάβουν.

2. Εκκινεί μια ασύγχρονη λειτουργία λήψης των συνοριακών δεδομένων που θα χρειαστεί για τους υπολογισμούς που σχετίζονται με τον *επόμενο* υπερκόμβο.

3. Εκτελεί τους υπολογισμούς που σχετίζονται με τις επαναλήψεις που περικλείει ο *τρέχων* υπερκόμβος.

4. Διασφαλίζει την επιτυχή ολοκλήρωση των ασύγχρονων λειτουργιών επικοινωνίας, δηλαδή τόσο της αποστολής όσο και της λήψης δεδομένων, για να μεταβεί στην επεξεργασία του επόμενου υπερκόμβου.



Σχήμα 4.2: *Παράλληλη εκτέλεση τρισδιάστατου αλγορίθμου με χρήση 9 διεργασιών. Ο αλγόριθμος απεικονίζεται στις διαθέσιμες διεργασίες ως προς τις διαστάσεις j_1, j_2, ενώ κατά την j_3 πραγματοποιείται ακολουθιακή εκτέλεση των υπερκόμβων σε κάθε διεργασία. Αριστερά φαίνεται η χρονοδρομολόγηση σωλήνωσης που επιτυγχάνεται, ενώ δεξιά παρατηρούμε την απεικόνιση των υπερκόμβων στο 3×3 πλέγμα των 9 διαθέσιμων διεργασιών, καθώς και την ανάγκη αποστολής δεδομένων από τη διεργασία $\vec{p}_5$ προς τις $\vec{p}_6$ και $\vec{p}_8$.*

Αναλυτικότερα, σε κάθε διεργασία αντιστοιχίζεται ένα μοναδικό διάνυσμα $\vec{p} = (p_1, \ldots, p_N)$ διάστασης N, ενώ οι διαφορετικοί υπερκόμβοι ταυτοποιούνται από τα διαφορετικά στιγμιότυπα του διανύσματος $\overrightarrow{tile} = (tile_1, \ldots, tile_N, tile_{N+1})$ διάστασης $N+1$. Οι N εξωτερικές συνιστώσες του διανύσματος $\overrightarrow{tile}$ ενός υπερκόμβου καθορίζουν την διεργασία-ιδιοκτήτη $\vec{p}$ του υπερκόμβου, ενώ η πλέον εσωτερική συνιστώσα $tile_{N+1}$ απαριθμεί την ακολουθία των υπερκόμβων που ανατίθενται στη διεργασία $\vec{p}$. Η γραμμή

$$\texttt{Compute}(\overrightarrow{tile})$$

αποτελεί συντομογραφικό συμβολισμό των γραμμών 5-13 του αλγορίθμου 4.3, και συνοψίζει τον υπολογισμό των επαναλήψεων $\vec{j} = (j_1, \ldots, j_{N+1})$ που περικλείονται στον τρέχοντα υπερκόμβο $\overrightarrow{tile}$. Η παράμετρος z υποδηλώνει το ύψος του υπερκόμβου κατά μήκος της διάστασης εκτέλεσης και έμμεσα καθορίζει τον κόκκο του παραλληλισμού. Έτσι, σχετικά υψηλές τιμές της παραμέτρου z σε σχέση με τη διάσταση Z του χώρου επαναλήψεων συνεπάγονται λιγότερο συχνή επικοινωνία, καθώς επίσης και παραλληλισμό χονδρού κόκκου. Από την άλλη πλευρά, σχετικά χαμηλές τιμές της παραμέτρου z (σε σχέση με τη διάσταση Z) απαιτούν συχνότερη επικοινωνία και ταυτόχρονα επιτρέπουν την επίτευξη παραλληλισμού λεπτού κόκκου. Στο πλαίσιο της παρούσας εργασίας θα θεωρήσουμε την τιμή του z σαν παράμετρο καθοριζόμενη από το χρήστη, ανάλογα με τη φύση της υφιστάμενης αρχιτεκτονικής (υλικό

και δίκτυο διασύνδεσης), καθώς και το είδος της υπό παραλληλοποίηση εφαρμογής. Στις πειραματικές μετρήσεις που θα ακολουθήσουν θα θεωρήσουμε μεταβλητό κόκκο παραλληλισμού, ώστε η βέλτιστη τιμή του ύψους z του υπερκόμβου να προσδιορίζεται πειραματικά μέσω των μετρήσεων αυτών.

Αξιοποιώντας το μετασχηματισμό υπερκόμβων, η κατανομή των υπολογισμών μεταξύ των επεξεργαστών μπορεί να γίνει με απλό τρόπο, όπως φαίνεται στις γραμμές 1-3 του αλγορίθμου 4.4. Έτσι, κάθε διεργασία $\vec{p} = (p_1, \ldots, p_N)$ αναλαμβάνει την ακολουθία των υπερκόμβων που ταυτοποιούνται από διανύσματα $\overrightarrow{tile}$ της μορφής $(p_1, \ldots, p_N, tile_{N+1})$. Το διάνυσμα $\vec{p}$ της κάθε διεργασίας μπορεί να προκύψει από το μοναδικό αναγνωριστικό που της έχει αποδοθεί από τη βιβλιοθήκη ανταλλαγής μηνυμάτων, π.χ. το βαθμό της (`rank`) εφόσον πρόκειται για τη βιβλιοθήκη MPI. Επίσης, η κατανομή των δεδομένων υλοποιείται έμμεσα μέσω της κατανομής των υπολογισμών και της υιοθέτησης του κανόνα *υπολογιστή-ιδιοκτήτη* (*computer-owns*): η διεργασία που υπολογίζει την τιμή κάποιου δεδομένου θεωρείται και ιδιοκτήτης του συγκεκριμένου δεδομένου, συνεπώς θα πρέπει να κοινοποιήσει την τιμή του σε όποια διεργασία τη χρειαστεί.

Το σύνολο $\mathbb{S}_{\vec{p}}$ υποδηλώνει τις έγκυρες κατευθύνσεις αποστολής της διεργασίας $\vec{p}$. Δεδομένου ότι έχουμε θεωρήσει διαγώνιο πίνακα εξαρτήσεων δεδομένων, κάθε διεργασία χρειάζεται να επικοινωνεί μόνο με γειτονικές, εφαπτόμενες διεργασίες προς τις μοναδιαίες διευθύνσεις επικοινωνίας. Έτσι, αν για μια διεύθυνση επικοινωνίας $\overrightarrow{dir} = (0, \ldots, 1, \ldots, 0)$ και μια διεργασία $\vec{p}$ ισχύει $\overrightarrow{dir} \in \mathbb{S}_{\vec{p}}$, τότε η $\vec{p}$ είναι μη συνοριακή διεργασία και πρέπει να αποστείλει δεδομένα προς τη διεργασία $\vec{p} + \overrightarrow{dir}$. Ομοίως, το σύνολο $\mathbb{R}_{\vec{p}}$ αντιστοιχεί στις έγκυρες διευθύνσεις λήψης της διεργασίας $\vec{p}$, καθώς η $\vec{p}$ πρέπει να λάβει δεδομένα από τη διεργασία $\vec{p} - \overrightarrow{dir}$ αν $\overrightarrow{dir} \in \mathbb{R}_{\vec{p}}$. Τα σύνολα $\mathbb{S}_{\vec{p}}$ και $\mathbb{R}_{\vec{p}}$ καθορίζονται από την επιλεγόμενη τοπολογία διεργασιών για την απεικόνιση του παράλληλου αλγορίθμου.

Το παραπάνω σχήμα παραλληλοποίησης ενός αλγορίθμου φωλιασμένων βρόχων διάστασης $N+1$ και απεικόνισης σε μια παράλληλη αρχιτεκτονική ως προς τις N εξωτερικές διαστάσεις διευκολύνεται σημαντικά αναφορικά με την κατανομή των υπολογισμών αν σε κάθε διεργασία αποδοθεί ένα αναγνωριστικό $\vec{p}$ υπό τη μορφή διανύσματος διάστασης N (σε αντιδιαστολή π.χ. με το βαθμωτό `rank` που αποδίδει το MPI). Η επιλογή αυτή επιτρέπει την εύκολη ταυτοποίηση των υπερκόμβων μιας δεδομένης διεργασίας, καθώς και τη γενικότερη συσχέτιση του χώρου επαναλήψεων με τις διαθέσιμες διεργασίες και κατ' επέκταση την υφιστάμενη αρχιτεκτονική. Η διανυσματική ταυτοποίηση των διεργασιών εισάγει την αναγκαιότητα θεώρησης του πλήθους των διαθέσιμων διεργασιών, έστω P, υπό τη μορφή *N-διάστατης εικονικής τοπολογίας* $(P_1, \ldots, P_N)$, ώστε

$$P = \prod_{i=1}^{N} P_i \tag{4.3}$$

Εφόσον οριστεί μια τέτοια τοπολογία απεικόνισης, σε κάθε διεργασία μπορεί να αποδοθεί ένα μονα-

δικό αναγνωριστικό της μορφής $\vec{p} = (p_1, \ldots, p_N)$ με $0 \leq p_i \leq P_i - 1, 1 \leq i \leq N$. Είναι προφανές όμως ότι η παραγοντοποίηση του πλήθους των διεργασιών P σύμφωνα με την (4.3) μπορεί στη γενική περίπτωση να γίνει με περισσότερους του ενός τρόπους, οδηγώντας έτσι σε εναλλακτικές τοπολογίες απεικόνισης. Η επιλογή μιας κατάλληλης τοπολογίας διεργασιών αποτελεί το αντικείμενο της επόμενης ενότητας.

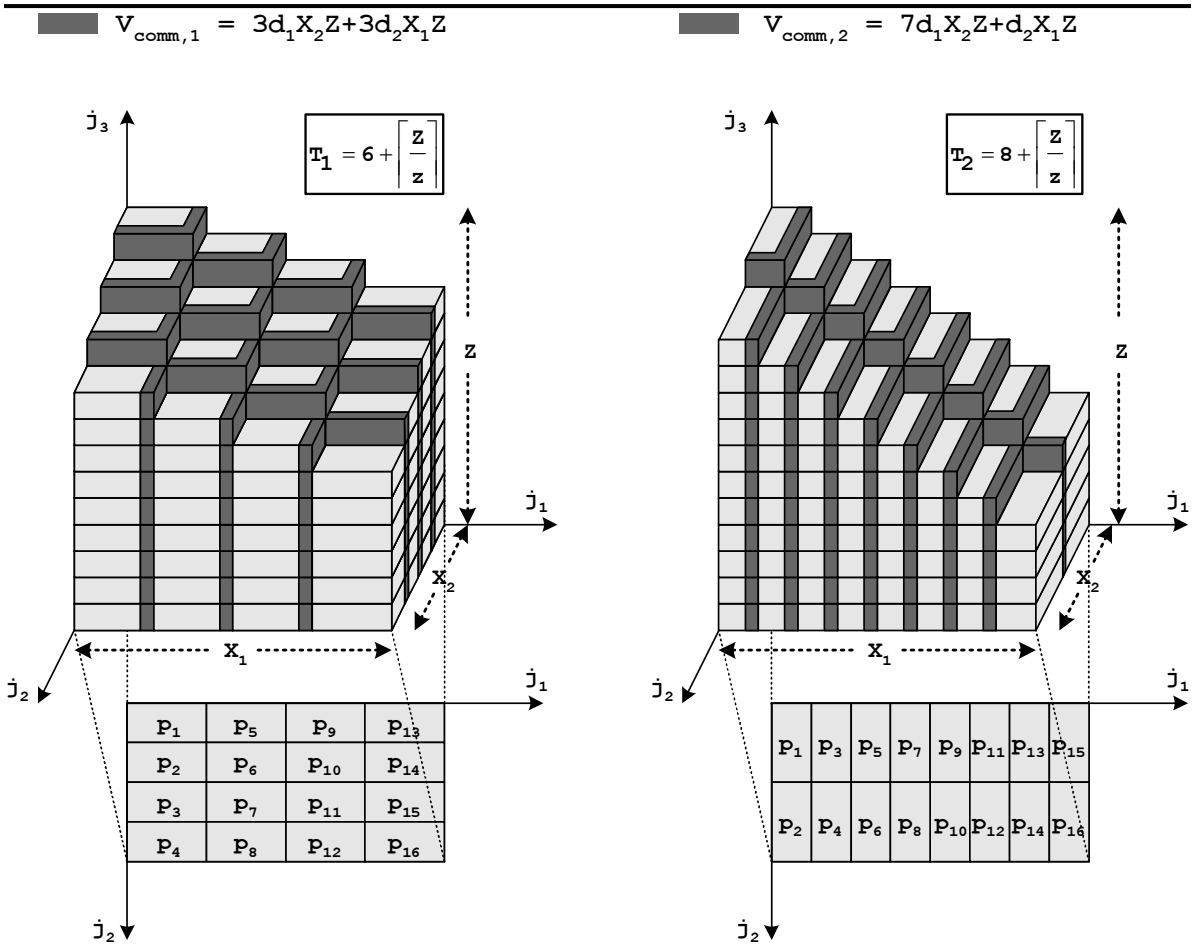
4.2 Απεικόνιση σε Διεργασίες

Σύμφωνα με τη συνήθη προσέγγιση [HS98, HS99, HS02, HCF97, HCF99, HCF03], δοθέντων P διεργασιών προς απεικόνιση ενός παραλληλοποιημένου επαναληπτικού αλγορίθμου διάστασης $N + 1$, επιλέγεται ένα *πλέγμα διεργασιών* $P_1 \times \cdots \times P_N$, που αποτελεί μια αποδεκτή λύση του ακόλουθου προβλήματος βελτιστοποίησης:

$$\left.\begin{array}{rcl} P_i & \rightarrow & \sqrt[N]{P} \\ P & = & \prod_{i=1}^{N} P_i \\ P_i & \in & \mathbb{N} \end{array}\right\}$$

Το πλεονέκτημα της ανωτέρω τοπολογίας είναι πως ελαχιστοποιεί την αρχική καθυστέρηση διάδοσης του παράλληλου αλγορίθμου, καθώς διασφαλίζει ότι η πιο «απομακρυσμένη» διεργασία θα ξεκινήσει την εκτέλεση των υπολογισμών της το ταχύτερο δυνατό. Παρότι η επιλογή αυτή αποτελεί τη συνήθη προσέγγιση για τον καθορισμό της τοπολογίας διεργασιών, παρουσιάζει σημαντικά μειονεκτήματα για τους υπό εξέταση αλγορίθμους. Πιο συγκεκριμένα, κατά τον προσδιορισμό της τοπολογίας των διεργασιών με την παραπάνω μέθοδο δεν λαμβάνονται υπόψη τόσο ο χώρος επαναλήψεων όσο και οι εξαρτήσεις τού υπό παραλληλοποίηση αλγορίθμου. Κατά συνέπεια, ενδέχεται η συγκεκριμένη τοπολογία να επιβάλλει σημαντικά αυξημένες απαιτήσεις όσον αφορά στην επικοινωνία μεταξύ διεργασιών, σε σχέση με πιο κατάλληλες, εναλλακτικές έγκυρες τοπολογίες, που θα μπορούσαν δυνητικά να εφαρμοστούν.

Για παράδειγμα, έστω ότι θέλουμε να παραλληλοποιήσουμε ένα τρισδιάστατο αλγόριθμο χρησιμοποιώντας 16 διεργασίες (βλ. σχήμα 4.3). Η απεικόνιση του αλγορίθμου στο πλέγμα των διεργασιών υλοποιείται με διαμέριση του επιπέδου $j_1 j_2$ του χώρου επαναλήψεων, ενώ κάθε διεργασία αναλαμβάνει την εκτέλεση μιας ακολουθίας υπερκόμβων κατά μήκος της διάστασης j_3 του χώρου. Η συνήθης προσέγγιση θα ήταν να θεωρήσουμε μια τοπολογία απεικόνισης 4×4, καθώς επιτρέπει στην πιο απομακρυσμένη διεργασία να εκκινήσει τους υπολογισμούς όταν η πρώτη εκτελεί τον έβδομο υπερκόμβο της. Έστω ότι ο αλγόριθμος χαρακτηρίζεται από ένα χώρο επαναλήψεων $X_1 \times X_2 \times Z$ με $X_1 = 4X_2$ και εξαρτήσεις δεδομένων $[d, 0, 0]^T$, $[0, d, 0]^T$ και $[0, 0, d]^T$. Στην περίπτωση αυτή, μια εναλλακτική τοπολογία απεικόνισης 8×2 θα ήταν ενδεχομένως προτιμότερη για αρχιτεκτονική κατανεμημένης μνήμης.



Σχήμα 4.3: *Σύγκριση των δεδομένων επικοινωνίας για δύο εναλλακτικές τοπολογίες απεικόνισης τρισδιάστατου αλγορίθμου*

Πράγματι, ο όγκος των συνολικών δεδομένων επικοινωνίας ισούται με

$$\begin{aligned} V_{comm,1} &= \underbrace{9\left(\frac{d_1X_2}{4}+\frac{d_2X_1}{4}\right)Z}_{\text{επικοινωνία κατά } j_1 \text{ και } j_2} + \underbrace{3\frac{d_1X_2}{4}Z}_{\text{επικοινωνία μόνο κατά } j_1} + \underbrace{3\frac{d_2X_1}{4}Z}_{\text{επικοινωνία μόνο κατά } j_2} \\ &= 3d_1X_2Z + 3d_2X_1Z \end{aligned}$$

για την περίπτωση της 4×4 τοπολογίας και με

$$\begin{aligned} V_{comm,2} &= \underbrace{7\left(\frac{d_1 X_2}{2} + \frac{d_2 X_1}{8}\right) Z}_{\text{επικοινωνία κατά } j_1 \text{ και } j_2} + \underbrace{7\frac{d_1 X_2}{2} Z}_{\text{επικοινωνία μόνο κατά } j_1} + \underbrace{1\frac{d_2 X_1}{8} Z}_{\text{επικοινωνία μόνο κατά } j_2} \\ &= 7 d_1 X_2 Z + d_2 X_1 Z \end{aligned}$$

στην περίπτωση της 8×2 τοπολογίας. Για $X_1 = 4X_2$ και $d_1 = d_2 = d$, η επιλογή της 8×2 τοπολογίας μειώνει το συνολικό όγκο των δεδομένων επικοινωνίας κατά

$$\frac{V_{comm,1} - V_{comm,2}}{V_{comm,1}} = \frac{3dX_2Z + 12dX_2Z - 7dX_2Z - 4dX_2Z}{3dX_2Z + 12dX_2Z} = \frac{4}{15} \text{ ή } 27\%$$

Γενικότερα, μπορούμε εύκολα να εξαγάγουμε ότι όταν $d_1 = d_2 = d$ θα είναι $V_{comm,2} < V_{comm,1}$ στις περιπτώσεις των χώρων επαναλήψεων $X_1 \times X_2 \times Z$ στις οποίες ισχύει $X_1 > 2X_2$. Για τέτοιους χώρους, η τοπολογία 8×2 επιτυγχάνει πάντοτε μικρότερο όγκο δεδομένων επικοινωνίας σε σχέση με την 4×4 τοπολογία διεργασιών.

Παρατηρούμε εντούτοις ότι η τοπολογία 8×2 θα επιτρέψει στην πιο απομακρυσμένη διεργασία να εκκινήσει τους υπολογισμούς της μόνο όταν η πρώτη εκτελεί τον ένατο υπερκόμβο της, αυξάνοντας έτσι τα συνολικά βήματα εκτέλεσης του παράλληλου αλγορίθμου κατά δύο σε σχέση με την τοπολογία 4×4. Αναλυτικότερα, η προτεινόμενη παραλληλοποίηση υλοποιεί γραμμική χρονοδρομολόγηση, που περιγράφεται από το διάνυσμα $\Pi = [1, 1, 1]$. Έτσι, στην τοπολογία 4×4 ο πρώτος υπερκόμβος $(p_1 = 0, p_2 = 0, tile_3 = 0)$ δρομολογείται για τη χρονική στιγμή $T_{1,first} = 0$, ενώ ο τελευταίος $(p_1 = 3, p_2 = 3, tile_3 = \lceil \frac{Z}{z} \rceil - 1)$ θα εκτελεστεί τη χρονική στιγμή $T_{1,last} = 6 + \lceil \frac{Z}{z} \rceil - 1$, οδηγώντας σε συνολικό αριθμό βημάτων εκτέλεσης $T_1 = 6 + \lceil \frac{Z}{z} \rceil$. Ομοίως αποδεικνύεται για τη δεύτερη τοπολογία 8×2 ότι το συνολικό πλήθος βημάτων εκτέλεσης ισούται με $T_2 = 8 + \lceil \frac{Z}{z} \rceil$, δηλαδή ο T_2 είναι κατά δύο βήματα μεγαλύτερος του T_1. Δεδομένου όμως ότι κάθε διεργασία θα πρέπει να αναλάβει την εκτέλεση μιας ακολουθίας υπερκόμβων επαρκούς πλήθους $\lceil \frac{Z}{z} \rceil$, ώστε να επιτευχθεί ικανοποιητική επίδοση στο φαινόμενο της σωλήνωσης για το παράλληλο πρόγραμμα, η διαφορά στα βήματα εκτέλεσης αναμένεται να είναι σχετικά μικρή, καθώς τα T_1 και T_2 καθορίζονται πρωτίστως από τον όρο $\lceil \frac{Z}{z} \rceil$ και λιγότερο από την αρχική καθυστέρηση διάδοσης του αλγορίθμου. Έτσι, η προαναφερθείσα επιβάρυνση των επιπλέον βημάτων αναμένεται να είναι αμελητέα συγκριτικά με τα οφέλη στους χρόνους επικοινωνίας, που αποκομίζονται σε *κάθε* βήμα του αλγορίθμου από την υιοθέτηση της τοπολογίας 8×2.

Παρακινούμενοι από την παραπάνω παρατήρηση, αναπτύξαμε μία γενική μέθοδο για τον προσδιορισμό μιας κατάλληλης τοπολογίας απεικόνισης, δοθέντων του χώρου επαναλήψεων και των εξαρτήσεων δεδομένων του επαναληπτικού αλγορίθμου. Η μεθοδολογία βασίζεται στο ακόλουθο λήμμα:

Λήμμα 4.1. *Έστω $X_1 \times \cdots \times X_N \times Z$ ο χώρος επαναλήψεων ενός επαναληπτικού αλγορίθμου διάστασης $N+1$, με εξαρτήσεις δεδομένων τις $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$. Έστω P το πλήθος των διεργασιών που χρησιμοποιούνται για την απεικόνιση του παράλληλου αλγορίθμου. Αν υπάρχουν μη αρνητικοί ακέραιοι $P_i \in \mathbb{N}$ τέτοιοι, ώστε*

$$P = \prod_{i=1}^{N} P_i \tag{4.4}$$

και

$$\frac{d_i P_i}{X_i} = \frac{d_j P_j}{X_j}, 1 \leq i, j \leq N \tag{4.5}$$

τότε η εικονική τοπολογία $P_1 \times \cdots \times P_N$ ελαχιστοποιεί την επικοινωνία μεταξύ διεργασιών κατά την απεικόνιση του παράλληλου αλγορίθμου στις P διεργασίες. Επίσης, η σχέση (4.5) είναι ισοδύναμη με την

$$P_j = \frac{X_j}{d_j} \sqrt[N]{\frac{P \prod_{i=1}^{N} d_i}{\prod_{i=1}^{N} X_i}}, 1 \leq j \leq N \tag{4.6}$$

Απόδειξη. Σύμφωνα με τη σχέση (4.4), ισχύει

$$P_N = \frac{P}{P_1 \times \cdots \times P_{N-1}} \tag{4.7}$$

Υπό τη συγκεκριμένη απεικόνιση, κάθε διεργασία αναλαμβάνει την εκτέλεση $\left\lceil \frac{X_i}{P_i} \right\rceil$ το πλήθος επαναλήψεων κατά μήκος της διεύθυνσης i, όπου $1 \leq i \leq N$. Χάριν απλότητας, υποθέτουμε ότι

$$\left\lceil \frac{X_i}{P_i} \right\rceil \simeq \frac{X_i}{P_i} \tag{4.8}$$

Λόγω των εξαρτήσεων δεδομένων του αλγορίθμου, κάθε διεργασία απαιτείται να αποστείλει προς την i-στή διεύθυνση $d_i Z \prod_{j=1, j \neq i}^{j=N} \frac{X_j}{P_j}$ το πλήθος δεδομένα. Κατά συνέπεια, ο συνολικός όγκος V_{comm} των

δεδομένων επικοινωνίας μιας διεργασίας μπορεί να υπολογιστεί από την ακόλουθη έκφραση:

$$\begin{aligned} V_{comm} &= d_1 Z \prod_{\substack{i=1 \\ i\neq 1}}^{N} \frac{X_i}{P_i} + \cdots + d_N Z \prod_{\substack{i=1 \\ i\neq N}}^{N} \frac{X_i}{P_i} \\ &= \frac{d_1 Z P_1}{X_1} \prod_{i=1}^{N} \frac{X_i}{P_i} + \cdots + \frac{d_N Z P_N}{X_N} \prod_{i=1}^{N} \frac{X_i}{P_i} \\ &= \frac{Z \prod_{i=1}^{N} X_i}{P} \left(\frac{d_1 P_1}{X_1} + \cdots + \frac{d_N P_N}{X_N} \right) \end{aligned} \tag{4.9}$$

Απαλείφοντας τον όρο P_N από τις (4.7), (4.9), προκύπτει

$$\begin{aligned} V_{comm} &= V_{comm}(P_1, \ldots, P_{N-1}) \\ &= \frac{Z \prod_{i=1}^{N} X_i}{P} \sum_{i=1}^{N-1} \frac{d_i P_i}{X_i} + \frac{d_N Z \prod_{i=1}^{N} X_i}{X_N P_1 \ldots P_{N-1}} \end{aligned} \tag{4.10}$$

Παρατηρούμε ότι ο όρος V_{comm} είναι κατ' ουσίαν συνάρτηση των μεταβλητών $P_1, \ldots, P_{N-1}$, δηλαδή $V_{comm} : \mathbb{N}^{N-1} \rightarrow \mathbb{R}$. Έστω $\overline{V}_{comm}$ η *συνεχής πραγματική επέκταση* της V_{comm}, που ορίζεται από τη σχέση (4.10) για $P_j \in \mathbb{R}, 1 \leq j \leq N$ ($\overline{V}_{comm} : \mathbb{R}^{N-1} \rightarrow \mathbb{R}$). Για ένα στάσιμο σημείο $(P_1, \ldots, P_{N-1})$ της $\overline{V}_{comm}$ και για $1 \leq j \leq N-1$, θα ισχύει

$$\frac{\partial \overline{V}_{comm}}{\partial P_j} = 0 \tag{4.11}$$

ή ισοδύναμα

$$\begin{aligned} \frac{d_j Z \prod_{i=1}^{N} X_i}{P X_j} - \frac{d_N Z \prod_{i=1}^{N} X_i}{X_N P_1 \ldots P_j^2 \ldots P_{N-1}} &= 0 \\ \frac{d_j \prod_{i=1}^{N} X_i}{P X_j} &= \frac{d_N P_N \prod_{i=1}^{N} X_i}{X_N P_j P} \\ \frac{d_j P_j}{X_j} &= \frac{d_N P_N}{X_N} \end{aligned} \tag{4.12}$$

Επίσης,

$$\frac{\partial^2 \overline{V}_{comm}}{\partial {P_j}^2} = \frac{2d_N Z \prod_{i=1}^{N} X_i}{X_N P_1 \dots P_j^3 \dots P_{N-1}} > 0 \tag{4.13}$$

Λόγω των (4.11) και (4.13), η $\overline{V}_{comm}$ παρουσιάζει ελάχιστο στο στάσιμο σημείο $(P_1, \dots, P_{N-1})$, και καθώς $P_i \in \mathbb{N}, 1 \leq i \leq N-1$ συνάγεται ότι στην ίδια θέση θα παίρνει ελάχιστη τιμή και η V_{comm}. Συμπεραίνουμε λοιπόν ότι το πλήθος των δεδομένων επικοινωνίας ελαχιστοποιείται κατά την εφαρμογή της τοπολογίας απεικόνισης $P_1 \times \dots \times P_N$, που πληροί τη σχέση (4.12). Τέλος, ισχύει

$$\frac{P \prod_{i=1}^{N} d_i}{\prod_{i=1}^{N} X_i} = \frac{d_1 P_1}{X_1} \dots \frac{d_N P_N}{X_N} \tag{4.14}$$

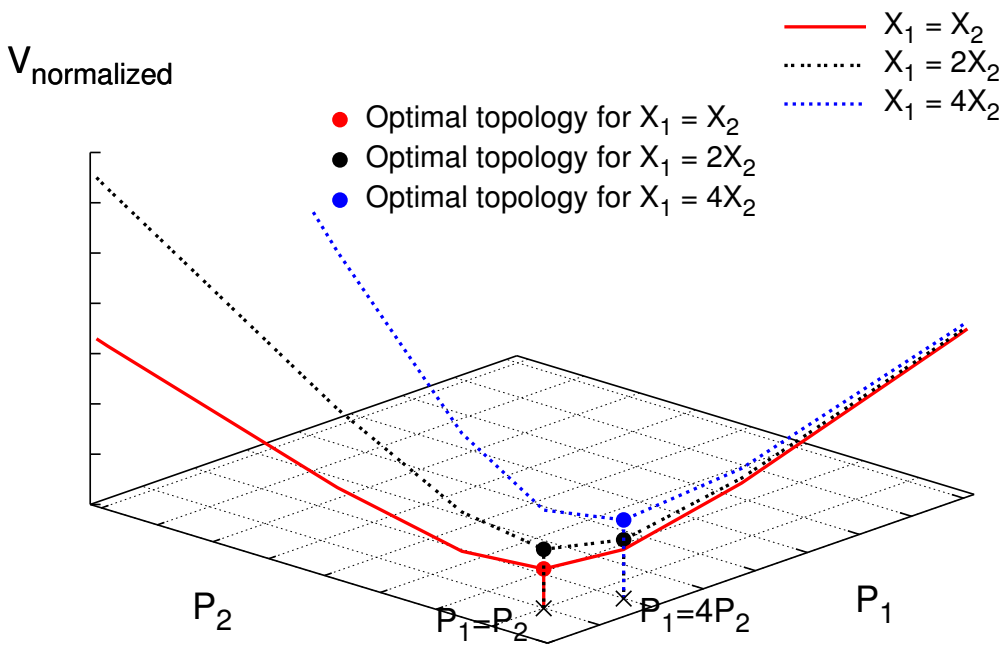
και δεδομένου ότι η V_{comm} ελαχιστοποιείται στη θέση $(P_1, \dots, P_N)$, η σχέση (4.14) μπορεί να γραφεί λόγω της (4.12) ως εξής:

$$\begin{aligned} \frac{P \prod_{i=1}^{N} d_i}{\prod_{i=1}^{N} X_i} &= \left(\frac{d_j P_j}{X_j}\right)^N \\ P_j &= \frac{X_j}{d_j} \sqrt[N]{\frac{P \prod_{i=1}^{N} d_i}{\prod_{i=1}^{N} X_i}} \end{aligned} \tag{4.15}$$

□

Παρατηρούμε ότι η (4.6) δεν ορίζει πάντα μια αποδεκτή τοπολογία απεικόνισης. Πράγματι, κάτι τέτοιο συμβαίνει στην περίπτωση που υπάρχει τουλάχιστον ένας ακέραιος j με $1 \leq j \leq N$ τέτοιος, ώστε για την αντίστοιχη συνιστώσα P_j που ορίζεται από την (4.6) να ισχύει $P_j \notin \mathbb{N}$. Εντούτοις, η μονοτονία της συνάρτησης $\overline{V}_{comm}$ διασφαλίζει ότι μια αποδεκτή λύση για την τοπολογία απεικόνισης στην κοντινή περιοχή του ελαχίστου της συνάρτησης, όπως αυτό καθορίζεται από τη σχέση (4.6), πρακτικά είτε θα ελαχιστοποιεί το κόστος επικοινωνίας, είτε θα παρέχει μια αποδοτική τοπολογία για το δεδομένο αλγόριθμο.

Σχηματικά, αναλόγως του χώρου επαναλήψεων και των εξαρτήσεων δεδομένων του αλγορίθμου, επιλέγεται μια τοπολογία διεργασιών που ελαχιστοποιεί το συνολικό όγκο των δεδομένων επικοινωνίας. Η διαδικασία είναι ισοδύναμη με την εύρεση ενός N-διάστατου ακέραιου σημείου $(P_1, \dots, P_N)$,



Σχήμα 4.4: *Επιλογή βέλτιστης τοπολογίας απεικόνισης (P_1, P_2) για τρισδιάστατο αλγόριθμο βάσει του κανονικοποιημένου κόστους επικοινωνίας*

το οποίο να ελαχιστοποιεί τη συνάρτηση όγκου επικοινωνίας V_{comm}. Το σχήμα 4.4 αναπαριστά τον προσδιορισμό μιας κατάλληλης δισδιάστατης τοπολογίας απεικόνισης για τρισδιάστατο αλγόριθμο με χώρο επαναλήψεων $X_1 \times X_2 \times Z$ και εξαρτήσεις δεδομένων $[d, 0]^T$, $[0, d]^T$. Για παράδειγμα, υποθέτοντας 64K διεργασίες σε ένα υπολογιστικό σύστημα μεγάλης κλίμακας, ένας τρισδιάστατος χώρος με $X_1 = 4X_2$ θα επωφελούταν περισσότερο από μία τοπολογία απεικόνισης 512×128. Αντίθετα, για χώρο επαναλήψεων με $X_1 = X_2$ προτείνεται η επιλογή μιας τοπολογίας διεργασιών 256×256, ενώ για την περίπτωση $X_1 = 2X_2$ οι δύο αυτές τοπολογίες είναι ισοδύναμες, σε ό,τι αφορά τον όγκο δεδομένων επικοινωνίας. Στην τελευταία περίπτωση, θα επιλέγαμε πιθανότατα την τοπολογία που ικανοποιεί τη σχέση $P_1 = P_2$ (εδώ 256×256), καθώς αυτή επιπλέον ελαχιστοποιεί την αρχική καθυστέρηση διάδοσης του παράλληλου προγράμματος, όπως εξηγήθηκε και παραπάνω.

4.3 Προσδιορισμός Τοπολογίας Διεργασιών Ελάχιστης Επικοινωνίας

Βασιζόμενοι στο λήμμα 4.1 υλοποιήσαμε μια ευριστική μέθοδο που αναζητεί εξαντλητικά μια αποδεκτή τοπολογία στην κοντινή περιοχή του ελαχίστου της $\overline{V}_{comm}$. Η μέθοδος υλοποιήθηκε σε γλώσσα προγραμματισμού C και ενσωματώθηκε ως αυτόματη βελτιστοποίηση σε όλες τις πειραματικές περιπτώσεις, που θελήσαμε να επαληθεύσουμε την επίδοση της τοπολογίας απεικόνισης ελάχιστης επικοινωνίας.

Η μέθοδος παρουσιάζεται στον αλγόριθμο 4.5. Ο αλγόριθμος δέχεται σαν είσοδο το συνολικό πλήθος των διαθέσιμων διεργασιών P, τον υπό εξέταση χώρο επαναλήψεων $\prod_{i=1}^{N} X_i Z$ και τις εξαρτήσεις δεδομένων $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$, ενώ παρέχει ως έξοδο μια προτεινόμενη τοπολογία διεργασιών $\prod_{i=1}^{N} P_i$ για την απεικόνιση του παράλληλου αλγορίθμου. Στη γραμμή 1 ο αλγόριθμος υπολογίζει με χρήση της (4.6) μια ακέραιη προσέγγιση $\prod_{i=1}^{N} P_i^{(0)}$ της θέσης ελαχίστου της $\overline{V}_{comm}$. Στις γραμμές 2-4 εξετάζεται κατά πόσο η $\prod_{i=1}^{N} P_i^{(0)}$ αποτελεί έγκυρη λύση του προβλήματος βελτιστοποίησης, παρέχει δηλαδή ως γινόμενο το επιθυμητό πλήθος διεργασιών P. Αν κάτι τέτοιο συμβαίνει, αντιγράφονται οι συνιστώσες $P_i^{(0)}$ στις αντίστοιχες P_i και πραγματοποιείται έξοδος από τον αλγόριθμο, καθώς στην περίπτωση αυτή η (4.6) παρείχε απευθείας μία έγκυρη τοπολογία απεικόνισης. Σε αντίθετη περίπτωση, χρησιμοποιώντας ως αφετηριακό διάνυσμα το $(P_1^{(0)}, \ldots, P_N^{(0)})$ πραγματοποιείται εξαντλητική αναζήτηση στην κοντινή περιοχή για εύρεση μιας αποδεκτής τοπολογίας απεικόνισης στις γραμμές 6-24.

Συγκεκριμένα, στο τμήμα του αλγορίθμου μεταξύ των γραμμών 7-17 σχηματίζονται από όλα τα διανύσματα $(P_1^{(n)}, \ldots, P_N^{(n)})$ του προηγούμενου βήματος όλα τα διανύσματα $(P_1^{(n+1)}, \ldots, P_N^{(n+1)})$, που διαφέρουν από τα πρώτα ακριβώς σε μία συνιστώσα κατά +1 ή -1 (βλ. γραμμή 9). Από αυτά απορρίπτονται οι συνδυασμοί εκείνοι, στους οποίους μία τουλάχιστον συνιστώσα έγκειται εκτός του διαστήματος $[1 \ldots P]$. Επιπλέον, μπορούν να απορριφθούν τα διανύσματα που αφενός δεν αντιστοιχούν σε περιπτώσεις ταλάντωσης της μεθόδου εκατέρωθεν της θέσης ελαχίστου και αφετέρου απομακρύνονται από αυτή σε σχέση με το διάνυσμα απ' το οποίο προήλθαν (γραμμές 13-15). Η συγκεκριμένη απαλοιφή των περιπτώσεων αυτών βασίζεται στη μονοτονία που έχει η συνάρτηση κόστους επικοινωνίας $\overline{V}_{comm}$ και μειώνει σημαντικά την πολυπλοκότητα της μεθόδου, καθώς από τους $2N$ νέους συνδυασμούς που παρέχει κάθε διάνυσμα απορρίπτονται οι μισοί. Τέλος, στις γραμμές 18-22 εξετάζεται αν μεταξύ των διανυσμάτων $(P_1^{(n+1)}, \ldots, P_N^{(n+1)})$ υπάρχουν έγκυρες λύσεις του προβλήματος της τοπολογίας. Εφόσον αυτό συμβαίνει, επιλέγεται εκείνη με το ελάχιστο κόστος επικοινωνίας και αντιγράφεται στην $\prod_{i=1}^{N} P_i$, ενώ η σημαία $valid$ τίθεται στην τιμή 1 για να πραγματοποιηθεί έξοδος από το βρόχο αναζήτησης. Σε αντίθετη περίπτωση, η αναζήτηση συνεχίζεται με την εξέταση όλων των δυνατών συνδυασμών του επόμενου επιπέδου.

Λαμβάνοντας υπόψη ότι η διάσταση N του αλγορίθμου είναι μικρή στις περιπτώσεις πραγματικών

Αλγόριθμος 4.5: Αλγόριθμος προσδιορισμού τοπολογίας διεργασιών ελάχιστης επικοινωνίας

Δεδομένα: Πλήθος διεργασιών P, χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, εξαρτήσεις δεδομένων

$$[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$$

Ζητούμενα: Τοπολογία διεργασιών $\prod_{i=1}^{N} P_i$

$P_j^{(0)} = \left\lfloor \frac{X_j}{d_j} \sqrt[N]{\frac{P \prod_{i=1}^{N} d_i}{\prod_{i=1}^{N} X_i}} \right\rfloor, 1 \leq j \leq N;$

if $\prod_{i=1}^{N} P_i^{(0)} = P$ **then**
  $P_i = P_i^{(0)}, 1 \leq i \leq N$; `return`;
endif
$valid = n = 0$; $V_{comm\_min} = MAX\_INF$;
while $valid = 0$ **do**
  foreach $P^{(n)}$ **do**
    for $j \leftarrow 1$ **to** N **do**
      $P^{(n+1)} = (P_1^{(n)}, \ldots, P_j^{(n)} \pm 1, \ldots, P_N^{(n)})$;
      if $\left(\prod_{i=1}^{N} P_i^{(n+1)} > P\right) \vee \left(\prod_{i=1}^{N} P_i^{(n+1)} < 1\right)$ **then**
        `discard(`$P^{(n+1)}$`); continue;`
      endif
      if $\left(\left(\left(\prod_{i=1}^{N} P_i^{(n+1)} - P\right)\left(\prod_{i=1}^{N} P_i^{(n)} - P\right) > 0\right) \wedge \left(|\prod_{i=1}^{N} P_i^{(n+1)} - P| > |\prod_{i=1}^{N} P_i^{(n)} - P|\right)\right)$ **then**
        `discard(`$P^{(n+1)}$`); continue;`
      endif
    endfor
  endforeach
  foreach $P^{(n+1)}$ **do**
    if $\left(\left(\prod_{i=1}^{N} P_i^{(n+1)} = P\right) \wedge \left(\texttt{Vcomm(}P^{(n+1)}\texttt{)} < V_{comm\_min}\right)\right)$ **then**
      $P_i = P_i^{(n+1)}, 1 \leq i \leq N$; $V_{comm\_min} =$ `Vcomm(`$P^{(n+1)}$`)`; $valid = 1$;
    endif
  endforeach
  $n = n + 1$;
endwhile

εφαρμογών, η πολυπλοκότητα της ευριστικής μεθόδου είναι πρακτικά αποδεκτή, παρότι θεωρητικά χαρακτηρίζεται ως εκθετική. Για να επαληθεύσουμε τον ισχυρισμό αυτό μετρήσαμε το χρόνο εκτέλεσης για τον καθορισμό της τοπολογίας ελάχιστης επικοινωνίας σε έναν Pentium III στα 800 MHz για πλήθος διεργασιών $100 \leq P \leq 1k$, όλους τους δυνατούς τετραδιάστατους χώρους επαναλήψεων της μορφής $(100 \ldots 10k) \times (100 \ldots 10k) \times (100 \ldots 10k) \times Z$ και εξαρτήσεις δεδομένων στο εύρος $[(1 \ldots 3, 0, 0, d), (0, 1 \ldots 3, 0, d'), (0, 0, 1 \ldots 3, d'')]$. Οι τιμές αυτές αντιστοιχούν σε προβλήματα και αρχιτεκτονικές μεγάλης κλίμακας, συνεπώς η πιστοποίηση της επίδοσης της μεθόδου με χρήση των συγκεκριμένων μετρήσεων παρέχει μια πραγματική ένδειξη της επιβάρυνσης που επιφέρει η μέθοδος σε ρεαλιστικές συνθήκες. Ο μέσος χρόνος υπολογισμού μιας έγκυρης τοπολογίας βάσει της προτεινόμενης τεχνικής ήταν 21 msec, ενώ σε καμία περίπτωση ο χρόνος εκτέλεσης δεν υπερέβη τα 0.9 sec.

4.4 Ισοδύναμα: Προσδιορισμός Ορθογώνιου Μετασχηματισμού Υπερκόμβων Ελάχιστης Επικοινωνίας

Στην παρούσα ενότητα θα επιχειρήσουμε μια διαφορετική προσέγγιση στο πρόβλημα του προσδιορισμού τοπολογίας διεργασιών ελάχιστης επικοινωνίας. Συγκεκριμένα, θα αναδείξουμε την ισοδυναμία του προβλήματος αυτού με το πρόβλημα του καθορισμού κατάλληλου ορθογώνιου μετασχηματισμού υπερκόμβων για τη διαμέριση του αρχικού αλγορίθμου. Όπως αναφέρθηκε, ο μετασχηματισμός υπερκόμβων τυγχάνει εκτενούς μελέτης στη διεθνή βιβλιογραφία. Αντίθετα όμως με τη δική μας προσέγγιση, σε κάποιες περιπτώσεις χρησιμοποιούνται μη ορθογώνιοι υπερκόμβοι, που δυσχεραίνουν σημαντικά την πρακτική εφαρμογή των τεχνικών στην SPMD παραλληλοποίηση αλγορίθμων. Ακόμα πιο σημαντική διαφορά συνιστά το γεγονός πως στη σχετική βιβλιογραφία το πλήθος των διεργασιών που διατίθενται για την παράλληλη εκτέλεση σπανίως θεωρείται εξ αρχής ως περιορισμός, καταλήγοντας έτσι σε διαφωνία μεταξύ του εκάστοτε προτεινόμενου μετασχηματισμού υπερκόμβων και της υφιστάμενης υπολογιστικής υποδομής.

Στη δική μας προσέγγιση, για τον προσδιορισμό αποδοτικού μετασχηματισμού υπερκόμβων δεχόμαστε τα ακόλουθα δεδομένα:

- Το *πλήθος των διαθέσιμων διεργασιών* είναι γνωστό και πεπερασμένο, έστω ίσο με P. Η παραδοχή αυτή αντανακλά τη ρεαλιστική ανάγκη να προσαρμόσουμε την παράλληλη υλοποίησή μας στην υφιστάμενη επεξεργαστική υποδομή, καθώς η τελευταία θέτει άλλωστε ένα άνω φράγμα στη μέγιστη επιτάχυνση του χρόνου εκτέλεσης, που μπορεί πρακτικά να επιτευχθεί.

- Θεωρούμε ένα δεδομένο *κόκκο παραλληλισμού*, έστω g, που ισοδυναμεί πρακτικά με το μέγεθος του υπερκόμβου, δηλαδή το πλήθος των επαναλήψεων που θα περικλείει ο κάθε υπερκόμβος.

Ο υπολογισμός του βέλτιστου κόκκου παραλληλισμού είναι ανοιχτό ερευνητικό ζήτημα και ξεφεύγει από το αντικείμενο της παρούσας διατριβής, μπορεί δε να καθοριστεί όπως θα δούμε με την επιλογή του ύψους του υπερκόμβου z, που επίσης θεωρήσαμε ως ελεύθερη παράμετρο στην ενότητα 4.2. Πάντως είναι σημαντικό να θεωρηθεί ο κόκκος παραλληλισμού ως δεδομένος περιορισμός, γιατί αν μια τέτοια πληροφορία διατίθεται ή μπορεί να εξαχθεί με κάποιο τρόπο για ένα συγκεκριμένο συνδυασμό αλγορίθμου-αρχιτεκτονικής, θα πρέπει να αξιοποιηθεί σε οποιαδήποτε τεχνική μετασχηματισμού υπερκόμβων. Επιπλέον, όπως θα δούμε, ακόμα κι αν διατηρήσουμε το g ως ελεύθερο άγνωστο, μπορούμε παρόλα αυτά να καταλήξουμε στον αναλυτικό προσδιορισμό ενός μετασχηματισμού υπερκόμβων ελάχιστης επικοινωνίας, λόγω του ότι η επικοινωνία θα αφορά τις N διαστάσεις ενός αλγορίθμου διάστασης $N+1$, αφήνοντας έτσι ένα βαθμό ελευθερίας στον προσδιορισμό του ζητούμενου πίνακα μετασχηματισμού H.

- Θεωρούμε τέλος ότι αναφερόμαστε σε ένα συγκεκριμένο αλγόριθμο φωλιασμένων βρόχων διάστασης $N+1$, δηλαδή κατά τα γνωστά θα θεωρήσουμε ένα δεδομένο *χώρο επαναλήψεων* $X_1 \times \cdots \times X_N \times Z$, καθώς και συγκεκριμένες *εξαρτήσεις δεδομένων* $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$.

Ο προτεινόμενος μετασχηματισμός υπερκόμβων για την ελαχιστοποίηση της επικοινωνίας μεταξύ των διεργασιών μπορεί να υπολογιστεί με χρήση του ακόλουθου λήμματος:

Λήμμα 4.2. *Έστω $X_1 \times \cdots \times X_N \times Z$ ο χώρος επαναλήψεων ενός επαναληπτικού αλγορίθμου διάστασης $N+1$, με εξαρτήσεις δεδομένων τις $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$. Έστω P το πλήθος των διεργασιών που χρησιμοποιούνται για την απεικόνιση του παράλληλου αλγορίθμου. Αν υπάρχουν μη αρνητικοί ακέραιοι $P_i \in \mathbb{N}$ που πληρούν τις* (4.4) *και* (4.5)*, τότε ο μετασχηματισμός υπερκόμβων που ορίζεται από τον πίνακα*

$$H = \begin{bmatrix} \frac{P_1}{X_1} & 0 & \ldots & 0 & 0 \\ & & \ldots & & \\ 0 & 0 & \ldots & \frac{P_N}{X_N} & 0 \\ 0 & 0 & \ldots & 0 & \frac{\prod_{i=1}^{N} X_i}{g \prod_{i=1}^{N} P_i} \end{bmatrix} \tag{4.16}$$

ελαχιστοποιεί την επικοινωνία μεταξύ διεργασιών κατά την παράλληλη εκτέλεση με χρήση του αλγορίθμου 4.4 με κόκκο παραλληλισμού g.

Απόδειξη. Είναι σαφές ότι ο προτεινόμενος μετασχηματισμός είναι ορθογώνιος, καθώς ο πίνακας H είναι διαγώνιος. Αρκεί να δείξουμε ότι ο μετασχηματισμός αυτός ισοδυναμεί με επιλογή τοπολογίας $P_1 \times \cdots \times P_N$ και χαρακτηρίζεται από κόκκο παραλληλισμού g. Ο προτεινόμενος υπερκόμβος έχει ως

πλευρές τις στήλες του αντιστρόφου του H, δηλαδή του πίνακα

$$H^{-1} = \begin{bmatrix} \frac{X_1}{P_1} & 0 & \dots & 0 & 0 \\ & & \dots & & \\ 0 & 0 & \dots & \frac{X_N}{P_N} & 0 \\ & & & & \\ 0 & 0 & \dots & 0 & \frac{g \prod_{i=1}^{N} P_i}{\prod_{i=1}^{N} X_i} \end{bmatrix}$$

Πρόκειται επομένως για ορθογώνιους υπερκόμβους μεγέθους

$$|H^{-1}| = \frac{X_1}{P_1} \cdot \dots \cdot \frac{X_N}{P_N} \cdot \frac{g \prod_{i=1}^{N} P_i}{\prod_{i=1}^{N} X_i} = g$$

και πλευράς μήκους $\frac{X_j}{P_j}$ κατά μήκος της διάστασης X_j του αλγορίθμου, διαιρώντας έτσι την τελευταία σε $X_j / \frac{X_j}{P_j} = P_j$ διεργασίες. Τέλος, παρατηρούμε ότι η εσωτερική διάσταση σειριακής εκτέλεσης του υπερκόμβου (ύψος z) σχετίζεται με τον κόκκο παραλληλισμού g μέσω της σχέσης:

$$z = \frac{g \prod_{i=1}^{N} P_i}{\prod_{i=1}^{N} X_i}$$

□

4.5 Χρονοδρομολόγηση με Επικάλυψη Επικοινωνίας - Υπολογισμών

Επανερχόμενοι στον αλγόριθμο 4.4, θα πρέπει να σχολιάσουμε ένα ακόμη πολύ επιθυμητό χαρακτηριστικό της συγκεκριμένης προσέγγισης παραλληλοποίησης: τη δυνατότητα επικάλυψης υπολογισμού και επικοινωνίας. Πιο συγκεκριμένα, σε κάθε υπερκόμβο που απαριθμείται από τη μεταβλητή ελέγχου $tile_{N+1}$, μία διεργασία $\vec{p} = (p_1, \dots, p_N)$ *ταυτόχρονα* εκτελεί τους υπολογισμούς που σχετίζονται με τον *τρέχοντα* υπερκόμβο $(p_1, \dots, p_N, tile_{N+1})$, λαμβάνει δεδομένα που απαιτούνται για τον υπολογισμό του *επόμενου* υπερκόμβου $(p_1, \dots, p_N, tile_{N+1} + 1)$, καθώς και αποστέλλει δεδομένα που παρήχθησαν κατά τους υπολογισμούς του *προηγούμενου* υπερκόμβου $(p_1, \dots, p_N, tile_{N+1} - 1)$.

Η εφαρμοζόμενη χρονοδρομολόγηση συγκεντρώνει τις εξής σημαντικές ιδιότητες:

- δεν παραβιάζει τις εξαρτήσεις δεδομένων του υπό παραλληλοποίηση αλγορίθμου

- επιτρέπει την επικάλυψη υπολογισμού και επικοινωνίας
- υλοποιεί παραλληλισμό χονδρού κόκκου, κατάλληλο για αρχιτεκτονικές κατανεμημένης μνήμης
- επιτυγχάνει ένα ικανοποιητικό συμβιβασμό μεταξύ του βαθμού παραλληλισμού και της αναγκαιότητας επικοινωνίας και συγχρονισμού

Η εν λόγω χρονοδρομολόγηση παρέχει απλά τη *δυνατότητα* για επικάλυψη επικοινωνίας και υπολογισμού. Η αξιοποίηση της δυνατότητας αυτής προϋποθέτει όμως προηγμένα χαρακτηριστικά επικοινωνίας του δικτύου διασύνδεσης, όπως υποστήριξη DMA επικοινωνίας με άμεση προσπέλαση της μνήμης, καθώς και *μηδενική αντιγραφή* (*zero-copy*) με σταθερή συσχέτιση φυσικών-εικονικών διευθύνσεων μνήμης. Για το τελευταίο εφαρμόζονται τεχνικές όπως η *απαγκίστρωση εικονικών σε φυσικές διευθύνσεις* (*pinned-down memory addresses*) ή *κλείδωμα μνήμης* (*memory locking*). Δυστυχώς, η πειραματική αξιολόγηση σε ένα δίκτυο διασύνδεσης που υλοποιεί τη στοίβα πρωτοκόλλων TCP/IP, όπως το Ethernet, σε συνδυασμό με την αντίστοιχη ADI-2 συσκευή της υλοποίησης του MPICH (κανάλι ch_p4), απαγορεύει την αξιοποίηση τέτοιων προηγμένων χαρακτηριστικών. Θα πρέπει πάντως να σημειωθεί ότι οι ίδιοι περιορισμοί ισχύουν και για τα υβριδικά μοντέλα προγραμματισμού, συνεπώς δεν αναμένεται να επηρεάσουν τη σύγκριση μεταξύ των διαφορετικών τεχνικών παράλληλου προγραμματισμού. Στον αντίποδα όμως, το συγκεκριμένο δεδομένο περιπλέκει τη θεωρητική μας ανάλυση, καθώς θα θεωρήσουμε εν γένει διακριτές, μη επικαλυπτόμενες φάσεις υπολογισμού και επικοινωνίας, παραδοχή που εν μέρει υποτιμά την αποδοτικότητα των λειτουργιών ανταλλαγής μηνυμάτων.

Αναλυτικότερα, ο συνολικός χρόνος εκτέλεσης T του παράλληλου προγράμματος υπό το μετασχηματισμό υπερκόμβων μπορεί θεωρητικά να προσεγγιστεί ως

$$T = N \times T_{tile} \tag{4.17}$$

όπου N ο συνολικός αριθμός βημάτων παράλληλης εκτέλεσης και T_{tile} ο μέσος χρόνος εκτέλεσης που απαιτείται για την περάτωση ενός βήματος και ταυτίζεται με το μέσο χρόνο εκτέλεσης υπερκόμβου. Ο συνολικός αριθμός παράλληλων βημάτων N μπορεί να προκύψει από τη σχέση

$$N = t_{last} - t_{first} + 1 \tag{4.18}$$

όπου t_{first} η χρονική στιγμή στην οποία δρομολογείται ο πρώτος υπερκόμβος που θα εκτελεστεί, ενώ t_{last} η χρονική στιγμή δρομολόγησης του τελευταίου υπερκόμβου. Βάσει της θεωρίας γραμμικής χρονοδρομολόγησης, η χρονική στιγμή t στην οποία δρομολογείται ένας υπερκόμβος $\vec{j'} = \lfloor H\vec{j} \rfloor$ παρέχεται

από τη σχέση

$$t = \left\lfloor \frac{\Pi \vec{j'} + t_0}{disp\Pi} \right\rfloor \tag{4.19}$$

όπου t_0 η σταθερά έναρξης και $disp\Pi$ η σταθερά μετατόπισης. Αναφορικά με τη θεωρητική ανάλυση της γραμμικής χρονοδρομολόγησης, ο ενδιαφερόμενος αναγνώστης μπορεί να βρει περισσότερα στα [SF91, AKPT99, Hod99, GSK01, KSG03, Σωτ04]. Εν προκειμένω, για την περίπτωσή μας αποδεικνύεται ότι $t_0 = 0$ και $disp\Pi = 1$. Αυτό που παρουσιάζει ιδιαίτερο ενδιαφέρον είναι αυτό καθ' αυτό το γραμμικό διάνυσμα δρομολόγησης Π (linear scheduling vector) που διαφοροποιείται από το κατά πόσο έχουμε ή όχι επικάλυψη υπολογισμών και επικοινωνίας. Έτσι, στην περίπτωση της μη επικαλυπτόμενης χρονοδρομολόγησης, προκύπτει ότι το βέλτιστο διάνυσμα δρομολόγησης ισούται με

$$\Pi_{non-overlap} = [1, 1, \ldots, 1, 1] \tag{4.20}$$

ενώ για την περίπτωση της επικαλυπτόμενης χρονοδρομολόγησης έχουμε

$$\Pi_{overlap} = [2, 2, \ldots, 2, 1] \tag{4.21}$$

Για το μέσο χρόνο εκτέλεσης υπερκόμβου, στην περίπτωση της μη επικαλυπτόμενης χρονοδρομολόγησης θεωρούνται διακριτές φάσεις υπολογισμού και επικοινωνίας. Συνεπώς, ο αντίστοιχος χρόνος εκτέλεσης υπερκόμβου $T_{non-overlap}$ προκύπτει αθροίζοντας το χρόνο υπολογισμού t_{comp} με το χρόνο επικοινωνίας $t_{startup} + t_{comm}$ (αρχικοποίηση+μετάδοση δεδομένων):

$$T_{non-overlap} = t_{comp} + t_{startup} + t_{comm} \tag{4.22}$$

Αντίθετα, στην περίπτωση πραγματικής επικάλυψης υπολογισμού και επικοινωνίας, ο μέσος χρόνος εκτέλεσης του υπερκόμβου ισούται με τη σταθερή συνιστώσα αρχικοποίησης της επικοινωνίας $t_{startup}$ αθροιζόμενη με την ποσότητα $max(t_{comp}, t_{comm})$ λόγω επικάλυψης:

$$T_{overlap} = t_{startup} + max(t_{comp}, t_{comm}) \tag{4.23}$$

Παρότι *σημασιολογικά* το σχήμα δρομολόγησης που υλοποιείται μέσω του αλγορίθμου 4.4 επιτρέπει την επικάλυψη υπολογισμού και επικοινωνίας, για τη θεωρητική μοντελοποίηση της συμπεριφοράς του συστήματος θα προτιμήσουμε τη χρήση των (4.20) και (4.22) έναντι των (4.21) και (4.23). Αν και η DMA λειτουργία μας επιτρέπει μέχρι κάποιου βαθμού να επιτύχουμε αρκετά υψηλούς ρυθμούς παροχής δικτύου, ο συνδυασμός του καναλιού ch_p4 του MPICH με τη στοίβα πρωτοκόλλων TCP/IP του Ethernet οδηγούν σε ένα σημαντικό αριθμό αντιγραφών των δεδομένων, απασχολώντας έτσι τον

επεξεργαστή όχι μόνο κατά τη σταθερή συνιστώσα αρχικοποίησης $t_{startup}$, αλλά τουλάχιστον και κατά μια συνιστώσα που είναι ανάλογη του μεγέθους του μηνύματος και είναι δύσκολο να μοντελοποιηθεί θεωρητικά. Για το σκοπό αυτό επιλέξαμε να προσεγγίσουμε σε θεωρητικό επίπεδο την επίδοση των μοντέλων παραλληλοποίησης με χρήση της μη επικαλυπτόμενης προσέγγισης, θεωρώντας παράλληλα, όπου χρειάστηκε, υψηλό ρυθμό παροχής δικτύου για να προσομοιάσουμε την περιορισμένη επικάλυψη που επιτυγχάνεται στη συγκεκριμένη αρχιτεκτονική υποδομή.

ΚΕΦΑΛΑΙΟ 5

Υβριδικό Παράλληλο Προγραμματιστικό Μοντέλο

Το υβριδικό προγραμματιστικό μοντέλο διαισθητικά ανταποκρίνεται αποδοτικότερα σε μια αρχιτεκτονική κατανεμημένης μοιραζόμενης μνήμης. Προς την κατεύθυνση αξιοποίησης μιας τέτοιας αρχιτεκτονικής υποδομής (π.χ. μιας συστοιχίας κόμβων συμμετρικής πολυεπεξεργασίας) η υβριδική προσέγγιση παραλληλοποίησης κάνει διάκριση μεταξύ της επικοινωνίας που εμπλέκει επεξεργαστές του ίδιου κόμβου μοιραζόμενης μνήμης (*intra-node communication*) και της επικοινωνίας μεταξύ επεξεργαστών διαφορετικών κόμβων (*inter-node communication*). Έτσι, στο εσωτερικό του ίδιου κόμβου αξιοποιείται η δυνατότητα μοιραζόμενης μνήμης, σε συνδυασμό με κατάλληλο συγχρονισμό. Αντίθετα, μεταξύ διαφορετικών κόμβων διατηρείται η επικοινωνία μέσω ανταλλαγής μηνυμάτων, στη μορφή που αυτή παρουσιάστηκε στο κεφάλαιο 4. Στο κεφάλαιο αυτό θα περιγραφούν υβριδικά μοντέλα παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων, που αξιοποιούν τη χρονοδρομολόγηση υπερεπιπέδων για την ελαχιστοποίηση του συγχρονισμού και την επίτευξη υψηλού βαθμού παραλληλίας. Θα αναφερθούμε σε υβριδικά μοντέλα παραλληλισμού τόσο λεπτού όσο και χονδρού κόκκου και θα αναπτύξουμε βασικές τεχνικές εξισορρόπησης του φορτίου μεταξύ των νημάτων, που αποδεικνύονται ιδιαίτερα χρήσιμες σε περιπτώσεις ελλιπούς πολυνηματικής υποστήριξης από τη βιβλιοθήκη ανταλλαγής μηνυμάτων.

5.1 Υβριδικό Μοντέλο

Σε όλες τις παραλλαγές του, το υβριδικό μοντέλο βασίζεται στη θεμελιώδη διάκριση μεταξύ των *νημάτων* (*threads*) και των *διεργασιών* (*processes*). Κάθε διεργασία του υβριδικού προγράμματος αποτελείται

από περισσότερα του ενός νήματα εκτέλεσης. Τα νήματα μοιράζονται τον κοινό χώρο διευθύνσεων της διεργασίας, αλλά παράλληλα κάθε ένα διατηρεί το δικό του μετρητή προγράμματος, σύνολο καταχωρητών και στοίβα δεδομένων. Το βασικό πλεονέκτημα των νημάτων σε σχέση με τις διεργασίες είναι η σημαντικά μικρότερη επιβάρυνση που συνεπάγονται κατά τη *μεταγωγή περιβάλλοντος* (*context switching*) λόγω του ότι επιφέρουν ελάχιστη ή καθόλου αναγκαιότητα διαχείρισης μνήμης. Το υβριδικό παράλληλο προγραμματιστικό μοντέλο στοχεύει στην αξιοποίηση της δυνατότητας επικοινωνίας μέσω του κοινού χώρου μνήμης που βλέπουν τα νήματα, αποσκοπώντας ταυτόχρονα στην ελαχιστοποίηση του μεταξύ τους συγχρονισμού, που απαιτείται για τη διασφάλιση ελεγχόμενης πρόσβασης στα μοιραζόμενα δεδομένα. Οι δύο αυτές παράμετροι συνιστούν και τις βασικές προτεραιότητες του σχήματος χρονοδρομολόγησης υπερεπιπέδων, που αποτελεί το αντικείμενο της επόμενης ενότητας.

Η πολυνηματική επεξεργασία χρησιμοποιείται κατά κόρον στον παράλληλο προγραμματισμό σε αρχιτεκτονικές μοιραζόμενης μνήμης. Το μοντέλο προγραμματισμού μοιραζόμενης μνήμης παρουσιάζει δύο πολύ βασικά πλεονεκτήματα: κατά πρώτον, είναι σχετικά απλό, δεδομένου ότι χρησιμοποιεί τη συνήθη σύνταξη της γλώσσας προγραμματισμού για την προσπέλαση του μοιραζόμενου χώρου μνήμης. Συγκριτικά με το μοντέλο ανταλλαγής μηνυμάτων, το πολυνηματικό μοντέλο δεν επιφέρει την προγραμματιστική πολυπλοκότητα που σχετίζεται με την κατανομή των δεδομένων και των υπολογισμών κατά την απεικόνιση του αλγορίθμου στην υφιστάμενη αρχιτεκτονική. Κατά δεύτερον, η πολυνηματική επεξεργασία παρέχει τη δυνατότητα βελτίωσης της απόδοσης σε υποδομή μοιραζόμενης μνήμης, τόσο μέσω ρητού διαχωρισμού της *επικοινωνίας* και του *συγχρονισμού* όσο και μέσω ελαχιστοποίησης της *ενδιάμεσης αποθήκευσης* και της *αντιγραφής δεδομένων*. Στο μοντέλο ανταλλαγής μηνυμάτων, η χρήση λειτουργιών επικοινωνίας για την ανταλλαγή δεδομένων μεταξύ διεργασιών συχνά επιβάλλει έμμεσα και μια συγκεκριμένη χρονική ακολουθία εκτέλεσης, τουλάχιστον σε ό,τι αφορά τις εμπλεκόμενες διεργασίες [GLT99]. Για παράδειγμα, για την ανταλλαγή ενός μηνύματος θα πρέπει η διεργασία-αποστολέας να καλέσει μια ρουτίνα αποστολής, ενώ η διεργασία-παραλήπτης να απαντήσει με μια κατάλληλη ρουτίνα λήψης. Εάν για κάποιο λόγο η ρουτίνα αποστολής μπορεί να επιστρέψει μόνο εφόσον βεβαιωθεί ότι τα δεδομένα έχουν παραληφθεί επιτυχώς (π.χ. λόγω έλλειψης πόρων για ενδιάμεση αποθήκευση), τότε οι δύο διεργασίες θα πρέπει ουσιαστικά να συγχρονιστούν, γεγονός που έχει δυσμενή επίδραση στην απόδοση του προγράμματος. Η πολυνηματική επεξεργασία επιτρέπει τον διαχωρισμό της πρόσβασης σε ιδιωτική και μοιραζόμενη μνήμη μέσω κατάλληλου ρητού (explicit) συγχρονισμού, που μπορεί να χρησιμοποιείται επιλεκτικά από τον προγραμματιστή μόνο εφόσον κρίνεται σημασιολογικά απαραίτητος, χωρίς να επιβαρύνει έτσι την επίδοση περισσότερο απ' ό,τι απαιτείται. Επίσης, ο συνδυασμός του συγχρονισμού και της μοιραζόμενης μνήμης επιτρέπει την εξάλειψη όλων των ενδιάμεσων αντιγραφών δεδομένων, που επιβάλλει το μοντέλο ανταλλαγής μηνυμάτων.

Για να εφαρμοστεί το πολυνηματικό μοντέλο προγραμματισμού, είναι σαφές ότι προϋποτίθεται υφι-

στάμενη αρχιτεκτονική μοιραζόμενης μνήμης. Όπως αναλύθηκε στην ενότητα 2.1.3, οι αρχιτεκτονικές αμιγώς μοιραζόμενης μνήμης υστερούν σε σχέση με τις αντίστοιχες κατανεμημένης μνήμης σε ό,τι αφορά στην επεκτασιμότητα, καθώς οι περιορισμοί του ρυθμού παροχής δεδομένων που επιβάλλει η χρήση ενός κεντρικού συστήματος μνήμης σε συνδυασμό με τη δυσκολία αποδοτικής κλιμάκωσης ενός πρωτοκόλλου συνάφειας κρυφής μνήμης μεταξύ των επεξεργαστών επιτρέπουν την επέκταση μιας τέτοιας αρχιτεκτονικής κατά μέγιστο σε λίγες δεκάδες επεξεργαστών. Εντούτοις, ο συνδυασμός των δύο βασικών αρχιτεκτονικών προς τη σύνθεση μιας νέας, υβριδικής αρχιτεκτονικής κατανεμημένης μοιραζόμενης μνήμης επιτυγχάνει να συγκεράσει τα επιθυμητά στοιχεία κάθε αρχιτεκτονικής, παρακάμπτοντας ταυτόχρονα αρκετούς από τους δυσμενείς περιορισμούς. Πέρα από τις υπόλοιπες ιδιαιτερότητές τους, οι αρχιτεκτονικές αυτές παρέχουν τη δυνατότητα τόσο για πολυνηματική επεξεργασία στο εσωτερικό κάθε κόμβου όσο και για ανταλλαγή μηνυμάτων μεταξύ διαφορετικών κόμβων, επιτρέποντας έτσι την ανάπτυξη ενός νέου προγραμματιστικού μοντέλου, που συχνά χαρακτηρίζεται ως *υβριδικό προγραμματιστικό μοντέλο*. Το μοντέλο αυτό αποσκοπεί στην αξιοποίηση της διεπίπεδης ιεραρχίας που ενυπάρχει σε τέτοια συστήματα, με απώτερο στόχο τη μείωση του συνολικού χρόνου εκτέλεσης του παράλληλου προγράμματος, κυρίως μέσω μείωσης της απαιτούμενης επικοινωνίας.

Μολονότι η αξιοποίηση της υφιστάμενης αρχιτεκτονικής αποτελεί βασική επιδίωξη του υβριδικού παράλληλου προγραμματιστικού μοντέλου, τα συγκριτικά θεωρητικά πλεονεκτήματα του τελευταίου δεν εξαντλούνται στο στοιχείο αυτό. Στην εργασία [SB01] γίνεται εκτενής αναφορά σε αρκετά από τα πλεονεκτήματα του υβριδικού παράλληλου προγραμματισμού, τα οποία συνοψίζουμε εδώ χάριν πληρότητας, μαζί με περαιτέρω συμπεράσματα που προέκυψαν κατά την ενασχόλησή μας με το ζήτημα της υβριδικής παραλληλοποίησης:

1. Σε κάποιες περιπτώσεις, η μονολιθική υλοποίηση ανταλλαγής μηνυμάτων μπορεί να παρουσιάζει μειωμένη *επεκτασιμότητα* για διαφορετικούς λόγους απ' ότι η ισοδύναμη πολυνηματική υλοποίηση για περιβάλλον μοιραζόμενης μνήμης. Για παράδειγμα, είναι πιθανό η πρώτη να εμφανίζει προβλήματα στην αποδοτική εξισορρόπηση του υπολογιστικού φορτίου, ενώ η δεύτερη να αντιμετωπίζει προβλήματα συμφόρησης κατά την προσπέλαση μεγάλου όγκου δεδομένων στο κοινό υποσύστημα μνήμης. Η χρήση υβριδικής παραλληλοποίησης επιτρέπει την άμβλυνση αμφότερων προβλημάτων, καθώς η κατανεμημένη μνήμη διασφαλίζει πολλαπλάσια δυνατότητα παροχής δεδομένων σε σχέση με το μονολιθικό πολυνηματικό μοντέλο, ενώ η χρήση πολυνηματικής επεξεργασίας στο εσωτερικό μιας διεργασίας προσφέρει ευελιξία ως προς την αποτελεσματικότερη εξισορρόπηση του φορτίου κατά το χρόνο εκτέλεσης.

2. Στην περίπτωση παραλληλοποίησης εφαρμογών που χαρακτηρίζονται εγγενώς από σχετικά μικρή δυνατότητα στατικής εκτίμησης του συνολικού υπολογιστικού φορτίου, η πολυνηματική επεξεργασία πλεονεκτεί διευκολύνοντας το *δυναμικό επαναπροσδιορισμό της κατανομής των δεδο-*

μένων και των υπολογισμών, καθώς δεν επιβαρύνει με την επιπλέον επικοινωνία που θα χρειαζόμασταν στο μονολιθικό μοντέλο ανταλλαγής μηνυμάτων. Σε τέτοιες εφαρμογές το υβριδικό μοντέλο μπορεί να επιτύχει αποδοτική δυναμική κατανομή των δεδομένων και των υπολογισμών μεταξύ των νημάτων στο εσωτερικό της διεργασίας, ενώ διατηρείται παράλληλα η χρήση ανταλλαγής μηνυμάτων για την επικοινωνία μεταξύ των κόμβων του κατανεμημένου περιβάλλοντος.

3. Γενικά, ο βέλτιστος κόκκος παραλληλισμού υπαγορεύεται τόσο από τη φύση των υπολογισμών του προβλήματος όσο και από την υφιστάμενη αρχιτεκτονική και ιδιαίτερα τις επιβαρύνσεις στην επικοινωνία που σχετίζονται με την τελευταία. Σε περιπτώσεις λοιπόν που από αλγοριθμικής πλευράς απαιτείται *παραλληλισμός λεπτού κόκκου*, η πολυνηματική επεξεργασία μπορεί να παρέχει πολύ καλύτερα αποτελέσματα, καθώς κατά τη μονολιθική προσέγγιση με ανταλλαγή μηνυμάτων το κόστος της επικοινωνίας γίνεται εμφανές στη συνολική επίδοση. Η χρήση πολυνηματικής επεξεργασίας στο εσωτερικό της διεργασίας μας επιτρέπει να επιτύχουμε τον επιθυμητό λεπτό κόκκο παραλληλισμού, αμβλύνοντας έτσι τα μειονεκτήματα της αναπόφευκτης χρήσης ανταλλαγής μηνυμάτων στις περιπτώσεις αρχιτεκτονικών κατανεμημένης μνήμης.

4. Τα κατανεμημένα περιβάλλοντα μνήμης μπορεί να επιτρέπουν την αποφυγή της συμφόρησης που παρατηρείται κατά την ταυτόχρονη προσπέλαση δεδομένων σε ένα κοινό υποσύστημα μοιραζόμενης μνήμης, αλλά συνεπάγονται συχνά και την *διατήρηση πολλαπλών αντιγράφων* των δεδομένων εκείνων, που επιθυμούν να προσπελάσουν ταυτόχρονα περισσότερες διεργασίες. Το υβριδικό μοντέλο παραλληλοποίησης μπορεί να πετύχει έναν αποδοτικό συμβιβασμό μεταξύ της κατανομής των δεδομένων για την ταυτόχρονη αξιοποίηση πολλαπλών μονοπατιών εισόδου/εξόδου και της αποδοτικής αξιοποίησης των διαθέσιμων πόρων για την αντιμετώπιση αλγορίθμων υψηλής χωρικής πολυπλοκότητας.

5. Είναι προφανές ότι η επίδοση που επιτυγχάνεται στο μοντέλο ανταλλαγής μηνυμάτων εξαρτάται από τη συγκεκριμένη υλοποίηση της βιβλιοθήκης επικοινωνίας που χρησιμοποιείται. Σε ορισμένες περιπτώσεις ενδέχεται η χρησιμοποιούμενη βιβλιοθήκη ανταλλαγής μηνυμάτων να επιβάλλει περιορισμούς π.χ. στο *μέγιστο δυνατό πλήθος διεργασιών* που μπορούν να εκκινήσουν, ή ακόμα να υλοποιεί μη αποδοτικά την *επικοινωνία στο εσωτερικό του κόμβου μοιραζόμενης μνήμης*. Στις περιπτώσεις αυτές η χρήση του υβριδικού μοντέλου μπορεί να μας επιτρέψει να παρακάμψουμε αρκετούς τέτοιους περιορισμούς, π.χ. συμπληρώνοντας με νήματα τον επιθυμητό αριθμό μονάδων παράλληλης εκτέλεσης όταν ο τελευταίος υπερβαίνει το μέγιστο επιτρεπτό πλήθος διεργασιών, ή κάνοντας διάκριση μεταξύ τοπικής και απομακρυσμένης επικοινωνίας, όπως στην παρούσα εργασία.

6. Σε πολλές περιπτώσεις, το υφιστάμενο δικτυακό πρωτόκολλο και το δίκτυο διασύνδεσης συνε-

πάγονται σημαντική *επιβάρυνση αρχικοποίησης* για κάθε μήνυμα που αποστέλλεται. Η χρήση του υβριδικού μοντέλου, όπως θα δούμε και στην ενότητα 5.3, επιτρέπει την αποστολή μικρότερου αριθμού μηνυμάτων, μειώνοντας έτσι το σχετικό κόστος αρχικοποίησης.

Η παραλληλοποίηση των υπό εξέταση αλγοριθμικών περιγραφών φωλιασμένων βρόχων με χρήση του υβριδικού μοντέλου απαιτεί κατάλληλο συγχρονισμό για τη διατήρηση των εξαρτήσεων δεδομένων του αλγορίθμου κατά την εκτέλεση του προγράμματος. Ο απαιτούμενος συγχρονισμός διασφαλίζεται από την εφαρμογή κατάλληλου σχήματος χρονοδρομολόγησης των επαναλήψεων, της χρονοδρομολόγησης υπερεπιπέδων.

5.1.1 Χρονοδρομολόγηση Υπερεπιπέδων

Η χρονοδρομολόγηση σωλήνωσης που χρησιμοποιήσαμε στην περίπτωση του μοντέλου ανταλλαγής μηνυμάτων δεν αντιμετωπίζει με τον πλέον αποδοτικό τρόπο την παράλληλη εκτέλεση των υπό εξέταση αλγορίθμων σε διεπίπεδες αρχιτεκτονικές μέσω του υβριδικού μοντέλου. Πράγματι, το σχήμα χρονοδρομολόγησης που εφαρμόσαμε στο κεφάλαιο 4 θεωρεί ως ισότιμες τις μονάδες εκτέλεσης του παράλληλου προγράμματος και κατ' επέκταση τους υπερκόμβους που εκτελούνται σε μια δεδομένη χρονική στιγμή. Στο υβριδικό μοντέλο όμως κάνουμε διάκριση μεταξύ δύο διαφορετικών μονάδων εκτέλεσης, των διεργασιών και των νημάτων, καθώς επίσης διαφοροποιούμε το συγχρονισμό μέσω της μοιραζόμενης μνήμης από την επικοινωνία μέσω του δικτύου διασύνδεσης. Για το σκοπό αυτό, σε όλα τα υβριδικά μοντέλα θα υιοθετήσουμε ένα πιο κατάλληλο σχήμα χρονοδρομολόγησης των υπερκόμβων, τη χρονοδρομολόγηση υπερεπιπέδων, που περιγράφεται στις εργασίες [ASTK02, Αθα05] και στο παρόν κεφάλαιο το εξειδικεύουμε για την περίπτωση ορθογώνιων χώρων και υπερκόμβων.

Η *χρονοδρομολόγηση υπερεπιπέδων* (*hyperplane scheduling*) ομαδοποιεί τους υπερκόμβους, που έχουν ανατεθεί στα νήματα μιας δεδομένης διεργασίας, σε *ομάδες* (*groups*), που μπορούν να εκτελεστούν ταυτόχρονα. Κάθε ομάδα αποτελείται από εκείνους τους υπερκόμβους, των οποίων η εκτέλεση μπορεί να ανατεθεί σε ένα ισάριθμο πλήθος νημάτων T, διατηρώντας όμως παράλληλα τις εξαρτήσεις δεδομένων του αρχικού αλγορίθμου. Υπό μία έννοια, κάθε ομάδα μπορεί να θεωρηθεί ως μια διακριτή χρονική στιγμή της ακολουθίας εκτέλεσης μιας διεργασίας, που καθορίζει ποια νήματα της εν λόγω διεργασίας θα πρέπει να εκτελέσουν κάποιον υπερκόμβο τη δεδομένη χρονική στιγμή και ποια όχι.

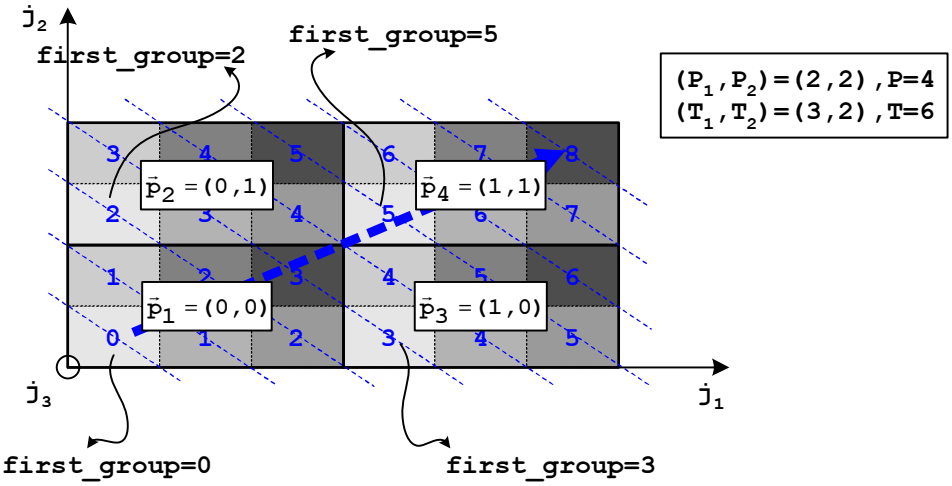
Η χρονοδρομολόγηση υπερεπιπέδων θεωρεί τη γενική περίπτωση εξαρτήσεων προς όλες τις μοναδιαίες κατευθύνσεις του χώρου, αφού όλες οι άλλες εξαρτήσεις μπορούν να εκφραστούν ως γραμμικός συνδυασμός των μοναδιαίων διανυσμάτων εξάρτησης. Θεωρούμε μόνο την ανάγκη επικοινωνίας μεταξύ γειτονικών διεργασιών, όπως άλλωστε και ρεαλιστικά συμβαίνει στη συντριπτική πλειοψηφία των αλγορίθμων φωλιασμένων βρόχων, για την ανταλλαγή π.χ. συνοριακών τιμών ή επιφανειών. Η χρονοδρομολόγηση υπερεπιπέδων επιδιώκει την ελαχιστοποίηση τόσο του συνολικού αριθμού των βημάτων

Αλγόριθμος 5.1: Χρονοδρομολόγηση υπερεπιπέδων

Δεδομένα: Αλγόριθμος (`Compute`), χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, διεργασία $\vec{p}$, νήμα $\vec{t}$

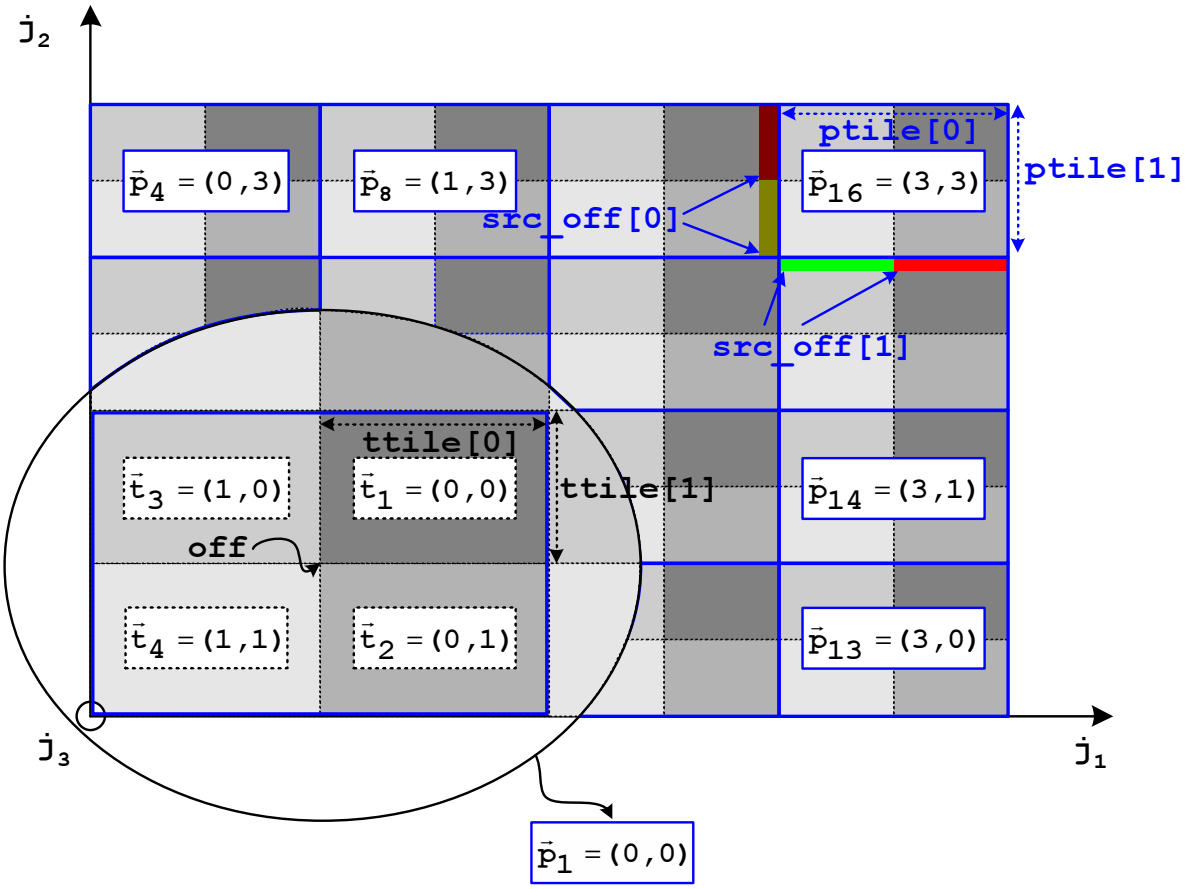
```
1  #pragma omp parallel
2  begin
3     for i ← 1 to N do
4        tile_i = p_i T_i + T_i − 1 − t_i;
5     endfor
6     foreach group ∈ G_p⃗ do
7        tile_{N+1} = group − Σ_{i=1}^{N} tile_i;
8        if 0 ≤ tile_{N+1} ≤ ⌈Z/z⌉ − 1 then
9           Compute(tile⃗);
10       endif
11       #pragma omp barrier
12    endforeach
13 end
```

εκτέλεσης του υβριδικού προγράμματος όσο και του απαιτούμενου συγχρονισμού μεταξύ των νημάτων.



Σχήμα 5.1: *Χρονοδρομολόγηση υπερεπιπέδων με 4 διεργασίες σε τοπολογία 2×2 και 6 νήματα ανά διεργασία σε τοπολογία 3×2. Οι πλάγιες διακεκομμένες γραμμές αντιστοιχούν στα διαφορετικά υπερεπίπεδα, ενώ η μεταβλητή `first_group` αντιστοιχεί στο υπερεπίπεδο εκκίνησης του πρώτου νήματος κάθε διεργασίας.*

Σχηματικά, η χρονοδρομολόγηση υπερεπιπέδων μπορεί να επιτευχθεί βάσει του αλγορίθμου 5.1. Θεωρείται ότι έχει επιλεγεί τόσο μια τοπολογία διεργασιών $\prod_{i=1}^{N} P_i$ όσο και μια τοπολογία νημάτων



Σχήμα 5.2: *Απεικόνιση τρισδιάστατου αλγορίθμου σε 16 διεργασίες $\times$ 4 νήματα ανά διεργασία κατά τη χρονοδρομολόγηση υπερεπιπέδων. Επιλέγεται τοπολογία 4×4 τόσο για τις διεργασίες όσο και για τα νήματα. Στο κάτω αριστερά μέρος φαίνεται σε λεπτομέρεια η διεργασία $\vec{p}_1$, για να αναδειχθεί ο ανάστροφος τρόπος απεικόνισης της τοπολογίας των νημάτων $\vec{t}_1$-$\vec{t}_4$. Επιπλέον, ενδεικτικά παρατίθενται τέσσερις πίνακες που είναι ιδιαίτερα χρήσιμοι για την αφαιρετική υλοποίηση του παράλληλου υβριδικού προγράμματος, ήτοι οι* `ptile` *(διαστάσεις υπερκόμβου διεργασίας),* `ttile` *(διαστάσεις υπερκόμβου νήματος),* `off` *(αρχική θέση υπερκόμβου νήματος) και* `src_off` *(αρχικές θέσεις δεδομένων επικοινωνίας γειτονικών διεργασιών).*

$\prod_{i=1}^{N} T_i$. Κάθε υπερκόμβος ταυτοποιείται από ένα διάνυσμα $\overrightarrow{tile}$ διάστασης $N+1$, όπου οι N πρώτες συνιστώσες σχετίζονται τόσο με την ιδιοκτήτρια διεργασία $\vec{p}$ όσο και με το συγκεκριμένο νήμα $\vec{t}$, ενώ η πιο εσωτερική απαριθμεί τις διακριτές χρονικές στιγμές. Το $\mathbb{G}_{\vec{p}}$ είναι το σύνολο όλων των χρονικών στιγμών του διαστήματος εκτέλεσης της διεργασίας $\vec{p}$ και καθορίζεται τόσο από την τοπολογία

απεικόνισης των διεργασιών όσο και από αυτή των νημάτων. Τυπικά, το σύνολο $\mathbb{G}_{\vec{p}}$ ορίζεται ως εξής:

$$\mathbb{G}_{\vec{p}} = \left\{ group \in \mathbb{N} | \sum_{i=1}^{N} p_i T_i \leq group \leq \sum_{i=1}^{N} p_i T_i + \sum_{i=1}^{N} \{T_i - 1\} + \left\lceil \frac{Z}{z} \right\rceil - 1 \right\}$$

Σε κάθε χρονική στιγμή $group$ της εκτέλεσης της διεργασίας $\vec{p}$, κάθε νήμα $\vec{t}$ υπολογίζει έναν *υποψήφιο* προς εκτέλεση υπερκόμβο $\overrightarrow{tile}$ και αποτιμά μια συνθήκη τύπου **if** για να καθορίσει εάν το $\overrightarrow{tile}$ αντιστοιχεί σε κάποιο *έγκυρο* υπερκόμβο που πρέπει να εκτελεστεί στην τρέχουσα χρονική στιγμή. Ουσιαστικά, κάθε υπερκόμβος ταυτοποιείται από μια τριάδα $(\vec{p}, \vec{t}, tile_{N+1})$, στοιχείο που θα αξιοποιήσουμε στα υβριδικά προγράμματα. Μια οδηγία `barrier` διασφαλίζει ότι όλα τα νήματα συγχρονίζονται μετά την εκτέλεση των έγκυρων υπερκόμβων, ώστε να μπορεί να εκκινήσει η επόμενη χρονική στιγμή.

Στο σχήμα 5.1 απεικονίζεται σχηματικά η χρονοδρομολόγηση υπερεπιπέδων για την περίπτωση 4 διεργασιών, 6 νημάτων ανά διεργασία και τρισδιάστατου αλγορίθμου. Ο αλγόριθμος απεικονίζεται στις μονάδες εκτέλεσης (διεργασίες, νήματα) κατά το επίπεδο $j_1 j_2$, ενώ κάθε νήμα πραγματοποιεί ακολουθιακή εκτέλεση κατά μήκος της διάστασης j_3, κάθετα στο επίπεδο του χαρτιού. Κάθε διεργασία ταυτοποιείται από ένα δισδιάστατο διάνυσμα $\vec{p} = (p_1, p_2)$, που της αποδίδεται έμμεσα από τη βιβλιοθήκη ανταλλαγής μηνυμάτων. Σε κάθε νήμα αναγράφεται η τιμή $group$ του υπερεπιπέδου εκείνου, στο οποίο το νήμα θα προβεί στην εκτέλεση του πρώτου υπερκόμβου του. Η `first_group` παρέχει την τιμή της παραμέτρου $group$ για το πρώτο νήμα της διεργασίας και ουσιαστικά σηματοδοτεί το υπερεπίπεδο στο οποίο εκκινεί η εκτέλεση των υπολογισμών της εν λόγω διεργασίας.

Τέλος, στο σχήμα 5.2 απεικονίζεται η ανάθεση των υπερκόμβων στα διαθέσιμα νήματα. Η αντιστοίχιση υπερκόμβων $tile$ σε νήματα $\vec{t}$ μέσω της εντολής ανάθεσης

$$tile_i = p_i T_i + T_i - 1 - t_i$$

του αλγορίθμου 5.1 οδηγεί στην «ανάστροφη» απεικόνιση της τοπολογίας των νημάτων, που φαίνεται στο σχήμα. Η επιλογή αυτή θα αιτιολογηθεί στην ενότητα 5.2.

5.1.2 Πολυνηματική Υποστήριξη

Μια θεμελιώδης παράμετρος για τη δυνατότητα υβριδικής παραλληλοποίησης είναι ο βαθμός υποστήριξης της πολυνηματικής επεξεργασίας από τη χρησιμοποιούμενη βιβλιοθήκη ανταλλαγής μηνυμάτων. Οι βιβλιοθήκες ανταλλαγής μηνυμάτων κατά κανόνα δεν υποστηρίζουν πλήρως την πολυνηματική επεξεργασία, είτε μη επιτρέποντας καν την ύπαρξη πολλαπλών νημάτων εκτέλεσης είτε περιορίζοντας/απαγορεύοντας την ταυτόχρονη κλήση ρουτινών ανταλλαγής μηνυμάτων από πολλαπλά νήματα. Το γεγονός αυτό οφείλεται στο ότι οι υλοποιήσεις των βιβλιοθηκών ανταλλαγής μηνυμάτων δεν πα-

ρέχουν *ασφάλεια νήματος* (*thread-safety*): ορισμένα τμήματα του κώδικα της βιβλιοθήκης δεν μπορούν να κληθούν ταυτόχρονα από πολλαπλά νήματα εκτέλεσης, κυρίως γιατί δεν έχει ληφθεί μέριμνα ώστε η προσπέλαση μοιραζόμενων δομών να γίνεται μέσω ατομικών λειτουργιών. Για παράδειγμα, η προσπέλαση της ουράς μηνυμάτων από δύο νήματα για την επεξεργασία αιτήσεων επικοινωνίας ενδέχεται να οδηγήσει σε συνθήκες ανταγωνισμού και τελικά εσφαλμένη λειτουργία. Η υλοποίηση πλήρους πολυνηματικής υποστήριξης συνεπάγεται εν γένει επιπτώσεις στην επίδοση ενός προγράμματος, π.χ. λόγω της αναπόφευκτης χρήσης *μηχανισμών κλειδώματος* (*locking*) και *κρίσιμων περιοχών* (*critical sections*) για τη διασφάλιση της απαιτούμενης ατομικότητας σε συγκεκριμένες λειτουργίες.

Στη βιβλιογραφία γίνεται αναφορά κυρίως σε πέντε διακριτά επίπεδα πολυνηματικής υποστήριξης:

1. *single*

   Η βιβλιοθήκη ανταλλαγής μηνυμάτων δεν υποστηρίζει πολυνηματική επεξεργασία. Είναι προφανές ότι σε τέτοια περίπτωση η υλοποίηση υβριδικών προγραμματιστικών μοντέλων είναι αδύνατη.

2. *masteronly*

   Η βιβλιοθήκη ανταλλαγής μηνυμάτων επιτρέπει πολυνηματική επεξεργασία, αλλά περιορίζει τη δυνατότητα κλήσης ρουτινών ανταλλαγής μηνυμάτων μόνο εκτός της δυναμικής εμβέλειας των παράλληλων περιοχών πολυνηματικής επεξεργασίας. Η περίπτωση αυτή ουσιαστικά επιτρέπει την εφαρμογή μόνο του υβριδικού μοντέλου λεπτού κόκκου (όχι δηλαδή του εναλλακτικού μοντέλου χονδρού κόκκου).

3. *funneled*

   Η βιβλιοθήκη ανταλλαγής μηνυμάτων επιτρέπει πολυνηματική επεξεργασία, αλλά μόνο το πρωτεύον νήμα μπορεί να καλέσει λειτουργίες ανταλλαγής μηνυμάτων. Ταυτόχρονα, τα υπόλοιπα νήματα μπορούν να εκτελούν άλλες λειτουργίες. Η περίπτωση αυτή αποτελεί τη συνηθέστερη δυνατότητα που παρέχεται από τις υπάρχουσες ελεύθερες βιβλιοθήκες ανταλλαγής μηνυμάτων και επιτρέπει την υλοποίηση αμφότερων των υβριδικών μοντέλων παραλληλοποίησης που περιγράφηκαν παραπάνω.

4. *serialized*

   Η βιβλιοθήκη ανταλλαγής μηνυμάτων επιτρέπει πολυνηματική επεξεργασία, και μάλιστα όλα τα νήματα μπορούν να καλούν λειτουργίες ανταλλαγής μηνυμάτων, στο βαθμό που διασφαλίζεται η σειριοποίηση του φαινομένου αυτού. Με άλλα λόγια, δεν επιτρέπεται σε περισσότερα του ενός νήματος να καλούν την ίδια χρονική στιγμή ρουτίνες ανταλλαγής μηνυμάτων, αλλά δεν υπάρχει

περιορισμός ως προς την ταυτότητα του νήματος που μπορεί να αναλάβει το ρόλο αυτό σε κάθε χρονική στιγμή (δεν χρειάζεται π.χ. να πρόκειται για το πρωτεύον νήμα).

5. *multiple*

   Η βιβλιοθήκη ανταλλαγής μηνυμάτων παρέχει πλήρη πολυνηματική υποστήριξη. Κάθε νήμα μπορεί να καλεί λειτουργίες ανταλλαγής μηνυμάτων, χωρίς κανένα απολύτως περιορισμό.

Κάθε κατηγορία αποτελεί υπερσύνολο όλων των προηγουμένων. Στην παρούσα φάση, οι πιο δημοφιλείς βιβλιοθήκες ανταλλαγής μηνυμάτων ανοιχτού κώδικα παρέχουν πολυνηματική υποστήριξη μέχρι το funneled επίπεδο, ενώ μόνο λίγες εμπορικές κλειστού τύπου υλοποιήσεις επιτρέπουν πλήρη πολυνηματική υποστήριξη (περίπτωση multiple). Συνέπεια του γεγονότος αυτού είναι πως οι περισσότερες υβριδικές υλοποιήσεις, που έχουν μέχρι σήμερα προταθεί ή υλοποιηθεί, περιορίζονται στα τρία πρώτα επίπεδα πολυνηματικής υποστήριξης. Το δεύτερο επίπεδο πολυνηματικής υποστήριξης (masteronly) εμφανίζεται συχνά στη σχετική βιβλιογραφία, καθώς η υβριδική παραλληλοποίηση επιτυγχάνεται συνήθως με επαυξητική παραλληλοποίηση των υπολογιστικά απαιτητικών τμημάτων με ομαδοποίηση και κατανομή των υπολογισμών μεταξύ των νημάτων. Η προσέγγιση αυτή χαρακτηρίζεται ως υβριδικός παραλληλισμός λεπτού κόκκου και επιτυγχάνεται συνήθως με άμεσο τρόπο περικλείοντας τους κατάλληλους βρόχους με λειτουργίες κατανομής εργασίας της πολυνηματικής διεπαφής. Από την άλλη μεριά, ο υβριδικός παραλληλισμός χονδρού κόκκου με πολυνηματική επεξεργασία SPMD τύπου αναφέρεται σπανιότερα στη διεθνή βιβλιογραφία, λόγω της προγραμματιστικής πολυπλοκότητας που συνεπάγεται.

5.1.3 Υβριδικό Μοντέλο Λεπτού Κόκκου

Το υβριδικό προγραμματιστικό μοντέλο λεπτού κόκκου (fine-grain ή masteronly) αποτελεί τη δημοφιλέστερη προσέγγιση για υβριδική παραλληλοποίηση, παρότι όπως θα δούμε θέτει σημαντικούς περιορισμούς στην επίτευξη υψηλής απόδοσης. Η δημοφιλία του υβριδικού μοντέλου λεπτού κόκκου οφείλεται κυρίως στην προγραμματιστική απλότητά του: στις περισσότερες περιπτώσεις μπορεί να προκύψει άμεσα από το μοντέλο ανταλλαγής μηνυμάτων με περαιτέρω *επαυξητικό παραλληλισμό* (*incremental parallelization*) των υπολογιστικά απαιτητικών τμημάτων. Έτσι, δεν απαιτεί ιδιαίτερη τροποποίηση της δομής ενός υπάρχοντος κώδικα ανταλλαγής μηνυμάτων και μπορεί να υλοποιηθεί σχετικά απλά, αρχικά με ανάλυση της απόδοσης του κώδικα στα επιμέρους τμήματα και ακολούθως με πρόσθετη παραλληλοποίηση των περισσότερο «βαριών» τμημάτων με χρήση πολυνηματικής επεξεργασίας. Θα πρέπει επίσης να αναφερθεί ότι συχνά η υβριδική παραλληλοποίηση λεπτού κόκκου συνιστά τη μόνη εφικτή δυνατότητα υβριδικής παραλληλοποίησης σε περιπτώσεις που η βιβλιοθήκη ανταλλαγής μηνυμάτων

επιτρέπει την κλήση ρουτινών επικοινωνίας μόνο εκτός των περιοχών πολυνηματικής επεξεργασίας (επίπεδο masteronly πολυνηματικής υποστήριξης, βλ. ενότητα 5.1.2).

Το τίμημα για την προγραμματιστική απλότητα του υβριδικού μοντέλου λεπτού κόκκου εντοπίζεται κυρίως στους σημαντικούς περιορισμούς που επιβάλλει στην προσπάθεια επίτευξης υψηλής απόδοσης. Κατά κύριο λόγο, η αποδοτικότητα του μοντέλου αυτού είναι άρρηκτα συνυφασμένη με το ποσοστό του κώδικα, για το οποίο εφαρμόζεται ο επαυξητικός πολυνηματικός παραλληλισμός. Σύμφωνα με το νόμο του Amdahl, καθώς η ανταλλαγή μηνυμάτων στο μοντέλο αυτό περιορίζεται εκτός των περιοχών πολυνηματικής επεξεργασίας, κατά την ανταλλαγή μηνυμάτων όλα τα νήματα πλην του πρωτεύοντος παραμένουν ανενεργά, κατασπαταλώντας μέρος της υπολογιστικής ισχύος του συστήματος και επιτυγχάνοντας φτωχή εξισορρόπηση φορτίου. Επιπροσθέτως, το μοντέλο λεπτού κόκκου εμφανίζει την επιβάρυνση της επαναρχικοποίησης των πολυνηματικών δομών, καθώς τα νήματα αρχικοποιούνται και τερματίζονται κατά την επαναλαμβανόμενη είσοδο και έξοδο προς και από παράλληλες περιοχές πολυνηματικής επεξεργασίας. Στα παραπάνω μειονεκτήματα του υβριδικού μοντέλου λεπτού κόκκου θα πρέπει κανείς να προσθέσει πως ο επαυξητικός παραλληλισμός αποτελεί μια αρκετά περιοριστική προσέγγιση παραλληλοποίησης και κρίνεται ανεπαρκής για αρκετούς πραγματικούς αλγορίθμους, στους οποίους είτε δεν υπάρχουν οι απαιτούμενοι επαναληπτικοί βρόχοι, είτε δεν μπορούν να συμπεριληφθούν άμεσα σε παράλληλες περιοχές πολυνηματικής επεξεργασίας. Αποτελεί άλλωστε κοινό τόπο πως η επιτυχία του προγραμματιστικού μοντέλου ανταλλαγής μηνυμάτων μπορεί σε μεγάλο βαθμό να αποδοθεί στη γενικότητα του SPMD προγραμματιστικού μοντέλου, το οποίο μπορεί να υποκατασταθεί μόνο μερικώς από το υβριδικό μοντέλο λεπτού κόκκου.

Η προτεινόμενη υβριδική παραλληλοποίηση λεπτού κόκκου για την περίπτωση επαναληπτικών αλγορίθμων φωλιασμένων βρόχων μπορεί να υλοποιηθεί βάσει του σχήματος του αλγορίθμου 5.2. Η χρονοδρομολόγηση υπερεπιπέδων συνδυάζεται με το μοντέλο ανταλλαγής μηνυμάτων για την περαιτέρω πολυνηματική παραλληλοποίηση του υπολογιστικού τμήματος του βρόχου με την επαυξητική προσέγγιση. Όλη η απαιτούμενη επικοινωνία για την ανταλλαγή μηνυμάτων διεξάγεται εκτός της πολυνηματικής περιοχής (γραμμές 2-8 και 19-22) ενώ η πολυνηματική παραλληλοποίηση του υπολογισμού υλοποιείται στις γραμμές 9-18. Συγκεκριμένα, στις γραμμές 2-8 ομαδοποιούνται και αποστέλλονται τα δεδομένα που υπολογίστηκαν κατά το προηγούμενο υπερεπίπεδο $group - 1$, ενώ πραγματοποιείται λήψη των δεδομένων που θα χρειαστούμε κατά τους υπολογισμούς του επόμενου υπερεπιπέδου $group + 1$. Όλες οι λειτουργίες αποστολής και λήψης είναι ασύγχρονες, ώστε να επιτρέψουν την επικάλυψη της επικοινωνίας με τους υπολογισμούς του τρέχοντος υπερεπιπέδου $group$, εφόσον παρέχεται η σχετική δυνατότητα από το υλικό. Σε γενικές γραμμές, η επικοινωνία ακολουθεί τη λογική του μοντέλου ανταλλαγής μηνυμάτων, με τη μόνη διαφορά ότι αναφέρεται σε υπερεπίπεδα που αφορούν εν γένει T υπερκόμβους ισάριθμων νημάτων, σε αντιδιαστολή με την επικοινωνία ανά υπερκόμβο διεργασίας

Αλγόριθμος 5.2: Υβριδικό προγραμματιστικό μοντέλο λεπτού κόκκου

Δεδομένα: Αλγόριθμος (`Compute`), χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, διεργασία $\vec{p}$, νήμα $\vec{t}$

```
1  foreach group ∈ G_p⃗ do
2    foreach dir⃗ ∈ S_p⃗ do
3      Pack(snd_buf [dir⃗],group − 1,p⃗);
4      MPI_Isend(snd_buf [dir⃗],dest(p⃗ + dir⃗));
5    endforeach
6    foreach dir⃗ ∈ R_p⃗ do
7      MPI_Irecv(recv_buf [dir⃗],src(p⃗ − dir⃗));
8    endforeach
9    #pragma omp parallel
10   begin
11     for i ← 1 to N do
12       tile_i = p_i T_i + T_i − 1 − t_i;
13     endfor
14     tile_{N+1} = group − Σ_{i=1}^{N} tile_i;
15     if 0 ≤ tile_{N+1} ≤ ⌈Z/z⌉ − 1 then
16       Compute(tile⃗);
17     endif
18   end
19   MPI_Waitall;
20   foreach dir⃗ ∈ R_p⃗ do
21     Unpack(recv_buf [dir⃗],group + 1,p⃗);
22   endforeach
23 endforeach
```

που είχαμε στο μονολιθικό μοντέλο.

Παρατηρούμε τέλος ότι δεν απαιτείται ρητή λειτουργία barrier για το συγχρονισμό των νημάτων, καθώς κάτι τέτοιο επιτυγχάνεται έμμεσα κατά την έξοδο από την παράλληλη πολυνηματική περιοχή. Είναι επίσης αξιοσημείωτο ότι μόνο το τμήμα του κώδικα μεταξύ των γραμμών 9-18 αξιοποιεί πλήρως την υφιστάμενη επεξεργαστική υποδομή, γεγονός που περιορίζει σημαντικά την παράλληλη απόδοση του προγράμματος. Πράγματι, μόνο στο συγκεκριμένο κομμάτι του κώδικα έχουμε παράλληλη εκτέλεση σε όλους τους διαθέσιμους επεξεργαστές, καθώς αποτελεί κοινή πρακτική το γινόμενο του αριθμού των διεργασιών P επί το πλήθος των νημάτων T που εκκινεί κάθε διεργασία να ισούται με το σύνολο των επεξεργαστών για πλήρη εκμετάλλευση της υποδομής. Έτσι, παρότι το υβριδικό μοντέλο μειώνει γενικά το συνολικό *όγκο των δεδομένων* που πρέπει να ανταλλαγούν μεταξύ των διεργασιών λόγω της

μερικής δυνατότητας πρόσβασης σε κοινή μνήμη, γίνεται φανερό ότι κατά την υβριδική προσέγγιση λεπτού κόκκου υποπολλαπλασιάζονται κατά ένα παράγοντα T και οι *μονάδες επικοινωνίας*, δηλαδή οι διαθέσιμες διεργασίες που μπορούν να καλέσουν ρουτίνες ανταλλαγής μηνυμάτων. Για το λόγο αυτό, ήδη σε θεωρητικό επίπεδο είναι αμφίβολο κατά πόσο το υβριδικό μοντέλο λεπτού κόκκου θα πλεονεκτεί τελικά του αντίστοιχου μοντέλου ανταλλαγής μηνυμάτων, καθώς η σχετική υπεροχή της μίας ή της άλλης προσέγγισης φαίνεται να εξαρτάται από το κατά πόσο η μείωση των δεδομένων επικοινωνίας μπορεί να υπερκεράσει την αντίστοιχη μείωση του αριθμού των διεργασιών που υλοποιούν την επικοινωνία αυτή. Το γεγονός αυτό έχει επισημανθεί στη διεθνή βιβλιογραφία [RW03] και αποτελεί το βασικότερο μειονέκτημα του υβριδικού μοντέλου λεπτού κόκκου για παράλληλες εφαρμογές με υψηλές ανάγκες επικοινωνίας.

5.1.4 Υβριδικό Μοντέλο Χονδρού Κόκκου - Funneled

Το υβριδικό μοντέλο χονδρού κόκκου (coarse-grain) διαφοροποιείται από το αντίστοιχο του λεπτού κόκκου κυρίως ως προς το ότι τα νήματα αρχικοποιούνται μόνο μία φορά στην αρχή του προγράμματος και τα αναγνωριστικά τους (thread ids) χρησιμοποιούνται για τη διαφοροποίηση της ροής εκτέλεσής τους, όπως ακριβώς συμβαίνει και στο SPMD προγραμματιστικό μοντέλο. Αναλυτικότερα, τα αναγνωριστικά των νημάτων βοηθούν στον καθορισμό τόσο του υπολογιστικού φορτίου τους όσο και των δεδομένων επικοινωνίας τους. Η ανταλλαγή μηνυμάτων μεταξύ των διεργασιών διαφορετικών πολυεπεξεργαστικών κόμβων λαμβάνει χώρα εντός της δυναμικής εμβέλειας των περιοχών πολυνηματικής επεξεργασίας, αλλά συνήθως διεκπεραιώνεται αποκλειστικά από το πρωτεύον νήμα, όπως υπαγορεύει η συνήθης περιορισμένη πολυνηματική υποστήριξη επιπέδου funneled της βιβλιοθήκης ανταλλαγής μηνυμάτων.

Σε γενικές γραμμές, το υβριδικό μοντέλο χονδρού κόκκου αναπληρώνει τη σχετικά υψηλότερη προγραμματιστική πολυπλοκότητα με τη δυνατότητα επίτευξης ανώτερης επίδοσης σε σχέση με το υβριδικό μοντέλο λεπτού κόκκου. Το βασικό πλεονέκτημα της προσέγγισης αυτής έγκειται στο ότι επιτρέπει την πραγματοποίηση κλήσεων ρουτινών ανταλλαγής μηνυμάτων εντός της εμβέλειας πολυνηματικών περιοχών, αντίθετα με το μοντέλο λεπτού κόκκου. Έτσι, η υβριδική προσέγγιση χονδρού κόκκου παρέχει τη δυνατότητα *επικάλυψης* της πολυνηματικής επεξεργασίας με την επικοινωνία μέσω ανταλλαγής μηνυμάτων. Επιπλέον, το μοντέλο χονδρού κόκκου επιτρέπει την αποφυγή της πρόσθετης επιβάρυνσης που σχετίζεται με την επαναλαμβανόμενη αρχικοποίηση και αναστολή των πολυνηματικών δομών, καθώς τα νήματα αρχικοποιούνται μία μόνο φορά στην αρχή του προγράμματος. Εξίσου σημαντική κρίνεται η δυνατότητα που παρέχει το μοντέλο χονδρού κόκκου για την υλοποίηση γενικότερων σχημάτων παραλληλίας, σε αντιδιαστολή με το περιοριστικό μοντέλο λεπτού κόκκου.

Στη διεθνή βιβλιογραφία γίνεται διάκριση μεταξύ του *funneled* υβριδικού μοντέλου χονδρού κόκ-

κου, στο οποίο κλήσεις ανταλλαγής μηνυμάτων γίνονται μόνο από το πρωτεύον νήμα, και του *multiple* υβριδικού μοντέλου χονδρού κόκκου, όπου όλα τα νήματα μπορούν να διεκπεραιώνουν επικοινωνία με μηνύματα. Ουσιαστικά, οι δύο εναλλακτικές υβριδικές υλοποιήσεις χονδρού κόκκου αντιστοιχούν στα ομώνυμα επίπεδα πολυνηματικής υποστήριξης που παρουσιάστηκαν στην ενότητα 5.1.2. Στην παρούσα ενότητα θα προτείνουμε μια funneled προσέγγιση υβριδικής παραλληλοποίησης αλγορίθμων φωλιασμένων βρόχων, ενώ η multiple παραλλαγή θα αποτελέσει αντικείμενο της επόμενης ενότητας.

Στην περίπτωση της συνήθους funneled προσέγγισης, η προφανής υλοποίηση του υβριδικού μοντέλου χονδρού κόκκου αντιμετωπίζει πρόβλημα στην αποτελεσματική εξισορρόπηση του φορτίου μεταξύ των νημάτων. Πιο συγκεκριμένα, μια ισοκατανομή του υπολογιστικού φορτίου μεταξύ των διαθέσιμων νημάτων θα επιβάρυνε περισσότερο το πρωτεύον νήμα, το οποίο επιπλέον έχει και την αποκλειστική αρμοδιότητα της περάτωσης της επικοινωνίας μέσω ανταλλαγής μηνυμάτων. Καθίσταται συνεπώς σαφές πως προκειμένου να είναι αποτελεσματικό το υβριδικό μοντέλο χονδρού κόκκου απαιτείται ειδική μέριμνα κατά την εξισορρόπηση του φορτίου μεταξύ των νημάτων, έτσι ώστε το πρωτεύον νήμα να αναλάβει αναλογικά μικρότερο φόρτο υπολογισμού σε σχέση με τα υπόλοιπα νήματα, κατά τρόπο που να εξισώνονται οι συνολικοί χρόνοι εκτέλεσης υπερκόμβου (υπολογισμός+επικοινωνία) για κάθε νήμα.

Σχηματικά, το funneled υβριδικό μοντέλο χονδρού κόκκου μπορεί να υλοποιηθεί για την περίπτωση αλγοριθμικών περιγραφών φωλιασμένων βρόχων όπως στον αλγόριθμο 5.3. Η επικοινωνία μεταξύ διεργασιών διαφορετικών πολυεπεξεργαστικών κόμβων (γραμμές 10-18 και 25-30) διεξάγεται από το πρωτεύον νήμα, ανά κατεύθυνση επικοινωνίας και ανά νήμα-ιδιοκτήτη των δεδομένων επικοινωνίας. Συγκεκριμένα, στις γραμμές 10-18 το πρωτεύον νήμα αναλαμβάνει να ομαδοποιήσει τα συνοριακά δεδομένα που υπολογίστηκαν κατά το προηγούμενο υπερεπίπεδο από όλα τα νήματα σε ένα μήνυμα ανά κατεύθυνση επικοινωνίας $\overrightarrow{dir} \in \mathbb{S}_{\vec{p}}$ της διεργασίας $\vec{p}$, ενώ αντίστοιχα διαμορφώνεται και η διαδικασία λήψης. Αποστέλλοντας ένα μόνο μήνυμα `snd_buf`$[\vec{dir}]$ σε κάθε γειτονική διεργασία $\vec{p} + \overrightarrow{dir}$ επιτυγχάνουμε να ελαχιστοποιήσουμε την αρχική καθυστέρηση του δικτύου διασύνδεσης, που γενικά αυξάνει με το πλήθος των αποστελλόμενων μηνυμάτων. Τα τμήματα κώδικα που αφορούν σε ανταλλαγή μηνυμάτων περικλείονται από τη σχετική λειτουργία `master`, ώστε να εκτελεστούν μόνο από το πρωτεύον νήμα. Η διασφάλιση αυτού του σχήματος επικοινωνίας επιφέρει πρόσθετη επιβάρυνση, τόσο σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων όσο και συγκριτικά με το υβριδικό μοντέλο λεπτού κόκκου. Η έκφραση `bal`$(\vec{p}, \vec{t})$, που εμφανίζεται στο υπολογιστικό τμήμα του αλγορίθμου, αναφέρεται στην υλοποίηση σχήματος εξισορρόπησης φορτίου για το νήμα $\vec{t}$ της διεργασίας $\vec{p}$ και θα αναλυθεί εκτενέστερα στην ενότητα 5.2. Τέλος, παρατηρούμε ότι το υβριδικό μοντέλο χονδρού κόκκου απαιτεί ρητό συγχρονισμό των νημάτων με τη βοήθεια λειτουργίας **barrier**, κάτι που όμως αναμένεται να επιφέρει μικρή μόνο επιβάρυνση σε σχέση με το όφελος που αντλείται από την αποφυγή της επαναλαμβανόμενης αρχικοποίησης των πολυνηματικών δομών, όπως συμβαίνει στην περίπτωση του μοντέλου λεπτού

Αλγόριθμος 5.3: Υβριδικό προγραμματιστικό μοντέλο χονδρού κόκκου - funneled

Δεδομένα: Αλγόριθμος (`Compute`), χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, διεργασία $\vec{p}$, νήμα $\vec{t}$

```
1  #pragma omp parallel
2  begin
3    for i ← 1 to N do
4      tile_i = p_i T_i + T_i − 1 − t_i;
5    endfor
6    foreach group ∈ G_p⃗ do
7      tile_{N+1} = group − Σ_{i=1}^{N} tile_i;
8      #pragma omp master
9      begin
10       foreach dir⃗ ∈ S_p⃗ do
11         for th ← 1 to T do
12           Pack(snd_buf [dir⃗],group − 1,p⃗,th);
13         endfor
14         MPI_Isend(snd_buf [dir⃗],dest(p⃗ + dir⃗));
15       endforeach
16       foreach dir⃗ ∈ R_p⃗ do
17         MPI_Irecv(recv_buf [dir⃗],src(p⃗ − dir⃗));
18       endforeach
19     end
20     if 0 ≤ tile_{N+1} ≤ ⌈Z/z⌉ − 1 then
21       Compute(tile⃗,bal(p⃗,t⃗));
22     endif
23     #pragma omp master
24     begin
25       MPI_Waitall;
26       foreach dir⃗ ∈ R_p⃗ do
27         for th ← 1 to T do
28           Unpack(recv_buf [dir⃗],group + 1,p⃗,th);
29         endfor
30       endforeach
31     end
32     #pragma omp barrier
33   endforeach
34 end
```

κόκκου.

5.1.5 Υβριδικό Μοντέλο Χονδρού Κόκκου - Multiple

Στην περίπτωση που η χρησιμοποιούμενη βιβλιοθήκη ανταλλαγής μηνυμάτων παρέχει πλήρη πολυνηματική υποστήριξη, καθίσταται εφικτή μια εναλλακτική υλοποίηση του υβριδικού μοντέλου χονδρού κόκκου. Έτσι, αν η βιβλιοθήκη ανταλλαγής μηνυμάτων επιτρέπει την κλήση ρουτινών επικοινωνίας από όλα τα νήματα, μπορούμε να υιοθετήσουμε μια προγραμματιστικά απλούστερη προσέγγιση, στην οποία κάθε νήμα αναλαμβάνει την πλήρωση των ιδίων αναγκών επικοινωνίας. Μια τέτοια παραλλαγή του υβριδικού μοντέλου χονδρού κόκκου κατονομάζεται συχνά ως multiple και χαρακτηρίζεται εγγενώς από περισσότερο ισορροπημένη κατανομή του συνολικού φορτίου επικοινωνίας μεταξύ των νημάτων, σε αντιδιαστολή με τη funneled παραλλαγή που επιβαρύνει σχετικά μόνο το πρωτεύον νήμα.

Η υλοποίηση της επικοινωνίας μεταξύ των απομακρυσμένων νημάτων στο multiple μοντέλο είναι μη τετριμμένη διαδικασία. Πράγματι, καθώς οι βιβλιοθήκες ανταλλαγής μηνυμάτων θεωρούν μόνο τις διεργασίες ως επικοινωνούσες οντότητες, η επικοινωνία σε επίπεδο νημάτων δεν μπορεί να γίνει άμεσα, με κλήση ρουτινών της βιβλιοθήκης. Για παράδειγμα, σύμφωνα με το πρότυπο MPI δεν υπάρχει άμεσος τρόπος να απευθυνθεί ένα νήμα $\vec{t_i}$ της διεργασίας $\vec{p}$ σε ένα νήμα $\vec{t_j}$ της απομακρυσμένης διεργασίας $\vec{p'}$. Ανταλλαγή μηνύματος μπορεί να διεκπεραιωθεί μόνο μεταξύ των $\vec{p}$ και $\vec{p'}$ και όχι σε ένα πιο λεπτομερές επίπεδο μεταξύ νημάτων. Για να παρακάμψουμε αυτήν τη δυσκολία, είναι δυνατό να χρησιμοποιήσουμε την *ετικέτα* (*tag*) του μηνύματος MPI ώστε να ενσωματώσουμε έμμεσα στο φάκελο του μηνύματος την αντιστοιχία μεταξύ τοπικού και απομακρυσμένου νήματος. Έτσι, ένα μήνυμα που αποστέλλεται από το νήμα $\vec{t_i}$ μέσω κλήσης ρουτίνας αποστολής από την ιδιοκτήτρια διεργασία $\vec{p}$ θα αντιστοιχιστεί με την κατάλληλη κλήση ρουτίνας λήψης στη διεργασία $\vec{p'}$.

Ο αλγόριθμος 5.4 συνοψίζει τα κυριότερα σημεία της multiple υβριδικής υλοποίησης χονδρού κόκκου. Ο αλγόριθμος είναι παρόμοιος με τον 5.3 της funneled περίπτωσης, χωρίς όμως τις οδηγίες **master**, καθώς όλα τα νήματα -και όχι μόνο το πρωτεύον- συνεισφέρουν στην επικοινωνία με απομακρυσμένες διεργασίες. Τα σύνολα επικοινωνίας $\mathbb{S}_{\vec{p}}$ και $\mathbb{R}_{\vec{p}}$ έχουν αντικατασταθεί από τα $\mathbb{S}_{\vec{p},\vec{t}}$ και $\mathbb{R}_{\vec{p},\vec{t}}$, καθώς στην επικοινωνία πλέον μετέχουν ενεργά και τα νήματα. Για παράδειγμα, το σύνολο $\mathbb{S}_{\vec{p},\vec{t}}$ αντιστοιχεί σε όλες τις έγκυρες διευθύνσεις μετάδοσης δεδομένων για το νήμα $\vec{t}$ της ιδιοκτήτριας διεργασίας $\vec{p}$. Συγκεκριμένα, αν για μια κατεύθυνση επικοινωνίας $\overrightarrow{dir}$ ισχύει $\overrightarrow{dir} \in \mathbb{S}_{\vec{p},\vec{t}}$, τότε δεδομένα που υπολογίζονται από το νήμα $\vec{t}$ της μη συνοριακής διεργασίας $\vec{p}$ πρέπει να αποσταλούν στη διεργασία $\vec{p} + \overrightarrow{dir}$. Ομοίως, αν $\overrightarrow{dir} \in \mathbb{R}_{\vec{p},\vec{t}}$, τότε η διεργασία $\vec{p}$ πρέπει να λάβει δεδομένα από τη γειτονική της $\vec{p} - \overrightarrow{dir}$, λόγω εξαρτήσεων που σχετίζονται με υπολογισμούς δεδομένων του νήματος $\vec{t}$ της $\vec{p}$. Λόγω της απεικόνισης της τοπολογίας νημάτων που επιλέγεται (βλ. σχήμα 5.2), το νήμα $\vec{t} = (t_1, \ldots, t_N)$ της διεργασίας $\vec{p}$ πρέπει να στείλει δεδομένα προς όλες τις κατευθύνσεις επικοινωνίας $j \in \mathbb{S}_{\vec{p}}$, για τις οποίες ισχύει

$$t_j = 0$$

Αλγόριθμος 5.4: Υβριδικό προγραμματιστικό μοντέλο χονδρού κόκκου - multiple

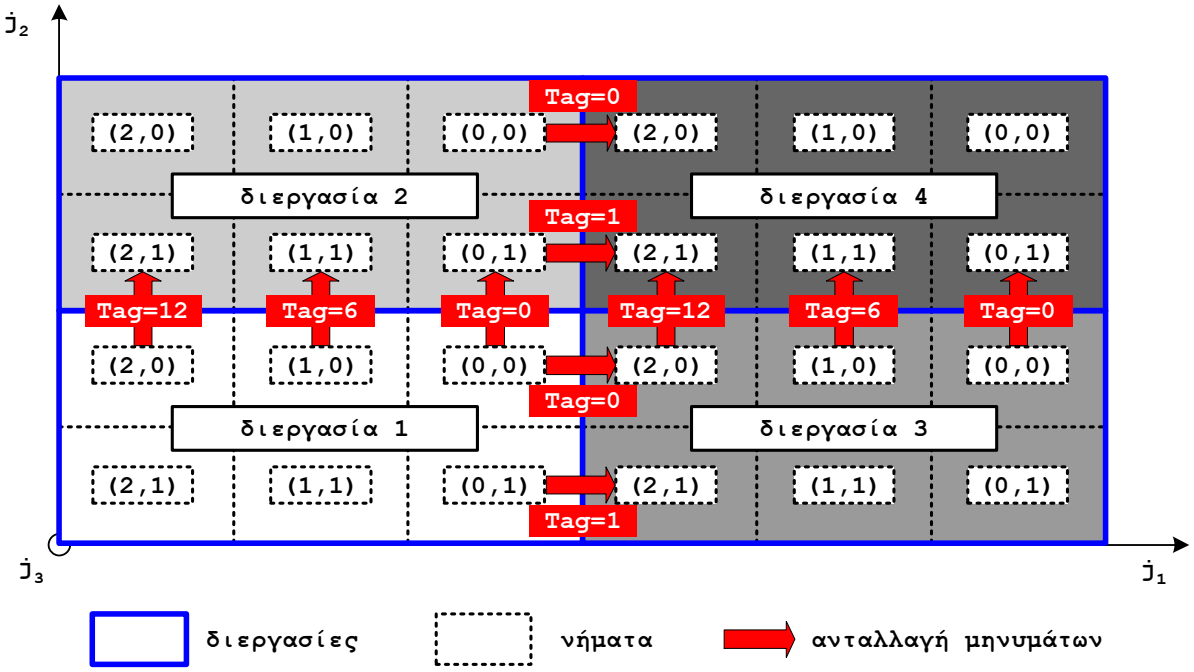
Δεδομένα: Αλγόριθμος (`Compute`), χώρος επαναλήψεων $\prod_{i=1}^{N} X_i Z$, διεργασία $\vec{p}$, νήμα $\vec{t}$

```
1  #pragma omp parallel
2  begin
3    for i ← 1 to N do
4      tile_i = p_i T_i + T_i − 1 − t_i;
5    endfor
6    foreach group ∈ G_p⃗ do
7      tile_{N+1} = group − Σ_{i=1}^{N} tile_i;
8      foreach dir⃗ ∈ S_{p⃗,t⃗} do
9        Pack(snd_buf [dir⃗],group − 1,p⃗,t⃗);
10       MPI_Isend(snd_buf [dir⃗],dest(p⃗ + dir⃗),tag(t⃗));
11     endforeach
12     foreach dir⃗ ∈ R_{p⃗,t⃗} do
13       MPI_Irecv(recv_buf [dir⃗],src(p⃗ − dir⃗),tag(t⃗));
14     endforeach
15     if 0 ≤ tile_{N+1} ≤ ⌈Z/z⌉ − 1 then
16       Compute(tile⃗);
17     endif
18     MPI_Waitall;
19     foreach dir⃗ ∈ R_{p⃗,t⃗} do
20       Unpack(recv_buf [dir⃗],group + 1,p⃗,t⃗);
21     endforeach
22     #pragma omp barrier
23   endforeach
24 end
```

και να λάβει δεδομένα από όλες τις κατευθύνσεις $j \in \mathbb{R}_{\vec{p}}$, για τις οποίες ισχύει

$$t_j = T_j - 1$$

Καθώς όλα τα νήματα καλούν ρουτίνες ανταλλαγής μηνυμάτων βάσει των συνόλων $\mathbb{S}_{\vec{p},\vec{t}}$ και $\mathbb{R}_{\vec{p},\vec{t}}$, η παράμετρος `tag()` αναπαριστά το ρόλο της ετικέτας του μηνύματος για τη διάκριση ζευγών μηνυμάτων. Συγκεκριμένα, ας υποθέσουμε ότι κάθε διεργασία εκκινεί T νήματα σε τοπολογία $\prod_{i=1}^{N} T_i$, οπότε το τυχόν νήμα έχει αναγνωριστικό της μορφής $(t_1, \ldots, t_N)$ με $0 \leq t_i \leq T_i - 1$, $1 \leq i \leq N$. Βάσει του σχήματος απεικόνισης των νημάτων που επιλέγουμε, το νήμα $(t_1, \ldots, t_j = 0, \ldots, t_N)$ μιας μη συνοριακής διεργασίας πρέπει να αποστείλει δεδομένα που απαιτούνται για υπολογισμούς του νήματος



Σχήμα 5.3: *Επικοινωνία μεταξύ διεργασιών στο multiple υβριδικό μοντέλο χονδρού κόκκου. Ο αλγόριθμος απεικονίζεται σε 4 διεργασίες σε τοπολογία 2×2, όπου κάθε διεργασία εκκινεί 6 νήματα σε τοπολογία 3×2. Η επικοινωνία διεξάγεται θεωρητικά μόνο μεταξύ των 4 διεργασιών, αλλά η χρήση διαφορετικών ετικετών στα μηνύματα επιτρέπει έμμεσα την υλοποίηση επικοινωνίας μεταξύ των νημάτων.*

$(t_1, \ldots, t_j = T_j - 1, \ldots, t_N)$ της γειτονικής διεργασίας κατά τη διεύθυνση j. Η αντιστοιχία μεταξύ αυτών των μηνυμάτων μπορεί να επιτευχθεί αν επιλέξουμε κατάλληλη ετικέτα που να μπορεί να υποδηλώσει την ταυτότητα του νήματος αποστολέα και να υποδείξει το αντίστοιχο νήμα παραλήπτη. Κάτι τέτοιο μπορεί να επιτευχθεί π.χ. αν αντιστοιχίσουμε το διάνυσμα $(t_1, \ldots, 0, \ldots, t_N)$ του αναγνωριστικού αποστολέα σε αριθμό Tag αριθμητικού συστήματος βάσης T. Πράγματι, αφού $0 \leq t_i \leq T_i - 1$, θα υπάρχει αμφιμονοσήμαντη αντιστοιχία μεταξύ του βαθμωτού αριθμού ετικέτας και του διανυσματικού αναγνωριστικού του νήματος. Έτσι, επιλέγουμε

$$Tag = \sum_{i=1}^{N} t_i T^{N-i} \tag{5.1}$$

Για παράδειγμα, στο σχήμα 5.3 απεικονίζεται η πολυνηματική επικοινωνία στο multiple υβριδικό μοντέλο χονδρού κόκκου για 4 διεργασίες και 6 νήματα ανά διεργασία. Το νήμα $(1, 0)$ της διεργασίας 1 θα καλέσει συνάρτηση αποστολής κατά τη διεύθυνση j_2 (αφού η δεύτερη συνιστώσα του νήματος είναι μηδενική) και θα χρησιμοποιήσει ως ετικέτα την $1 \times 6^1 + 0 \times 6^0 = 6$. Αντίστοιχα, το νήμα $(1, 1)$

της διεργασίας 2 θα καλέσει ρουτίνα λήψης κατά τη διεύθυνση j_2 (αφού η δεύτερη συνιστώσα ισούται με $T_2 - 1 = 1$) με την ίδια ετικέτα, επιτυγχάνοντας έτσι έμμεσα την αντιστοιχία των λειτουργιών αποστολής και λήψης. Πράγματι, παρατηρούμε ότι μεταξύ των διεργασιών 1 και 2 ανταλλάσσονται συνολικά τρία μηνύματα με ισάριθμες διαφορετικές ετικέτες (0, 6 και 12), επιτρέποντας έτσι τη λήψη των δεδομένων από το κατάλληλο νήμα στη διεργασία παραλήπτη. Τέλος, θα πρέπει να σημειωθεί πως η προτεινόμενη μέθοδος multiple επικοινωνίας είναι γενική και μπορεί να εφαρμοστεί στην N-διάστατη περίπτωση multiple υβριδικής παραλληλοποίησης χονδρού κόκκου. Παρότι για τον υπολογισμό μικρότερων τιμών ετικετών θα μπορούσε να χρησιμοποιηθεί περισσότερο αποδοτική μέθοδος κωδικοποίησης, επιλέξαμε τη συγκεκριμένη λόγω της απλότητάς της και της χαμηλής καθυστέρησης που επιφέρει στο χρόνο εκτέλεσης.

Το multiple υβριδικό μοντέλο χονδρού κόκκου εμφανίζεται σπανιότερα στη διεθνή βιβλιογραφία λόγω της περιορισμένης ύπαρξης υλοποιήσεων MPI που παρέχουν πολυνηματική υποστήριξη επιπέδου `MPI_THREAD_MULTIPLE`. Στην παρούσα διατριβή, κάθε αόριστη αναφορά στο υβριδικό μοντέλο χονδρού κόκκου θα υπονοεί πάντοτε τη funneled εκδοχή, ενώ αντίθετα η multiple υβριδική υλοποίηση θα μνημονεύεται μόνο ρητά.

5.2 Εξισορρόπηση Φορτίου μεταξύ των Νημάτων

Κατά την επισκόπηση των υβριδικών μοντέλων παρατηρήσαμε πως το υβριδικό μοντέλο χονδρού κόκκου επιτρέπει την επικάλυψη της απαραίτητης επικοινωνίας με ωφέλιμους υπολογισμούς, επιτυγχάνοντας θεωρητικά καλύτερη επίδοση από το αντίστοιχο μοντέλο λεπτού κόκκου. Όμως, καθώς οι υπάρχουσες υλοποιήσεις βιβλιοθηκών ανταλλαγής μηνυμάτων επιτρέπουν κατά κανόνα μόνο στο πρωτεύον νήμα να πραγματοποιεί κλήσεις επικοινωνίας με απομακρυσμένες διεργασίες, το υβριδικό μοντέλο χονδρού κόκκου χαρακτηρίζεται εγγενώς από αναποτελεσματική εξισορρόπηση του φορτίου μεταξύ των νημάτων. Πράγματι, αν εφαρμόσουμε στη funneled προσέγγιση ισοκατανομή του συνολικού υπολογιστικού φορτίου της διεργασίας μεταξύ των διαθέσιμων νημάτων αυτής, το πρωτεύον νήμα θα επιβαρυνθεί αναπόφευκτα περισσότερο από τα υπόλοιπα νήματα, αφού πέραν του υπολογισμού θα πρέπει να διεκπεραιώσει και τις ανάγκες επικοινωνίας για ανταλλαγή μηνυμάτων με άλλες διεργασίες. Ακόμα κι αν διασφαλίζεται πλήρης πολυνηματική υποστήριξη από τη μεριά της βιβλιοθήκης ανταλλαγής μηνυμάτων (multiple εκδοχή), η ελεύθερη κλήση ρουτινών επικοινωνίας σε πολυνηματικό περιβάλλον απαιτεί τη χρήση κλειδωμάτων και κρίσιμων περιοχών στην υλοποίηση της βιβλιοθήκης ανταλλαγής μηνυμάτων, που επιβαρύνουν και περιορίζουν την απόδοση του προγράμματος. Κατά συνέπεια, η επίδοση του υβριδικού προγραμματιστικού μοντέλου χονδρού κόκκου συναρτάται άμεσα με την υλοποίηση αποδοτικού σχήματος εξισορρόπησης φορτίου μεταξύ των νημάτων, που να ανταπεξέρχεται στους

περιορισμούς που θέτει η βιβλιοθήκη ανταλλαγής μηνυμάτων, αποφεύγοντας παράλληλα και τις επιβαρύνσεις που συνεπάγεται η απαίτηση πλήρους πολυνηματικής υποστήριξης. Σε αντίθετη περίπτωση, αν δηλαδή ανατεθούν αδιακρίτως ισομερή τμήματα υπολογισμού σε όλα τα νήματα, το πρωτεύον νήμα θα αναλάβει τελικά αναπόφευκτα μεγαλύτερο συνολικό φορτίο εκτέλεσης, επιδρώντας έτσι δυσμενώς στο συνολικό χρόνο εκτέλεσης του προγράμματος.

Η χρονοδρομολόγηση υπερεπιπέδων καθιστά εφικτό ένα πολύ πιο αποδοτικό σχήμα εξισορρόπησης φορτίου μεταξύ των νημάτων: εφόσον σε κάθε υπερεπίπεδο οι υπολογισμοί που πραγματοποιούνται είναι ανεξάρτητοι της επικοινωνίας που λαμβάνει χώρα, καθώς η τελευταία αφορά λήψη δεδομένων για το επόμενο υπερεπίπεδο και αποστολή δεδομένων που υπολογίστηκαν κατά το προηγούμενο υπερεπίπεδο, υπάρχει δυνατότητα κατάλληλης αναδιανομής των υπολογισμών αυτών μεταξύ των νημάτων. Έτσι, το πρωτεύον νήμα θα μπορούσε να αναλάβει ένα σχετικά μικρότερο υπολογιστικό φορτίο συγκριτικά με τα υπόλοιπα νήματα, αποσκοπώντας τελικά σε μια ομοιόμορφη κατανομή του συνολικού φορτίου της παράλληλης εκτέλεσης (υπολογισμός+επικοινωνία) μεταξύ όλων των διαθέσιμων νημάτων.

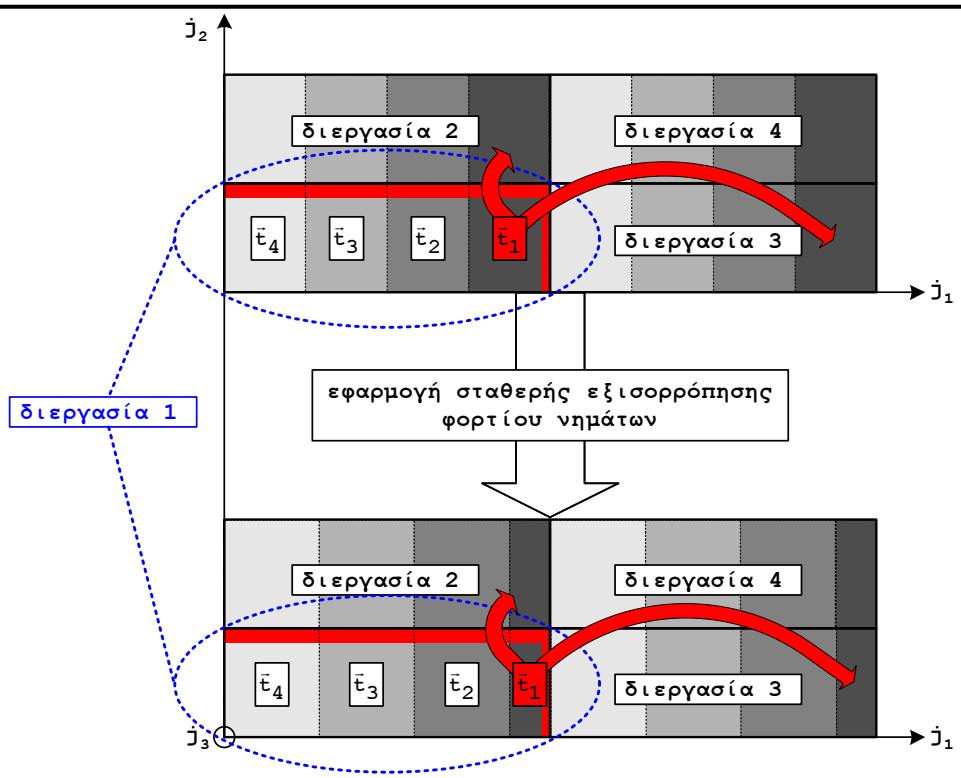
Στο πλαίσιο της παρούσας εργασίας υλοποιήσαμε σχήματα τόσο *στατικής* όσο και *δυναμικής* εξισορρόπησης φορτίου. Σύμφωνα με το πρώτο, η εξισορρόπηση φορτίου εφαρμόζεται κατά το χρόνο μεταγλώττισης (compile time) βάσει θεωρητικής εκτίμησης της συμπεριφοράς του συστήματος και ιδιαίτερα του σχετικού κόστους υπολογισμού και επικοινωνίας. Για το σκοπό αυτό περιοριστήκαμε στη μελέτη της επίδρασης μόνο θεμελιωδών συστημικών χαρακτηριστικών, όπως ο μέσος χρόνος εκτέλεσης ανά επανάληψη, η αρχική καθυστέρηση και ο ρυθμός παροχής δεδομένων του δικτύου διασύνδεσης, ώστε να διατηρήσουμε την προτεινόμενη μεθοδολογία κατά το δυνατόν απλή και εφαρμόσιμη. Εναλλακτικά, η δυναμική εξισορρόπηση φορτίου υιοθετεί μια περισσότερο παρεμβατική προσέγγιση, κατά την οποία τα σχετικά κόστη υπολογισμού και επικοινωνίας του αλγορίθμου δειγματοληπτούνται κατά το χρόνο εκτέλεσης (run-time), και δεν είναι αναγκαίο να γίνουν a priori στατικές παραδοχές ή εκτιμήσεις.

Στις ακόλουθες ενότητες θα αναφερθούμε αναλυτικά στα συνολικά τρία προτεινόμενα σχήματα εξισορρόπησης φορτίου των νημάτων, ήτοι δύο σχήματα στατικής εξισορρόπησης (σταθερό και μεταβλητό), καθώς και ένα σχήμα δυναμικής εξισορρόπησης φορτίου.

5.2.1 Στατική Εξισορρόπηση Φορτίου

Στο πλαίσιο της παρούσας έρευνας υλοποιήθηκαν δύο σχήματα στατικής εξισορρόπησης φορτίου μεταξύ των νημάτων για το υβριδικό μοντέλο χονδρού κόκκου. Το πρώτο απαιτεί τον υπολογισμό ενός σταθερού συντελεστή, που εφαρμόζεται από κοινού σε όλες τις διεργασίες. Ο συντελεστής αυτός καθορίζει το ποσοστό επί του ισοκατανεμημένου υπολογιστικού φορτίου, που θα πρέπει να αναλάβει το

πρωτεύον νήμα, ενώ το λοιπό υπολογιστικό φορτίο διαμοιράζεται ομοιόμορφα στα υπόλοιπα νήματα. Για παράδειγμα, συντελεστής εξισορρόπησης 100% συνεπάγεται ισοκατανομή του υπολογιστικού φορτίου σε όλα τα διαθέσιμα νήματα, ενώ ένας συντελεστής 50% υποδηλώνει ότι το πρωτεύον νήμα θα πρέπει να αναλάβει το μισό υπολογιστικό φορτίο σε σχέση με εκείνο που θα του αναλογούσε σε περίπτωση ομοιόμορφης κατανομής των υπολογισμών. Θα αναφερόμαστε στο σχήμα αυτό ως *σταθερή εξισορρόπηση φορτίου*. Για τον υπολογισμό ενός κατάλληλου συντελεστή εξισορρόπησης φορτίου θα θεωρήσουμε μια μη συνοριακή διεργασία, που πρέπει να επικοινωνήσει προς όλες τις διαστάσεις του N-διάστατου χώρου των διεργασιών, ώστε να αποστείλει δεδομένα προς όλες τις γειτονικές της διεργασίες. Για μια τέτοια διεργασία καθορίζουμε το συντελεστή εξισορρόπησης φορτίου για το πρωτεύον νήμα, που επιτυγχάνει την εξίσωση των συνολικών χρόνων εκτέλεσης υπερκόμβου για όλα τα νήματα.



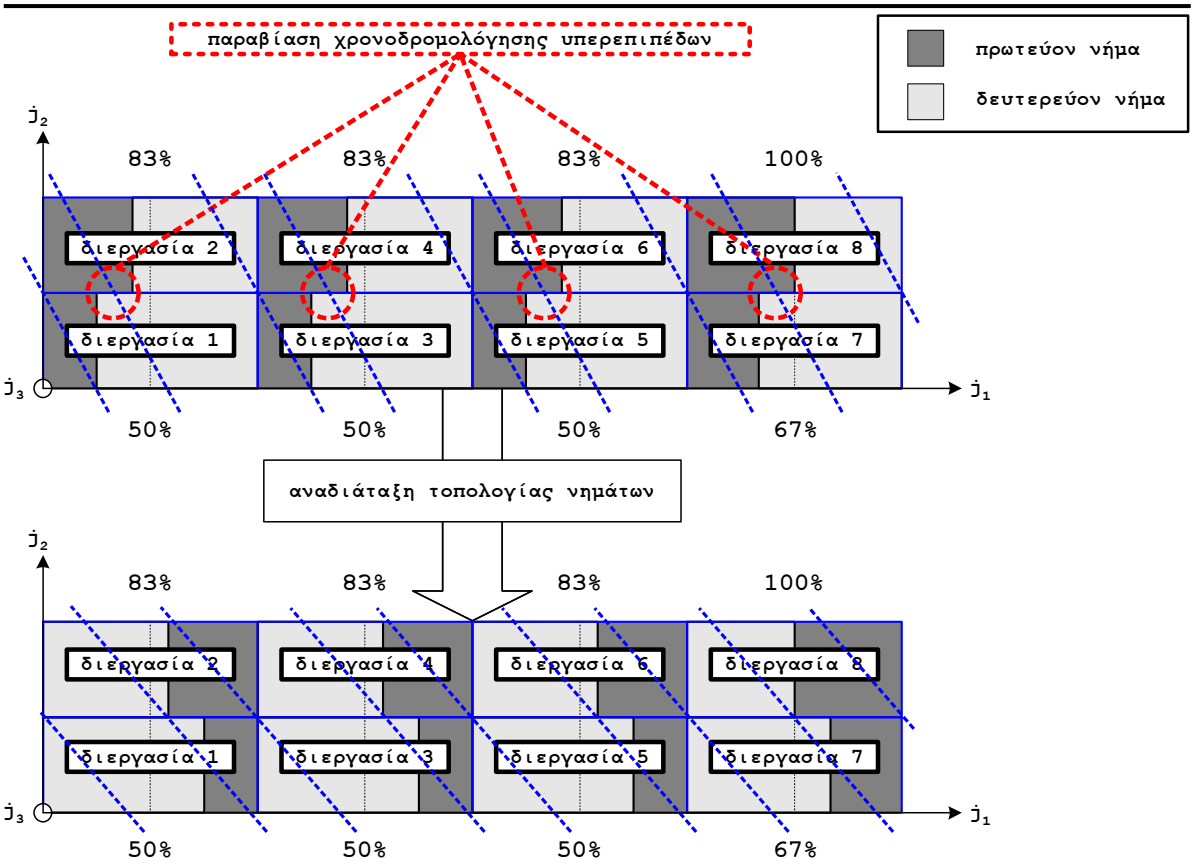
Σχήμα 5.4: *Σταθερή εξισορρόπηση φορτίου για 4 διεργασίες σε 2×2 τοπολογία και 4 νήματα ανά διεργασία σε 4×1 τοπολογία. Το πρωτεύον νήμα $\vec{t}_1$ αναλαμβάνει μικρότερο φορτίο από τα υπόλοιπα για να εξισωθούν οι συνολικοί χρόνοι υπολογισμού και επικοινωνίας όλων των νημάτων. Παρατηρούμε ότι ο ίδιος συντελεστής εξισορρόπησης φορτίου εφαρμόζεται σε όλες τις διεργασίες, παρότι λ.χ. η διεργασία 4 δεν αποστέλλει δεδομένα προς άλλη διεργασία.*

Η εφαρμογή σταθερής εξισορρόπησης φορτίου απεικονίζεται στο σχήμα 5.4. Στο πάνω μέρος απεικονίζεται το απλό υβριδικό μοντέλο χονδρού κόκκου για την περίπτωση 4 διεργασιών και τρισδιά-

στατου αλγορίθμου, όπου κάθε διεργασία εκκινεί 4 νήματα. Παρατηρούμε ότι σε κάθε διεργασία το πρωτεύον νήμα $\vec{t_1}$ παρουσιάζεται πιο επιβαρυμένο από τα υπόλοιπα 3 νήματα, αφού εκτός του υπολογιστικού του φορτίου οφείλει να αναλάβει και την επικοινωνία με το ομότιμο νήμα κάθε γειτονικής διεργασίας. Κατά την εφαρμογή σταθερής εξισορρόπησης φορτίου, σε όλες τις διεργασίες το εκάστοτε πρωτεύον νήμα αναλαμβάνει ένα σχετικά μικρότερο όγκο υπολογισμών, με απώτερο στόχο την εξίσωση των συνολικών χρόνων εκτέλεσης των νημάτων. Ο λόγος για τον οποίο πραγματοποιείται απεικόνιση του υπερκόμβου της διεργασίας στα διαθέσιμα νήματα με τη δοθείσα ακολουθία $\vec{t_4} - \vec{t_1}$ έχει να κάνει με ζητήματα ομοιομορφίας προς τα υπόλοιπα σχήματα εξισορρόπησης φορτίου (μεταβλητό, δυναμικό), και θα αναλυθεί στη συνέχεια.

Το δεύτερο σχήμα εξισορρόπησης φορτίου χαρακτηρίζεται ως *μεταβλητή εξισορρόπηση φορτίου*. Διαφοροποιώντας ελαφρά τη συλλογιστική του σχήματος σταθερής εξισορρόπησης φορτίου, παρατηρούμε ότι οι συνοριακές διεργασίες επιβαρύνονται αναλογικά λιγότερο με επικοινωνία μέσω ανταλλαγής μηνυμάτων, κάτι που δεν λαμβάνεται υπόψη στο σταθερό σχήμα εξισορρόπησης. Για το σκοπό αυτό, το μεταβλητό σχήμα εξισορρόπησης φορτίου αγνοεί για κάθε διεργασία τις κατευθύνσεις επικοινωνίας που τέμνουν τα όρια του καθολικού χώρου επαναλήψεων του αλγορίθμου, καθώς αυτές ουσιαστικά δεν αντιστοιχούν σε αποστολή μηνυμάτων και κατά συνέπεια δεν επιβαρύνουν την επικοινωνία της εν λόγω διεργασίας. Έτσι, για κάθε διεργασία υπολογίζεται ένας διαφορετικός συντελεστής εξισορρόπησης φορτίου των νημάτων, αφού το πρωτεύον νήμα πρέπει να ελαφρύνεται υπολογιστικά λιγότερο ή περισσότερο για συνοριακές και μη συνοριακές διεργασίες, αντίστοιχα. Επιπλέον, το SPMD προγραμματιστικό μοντέλο διευκολύνει την εξισορρόπηση του φορτίου των νημάτων με εφαρμογή διαφορετικού συντελεστή για κάθε διεργασία, όπως απαιτεί το σχήμα μεταβλητής εξισορρόπησης.

Στο σχήμα 5.5 απεικονίζεται η εφαρμογή του σχήματος μεταβλητής εξισορρόπησης φορτίου για 8 διεργασίες και 2 νήματα ανά διεργασία. Στην περίπτωση αυτή υπολογίζεται διαφορετικός συντελεστής εξισορρόπησης για συνοριακές και μη συνοριακές διεργασίες, αφού βλέπουμε ότι σε όλες τις μη συνοριακές διεργασίες στο πρωτεύον νήμα ανατίθεται το 1/4 του συνολικού υπολογιστικού φορτίου της διεργασίας (συντελεστής εξισορρόπησης 50%), ενώ στις συνοριακές επιλέγεται συντελεστής 67%, 83% ή ακόμα και καθόλου εξισορρόπηση φορτίου (συντελεστής 100%). Παρατηρούμε ότι στην περίπτωση που επιλέξουμε ακολουθία νημάτων $\vec{t_1} - \vec{t_2}$ (άνω σχήμα) παραβιάζεται η χρονοδρομολόγηση υπερεπιπέδων, αφού λ.χ. κατά το δεύτερο υπερεπίπεδο το πρωτεύον νήμα της διεργασίας 2 δεν διαθέτει όλα τα συνοριακά δεδομένα που χρειάζεται για τους υπολογισμούς του, καθώς κάποια από αυτά θα υπολογιστούν στο ίδιο υπερεπίπεδο από το δευτερεύον νήμα της διεργασίας 1. Αντίθετα, κατά την αντίστροφη ακολουθία απεικόνισης νημάτων $\vec{t_2} - \vec{t_1}$, γίνεται σε κάποιες περιπτώσεις πρώιμη αποστολή δεδομένων, όπως π.χ. αυτά που υπολογίζει το δευτερεύον νήμα της διεργασίας 1 κατά το πρώτο υπερεπίπεδο και αποστέλλονται κατά το δεύτερο υπερεπίπεδο, ενώ στην πραγματικότητα απαιτούνται σε



Σχήμα 5.5: *Μεταβλητή εξισορρόπηση φορτίου για 8 διεργασίες σε 4×2 τοπολογία και 2 νήματα ανά διεργασία σε 2×1 τοπολογία. Το πρωτεύον νήμα αναλαμβάνει μικρότερο φορτίο από το δευτερεύον, αλλά γενικά οι συντελεστές εξισορρόπησης φορτίου αυξάνουν σε συνοριακές διεργασίες που δεν αποστέλλονται δεδομένα προς μία ή περισσότερες διευθύνσεις (π.χ. στη διεργασία 8 δεν εφαρμόζεται εξισορρόπηση φορτίου και οι υπολογισμοί ισοκατανέμονται μεταξύ των νημάτων). Παρατηρούμε ότι η προσέγγιση του άνω σχήματος παραβιάζει τη χρονοδρομολόγηση υπερεπιπέδων, καθώς π.χ. στο δεύτερο υπερεπίπεδο το πρωτεύον νήμα της διεργασίας 2 θα χρειαζόταν τιμές που θα υπολογίσει στο ίδιο υπερεπίπεδο το δευτερεύον νήμα της διεργασίας 1.*

υπολογισμούς του πρωτεύοντος νήματος της διεργασίας 2 κατά το τρίτο υπερεπίπεδο. Η ακολουθία νημάτων $\vec{t}_2 - \vec{t}_1$ διατηρεί την εγκυρότητα της χρονοδρομολόγησης υπερεπιπέδων, καθώς λόγω του ότι οι συντελεστές εξισορρόπησης μειώνονται για συνοριακές διεργασίες, ενδέχεται να αποστέλλονται δεδομένα είτε έγκαιρα είτε πρόωρα, αλλά σε καμία περίπτωση καθυστερημένα.

Συνοψίζοντας τα παραπάνω, σε όλες τις περιπτώσεις υβριδικού μοντέλου παραλληλοποίησης αλγορίθμου φωλιασμένων βρόχων διάστασης $N+1$ που εφαρμόζεται κάποιο σχήμα εξισορρόπησης φορτίου,

θεωρείται ότι τα T νήματα αξιοποιούνται σε μια *γραμμική* N-διάστατη τοπολογία της μορφής

$$1 \times \ldots 1 \times T \times 1 \times \cdots \times 1$$

και εξισορρόπηση φορτίου εφαρμόζεται κατά τη διάσταση της τοπολογίας που εμφανίζεται η συνιστώσα T, και στην οποία στο εξής θα αναφερόμαστε ως *διάσταση εξισορρόπησης*. Επιπλέον, κατά τη διάσταση αυτή θεωρούμε την ακολουθία $\vec{t_T} - \vec{t_1}$ των T νημάτων, ώστε να μην παραβιάζεται η χρονοδρομολόγηση υπερεπιπέδων.

Αμφότερα τα προτεινόμενα σχήματα εξισορρόπησης φορτίου μπορούν να βασιστούν στο ακόλουθο λήμμα για τον υπολογισμό του μοναδικού συντελεστή εξισορρόπησης (περίπτωση σταθερής εξισορρόπησης) ή των διαφορετικών συντελεστών (μεταβλητή εξισορρόπηση φορτίου):

Λήμμα 5.1. *Έστω $X_1 \times \cdots \times X_N \times Z$ ο χώρος επαναλήψεων ενός επαναληπτικού αλγορίθμου διάστασης $N+1$ με εξαρτήσεις δεδομένων τις $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$. Έστω P το πλήθος των διεργασιών που χρησιμοποιούνται για την απεικόνιση του παράλληλου αλγορίθμου και T το αντίστοιχο πλήθος των νημάτων που διατίθενται ανά διεργασία, υπό το υβριδικό μοντέλο χονδρού κόκκου. Ο συνολικός χρόνος εκτέλεσης του παράλληλου αλγορίθμου ελαχιστοποιείται αν στο πρωτεύον νήμα ανατεθεί ένα ποσοστό $\frac{bal}{T}$ του συνολικού υπολογιστικού φορτίου της διεργασίας, όπου*

$$bal = 1 - \frac{T-1}{t_{comp}\left(\frac{Xz}{P}\right)} \sum_{\substack{i=1 \\ i \in \mathbb{S}_{\vec{p}}}}^{N} t_{comm}\left(\frac{d_i P_i X z}{X_i P}\right) \tag{5.2}$$

$t_{comp}(x)$ *ο χρόνος υπολογισμού x επαναλήψεων*

$t_{comm}(x)$ *ο χρόνος μετάδοσης ενός μηνύματος x στοιχείων*

z *το ύψος του υπερκόμβου για κάθε βήμα εκτέλεσης του παράλληλου αλγορίθμου*

$\mathbb{S}_{\vec{p}}$ *οι έγκυρες κατευθύνσεις αποστολής δεδομένων της διεργασίας $\vec{p}$*

X *ίσο με* $\prod_{i=1}^{N} X_i$

Απόδειξη. Χάριν απλότητας και χωρίς βλάβη της γενικότητας, υποθέτουμε ότι όλες οι διαιρέσεις οδηγούν σε ακέραιο πηλίκο, ώστε να μην περιπλέξουμε την ανάλυσή μας με την εισαγωγή τελεστών $ceil$ και $floor$. Έτσι, σε κάθε διεργασία ανατίθεται η εκτέλεση $\frac{Z}{z}$ το πλήθος υπερκόμβων, καθένας από τους οποίους αποτελείται από $\frac{Xz}{P}$ το πλήθος επαναλήψεις. Επιπλέον, καθώς το πρωτεύον νήμα αναλαμβάνει την εκτέλεση ενός ποσοστού $\frac{bal}{T}$ του υπολογιστικού φορτίου της διεργασίας (ή ισοδύναμα, $bal\%$ του επί ισοκατανεμημένου φορτίου), σε κάθε ένα από τα υπόλοιπα $T-1$ θα ανατεθεί η εκτέλεση ποσοστού

$\frac{T-bal}{T(T-1)}$ του υπολογιστικού φορτίου της διεργασίας. Επειδή μόνο το πρωτεύον νήμα θα υλοποιήσει την επικοινωνία μεταξύ των διεργασιών μέσω ανταλλαγής μηνυμάτων, θα πρέπει να λαμβάνει μέριμνα τόσο για τα δικά του συνοριακά δεδομένα επικοινωνίας όσο και για τα δεδομένα επικοινωνίας των υπολοίπων νημάτων. Ο συνολικός χρόνος εκτέλεσης του αλγορίθμου μπορεί να προσεγγιστεί ως το γινόμενο του συνολικού αριθμού υπολογιστικών βημάτων επί το χρόνο εκτέλεσης T_{tile}, που απαιτείται για τον υπολογισμό ενός υπερκόμβου σε κάθε βήμα. Ισχύει

$$T_{tile} = max\left(T_{tile}^{m}, T_{tile}^{o}\right) \tag{5.3}$$

όπου

$$T_{tile}^{m} = t_{comp}\left(\frac{bal}{T}\frac{Xz}{P}\right) + \sum_{\substack{i=1 \\ i\in\mathbb{S}_{\vec{p}}}}^{N} t_{comm}\left(\frac{d_i P_i Xz}{X_i P}\right) \tag{5.4}$$

ο χρόνος εκτέλεσης υπερκόμβου του πρωτεύοντος νήματος και

$$T_{tile}^{o} = t_{comp}\left(\frac{T-bal}{T(T-1)}\frac{Xz}{P}\right) \tag{5.5}$$

ο χρόνος εκτέλεσης υπερκόμβου ενός μη πρωτεύοντος νήματος.

Η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του παράλληλου αλγορίθμου ισοδυναμεί πρακτικά με την ελαχιστοποίηση του χρόνου εκτέλεσης υπερκόμβου, αφού ο αριθμός των συνολικών βημάτων εξαρτάται κυρίως από το πλήθος των υπερκόμβων ανά διεργασία και δεν επηρεάζεται από την κατανομή του φορτίου μεταξύ των νημάτων. Ο χρόνος εκτέλεσης υπερκόμβου, όπως προκύπτει από την (5.3), ελαχιστοποιείται για

$$T_{tile}^{m} = T_{tile}^{o} \tag{5.6}$$

Πράγματι, αν αυτό δεν συμβαίνει, αν δηλαδή $T_{tile}^{m} \neq T_{tile}^{o}$, μπορεί πάντα να επιτευχθεί μια πιο αποτελεσματική εξισορρόπηση φορτίου με την ανάθεση επιπλέον υπολογιστικού φορτίου στα λιγότερο επιφορτισμένα νήματα. Υποθέτοντας κατά προσέγγιση ότι ο χρόνος υπολογισμού t_{comp} είναι γραμμική συνάρτηση του πλήθους των επαναλήψεων, δηλαδή ότι ισχύει $t_{comp}(ax) = at_{comp}(x)$, ο συνδυασμός των (5.4), (5.5) και (5.6) δίνει την (5.2).

□

Στην πράξη, για να αξιοποιήσουμε τον υπολογιζόμενο συντελεστή εξισορρόπησης φορτίου με χρήση του λήμματος 5.1 (πραγματικός αριθμός) για τον καθορισμό των ακεραίων διαστάσεων των υπερκόμβων των νημάτων εργαζόμαστε ως εξής: έστω bal ο υπολογιζόμενος συντελεστής εξισορρόπησης φορτίου από την (5.2) και $s_1 \times \cdots \times s_N \times z$ οι διαστάσεις του υπερκόμβου που αναλαμβάνει κάθε νήμα

κατά την ομοιόμορφη διαμέριση του υπερκόμβου της διεργασίας, με z το ελεύθερο ύψος υπερκόμβου που καθορίζει εν τέλει τον κόκκο παραλληλισμού. Αν $\prod_{i=1}^{N} P_i$ η τοπολογία των διεργασιών, $\prod_{i=1}^{N} T_i$ η τοπολογία των νημάτων και $\prod_{i=1}^{N} X_i Z$ ο χώρος επαναλήψεων του αλγορίθμου διάστασης $N+1$, θα ισχύει κατά την ισοκατανομή (χωρίς εξισορρόπηση φορτίου)

$$s_i = \left\lceil \frac{X_i}{P_i T_i} \right\rceil, 1 \leq i \leq N \tag{5.7}$$

Αν υποθέσουμε ότι bal_dim είναι η διάσταση εξισορρόπησης, δηλαδή $T_{bal\_dim} = T$ και $T_i = 1$ για $i \neq bal\_dim$, τότε οι διαστάσεις του υπερκόμβου διαφοροποιούνται για ένα μη πρωτεύον νήμα κατά τη σχέση

$$s_i^o = \begin{cases} \left\lceil \frac{X_i}{P_i T_i} \right\rceil, & i \neq bal\_dim \\ (int)\frac{1}{T-1}\left(\left\lceil \frac{X_i}{P_i} \right\rceil - \frac{bal}{T} \left\lceil \frac{X_i}{P_i} \right\rceil \right), & i = bal\_dim \end{cases} \tag{5.8}$$

όπου με τον τελεστή (int) συμβολίζεται η πράξη της στρογγυλοποίησης ενός πραγματικού αριθμού στον πλησιέστερο ακέραιο. Αντίστοιχα, για το πρωτεύον νήμα οι διαστάσεις του υπερκόμβου διαμορφώνονται σύμφωνα με τη σχέση

$$s_i^m = \begin{cases} \left\lceil \frac{X_i}{P_i T_i} \right\rceil, & i \neq bal\_dim \\ \left\lceil \frac{X_i}{P_i} \right\rceil - (T-1)s_i^o, & i = bal\_dim \end{cases} \tag{5.9}$$

Έτσι, το πρωτεύον νήμα θα αναλάβει την εκτέλεση υπερκόμβων διαστάσεων $s_1^m \times \cdots \times s_N^m \times z$, ενώ κάθε ένα από τα υπόλοιπα νήματα εκτελεί τους υπολογισμούς που περικλείονται από υπερκόμβους μεγέθους $s_1^o \times \cdots \times s_N^o \times z$. Ο προσδιορισμός των s_i^m και s_i^o μέσω των (5.9) και (5.8) επιτρέπει αφενός την ισοκατανομή του υπολογιστικού φορτίου για τα μη πρωτεύοντα νήματα, και αφετέρου την εξισορρόπηση του φορτίου του πρωτεύοντος νήματος κοντά στον επιθυμητό συντελεστή bal.

5.2.2 Δυναμική Εξισορρόπηση Φορτίου

Εναλλακτικά της εφαρμογής στατικού σχήματος εξισορρόπησης του φορτίου βάσει θεωρητικής μοντελοποίησης της συμπεριφοράς υλικού και λογισμικού και της εκτίμησης του σχετικού κόστους υπολογισμού και επικοινωνίας, είναι δυνατό να υπολογίζουμε τους συντελεστές εξισορρόπησης φορτίου δυναμικά κατά το χρόνο εκτέλεσης. Αρχικά, θα μπορούσαμε να μετράμε τους χρόνους υπολογισμού και επικοινωνίας των νημάτων για ένα μικρό διάστημα της παράλληλης εκτέλεσης του προγράμματος. Στη συνέχεια, οι χρόνοι αυτοί μπορούν να αξιοποιηθούν για τη διόρθωση της τιμής των συντελεστών εξισορρόπησης φορτίου και την εφαρμογή ενός ενδεχομένως πιο αποδοτικού σχήματος εξισορρόπησης φορτίου.

Πιο συγκεκριμένα, έστω ότι εφαρμόζουμε έναν αρχικό συντελεστή bal για την εξισορρόπηση του φορτίου μεταξύ T νημάτων κάποιας διεργασίας. Η τιμή του αρχικού συντελεστή bal μπορεί να υπολογιστεί με χρήση κάποιου σχήματος στατικής εξισορρόπησης φορτίου, όπως αυτά που παρουσιάστηκαν στην ενότητα 5.2.1. Υποθέτοντας παράλληλη εκτέλεση με χρήση P διεργασιών, ο στόχος μας είναι η δειγματοληψία των μερικών χρόνων υπολογισμού και επικοινωνίας για τουλάχιστον PT υπερεπίπεδα, ώστε το κυματομέτωπο εκτέλεσης να φτάσει μέχρι την πιο απομακρυσμένη διεργασία, καθώς επιθυμούμε να καταγράψουμε τη συμπεριφορά του προγράμματος πρωτίστως όταν όλα τα στάδια της σωλήνωσης είναι πλήρη. Στο εξής, θα αναφερόμαστε στην περίοδο κατά την οποία καταγράφουμε τους χρόνους υπολογισμού και επικοινωνίας του παράλληλου προγράμματος με τον όρο *περίοδο δειγματοληψίας*. Βάσει των χρονικών μετρήσεων που λαμβάνονται κατά την περίοδο δειγματοληψίας, μπορούμε από τον αρχικό συντελεστή εξισορρόπησης φορτίου bal να υπολογίσουμε έναν πιο κατάλληλο συντελεστή bal' με χρήση του ακόλουθου λήμματος:

Λήμμα 5.2. *Έστω συντελεστής bal για την εξισορρόπηση του φορτίου T νημάτων και t^m_{comp}, t^m_{comm} οι μέσοι χρόνοι υπολογισμού και επικοινωνίας υπερκόμβου για το πρωτεύον νήμα, όπως καταγράφονται κατά το χρόνο εκτέλεσης. Ένας περισσότερο αποδοτικός συντελεστής εξισορρόπησης φορτίου bal' μπορεί να υπολογιστεί με χρήση της ακόλουθης σχέσης:*

$$bal' = 1 - bal\frac{T-1}{T}\frac{t^m_{comm}}{t^m_{comp}} \tag{5.10}$$

Απόδειξη. Υποθέτουμε ότι η αρχική εξισορρόπηση φορτίου που επιτυγχάνεται με την εφαρμογή του συντελεστή bal είναι μη βέλτιστη, δηλαδή για τους χρόνους εκτέλεσης υπερκόμβου πρωτεύοντος (T^m_{tile}) και μη πρωτεύοντος νήματος (T^o_{tile}) ισχύει

$$\begin{aligned} T^m_{tile} &\neq T^o_{tile} \Rightarrow \\ t^m_{comp} + t^m_{comm} &\neq t^o_{comp} \end{aligned}$$

Η αστοχία της στατικής εξισορρόπησης φορτίου πιθανότατα οφείλεται σε ανακριβή θεωρητική μοντελοποίηση της υφιστάμενης αρχιτεκτονικής, καθώς και στις διάφορες προσεγγίσεις που θεωρήσαμε στην εκτίμηση της προγραμματιστικής επίδοσης (π.χ. γραμμικό κόστος υπολογισμού, που αγνοεί τα φαινόμενα τοπικότητας αναφοράς στην ιεραρχία μνήμης). Ο στόχος μας είναι ο προσδιορισμός ενός νέου συντελεστή bal', που ιδεατά θα εξισώνει το μέσο χρόνο εκτέλεσης υπερκόμβου για όλα τα νήματα, δηλαδή

$$t'^m_{comp} + t'^m_{comm} = t'^o_{comp} \tag{5.11}$$

Υποθέτουμε ότι ο χρόνος που απαιτείται για τον υπολογισμό ενός υπερκόμβου είναι ανάλογος προς το

πλήθος των επαναλήψεων που περικλείονται από το συγκεκριμένο υπερκόμβο. Τυπικά δηλαδή θεωρούμε

$$t^m_{comp} \sim \frac{bal}{T}\frac{\prod_{i=1}^{N} X_i z}{P} \tag{5.12}$$

$$t^o_{comp} \sim \frac{T-bal}{T(T-1)}\frac{\prod_{i=1}^{N} X_i z}{P} \tag{5.13}$$

Λόγω της (5.12) ισχύει

$$t'^m_{comp} = \frac{bal'}{bal} t^m_{comp} \tag{5.14}$$

ενώ η (5.13) συνεπάγεται

$$t'^o_{comp} = \frac{T-bal'}{T-bal} t^o_{comp} \tag{5.15}$$

Επιπλέον, δεχόμαστε ότι ο χρόνος επικοινωνίας δεν διαφοροποιείται κατά την αναδιανομή του υπολογιστικού φορτίου μεταξύ των νημάτων, δηλαδή δεχόμαστε ότι

$$t'^m_{comm} \simeq t^m_{comm} \tag{5.16}$$

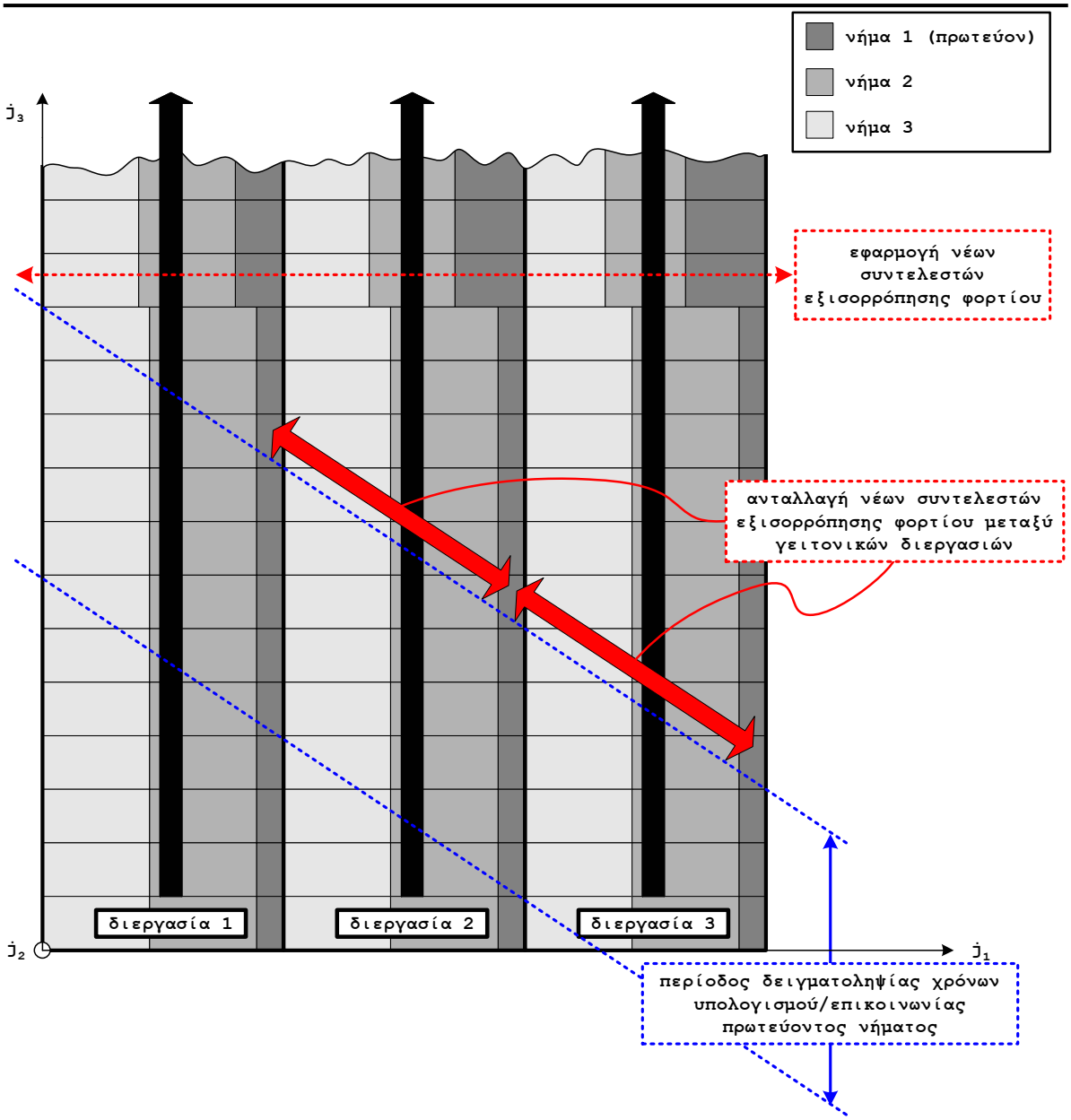
Συνδυάζοντας τις (5.14), (5.15) και (5.16), από την (5.11) προκύπτει

$$\begin{aligned} \frac{bal'}{bal} t^m_{comp} + t^m_{comm} &= \frac{T-bal'}{T-bal} t^o_{comp} \Rightarrow \\ bal' &= \frac{balTt^o_{comp} - bal(T-bal)t^m_{comm}}{(T-bal)t^m_{comp} + balt^o_{comp}} \end{aligned} \tag{5.17}$$

Τέλος, προσεγγίζοντας το χρόνο υπολογισμού ενός μη πρωτεύοντος νήματος t^o_{comp} με $\frac{T-bal}{bal(T-1)} t^m_{comp}$, λόγω των (5.12) και (5.13) μπορούμε εύκολα να συμπεράνουμε την (5.10).

□

Η βασική ιδέα της δυναμικής εξισορρόπησης φορτίου απεικονίζεται στο σχήμα 5.6, όπου η προτεινόμενη μεθοδολογία εφαρμόζεται κατά την υβριδική παραλληλοποίηση χονδρού κόκκου τρισδιάστατου αλγορίθμου φωλιασμένων βρόχων. Ο αλγόριθμος απεικονίζεται στις διαθέσιμες διεργασίες κατά το επίπεδο $j_1 j_2$, και κάθε διεργασία εκκινεί νήματα που αναλαμβάνουν ακολουθιακή εκτέλεση κατά μήκος της διάστασης j_3. Το παράλληλο πρόγραμμα διαιρείται σε δύο περιόδους, την κατά το δυνατόν μικρή περίοδο δειγματοληψίας και την κυρίως περίοδο εκτέλεσης. Κατά την περίοδο δειγματοληψίας μετρώνται οι χρόνοι υπολογισμού και επικοινωνίας του πρωτεύοντος νήματος, βάσει των οποίων υπολογίζεται ένας νέος υποψήφιος συντελεστής εξισορρόπησης φορτίου με χρήση της (5.10). Οι συντελεστές αυτοί ανταλλάσσονται μεταξύ των γειτονικών διεργασιών, και εφόσον διασφαλιστεί ότι οι



Σχήμα 5.6: *Δυναμική εξισορρόπηση φορτίου. Κατά την περίοδο δειγματοληψίας μετρώνται οι χρόνοι υπολογισμού και επικοινωνίας του πρωτεύοντος νήματος, ώστε να επαναπροσδιοριστούν οι συντελεστές εξισορρόπησης φορτίου.*

νέοι συντελεστές δεν παραβιάζουν τη χρονοδρομολόγηση υπερεπιπέδων αποφασίζεται η ταυτόχρονη εφαρμογή τους σε όλες τις διεργασίες το ταχύτερο δυνατό. Η εγκυρότητα των νέων συντελεστών εξισορρόπησης φορτίου έγκειται στο κατά πόσο οδηγούν απλώς στην πρόωρη ανταλλαγή συνοριακών

δεδομένων, χωρίς όμως σε καμία περίπτωση να καθυστερούν την αποστολή δεδομένων επικοινωνίας που κρίνονται απαραίτητα στη ροή της χρονοδρομολόγησης υπερεπιπέδων.

Η δυναμική εξισορρόπηση φορτίου συνδυάζει τη θεωρητική μοντελοποίηση της απόδοσης του προγράμματος με τη συνεκτίμηση της πραγματικής επίδοσης του τελευταίου κατά το χρόνο εκτέλεσης. Στη χειρότερη περίπτωση μπορεί να αποφασιστεί ότι η προτεινόμενη διόρθωση των συντελεστών εξισορρόπησης δεν είναι συμβατή με τη χρονοδρομολόγηση υπερεπιπέδων, οπότε απλά διατηρείται η αρχική στατική εξισορρόπηση φορτίου. Φυσικά, είναι προφανές ότι η δυναμική εξισορρόπηση φορτίου θα επιφέρει πρόσθετη επιβάρυνση στο πρόγραμμα, τόσο λόγω των μετρητών χρόνου που εισάγει όσο και εξαιτίας της επιπλέον επικοινωνίας που επιβάλλει για ανταλλαγή των νέων συντελεστών. Επίσης, κατά την απόδειξη του λήμματος 5.2 θεωρήσαμε κάποιες περαιτέρω παραδοχές, παρότι αυτές βασίζονται σε δειγματοληψία της συμπεριφοράς του συστήματος με τον εκάστοτε εξεταζόμενο αλγόριθμο και πραγματικό χώρο επαναλήψεων. Έτσι, θεωρήσαμε ότι το κόστος υπολογισμού είναι ανάλογο του πλήθους των επαναλήψεων, υπόθεση που εν μέρει αγνοεί φαινόμενα τοπικότητας αναφοράς, αν και σε μικρότερο βαθμό απ' ότι στον αντίστοιχο υπολογισμό της στατικής προσέγγισης. Τέλος, η παραδοχή περί σταθερού χρόνου επικοινωνίας παραβλέπει μερικώς τη δυνατότητα επικάλυψης της επικοινωνίας με τον υπολογισμό, θεωρώντας τα δύο τελευταία κατά προσέγγιση ασυσχέτιστες έννοιες.

5.3 Σύγκριση Μοντέλου Ανταλλαγής Μηνυμάτων με Υβριδικό Μοντέλο

Έστω T_{mp} ο συνολικός χρόνος εκτέλεσης του παράλληλου μονολιθικού προγράμματος ανταλλαγής μηνυμάτων και T_{hyb} ο αντίστοιχος χρόνος του ισοδύναμου υβριδικού. Στην ενότητα αυτή θα συγκρίνουμε σε θεωρητικό επίπεδο τους δύο αυτούς χρόνους, αποσκοπώντας ουσιαστικά σε μια θεωρητική σύγκριση των δύο προγραμματιστικών μοντέλων. Θα πρέπει πάντως να σημειωθεί ότι η ανάλυση της συγκεκριμένης ενότητας θα είναι σχετικά απλουστευμένη και μέχρι κάποιου βαθμού εξιδανικευμένη, υπό την έννοια ότι θα μας απασχολήσει μόνο η μοντελοποίηση του υπολογισμού και της επικοινωνίας των παράλληλων προγραμμάτων και όχι π.χ. λοιπές επιβαρύνσεις της βιβλιοθήκης ανταλλαγής μηνυμάτων, της διεπαφής πολυνηματικής επεξεργασίας, του εφαρμοζόμενου σχήματος χρονοδρομολόγησης κ.ο.κ.

Έστω ένας αλγόριθμος φωλιασμένων βρόχων διάστασης $N+1$, με χώρο επαναλήψεων $X_1 \times \cdots \times X_N \times Z$ και εξαρτήσεις δεδομένων $[d_1, \ldots, 0]^T, \ldots, [0, \ldots, d_{N+1}]^T$, τον οποίο θέλουμε να παραλληλοποιήσουμε τόσο με χρήση του μονολιθικού μοντέλου ανταλλαγής μηνυμάτων σε P_{mp} διεργασίες όσο και μέσω του υβριδικού προγραμματιστικού μοντέλου με P_{hyb} διεργασίες και T νήματα ανά διεργασία. Έστω επίσης ότι έχουν επιλεγεί οι τοπολογίες απεικόνισης για τα νήματα και τις διεργασίες, δηλαδή οι

όροι P_{mp}, P_{hyb} και T έχουν παραγοντοποιηθεί ως εξής:

$$\begin{aligned} P_{mp} &= \prod_{i=1}^{N} P_{mp,i} \\ P_{hyb} &= \prod_{i=1}^{N} P_{hyb,i} \\ T &= \prod_{i=1}^{N} T_i \end{aligned}$$

Σε αμφότερες τις περιπτώσεις, δηλαδή τόσο στο μονολιθικό όσο και στο υβριδικό μοντέλο, η παραλληλοποίηση θα βασιστεί στο μετασχηματισμό υπερκόμβων. Θεωρούμε συνεπώς ότι κάθε διεργασία/νήμα αναλαμβάνει την εκτέλεση ακολουθίας υπερκόμβων ύψους z κατά μήκος της εσωτερικής διάστασης Z, που απαριθμούνται από τη μεταβλητή $tile$ με $0 \leq tile \leq \lceil \frac{Z}{z} \rceil - 1$.

Ο συνολικός χρόνος εκτέλεσης μπορεί να προσεγγιστεί ικανοποιητικά ως το γινόμενο του συνολικού αριθμού βημάτων εκτέλεσης επί το μέσο χρόνο εκτέλεσης ενός υπερκόμβου ύψους z. Έτσι, για την περίπτωση του προγράμματος ανταλλαγής μηνυμάτων έχουμε

$$T_{mp} = N_{mp} \times T_{mp,tile} \tag{5.18}$$

ενώ για το μοντέλο της υβριδικής παραλληλοποίησης θα ισχύει

$$T_{hyb} = N_{hyb} \times T_{hyb,tile} \tag{5.19}$$

όπου T_{mp}, N_{mp} και $T_{mp,tile}$ ο συνολικός χρόνος εκτέλεσης, το συνολικό πλήθος βημάτων και ο μέσος χρόνος εκτέλεσης υπερκόμβου στην περίπτωση του μοντέλου ανταλλαγής μηνυμάτων, ενώ T_{hyb}, N_{hyb} και $T_{hyb,tile}$ τα αντίστοιχα μεγέθη για το υβριδικό μοντέλο. Αναλυτικότερα, για το μοντέλο ανταλλαγής μηνυμάτων θα έχουμε

$$N_{mp} = t_{mp,last} - t_{mp,first} + 1 \tag{5.20}$$

όπου $t_{mp,first}$, $t_{mp,last}$ οι χρονικές στιγμές εκτέλεσης του πρώτου και του τελευταίου υπερκόμβου, αντίστοιχα. Είδαμε ότι ο μετασχηματισμός υπερκόμβων εφαρμόζεται κατά τέτοιο τρόπο, ώστε οι N εξωτερικές διαστάσεις του μετασχηματισμένου διανύσματος διάσχισης των υπερκόμβων να ταυτοποιούν την ιδιοκτήτρια διεργασία, ενώ η πλέον εσωτερική απαριθμεί τον τρέχοντα υπερκόμβο. Κάθε υπερκόμβος χαρακτηρίζεται συνεπώς από ένα αναγνωριστικό διάνυσμα $(\vec{p} = (p_1, \ldots, p_N), tile)$, και μπορεί εύκολα να δειχθεί με χρήση θεωρίας χρονοδρομολόγησης (βλ. ενότητα 4.5) ότι η χρονική

στιγμή της εκτέλεσής του παρέχεται από την έκφραση

$$t(\vec{p} = (p_1, \ldots, p_N), tile) = \sum_{i=1}^{N} p_i + tile$$

Συνεπώς, θα είναι

$$t_{mp,first} = t((0, \ldots, 0), 0) = 0$$

και

$$\begin{aligned} t_{mp,last} &= t\left((P_{mp,1} - 1, \ldots, P_{mp,N} - 1), \left\lceil \frac{Z}{z} \right\rceil - 1 \right) \\ &= \sum_{i=1}^{N} P_{mp,i} - N + \left\lceil \frac{Z}{z} \right\rceil - 1 \end{aligned}$$

Επομένως, βάσει της (5.20), προκύπτει

$$N_{mp} = \sum_{i=1}^{N} P_{mp,i} - N + \left\lceil \frac{Z}{z} \right\rceil \tag{5.21}$$

Για κάθε υπερκόμβο, μια διεργασία απασχολείται αφενός με την εκτέλεση κατά προσέγγιση $\prod_{i=1}^{N} X_i z / P_{mp}$ επαναλήψεων (υποθέτοντας προσεγγιστικά ομοιόμορφη κατανομή του συνολικού αριθμού επαναλήψεων σε όλες τις διεργασίες), ενώ παράλληλα θα πρέπει να αποστείλει δεδομένα προς τις γειτονικές της διεργασίες. Λόγω των εξαρτήσεων δεδομένων, κάθε διεργασία απαιτείται να αποστείλει σε κάθε βήμα $d_i \prod_{\substack{j=1 \\ j \neq i}}^{j=N} \frac{X_j}{P_{mp,j}} z$ το πλήθος δεδομένα προς τη διεύθυνση i. Θεωρώντας

- το μέσο χρόνο για τον υπολογισμό n επαναλήψεων ίσο με n φορές το χρόνο t_{comp}, που απαιτείται για τον υπολογισμό μίας επανάληψης και
- το χρόνο επικοινωνίας ως αποτελούμενο από δύο συνιστώσες, την πρακτικά σταθερή συνιστώσα αρχικοποίησης $t_{startup}$, καθώς και τη συνιστώσα διάδοσης που λαμβάνεται κατά προσέγγιση ανάλογη με το μέγεθος του μηνύματος με ένα συντελεστή αναλογίας t_{data}

προκύπτει

$$T_{mp,tile} = \frac{\prod_{i=1}^{N} X_i z}{P_{mp}} t_{comp} + N t_{startup} + \frac{\prod_{i=1}^{N} X_i z}{P_{mp}} \sum_{i=1}^{N} \left\{ \frac{d_i P_{mp,i}}{X_i} \right\} t_{data} \tag{5.22}$$

Ο συνδυασμός των (5.18), (5.21) και (5.22) προσεγγίζει θεωρητικά το χρόνο εκτέλεσης του παράλλη-

λου αλγορίθμου ανταλλαγής μηνυμάτων ως εξής:

$$T_{mp} = \left(\sum_{i=1}^{N} P_{mp,i} - N + \left\lceil \frac{Z}{z} \right\rceil \right) \left(\frac{\prod_{i=1}^{N} X_i z}{P_{mp}} t_{comp} + N t_{startup} + \frac{\prod_{i=1}^{N} X_i z}{P_{mp}} \sum_{i=1}^{N} \left\{ \frac{d_i P_{mp,i}}{X_i} \right\} t_{data} \right)$$

Στον αντίποδα, για το υβριδικό μοντέλο παράλληλης επεξεργασίας θα ισχύει

$$N_{hyb} = t_{hyb,last} - t_{hyb,first} + 1 \tag{5.23}$$

όπου $t_{hyb,first}$, $t_{hyb,last}$ οι χρονικές στιγμές εκτέλεσης του τοπολογικά πλησιέστερου («πρώτου») και του τοπολογικά πιο απομακρυσμένου («τελευταίου») υπερκόμβου, αντίστοιχα. Κάθε υπερκόμβος του υβριδικού προγράμματος ταυτοποιείται από μια τριάδα $(\vec{p} = (p_1, \ldots, p_N), \vec{t} = (t_1, \ldots, t_N), tile)$, που αντιστοιχεί σε (ιδιοκτήτρια διεργασία, νήμα ιδιοκτήτης, τρέχων υπερκόμβος) και βάσει θεωρίας γραμμικής δρομολόγησης θα εκτελεστεί στο βήμα $t(\vec{p} = (p_1, \ldots, p_N), \vec{t} = (t_1, \ldots, t_N), tile)$ που παρέχεται από τη σχέση

$$t(\vec{p} = (p_1, \ldots, p_N), \vec{t} = (t_1, \ldots, t_N), tile) = \sum_{i=1}^{N} p_i T_i + \sum_{i=1}^{N} t_i + tile$$

Με βάση την παραπάνω σχέση, θα είναι

$$t_{hyb,first} = t((0, \ldots, 0), (0, \ldots, 0), 0) = 0$$

καθώς επίσης και

$$\begin{aligned} t_{hyb,last} &= t\left((P_{hyb,1} - 1, \ldots, P_{hyb,N} - 1), (T_1 - 1, \ldots, T_N - 1), \left\lceil \frac{Z}{z} \right\rceil - 1 \right) \\ &= \sum_{i=1}^{N} P_{hyb,i} T_i - N + \left\lceil \frac{Z}{z} \right\rceil - 1 \end{aligned}$$

Συνεπώς, από τη σχέση (5.23) προκύπτει

$$N_{hyb} = \sum_{i=1}^{N} P_{hyb,i} T_i - N + \left\lceil \frac{Z}{z} \right\rceil \tag{5.24}$$

Χάριν γενικότητας δεν θα εστιάσουμε σε κάποια συγκεκριμένη προσέγγιση του υβριδικού μοντέλου

παράλληλου προγραμματισμού. Αντίθετα, θα θεωρήσουμε την ιδεατή περίπτωση υβριδικής παραλληλοποίησης, που επιτυγχάνει αφενός μεν απαλοιφή μέρους της ανταλλαγής μηνυμάτων, που αφορούν στην επικοινωνία μεταξύ νημάτων της ίδιας διεργασίας, αφετέρου δε τέλεια εξισορρόπηση του *συνολικού* φορτίου υπολογισμού και επικοινωνίας μεταξύ των διαθέσιμων νημάτων. Έτσι, ο συνολικός χρόνος εκτέλεσης του υπερκόμβου κάθε νήματος θα θεωρηθεί ίσος με το $\frac{1}{T}$ του χρόνου που απαιτείται αν θεωρήσουμε τον εικονικό υπερκόμβο της ιδιοκτήτριας διεργασίας, που αποτελείται από το σύνολο των υπερκόμβων όλων των νημάτων της διεργασίας. Θα έχουμε λοιπόν

$$T_{hyb,tile} = \frac{1}{T}\left(\frac{\prod_{i=1}^{N} X_i z}{P_{hyb}} t_{comp} + N t_{startup} + \frac{\prod_{i=1}^{N} X_i z}{P_{hyb}} \sum_{i=1}^{N} \left\{\frac{d_i P_{hyb,i}}{X_i}\right\} t_{data}\right) \tag{5.25}$$

Οι (5.19),(5.24) και (5.25) δίνουν για το χρόνο εκτέλεσης του παράλληλου υβριδικού αλγορίθμου την εξής προσέγγιση:

$$\boxed{T_{hyb} = \left(\sum_{i=1}^{N} P_{hyb,i} T_i - N + \left\lceil \frac{Z}{z} \right\rceil\right) \frac{1}{T}\left(\frac{\prod_{i=1}^{N} X_i z}{P_{hyb}} t_{comp} + N t_{startup} + \frac{\prod_{i=1}^{N} X_i z}{P_{hyb}} \sum_{i=1}^{N} \left\{\frac{d_i P_{hyb,i}}{X_i}\right\} t_{data}\right)}$$

Η σύγκριση των εξισώσεων για τους εκτιμώμενους χρόνους T_{mp} και T_{hyb} οδηγεί στις ακόλουθες παρατηρήσεις:

- Δεδομένης μιας συγκεκριμένης αρχιτεκτονικής υποδομής κατανεμημένης μοιραζόμενης μνήμης με συνολικά P επεξεργαστές, είναι λογικό να υποθέσουμε ότι

  $$P_{mp} = P_{hyb} T = P \tag{5.26}$$

  Η παραπάνω υπόθεση εκφράζει την πλήρη αξιοποίηση των διαθέσιμων πόρων, είτε αποκλειστικά μέσω διεργασιών, είτε μέσω συνδυασμού διεργασιών και νημάτων. Με βάση αυτήν την παραδοχή, οι δύο προσεγγίσεις μπορούν θεωρητικά να είναι εξίσου αποδοτικές σε ό,τι αφορά στην κατανομή του φορτίου υπολογισμού, είτε ανά διεργασία είτε ανά νήμα. Επίσης, μπορούμε να θεωρήσουμε προσεγγιστικά ότι

  $$\sum_{i=1}^{N} P_{mp,i} \simeq \sum_{i=1}^{N} P_{hyb,i} T_i \tag{5.27}$$

  συνθήκη που συνεπάγεται ότι απαιτείται ο ίδιος αριθμός συνολικών βημάτων εκτέλεσης και στις

δύο περιπτώσεις. Ακόμα κι αν αυτό δεν συμβαίνει, η ενδεχόμενη διαφορά στα βήματα εκτέλεσης θα είναι μικρή, αφού ο συνολικός αριθμός βημάτων καθορίζεται και στις δύο περιπτώσεις κυρίως από τον παράγοντα $\lceil \frac{Z}{z} \rceil$ και συνεπώς δεν αναμένεται να κρίνει τη συνολική σύγκριση στην επίδοση των δύο προγραμματιστικών μοντέλων.

- Το υβριδικό μοντέλο απαιτεί την ανταλλαγή περίπου T φορές λιγότερων μηνυμάτων μεταξύ όλων των διεργασιών σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων. Το γεγονός αυτό είναι ιδιαίτερα επιθυμητό, καθώς μειώνει τη συνολική επιβάρυνση λόγω αρχικής καθυστέρησης διάδοσης των μηνυμάτων, στοιχείο σημαντικό για την απόδοση σε αρχιτεκτονικές κατανεμημένης μνήμης.

- Στο υβριδικό μοντέλο κάθε διεργασία πρέπει να αποστείλει μεγαλύτερο συνολικό όγκο δεδομένων σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων. Παρότι κατά την υβριδική παραλληλοποίηση ο συνολικός όγκος των δεδομένων επικοινωνίας μειώνεται, καθώς απαλείφεται η επικοινωνία μεταξύ των νημάτων της ίδιας διεργασίας με το σχετικά μικρότερο κόστος επιπλέον συγχρονισμού, εντούτοις μειώνεται λόγω της παραδοχής (5.26) κατά ένα παράγοντα T και ο αριθμός των διαθέσιμων διεργασιών που υλοποιούν την επικοινωνία αυτή. Όμως, ο συμψηφισμός επικοινωνίας και υπολογισμού κατά την εφαρμογή σχήματος εξισορρόπησης φορτίου μεταξύ των νημάτων δίνει θεωρητικά πλεονέκτημα στο υβριδικό μοντέλο, καθώς συνολικά μειώνονται τόσο ο αριθμός των μηνυμάτων όσο και το πλήθος των δεδομένων επικοινωνίας, ενώ κατά προσέγγιση διατηρείται το ίδιο κόστος υπολογισμού. Έτσι, μία τεχνική εξισορρόπησης φορτίου που θα επιτύγχανε την ομοιόμορφη κατανομή του συνολικού φορτίου του υπερκόμβου μεταξύ όλων των διαθέσιμων νημάτων θα μπορούσε θεωρητικά να αναδείξει τα πλεονεκτήματα του υβριδικού μοντέλου στον τομέα της επικοινωνίας.

Συμπερασματικά, το υβριδικό μοντέλο παράλληλου προγραμματισμού φαίνεται αρκετά υποσχόμενο για τις αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης, όπως οι συστοιχίες πολυεπεξεργαστικών στοιχείων. Είναι πάντως σαφές ότι η αποδοτικότητα του υβριδικού μοντέλου βρίσκεται σε άμεση συνάρτηση με την επίδοση του σχήματος εξισορρόπησης φορτίου που απαιτείται για την ομοιόμορφη κατανομή του συνολικού φόρτου εργασίας μεταξύ των νημάτων. Επιπλέον, κατά την παραπάνω ανάλυση αγνοήσαμε για λόγους απλότητας τον συνυπολογισμό του κόστους του συγχρονισμού μεταξύ των νημάτων, καθώς και των υπόλοιπων επιβαρύνσεων που επιβάλλει το περιβάλλον πολυνηματικής επεξεργασίας. Είναι προφανές ότι τα στοιχεία αυτά αναμένεται να επιφέρουν στην πράξη πρόσθετες χρονικές επιβαρύνσεις στην επίδοση του υβριδικού μοντέλου.

ΚΕΦΑΛΑΙΟ 6

Πειραματική Αξιολόγηση

Αναμφίβολα, η εγκυρότερη μέθοδος πιστοποίησης της αποτελεσματικότητας των προτεινόμενων μεθοδολογιών και προγραμματιστικών βελτιστοποιήσεων είναι η πειραματική αξιολόγηση τους με χρήση πραγματικών εφαρμογών. Αντικείμενο του κεφαλαίου αυτού αποτελεί η συγκριτική αξιολόγηση του μονολιθικού μοντέλου ανταλλαγής μηνυμάτων και των υβριδικών μοντέλων παράλληλου προγραμματισμού σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Προς την κατεύθυνση της πειραματικής αξιολόγησης των προτεινόμενων βελτιστοποιήσεων πραγματοποιήθηκαν αναλυτικές μετρήσεις του παράλληλου χρόνου εκτέλεσης για μια ομάδα μετροπρογραμμάτων σε δύο συστοιχίες πολυεπεξεργαστικών στοιχείων με διαφορετικά χαρακτηριστικά ως προς την επεξεργαστική ισχύ και το δίκτυο διασύνδεσης των κόμβων. Τα μετροπρογράμματα αυτά συνδυάστηκαν με ποικιλόμορφους χώρους επαναλήψεων, ώστε να παρέχουν τελικά μια κατά το δυνατό αντιπροσωπευτική εικόνα της επίδρασης των προτεινόμενων τεχνικών βελτιστοποίησης σε επαναληπτικούς αλγορίθμους ποικίλων δεδομένων εισόδου και υπολογιστικών απαιτήσεων σε μια δημοφιλή παράλληλη αρχιτεκτονική, όπως είναι η συστοιχία πολυεπεξεργαστικών στοιχείων.

6.1 Μετροπρογράμματα

Στην κατεύθυνση της αξιολόγησης των επιμέρους προγραμματιστικών μοντέλων χρησιμοποιήθηκαν πέντε διαφορετικά μετροπρογράμματα, που αντιστοιχούν σε βασικούς υπολογιστικούς πυρήνες πραγματικών εφαρμογών. Έτσι, παραλληλοποιήθηκαν η μέθοδος ολοκλήρωσης ADI (Alternating Direction

Implicit - **ADI**), η εξίσωση διάχυσης σε δισδιάστατη επιφάνεια XY για χρόνο T σε μορφή XYT (Diffusion Equation, XYT form - **DE-XYT**), η δισδιάστατη εξίσωση διάχυσης σε μορφή TXY (**DE-TXY**), η εξίσωση μεταφοράς σε δισδιάστατη επιφάνεια XY για χρόνο T (2D Advective equation - **Adv2D**), καθώς και η τρισδιάστατη εξίσωση μεταφοράς σε χώρο XYZ για χρόνο T (**Adv3D**). Πιο αναλυτικά:

1. *ADI*. Η τεχνική ADI αποτελεί μέθοδο που χρησιμοποιείται για την επίλυση μερικών διαφορικών εξισώσεων [KK02]. Ουσιαστικά, ο αλγόριθμος που περιγράφει την μέθοδο ADI μπορεί να μοντελοποιηθεί ως τρισδιάστατος τέλεια φωλιασμένος βρόχος, που επιβάλλει μοναδιαίες εξαρτήσεις δεδομένων προς τις αντίστοιχες κατευθύνσεις του (τρισδιάστατου) χώρου επαναλήψεων. Δηλαδή ισχύει:

$$D_{ADI} = \left[\begin{array}{cc|c} \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

   Ο πάνω αριστερά 2×2 υποπίνακας που έχει σημειωθεί με έντονη γραφή αντιστοιχεί στον πίνακα εξαρτήσεων μεταξύ των διεργασιών, μέσω του οποίου μπορούν εύκολα να καθοριστούν οι σχετικές ανάγκες για επικοινωνία. Παρατηρούμε επίσης ότι η τελευταία εξάρτηση του αλγορίθμου, δηλαδή το τρίτο διάνυσμα-στήλη του πίνακα εξαρτήσεων, δεν συνεπάγεται αναγκαιότητα ανταλλαγής δεδομένων μέσω μηνυμάτων. Το γεγονός αυτό οφείλεται στο ότι κατά μήκος της πλέον εσωτερικής διάστασης κάθε διεργασία πραγματοποιεί σειριακή εκτέλεση των υπολογισμών και συνεπώς διαθέτει ήδη τα απαιτούμενα δεδομένα, χωρίς να απαιτείται σχετική επικοινωνία με τις γειτονικές διεργασίες. Ο παραπάνω τρόπος γραφής διευκολύνει την εποπτική εξαγωγή των διανυσμάτων επικοινωνίας από τα διανύσματα εξάρτησης δεδομένων, και θα υιοθετηθεί στη συνέχεια για όλες τις περιπτώσεις των αλγορίθμων.

   Ο πυρήνας ADI διαθέτει ένα χώρο επαναλήψεων της μορφής $X_1 \times X_2 \times Z$, όπου η διάσταση Z θεωρείται ως η μεγαλύτερη των τριών. Έτσι, η απεικόνιση του αλγορίθμου στις διαθέσιμες διεργασίες θα γίνει ως προς την επιφάνεια $x_1 x_2$, ενώ κάθε διεργασία θα εκτελεί σειριακά υπερκόμβους κατά μήκος της διάστασης z.

2. *DE-XYT*. Η διαφορική εξίσωση που περιγράφει το φαινόμενο της ασταθούς διάχυσης σε δισδιάστατο χώρο Ω με αρχικές και συνοριακές συνθήκες περιγράφεται τυπικά ως εξής:

$$\left.\begin{array}{rcl} \frac{\partial \Theta}{\partial t} & = & \nabla^2 \Theta \\ \Theta(x, y, t = 0) & = & f(x, y) \\ \Theta(x, y, t) & = & g(x, y, t) \text{ στο σύνορο } \partial\Omega \end{array}\right\}$$

   όπου Θ η ζητούμενη συνάρτηση θερμότητας, της οποίας τις τιμές επιθυμούμε να υπολογίσουμε.

Έστω X, Y τα μήκη των πλευρών του ορθογωνίου που περικλείει το πεδίο εφαρμογής Ω και T το χρονικό παράθυρο κατά τη διάρκεια του οποίου επιθυμούμε να παρακολουθήσουμε το φαινόμενο της διάχυσης. Η backward διακριτοποίηση της εξίσωσης διάχυσης με ακρίβεια δεύτερης τάξης ως προς το χώρο και πρώτης τάξης ως προς το χρόνο παρέχει επαναληπτικό αλγόριθμο με τον ακόλουθο πίνακα εξαρτήσεων:

$$D_{DE-XYT} = \left[\begin{array}{cc|c} \mathbf{3} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{3} & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

Ουσιαστικά, ο παραπάνω πίνακας εξαρτήσεων αντιστοιχεί σε εξαρτήσεις από την προηγούμενη χρονική στιγμή και μέχρι τρεις προηγούμενες γειτονικές θέσεις στο πλέγμα. Θα πρέπει να σημειωθεί ότι λόγω της φυσικής σημασιολογίας της διάχυσης, βάσει της οποίας η πληροφορία κατά μήκος ολόκληρου του συνόρου $\partial\Omega$ πρέπει να ληφθεί υπόψη για τον υπολογισμό των τιμών της συνάρτησης θερμότητας, η backward διακριτοποίηση με τη μονόπλευρη ροή του υπολογισμού δεν συνιστά ευσταθή μέθοδο επίλυσης του προβλήματος της διάχυσης θερμότητας. Σε κάθε περίπτωση όμως χρησιμοποιήθηκε στο πλαίσιο της πειραματικής αξιολόγησης, καθώς αποτελεί αφενός έναν τυπικό υπολογισμό διακριτοποιημένης ΜΔΕ, ενώ αφετέρου επιβάλλει μεγαλύτερες ανάγκες για επικοινωνία σε σχέση με την μέθοδο ADI λόγω των τριπλάσιων διανυσμάτων εξάρτησης μεταξύ των διεργασιών. Έτσι, ακόμα κι αν ο αλγόριθμος δεν διασφαλίζει την ευστάθεια της μεθόδου επίλυσης, μας επιτρέπει να διερευνήσουμε την επίδραση των παράλληλων βελτιστοποιήσεων καθώς αυξάνουν οι εγγενείς ανάγκες επικοινωνίας του υπό εξέταση προβλήματος.

Στον πυρήνα DE-XYT θεωρούμε ως εσωτερική διάσταση τη χρονική t, ώστε ο αλγόριθμος να διαμερίζεται χωρικά μεταξύ των διαθέσιμων διεργασιών ως προς το επίπεδο xy. Η υπόθεση αυτή δεν αποτελεί τεχνητή σύμβαση, αλλά αντίθετα αντιστοιχεί σε φυσικά προβλήματα στα οποία ενδιαφέρει η παρακολούθηση της εξέλιξης του φαινομένου σε μια σχετικά περιορισμένη επιφάνεια για μεγάλο χρονικό διάστημα. Επιπλέον, λόγω της φύσης των εξαρτήσεων δεδομένων (ομοιόμορφες, μη αρνητικές), είναι δυνατή η αντιμετάθεση των βρόχων προς την επίτευξη της επιθυμητής ακολουθίας φωλιάσματος.

3. *DE-TXY*. Ο πυρήνας αυτός αποτελεί παραλλαγή του DE-XYT και προκύπτει μέσω αντιμετάθεσης της διάταξης φωλιάσματος των βρόχων, ώστε ο χωρικός βρόχος που σαρώνει τη διάσταση y να βρίσκεται στην πλέον εσωτερική θέση. Η περίπτωση αυτή αντιστοιχεί στην παρακολούθηση της εξέλιξης του φαινομένου σε μια ορθογώνια επιφάνεια XY μεγάλης έκτασης με $Y > X$ για σχετικά μικρό χρονικό διάστημα. Η αντιμετάθεση των βρόχων διαφοροποιεί τις εξαρτήσεις

μεταξύ των διεργασιών ως εξής:

$$D_{DE-TXY} = \left[\begin{array}{cc|c} \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{3} & 0 \\ \hline 0 & 0 & 3 \end{array}\right]$$

Η επιλογή της παραλλαγής αυτής για τη διαδικασία της πειραματικής αξιολόγησης έγινε για την πιστοποίηση της επίδρασης των προτεινόμενων βελτιστοποιήσεων σε μεγαλύτερο εύρος πυρήνων υπολογισμού και αλγοριθμικών χαρακτηριστικών (εξαρτήσεις δεδομένων-χώρος επαναλήψεων), που ενδέχεται να επιβάλλονται από τη σημασιολογία και τη σχετική χωροχρονική πολυπλοκότητα του προβλήματος.

4. *Adv2D*. Η διακριτοποίηση της δισδιάστατης εξίσωσης μεταφοράς ακολουθεί τη διαδικασία που περιγράφηκε στο παράδειγμα 3.1. Χάριν πληρότητας, αναπαράγεται εδώ η τυπική περιγραφή της σχετικής ΜΔΕ:

$$\left.\begin{array}{rcl} \frac{\partial u}{\partial t} & = & -v_x \frac{\partial u}{\partial x} - v_y \frac{\partial u}{\partial y} \\ u(t=0,x,y) & = & f(x,y) \\ u(t,x,y) & = & g(t,x,y) \text{ στο σύνορο } \partial\Omega \end{array}\right\}$$

Όπως προκύπτει από την ανάλυση του παραδείγματος 3.1, ο αλγόριθμος Adv2D επιβάλλει μοναδιαίες εξαρτήσεις δεδομένων:

$$D_{Adv2D} = \left[\begin{array}{cc|c} \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

Κατά την πειραματική διαδικασία, θεωρήσαμε αυθαίρετα για τις συνιστώσες της ταχύτητας διάδοσης της διαταραχής ότι $v_x = v_y = 50$. Έτσι, βάσει της (3.28) για ομοιόμορφο πλέγμα χωρικής πυκνότητας διακριτοποίησης Δ επιλέγουμε κβάντο χρόνου Δt με

$$\Delta t \leq \frac{\Delta}{100} \tag{6.1}$$

5. *Adv3D*. Στην προσπάθεια διερεύνησης της αποτελεσματικότητας της προτεινόμενης τεχνικής για επιλογή κατάλληλης τοπολογίας διεργασιών, θεωρήσαμε επιπλέον και τη διακριτοποιημένη εκδοχή της εξίσωσης μεταφοράς σε τρεις διαστάσεις. Η εξίσωση μεταφοράς σε τρεις διαστάσεις

μοντελοποιείται μέσω της ακόλουθης ΜΔΕ με αρχικές και συνοριακές συνθήκες:

$$\left.\begin{aligned} \frac{\partial u}{\partial t} &= -v_x\frac{\partial u}{\partial x} - v_y\frac{\partial u}{\partial y} - v_z\frac{\partial u}{\partial z} \\ u(t=0,x,y,z) &= f(x,y,z) \\ u(t,x,y,z) &= g(t,x,y,z) \text{ στο σύνορο } \partial\Omega \end{aligned}\right\}$$

Η διακριτοποίηση της τρισδιάστατης εξίσωσης μεταφοράς ακολουθεί όμοια διαδικασία με εκείνη της δισδιάστατης περίπτωσης και οδηγεί στον ακόλουθο πίνακα εξαρτήσεων δεδομένων:

$$D_{Adv3D} = \left[\begin{array}{ccc|c} \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Στις τρεις διαστάσεις και για ομοιόμορφο πλέγμα με $\Delta x = \Delta y = \Delta z = \Delta$, η συνθήκη ευστάθειας Courant για την επιλογή του κβάντου χρόνου Δt διαμορφώνεται ως εξής:

$$\Delta t \leq \frac{\Delta}{\sqrt{3\left(v_x^2 + v_y^2 + v_z^2\right)}} \tag{6.2}$$

Θεωρώντας αυθαίρετα για τις συνιστώσες της ταχύτητας διάδοσης της διαταραχής ότι $v_x = v_y = v_z = 50$, προκύπτει η ακόλουθη συνθήκη για την επιλογή του κβάντου χρόνου:

$$\Delta t \leq \frac{\Delta}{150} \tag{6.3}$$

Η επιλογή της αξιολόγησης των προτεινόμενων μεθοδολογιών και τεχνικών με χρήση των ανωτέρω μετροπρογραμμάτων έγινε με γνώμονα το γεγονός πως τα τελευταία αποτελούν τυπικούς εκπρόσωπους των αλγορίθμων φωλιασμένων βρόχων. Τα εξεταζόμενα μετροπρογράμματα πληρούν τις επιθυμητές ιδιότητες της κατηγορίας των αλγορίθμων που μελετάμε, καθώς εναλλάσσουν φάσεις υπολογισμού με ενδιάμεσες φάσεις επικοινωνίας για την ανταλλαγή δεδομένων προς όλες τις διαστάσεις του χώρου επαναλήψεων. Συνεπώς, όλες οι παράλληλες υλοποιήσεις περιέχουν ένα σημαντικό όγκο επικοινωνίας, διευκολύνοντας έτσι τη σύγκριση εναλλακτικών προγραμματιστικών μοντέλων και τεχνικών παράλληλης βελτιστοποίησης σε αρχιτεκτονικές κατανεμημένης μνήμης.

6.2 Πειραματική Υποδομή

Χρησιμοποιήσαμε το MPI ως βιβλιοθήκη ανταλλαγής μηνυμάτων και το OpenMP ως περιβάλλον πολυνηματικής επεξεργασίας. Για τις εκτελέσεις των παράλληλων προγραμμάτων χρησιμοποιήθηκαν δύο συστοιχίες πολυεπεξεργαστικών στοιχείων, η συστοιχία twins και η συστοιχία xenons. Η πρώτη συστοιχία αποτελείται από 8 κόμβους, καθένας από τους οποίους διαθέτει δύο επεξεργαστές Pentium III με συχνότητα ρολογιού 800 MHz, 256 MB κύρια μνήμη, 32 KB κρυφή μνήμη επιπέδου 1 (16 KB για δεδομένα + 16 KB για εντολές), 256 KB κρυφή μνήμη επιπέδου 2 και λειτουργικό σύστημα Linux με πυρήνα 2.4.22. Η δεύτερη συστοιχία αποτελείται από 8 κόμβους, έκαστος με δύο επεξεργαστές Xeon στα 2.8 GHz, 2 GB κύρια μνήμη, 28 KB κρυφή μνήμη επιπέδου 1 (16 KB για δεδομένα + 12 KB για εντολές), 1 MB κρυφή μνήμη επιπέδου 2 και λειτουργικό σύστημα Linux με πυρήνα 2.6.13. Για την υποστήριξη των οδηγιών του OpenMP χρησιμοποιήθηκε ο μεταγλωττιστής C/C++ της Intel (έκδοση 8.1 στη συστοιχία twins, έκδοση 9 στα xenons) με τις ακόλουθες επιλογές βελτιστοποίησης: `-O3 -mcpu=pentiumpro -openmp -static`. Οι κόμβοι της συστοιχίας twins διασυνδέονται μέσω δικτύου FastEthernet (100 Mbps), ενώ οι κόμβοι της συστοιχίας xenons διασυνδέονται μέσω του ταχύτερου δικτύου Gigabit Ethernet. Τέλος, χρησιμοποιήθηκε η υλοποίηση MPICH (έκδοση 1.2.6 για τα twins, 1.2.7p1 για τα xenons) του MPI, κατάλληλα διαμορφωμένη για SMP συστοιχία, ώστε να πραγματοποιεί την επικοινωνία μεταξύ διεργασιών στο εσωτερικό του ίδιου κόμβου μέσω μοιραζόμενης μνήμης τύπου SYS V. Η εν λόγω έκδοση του MPICH εγγυάται επίπεδο funneled πολυνηματικής υποστήριξης, συνεπώς είναι σε θέση να υποστηρίξει όλα τα προτεινόμενα προγραμματιστικά μοντέλα πλην της υβριδικής παραλληλοποίησης χονδρού κόκκου τύπου multiple. Τα τεχνικά στοιχεία των δύο συστοιχιών συνοψίζονται χάριν εποπτικότητας στον πίνακα 6.1.

Χαρακτηριστικό	***Συστοιχία twins***	***Συστοιχία xenons***
Πλήθος κόμβων	8	8
Πλήθος επεξεργαστών/κόμβο	2	2
Επεξεργαστής	Intel Pentium III (Coppermine)	Intel Xeon
Συχνότητα ρολογιού	800 MHz	2.8 GHz
Κύρια μνήμη κόμβου	256 MB	2 GB
Κρυφή μνήμη επεξεργαστή	16 KB L1 I + 16 KB L1 D 256 KB L2	12 KB L1 I + 16 KB L1 D 1 MB L2
Δίκτυο διασύνδεσης	100 Mbps FastEthernet	Gigabit Ethernet
Λειτουργικό σύστημα	Linux (πυρήνας 2.4.26)	Linux (πυρήνας 2.6.13)
Μεταγλωττιστής C/C++	Intel C/C++ μεταγλωττιστής 8.1	Intel C/C++ μεταγλωττιστής 9
Βιβλιοθήκη MPI	MPICH 1.2.6	MPICH 1.2.7p1

Πίνακας 6.1: *Τεχνικά στοιχεία συστοιχιών twins και xenons*

Στις περισσότερες περιπτώσεις, χρησιμοποιήσαμε 16 διεργασίες για το μονολιθικό μοντέλο ανταλ-

λαγής μηνυμάτων και 8 διεργασίες με 2 νήματα ανά διεργασία για το υβριδικό. Στην περίπτωση της πειραματικής αξιολόγησης της τοπολογίας απεικόνισης διεργασιών πραγματοποιήθηκαν σε πολλές περιπτώσεις πρόσθετες εκτελέσεις με 12 διεργασίες, για την κατά το δυνατό περισσότερο αξιόπιστη επαλήθευση των πλεονεκτημάτων της προτεινόμενης βελτιστοποίησης με περισσότερους συνδυασμούς καρτεσιανών τοπολογιών. Στην περίπτωση του μοντέλου ανταλλαγής μηνυμάτων, ένα κατάλληλο αρχείο αντιστοίχισης διεργασιών σε κόμβους του MPICH (machinefile) χρησιμοποιήθηκε για να διασφαλίσει ότι δύο διεργασίες που βρίσκονται στον ίδιο SMP κόμβο θα επικοινωνούν μέσω μοιραζόμενης μνήμης. Για όλα τα πειραματικά αποτελέσματα ελήφθησαν τουλάχιστον τρεις ανεξάρτητες μετρήσεις και οι τιμές που παρουσιάζονται αποτελούν το μέσο όρο των μετρήσεων αυτών. Παρότι κατεβλήθη κάθε δυνατή προσπάθεια να λαμβάνονται μετρήσεις κατά την απουσία άλλων χρηστών των συστημάτων, δεν κατέστη δυνατό να αποφευχθεί εξ ολοκλήρου κάποιος θόρυβος στις γραφικές παραστάσεις. Όπως όμως θα δούμε και στη συνέχεια, οι σημαντικότερες αυξομειώσεις που επισημαίνονται στις πειραματικές καμπύλες είναι θεωρητικά αναμενόμενες και οφείλονται κυρίως σε ιδιαιτερότητες της επικοινωνίας.

6.3 Τοπολογία Διεργασιών

Πρώτος στόχος της πειραματικής αξιολόγησης ήταν η σύγκριση των χρόνων παράλληλης εκτέλεσης κατά την επιλογή αφενός της συνήθους τοπολογίας διεργασιών ελάχιστης καθυστέρησης διάδοσης και αφετέρου της προτεινόμενης τοπολογίας ελαχιστοποίησης του όγκου των δεδομένων επικοινωνίας. Για το σκοπό αυτό χρησιμοποιήθηκαν όλα τα μετροπρογράμματα που περιγράφηκαν στην ενότητα 6.1 σε συνδυασμό με διάφορους χώρους επαναλήψεων και κόκκους παραλληλισμού. Επιπλέον, πραγματοποιήθηκαν μετρήσεις και στις δύο συστοιχίες πολυεπεξεργαστικών στοιχείων, δηλαδή τόσο στη συστοιχία twins όσο και στην τεχνολογικά πιο προηγμένη συστοιχία xenons, ενώ θεωρήθηκαν οι περιπτώσεις 16 και ενίοτε 12 διαθέσιμων διεργασιών. Η χρήση μεγαλύτερου εύρους συνδυασμών σε ό,τι αφορά τόσο στους αλγορίθμους (εξαρτήσεις δεδομένων, χώροι επαναλήψεων, σώμα υπολογισμού) όσο και στην υφιστάμενη αρχιτεκτονική υποδομή (ταχύτητα επεξεργαστή, αρχική καθυστέρηση διάδοσης και ρυθμός παροχής δικτύου διασύνδεσης, πλήθος διαθέσιμων διεργασιών) έγινε με σκοπό την κατά το δυνατό περισσότερο αξιόπιστη εξαγωγή συμπερασμάτων αναφορικά με την επίδοση της προτεινόμενης τοπολογίας απεικόνισης.

Η συγκεκριμένη σειρά πειραμάτων έγινε με χρήση αποκλειστικά του μονολιθικού παράλληλου προγραμματιστικού μοντέλου, χωρίς δηλαδή τη μέτρηση επίδοσης σε κάποιο από τα εναλλακτικά υβριδικά μοντέλα. Η επιλογή αυτή έγινε καθαρά για πρακτικούς λόγους, ώστε να μπορούμε σε κάθε περίπτωση να αξιοποιήσουμε το μέγιστο διαθέσιμο πλήθος επεξεργαστών της εκάστοτε συστοιχίας με ισάριθμο πλήθος διεργασιών. Άλλωστε, οποιαδήποτε βελτιστοποίηση αφορά στην επιλογή αποδοτικής

τοπολογίας απεικόνισης διεργασιών είναι ορθογώνια με τις περαιτέρω βελτιστοποιήσεις, που μπορούν να υιοθετηθούν στο υβριδικό παράλληλο προγραμματιστικό μοντέλο, καθώς το τελευταίο προσφέρει τη δυνατότητα ανεξάρτητης επιλογής αφενός της τοπολογίας των διεργασιών και αφετέρου εκείνης των νημάτων.

Κατά τη λήψη των μετρήσεων ακολουθήσαμε την εξής πειραματική διαδικασία: για κάθε συνδυασμό αλγορίθμου, χώρου επαναλήψεων και τοπολογίας διεργασιών μετρήθηκε ο χρόνος εκτέλεσης για επαρκές εύρος κόκκων παραλληλισμού, δηλαδή για διαφορετικές τιμές του ύψους z του υπερκόμβου. Κάθε μέτρηση εκτελέστηκε τουλάχιστον τρεις φορές, για λόγους επαλήθευσης των αποτελεσμάτων και μείωσης του ανεπιθύμητου θορύβου. Η βασικότερη σύγκριση που θα μας απασχολήσει στη συνέχεια είναι εκείνη του ελάχιστου χρόνου που επετεύχθη για κάποιο ύψος υπερκόμβου σε κάθε περίπτωση τοπολογίας διεργασιών. Επιπλέον όμως, θα επιχειρήσουμε να αναλύσουμε το συνολικό χρόνο εκτέλεσης σε χρόνους υπολογισμού και επικοινωνίας, για να διερευνήσουμε κατά πόσο η ενδεχόμενη βελτίωση του χρόνου εκτέλεσης με την επιλογή της προτεινόμενης τοπολογίας οφείλεται πραγματικά σε ελαχιστοποίηση του χρόνου επικοινωνίας. Τέλος, σε λίγες περιπτώσεις θα εξετάσουμε και μετρήσεις χρόνου εκτέλεσης σε συνάρτηση με το ύψος του υπερκόμβου, για να φανεί εποπτικά ο βαθμός ομοιομορφίας που χαρακτηρίζει τη συμπεριφορά του προγράμματος για διαφορετικό κόκκο παραλληλισμού.

6.3.1 ADI

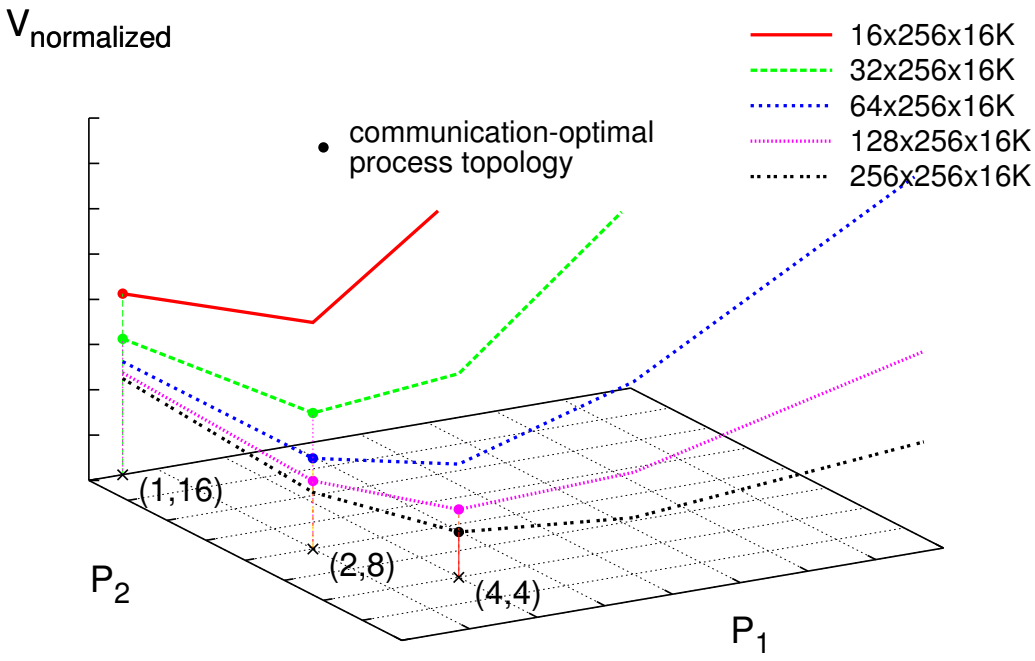
Το μετροπρόγραμμα ADI διαθέτει ένα χώρο επαναλήψεων $X_1 \times X_2 \times Z$. Θεωρήσαμε πέντε διαφορετικούς χώρους επαναλήψεων, ήτοι $16 \times 256 \times 16K$, $32 \times 256 \times 16K$, $64 \times 256 \times 16K$, $128 \times 256 \times 16K$ και $256 \times 256 \times 16K$. Πραγματοποιήθηκαν μετρήσεις τόσο με 16 όσο και με 12 διεργασίες για ύψη υπερκόμβου από 2 ως 200 με βήμα 2, καθώς σε αυτό το εύρος παρατηρήθηκε ο ελάχιστος χρόνος εκτέλεσης των προγραμμάτων. Επίσης, λήφθηκαν μετρήσεις τόσο στη συστοιχία twins όσο και στη συστοιχία xenons.

Χώρος επαναλήψεων	***16 διεργασίες***	***12 διεργασίες***
$16 \times 256 \times 16K$	1×16	1×12
$32 \times 256 \times 16K$	2×8	1×12
$64 \times 256 \times 16K$	2×8	2×6
$128 \times 256 \times 16K$	2×8	2×6
$256 \times 256 \times 16K$	4×4	3×4

Πίνακας 6.2: *Προτεινόμενες τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα ADI και συνολικό πλήθος διεργασιών 16 ή 12*

Ο πίνακας 6.2 συνοψίζει τις προτεινόμενες τοπολογίες απεικόνισης για τους διάφορους χώρους επαναλήψεων, τόσο για συνολικό πλήθος διεργασιών ίσο με 16 όσο και για 12 διεργασίες. Σε όλες τις

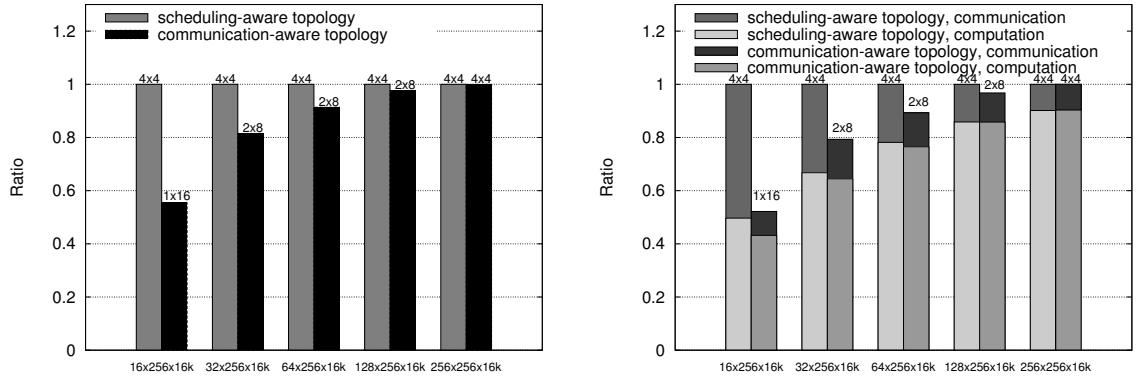
περιπτώσεις, οι χρόνοι εκτέλεσης που επιτυγχάνονται υπό την προτεινόμενη τοπολογία του πίνακα συγκρίνονται με εκείνους που λαμβάνονται κατά την υιοθέτηση της 4×4 τοπολογίας διεργασιών για 16 διεργασίες και της 3×4 τοπολογίας για 12 διεργασίες. Οι τοπολογίες αυτές ελαχιστοποιούν την αρχική καθυστέρηση εκκίνησης της πιο απομακρυσμένης διεργασίας κατά την εκτέλεση του παράλληλου προγράμματος. Για την περίπτωση των 12 διεργασιών επιλέγουμε την τοπολογία 3×4 έναντι της 4×3 παρότι αμφότερες είναι θεωρητικά ισοδύναμες όσον αφορά στην αρχική καθυστέρηση διάδοσης, καθώς η πρώτη ταιριάζει καλύτερα με τη μορφολογία των εξεταζόμενων χώρων επαναλήψεων, για τους οποίους ισχύει γενικά ότι $X_1 \leq X_2$.



Σχήμα 6.1: *Επιλογή τοπολογίας διεργασιών ελάχιστης επικοινωνίας (P_1, P_2) για πέντε χώρους επαναλήψεων. Για κάθε χώρο επαναλήψεων σχεδιάζεται η καμπύλη του κανονικοποιημένου κόστους επικοινωνίας, της οποίας οι θέσεις ελαχίστων αντιστοιχούν στην προτεινόμενη τοπολογία.*

Για την περίπτωση των 16 διεργασιών, οι τοπολογίες που ελαχιστοποιούν το συνολικό όγκο επικοινωνίας για κάθε χώρο επαναλήψεων απεικονίζονται στο σχήμα 6.1. Παρατηρούμε ότι για τους χώρους επαναλήψεων $16 \times 256 \times 16K$ και $64 \times 256 \times 16K$ οι προτεινόμενες τοπολογίες (1×16 και 2×8, αντίστοιχα) είναι βέλτιστες όσον αφορά στην ελαχιστοποίηση του όγκου της επικοινωνίας, και μπο-

ρούν να εξαχθούν άμεσα από την (4.6). Για τις άλλες δύο περιπτώσεις, η (4.6) δεν παρέχει μια έγκυρη ακέραιη τοπολογία, συνεπώς μια τέτοια πρέπει να αναζητηθεί με τη βοήθεια του αλγορίθμου 4.5. Στην ανάλυσή μας δεν παρέχουμε αποτελέσματα για χώρους επαναλήψεων της μορφής $X_1 \times X_2 \times Z$ με $X_1 > X_2$, καθώς λόγω συμμετρίας των αλγοριθμικών υπολογισμών παρατηρήθηκε ότι οι χώροι αυτοί παρουσιάζουν πανομοιότυπη συμπεριφορά με τους υπό εξέταση $X_2 \times X_1 \times Z$ χώρους.



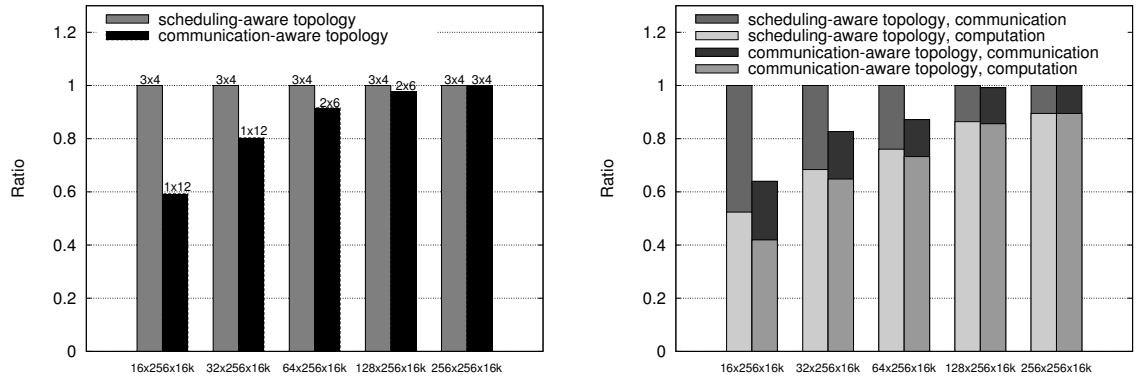
(α) συνολικός χρόνος εκτέλεσης

(β) ανάλυση χρόνου εκτέλεσης

Σχήμα 6.2: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (ADI, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία twins)*

Το σχήμα 6.2 συνοψίζει τους συνολικούς χρόνους εκτέλεσης για τη μέθοδο ADI στη συστοιχία twins με χρήση 16 διεργασιών, ενώ το σχήμα 6.3 απεικονίζει τους αντίστοιχους χρόνους για την περίπτωση των 12 διεργασιών. Σημειώνεται ότι για τη σύγκριση έχουν ληφθεί οι ελάχιστοι χρόνοι εκτέλεσης που επιτεύχθηκαν για κάποιο κόκκο παραλληλισμού, ενώ επίσης ως χρόνος εκτέλεσης υπονοείται η χρονική διάρκεια από την έναρξη των υπολογισμών στη χρονικά πρώτη διεργασία μέχρι την περάτωση των υπολογισμών στη χρονικά τελευταία διεργασία. Όλοι οι χρόνοι εκτέλεσης είναι κανονικοποιημένοι προς τους χρόνους της 4×4 τοπολογίας διεργασιών, καθώς αυτό διευκολύνει τη σύγκριση της σχετικής επίδοσης των εναλλακτικών τοπολογιών: η ποσοστιαία μείωση του χρόνου εκτέλεσης μπορεί να ληφθεί εποπτικά από ανάγνωση της διαφοράς ύψους των δύο ράβδων κάθε ζεύγους, όπου η πρώτη ράβδος αναπαριστά το μοναδιαίο χρόνο εκτέλεσης της τοπολογίας 4×4 και η δεύτερη σημειώνει τον αντίστοιχο κανονικοποιημένο χρόνο εκτέλεσης της εκάστοτε προτεινόμενης τοπολογίας.

Τα πειραματικά αποτελέσματα καταδεικνύουν μια σημαντική βελτίωση στην επίδοση με χρήση της τοπολογίας διεργασιών που ελαχιστοποιεί το συνολικό όγκο των δεδομένων επικοινωνίας, τόσο στην περίπτωση των 16 διεργασιών όσο και σε εκείνη των 12 διεργασιών. Η ποσοστιαία βελτίωση αυξάνεται καθώς μεταβαίνουμε από σχετικά συμμετρικούς χώρους επαναλήψεων προς λιγότερο συμμετρικούς



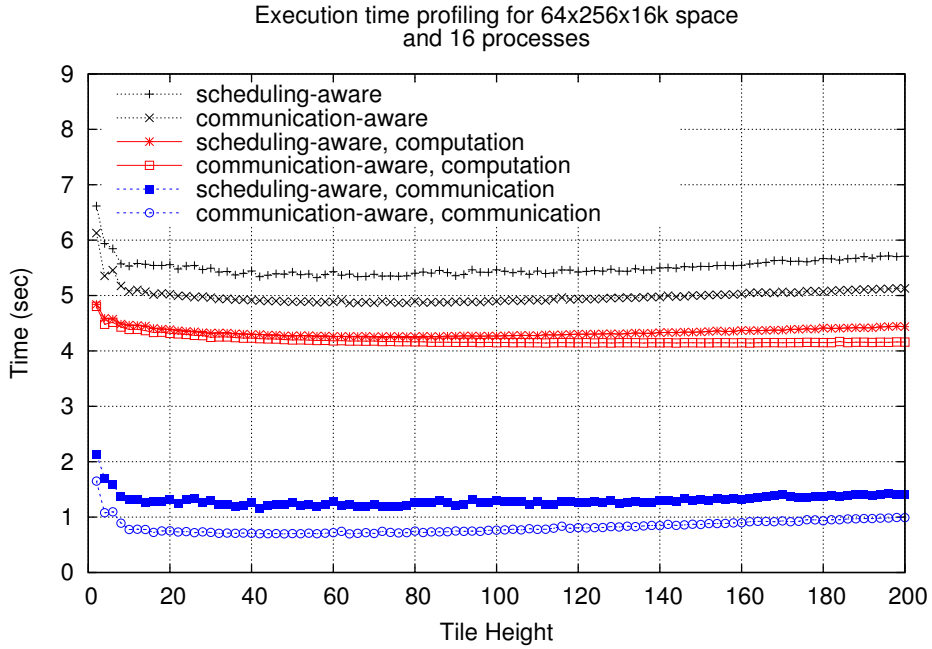
(α) συνολικός χρόνος εκτέλεσης (β) ανάλυση χρόνου εκτέλεσης

Σχήμα 6.3: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (ADI, διάφοροι χώροι επαναλήψεων, 12 διεργασίες, συστοιχία twins)*

χώρους. Για παράδειγμα, στις 16 διεργασίες παρατηρούμε μια βελτίωση πλέον του 45% για το χώρο $16 \times 256 \times 16K$, η οποία μειώνεται προοδευτικά σε 19% για το χώρο $32 \times 256 \times 16K$, 8% για το χώρο $64 \times 256 \times 16K$ και τελικά 2% στην περίπτωση του χώρου $128 \times 256 \times 16K$. Ομοίως, στην περίπτωση των 12 διεργασιών η ποσοστιαία βελτίωση ανέρχεται σε 41% για το χώρο $16 \times 256 \times 16K$, σε 20% για το χώρο $32 \times 256 \times 16K$, σε 9% για το χώρο $64 \times 256 \times 16K$ και τελικά σε 2% στην περίπτωση του χώρου $128 \times 256 \times 16K$. Στην περίπτωση του χώρου $256 \times 256 \times 16K$ η προτεινόμενη τοπολογία συμπίπτει με τη συνήθη τοπολογία ελαχιστοποίησης της αρχικής καθυστέρησης διάδοσης. Η προοδευτική μείωση της ποσοστιαίας βελτίωσης της επίδοσης του προγράμματος είναι αναμενόμενη και οφείλεται στο ότι η σχετική μείωση του όγκου των δεδομένων επικοινωνίας, που επιτυγχάνεται με χρήση της προτεινόμενης τοπολογίας, είναι μεγαλύτερη σε ασύμμετρους χώρους σε σχέση με τους περισσότερο συμμετρικούς. Πράγματι, για το χώρο επαναλήψεων $16 \times 256 \times 16K$ η υιοθέτηση της προτεινόμενης τοπολογίας μειώνει το συνολικό όγκο των δεδομένων επικοινωνίας κατά 71%. Στην περίπτωση του χώρου επαναλήψεων $32 \times 256 \times 16K$ η αντίστοιχη μείωση ισούται με 44%, ενώ στο χώρο επαναλήψεων $64 \times 256 \times 16K$ ο όγκος των δεδομένων επικοινωνίας μειώνεται κατά 27%. Στους χώρους επαναλήψεων $128 \times 256 \times 16K$ και $256 \times 256 \times 16K$ οι προτεινόμενες τοπολογίες οδηγούν στην ανταλλαγή ισότιμου όγκου δεδομένων με την τοπολογία 4×4.

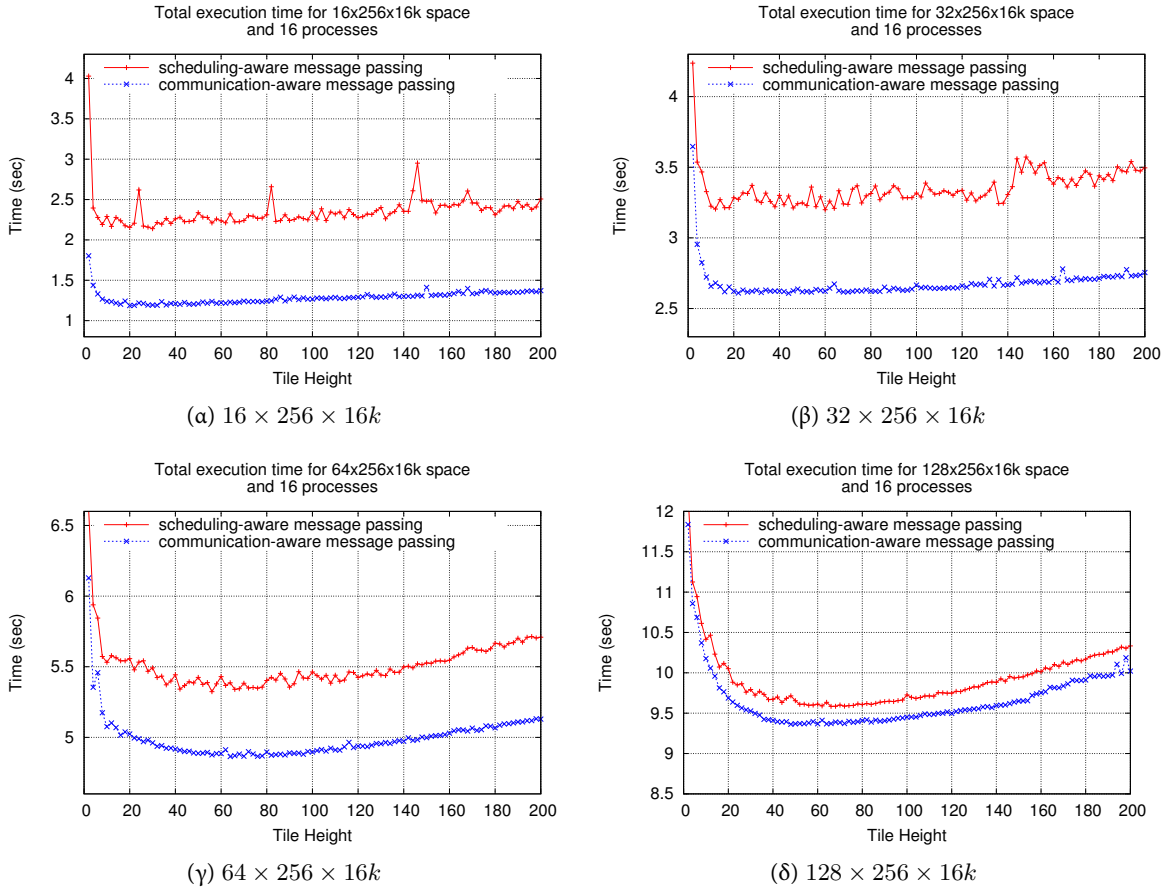
Διερευνώντας τα οφέλη που αποκομίσαμε κατά την υιοθέτηση της τοπολογίας ελάχιστης επικοινωνίας πραγματοποιήσαμε επιπλέον καταγραφή των επιμέρους χρόνων επικοινωνίας και υπολογισμού του παράλληλου προγράμματος. Έτσι, σε όλες τις διεργασίες μετρήσαμε το χρόνο που απαιτείται για την ολοκλήρωση των *υπολογισμών* ενός υπερκόμβου, ενώ επιπλέον καταγράψαμε και το συνολικό χρό-

νο *επικοινωνίας* που σχετίζεται τόσο με την κλήση ρουτινών MPI όσο και με την ομαδοποίηση/ταξινόμηση των δεδομένων επικοινωνίας. Συγκεκριμένα, στους χρόνους επικοινωνίας συμπεριλαμβάνονται οι χρόνοι ανταλλαγής μηνυμάτων, δηλαδή οι χρόνοι μετάδοσης και λήψης δεδομένων με χρήση ασύγχρονων λειτουργιών επικοινωνίας του MPICH και οι χρόνοι που απαιτούνται για τη διασφάλιση της επιτυχούς ολοκλήρωσης των παραπάνω λειτουργιών. Επιπλέον, στους χρόνους επικοινωνίας έχουν συμπεριληφθεί οι χρόνοι ομαδοποίησης των δεδομένων προς αποστολή, καθώς και οι χρόνοι τοποθέτησης των δεδομένων λήψης.



Σχήμα 6.4: *Παράσταση συνολικού χρόνου εκτέλεσης, μέγιστου χρόνου υπολογισμού και μέγιστου χρόνου επικοινωνίας για το μετροπρόγραμμα ADI, το χώρο επαναλήψεων* $64 \times 256 \times 16K$ *και 16 διεργασίες. Παρατηρούμε ότι οι χρόνοι υπολογισμού δεν διαφέρουν σημαντικά, ενώ αντίθετα το πλεονέκτημα που προσφέρει στο χρόνο επικοινωνίας η προτεινόμενη τοπολογία μεταφέρεται αυτούσιο και στο συνολικό χρόνο εκτέλεσης του προγράμματος.*

Στα σχήματα 6.2(β) και 6.3(β) παρουσιάζονται οι μέγιστοι χρόνοι υπολογισμού και επικοινωνίας, που προκύπτουν μέσω αναγωγής των επιμέρους χρόνων ως προς όλες τις διεργασίες και λήψη του μέγιστου χρόνου σε κάθε περίπτωση. Όλοι οι χρόνοι που παρουσιάζονται είναι κανονικοποιημένοι ως προς το άθροισμα {μέγιστος χρόνος υπολογισμού+μέγιστος χρόνος επικοινωνίας} που αντιστοιχεί στην περίπτωση της συνήθους τοπολογίας ελαχιστοποίησης των βημάτων παράλληλης εκτέλεσης. Το άθροισμα των επιμέρους χρόνων δεν ισούται κατ' ανάγκη με το συνολικό χρόνο εκτέλεσης, καθώς οι μέγιστοι επιμέρους χρόνοι υπολογισμού και επικοινωνίας που λαμβάνονται κατά την αναγωγή πιθα-



(α) $16 \times 256 \times 16k$ (β) $32 \times 256 \times 16k$

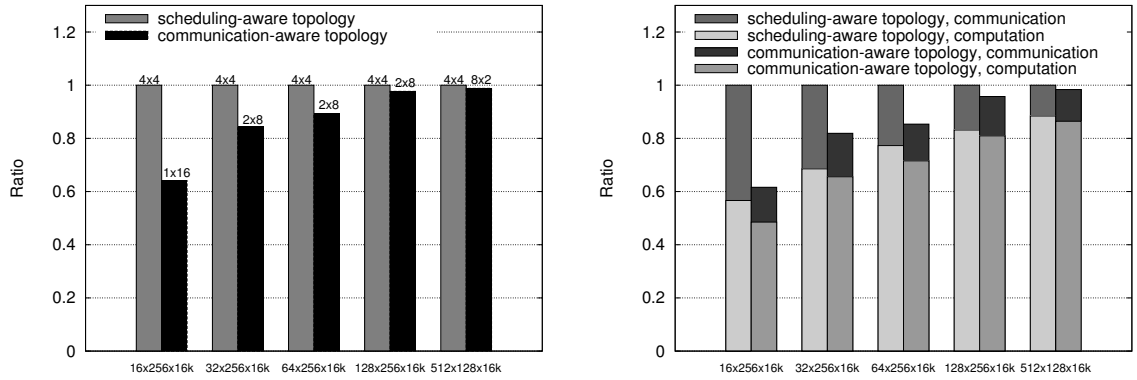
(γ) $64 \times 256 \times 16k$ (δ) $128 \times 256 \times 16k$

Σχήμα 6.5: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών για μεταβλητό κόκκο παραλληλισμού (ADI, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία twins)*

νότατα δεν αναφέρονται καν στην ίδια διεργασία. Εντούτοις, παρά τις σχετικά μικρές αποκλίσεις που παρατηρούνται στους χρόνους υπολογισμού και μπορούν να αποδοθούν σε φαινόμενα τοπικότητας αναφοράς δεδομένων στην ιεραρχία μνήμης, είναι εμφανές ότι το σχετικό πλεονέκτημα της προτεινόμενης τοπολογίας στο συνολικό χρόνο εκτέλεσης οφείλεται στην αντίστοιχη μείωση που επιτυγχάνεται στους χρόνους επικοινωνίας. Πράγματι, οι χρόνοι εκτέλεσης φανερώνουν ότι για τη δεδομένη αρχιτεκτονική και το συγκεκριμένο αλγόριθμο η τοπικότητα αναφοράς των δεδομένων πρακτικά δεν επηρεάζει τη συνολική απόδοση, παρότι οι χρόνοι υπολογισμού είναι εν γένει μικρότεροι κατά την επιλογή μετασχηματισμού υπερκόμβων, που διατηρεί το ίχνος προσπέλασης των δεδομένων εντός των ορίων της L2 κρυφής μνήμης. Καθότι στην παρούσα εργασία δίνεται έμφαση στην αξιοποίηση των χαρακτηριστικών επικοινωνίας σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης μέσω παράλληλων

προγραμματιστικών μοντέλων, οι χρόνοι υπολογισμού δεν θα αναλυθούν περαιτέρω. Όταν η μείωση του χρόνου επικοινωνίας είναι υψηλή, αντίστοιχα υψηλή προκύπτει και η μείωση του συνολικού χρόνου εκτέλεσης του προγράμματος. Η αναλογία αυτή καθίσταται περισσότερο σαφής κατά την παράσταση σε κοινούς άξονες του συνολικού χρόνου εκτέλεσης, του χρόνου υπολογισμού και του χρόνου επικοινωνίας για διάφορα ύψη υπερκόμβου, όπως στο σχήμα 6.4 για την περίπτωση του χώρου επαναλήψεων $64 \times 256 \times 16K$.

Όπως φαίνεται στο σχήμα 6.5, σε όλους τους χώρους επαναλήψεων παρατηρήσαμε ομοιόμορφη συμπεριφορά των χρόνων παράλληλης εκτέλεσης για το συνολικό εύρος τιμών του ύψους υπερκόμβου που εξετάσαμε. Συνεπώς, για τον αλγόριθμο ADI η προτεινόμενη τοπολογία απεικόνισης υπερέχει τόσο για λεπτομερή όσο και για χονδροειδή κόκκο παραλληλισμού, καθώς επιτυγχάνει σε όλες τις περιπτώσεις τους μικρότερους χρόνους εκτέλεσης.



(α) συνολικός χρόνος εκτέλεσης (β) ανάλυση χρόνου εκτέλεσης

Σχήμα 6.6: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (ADI, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία xenons)*

Τέλος, στο σχήμα 6.6 καταγράφονται οι χρόνοι εκτέλεσης για τις εναλλακτικές τοπολογίες στη συστοιχία των xenons. Παρότι η συστοιχία των xenons διαθέτει ένα δίκτυο διασύνδεσης κατά μία τάξη μεγέθους ταχύτερο σε σχέση με εκείνο των twins (Gigabit Ethernet έναντι FastEthernet), η υιοθέτηση της προτεινόμενης τοπολογίας παρέχει και πάλι υψηλή μείωση του συνολικού χρόνου εκτέλεσης λόγω της αντίστοιχης ποσοστιαίας μείωσης του χρόνου επικοινωνίας. Έτσι, ενώ σε απόλυτα μεγέθη τόσο οι συνολικοί χρόνοι παράλληλης εκτέλεσης όσο και οι επιμέρους χρόνοι υπολογισμού και επικοινωνίας μειώνονται κατά την εκτέλεση των προγραμμάτων στη συστοιχία xenons, το συγκριτικό πλεονέκτημα της προτεινόμενης τοπολογίας απεικόνισης διατηρείται, καθώς το σκέλος της επικοινωνίας εξακολουθεί να αποτελεί καθοριστική συνιστώσα της συνολικής επίδοσης του προγράμματος.

6.3.2 DE-XYT

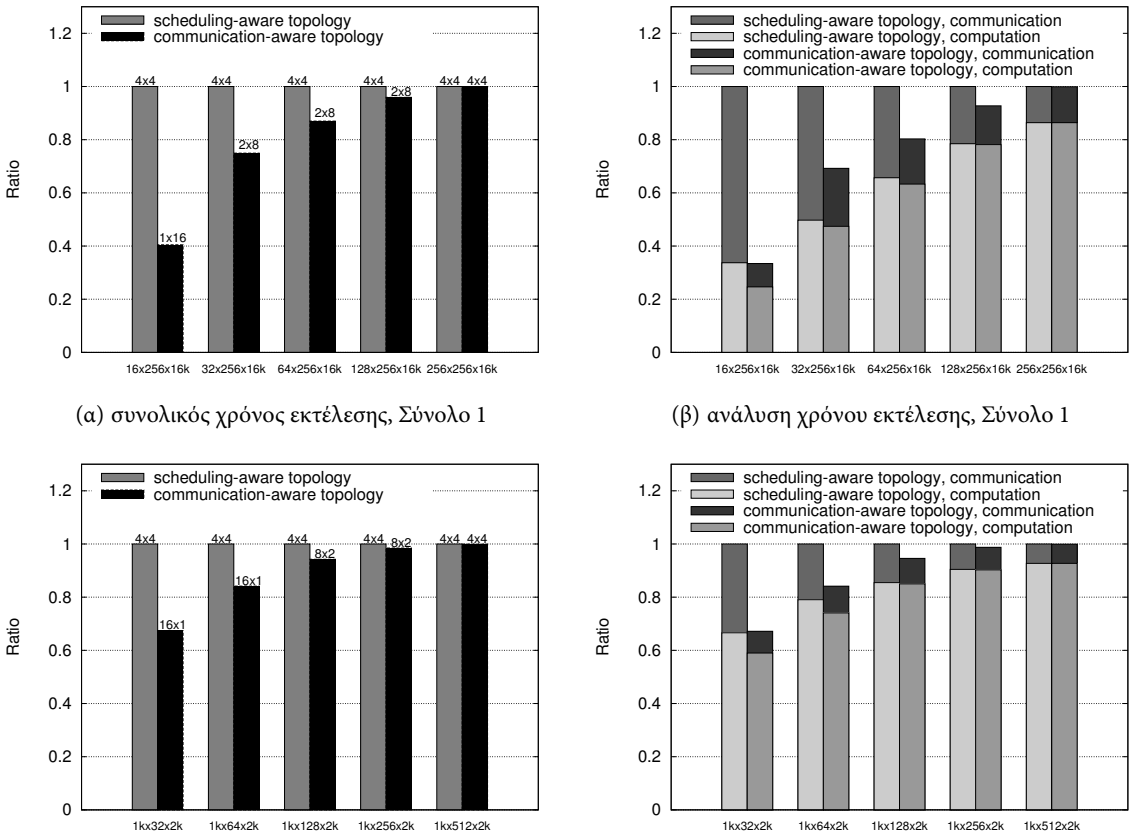
Στην περίπτωση του μετροπρογράμματος DE-XYT χρησιμοποιήθηκαν δύο ομάδες χώρων επαναλήψεων. Όπως και στην περίπτωση του ADI, χρησιμοποιήθηκαν οι χώροι $\{16, 32, 64, 128, 256\} \times 256 \times 16K$, οι οποίοι στο εξής χάριν συντομίας θα κατονομάζονται *Σύνολο 1*. Οι χώροι επαναλήψεων του Συνόλου 1 ευνοούν εγγενώς το φαινόμενο της σωλήνωσης και κατά συνέπεια το εφαρμοζόμενο σχήμα δρομολόγησης, καθώς η εσωτερική διάσταση της σειριακής εκτέλεσης είναι σημαντικά μεγαλύτερη από τις εξωτερικές διαστάσεις απεικόνισης. Επιπλέον, στην περίπτωση του Συνόλου 1 κάθε διεργασία αναλαμβάνει την εκτέλεση ενός υψηλού αριθμού υπερκόμβων κατά μήκος της σειριακής διάστασης, αμβλύνοντας έτσι τις επιπτώσεις της αρχικής καθυστέρησης διάδοσης του αλγορίθμου. Για τους λόγους αυτούς θεωρήσαμε επιπλέον ένα δεύτερο σύνολο χώρων επαναλήψεων (*Σύνολο 2*), στο οποίο η εσωτερική σειριακή διάσταση είναι συγκρίσιμη με τις διαστάσεις απεικόνισης. Συγκεκριμένα, θα εξετάσουμε επιπλέον τους χώρους επαναλήψεων $1K \times 32 \times 2K$, $1K \times 64 \times 2K$, $1K \times 128 \times 2K$, $1K \times 256 \times 2K$ και $1K \times 512 \times 2K$ (16 διεργασίες) ή $1K \times 384 \times 2K$ (12 διεργασίες, λόγω πρακτικών περιορισμών στους διαθέσιμους πόρους μνήμης). Η διερεύνηση των χώρων επαναλήψεων του Συνόλου 2 αντανακλά στο ιδιαίτερο φυσικό περιεχόμενο του αλγορίθμου, εάν αναλογιστεί κανείς πως οι δύο πρώτες διαστάσεις αντιστοιχούν σε χώρο και η τρίτη σε χρόνο. Ανάλογα με τη φύση του υπό εξέταση προβλήματος και τη διακριτοποίηση που εφαρμόζεται, είναι πιθανό να προκύψουν τέτοιοι χώροι επαναλήψεων συγκρίσιμης χωρικής και χρονικής πολυπλοκότητας.

Χώρος επαναλήψεων		***16 διεργασίες***	***12 διεργασίες***
Σύνολο 1	$16 \times 256 \times 16K$	1×16	1×12
	$32 \times 256 \times 16K$	2×8	1×12
	$64 \times 256 \times 16K$	2×8	2×6
	$128 \times 256 \times 16K$	2×8	2×6
	$256 \times 256 \times 16K$	4×4	3×4
Σύνολο 2	$1K \times 32 \times 2K$	16×1	12×1
	$1K \times 64 \times 2K$	16×1	12×1
	$1K \times 128 \times 2K$	8×2	12×1
	$1K \times 256 \times 2K$	8×2	6×2
	$1K \times 384 \times 2K$		6×2
	$1K \times 512 \times 2K$	4×4	

Πίνακας 6.3: *Προτεινόμενες τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-XYT και συνολικό πλήθος διεργασιών 16 ή 12*

Συνολικά, οι υπό εξέταση χώροι επαναλήψεων και οι προτεινόμενες τοπολογίες απεικόνισης για την παράλληλη εκτέλεση του μετροπρογράμματος DE-XYT σε 16 ή 12 διεργασίες αναγράφονται στον πίνακα 6.3. Στην περίπτωση των 16 διεργασιών γίνεται πάντα σύγκριση της προτεινόμενης τοπολογίας

με την τοπολογία 4×4. Στην περίπτωση των 12 διεργασιών συγκρίνουμε την εκάστοτε προτεινόμενη τοπολογία απεικόνισης με τις τοπολογίες 3×4 (Σύνολο 1) και 4×3 (Σύνολο 2), που επιτυγχάνουν την ελάχιστη καθυστέρηση διάδοσης για τους εξεταζόμενους χώρους επαναλήψεων.



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

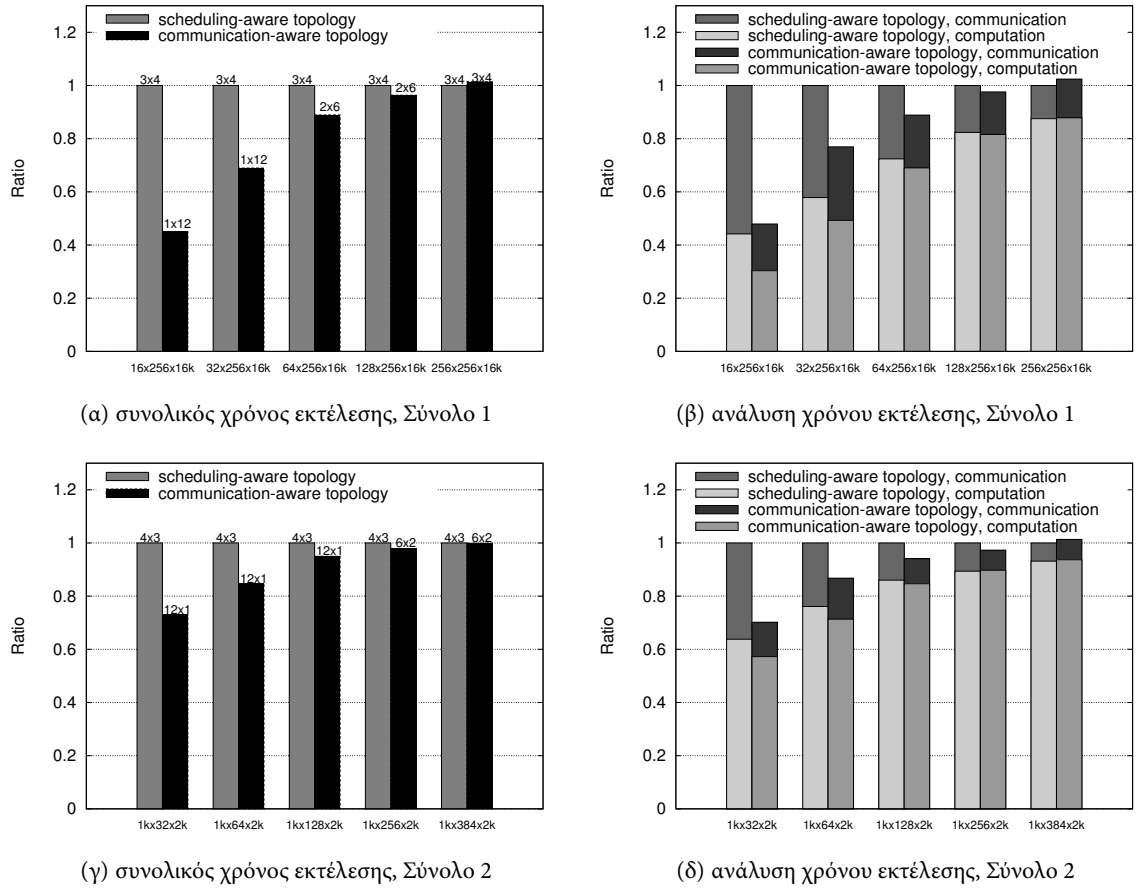
(β) ανάλυση χρόνου εκτέλεσης, Σύνολο 1

(γ) συνολικός χρόνος εκτέλεσης, Σύνολο 2

(δ) ανάλυση χρόνου εκτέλεσης, Σύνολο 2

Σχήμα 6.7: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (DE-XYT, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία twins)*

Τα πειραματικά αποτελέσματα του μετροπρογράμματος DE-XYT απεικονίζονται στο σχήμα 6.7 για 16 διεργασίες, ενώ στο σχήμα 6.8 καταγράφονται οι αντίστοιχες μετρήσεις για 12 διεργασίες. Όπως και στην περίπτωση του μετροπρογράμματος ADI, η βελτίωση του συνολικού χρόνου παράλληλης εκτέλεσης είναι ιδιαίτερα αισθητή στους περισσότερο ασύμμετρους χώρους επαναλήψεων. Έτσι, για τις 16 διεργασίες και το Σύνολο 1 έχουμε βελτίωση του χρόνου εκτέλεσης κατά 60% για το χώρο επαναλήψεων $16 \times 256 \times 16K$, 32% για το χώρο $32 \times 256 \times 16K$, 13% για το χώρο $64 \times 256 \times 16K$ και 5% για το χώρο $128 \times 256 \times 16K$. Ομοίως, στην περίπτωση του Συνόλου 2 παρατηρούμε μείωση του χρόνου



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) ανάλυση χρόνου εκτέλεσης, Σύνολο 1

(γ) συνολικός χρόνος εκτέλεσης, Σύνολο 2

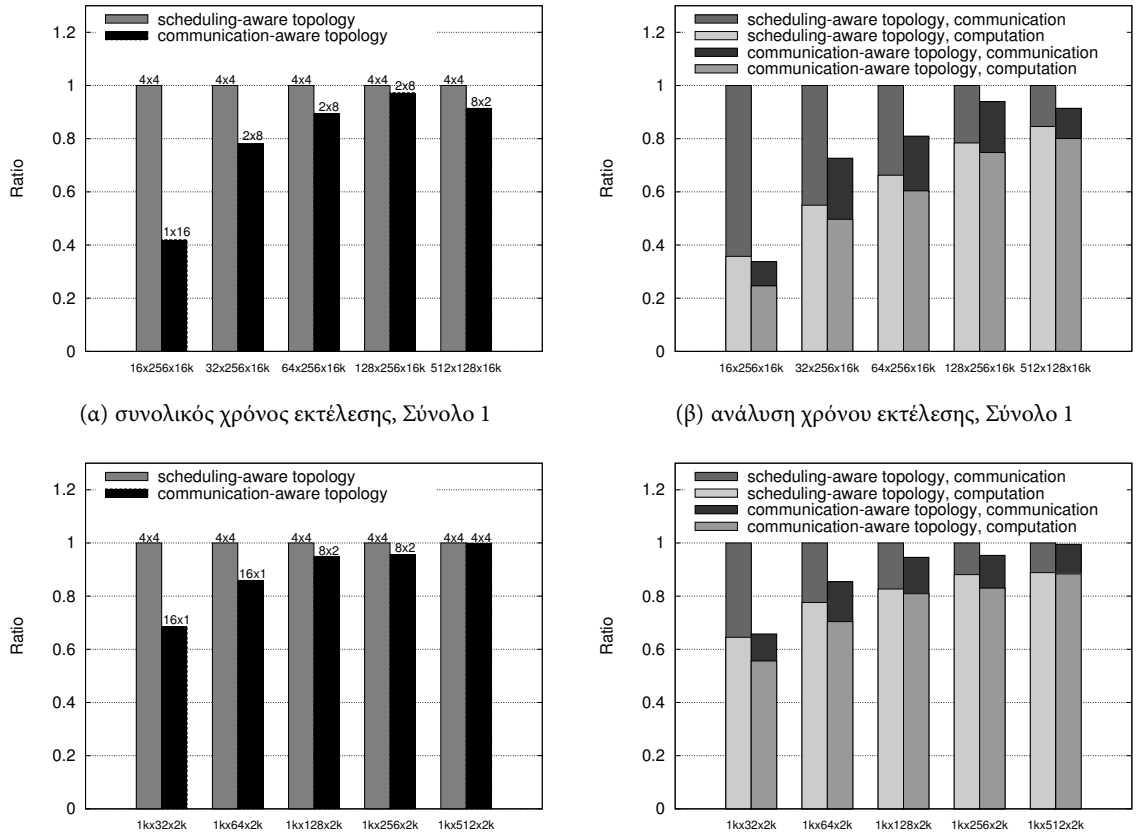
(δ) ανάλυση χρόνου εκτέλεσης, Σύνολο 2

Σχήμα 6.8: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (DE-XYT, διάφοροι χώροι επαναλήψεων, 12 διεργασίες, συστοιχία twins)*

εκτέλεσης κατά 33% για το χώρο $1K \times 32 \times 2K$, 16% για το χώρο $1K \times 64 \times 2K$, 6% για το χώρο $1K \times 128 \times 2K$ και τελικά 2% για το χώρο $1K \times 256 \times 2K$. Ανάλογη είναι η βελτίωση της επίδοσης και για την περίπτωση των 12 διεργασιών.

Από τα παραπάνω εξάγονται δύο βασικές παρατηρήσεις: κατά πρώτον, η ποσοστιαία βελτίωση στο Σύνολο 1 είναι πολύ μεγαλύτερη στην περίπτωση του μετροπρογράμματος DE-XYT απ' ό,τι στον αλγόριθμο ADI. Το γεγονός αυτό είναι αναμενόμενο, καθώς ο αλγόριθμος DE-XYT χαρακτηρίζεται εγγενώς από μεγαλύτερες ανάγκες επικοινωνίας, όπως άλλωστε πιστοποιούν τα διανύσματα εξάρτησης του αλγορίθμου. Έτσι, παρά τις όποιες διαφορές στις υπολογιστικές απαιτήσεις των δύο αλγορίθμων, μια βελτιστοποίηση που αποσκοπεί στην ελαχιστοποίηση της επιβάρυνσης επικοινωνίας αναμένεται να αποφέρει μεγαλύτερα οφέλη στην περίπτωση ενός αλγορίθμου με αυξημένες ανάγκες επικοινωνίας.

Κατά δεύτερον, παρατηρούμε ότι η μείωση του χρόνου εκτέλεσης είναι μικρότερη στην περίπτωση του Συνόλου 2 απ' ότι στο Σύνολο 1, όπως και θεωρητικά αναμενόταν, καθώς το Σύνολο 2 είναι περισσότερο ευαίσθητο στην αρχική καθυστέρηση διάδοσης σε σχέση με το Σύνολο 1. Εντούτοις, σε όλες τις περιπτώσεις που εξετάστηκαν η προτεινόμενη τοπολογία απεικόνισης επιτυγχάνει σημαντική μείωση του χρόνου εκτέλεσης σε σχέση με τη συνήθη τοπολογία ελάχιστης καθυστέρησης διάδοσης.



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) ανάλυση χρόνου εκτέλεσης, Σύνολο 1

(γ) συνολικός χρόνος εκτέλεσης, Σύνολο 2

(δ) ανάλυση χρόνου εκτέλεσης, Σύνολο 2

Σχήμα 6.9: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (DE-XYT, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία xenons)*

Στο σχήμα 6.9 αναπαρίστανται οι χρόνοι εκτέλεσης για τα Σύνολα 1 και 2 κατά την εκτέλεση του αλγορίθμου DE-XYT με 16 διεργασίες στη συστοιχία των xenons. Όπως και στη συστοιχία των twins, παρατηρούμε σημαντική βελτίωση της επίδοσης του προγράμματος, ιδιαίτερα για την περίπτωση περισσότερο ασύμμετρων χώρων επαναλήψεων και για το Σύνολο 1. Συνεπώς, τα πλεονεκτήματα της προτεινόμενης τοπολογίας μπορούν να αναδειχθούν ακόμα και στα πιο προηγμένα τεχνικά χαρακτη-

ριστικά της συστοιχίας των xenons.

6.3.3 DE-TXY

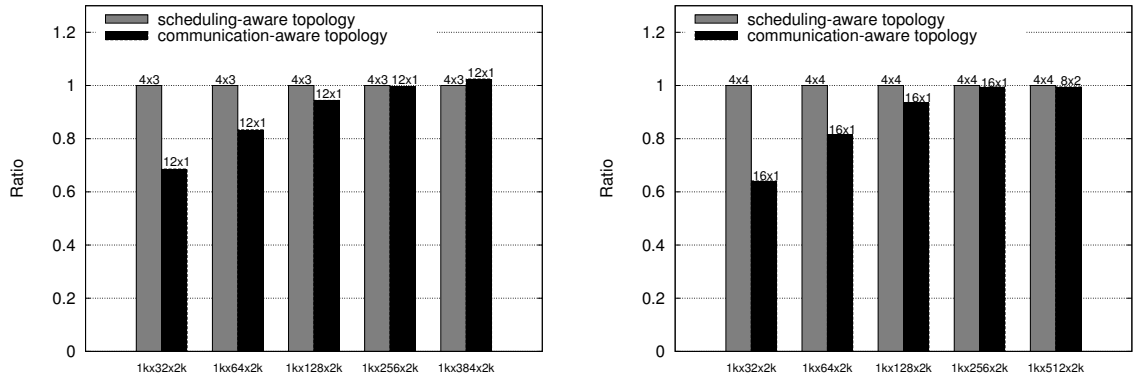
Το μετροπρόγραμμα DE-TXY διαθέτει χώρο επαναλήψεων $T \times X \times Y$, που αντιστοιχεί στην περίπτωση αυξημένης χωρικής πολυπλοκότητας σε σχέση με την αντίστοιχη χρονική. Ουσιαστικά, η περίπτωση αυτή εξετάστηκε αφενός για λόγους πληρότητας και αφετέρου για να διερευνηθεί η επίδραση που θα είχε η αντιμετάθεση των βρόχων, εφόσον η τελευταία επιβαλλόταν από τους συσχετισμούς της χωρικής και χρονικής πολυπλοκότητας του φυσικού προβλήματος. Όπως έχει ήδη αναφερθεί, υπό το εφαρμοζόμενο σχήμα δρομολόγησης συνίσταται η μετάθεση της μεγαλύτερης διάστασης του χώρου επαναλήψεων στην πλέον εσωτερική θέση φωλιάσματος, για να υπάρξει επαρκής εκμετάλλευση του φαινομένου της σωλήνωσης και να ελαχιστοποιηθεί η επιβάρυνση της αρχικής καθυστέρησης διάδοσης του υπολογισμού. Όμως, κατά πόσο η μεγαλύτερη αυτή διάσταση θα είναι κάποια από τις χωρικές διαστάσεις X, Y ή η χρονική διάσταση T, όπως είχαμε υποθέσει στη μορφή DE-XYT του αλγορίθμου, εξαρτάται από τη σημασιολογία και τις απαιτήσεις του αρχικού προβλήματος διάχυσης που καλούμαστε να λύσουμε.

Χώρος επαναλήψεων	***16 διεργασίες***	***12 διεργασίες***
$1K \times 32 \times 2K$	16×1	12×1
$1K \times 64 \times 2K$	16×1	12×1
$1K \times 128 \times 2K$	16×1	12×1
$1K \times 256 \times 2K$	16×1	12×1
$1K \times 384 \times 2K$		12×1
$1K \times 512 \times 2K$	8×2	

Πίνακας 6.4: *Προτεινόμενες τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-TXY και συνολικό πλήθος διεργασιών 16 ή 12*

Στην περίπτωση του μετροπρογράμματος DE-TXY εξετάστηκαν επιλεκτικά μόνο οι χώροι επαναλήψεων του Συνόλου 2, καθότι, όπως φάνηκε στην προηγούμενη ενότητα, οι χώροι επαναλήψεων του Συνόλου 1 επωφελούνται περισσότερο από την εφαρμογή της προτεινόμενης τοπολογίας απεικόνισης συγκριτικά με τους χώρους του Συνόλου 2. Έτσι, εξετάστηκαν οι χώροι επαναλήψεων $1K \times \{32, 64, 128, 256, 384/512\} \times 2K$ τόσο για 12 όσο και για 16 διεργασίες στη συστοιχία twins. Οι προτεινόμενες τοπολογίες απεικόνισης παρέχονται στον πίνακα 6.4, ενώ οι αντίστοιχοι κανονικοποιημένοι χρόνοι εκτέλεσης αναπαρίστανται με τα ραβδογράμματα του σχήματος 6.10. Στην περίπτωση των 12 διεργασιών γίνεται σύγκριση με την τοπολογία 4×3, ενώ για τις 16 διεργασίες χρησιμοποιείται ως τοπολογία αναφοράς η 4×4.

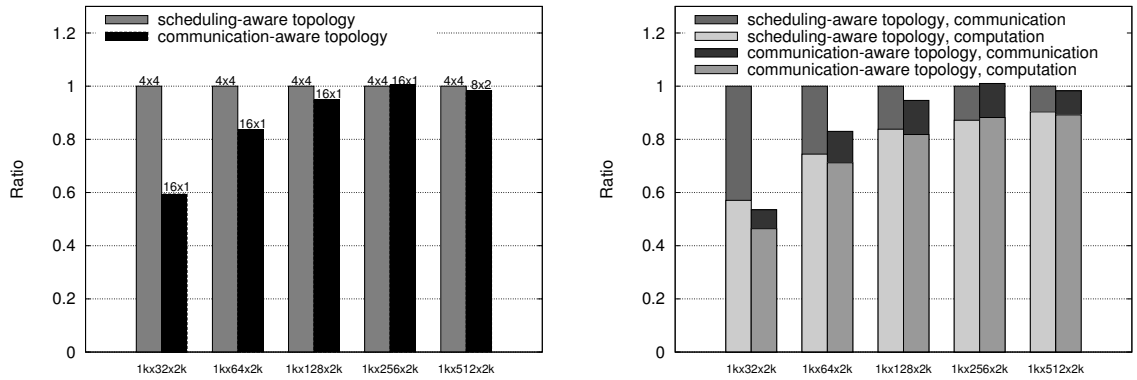
Όπως και στην περίπτωση της XYT μορφής του αλγορίθμου, παρατηρούμε σημαντική βελτίωση στους χρόνους εκτέλεσης κατά την υιοθέτηση της εκάστοτε προτεινόμενης τοπολογίας απεικόνισης.



(α) συνολικός χρόνος εκτέλεσης, 12 διεργασίες

(β) συνολικός χρόνος εκτέλεσης, 16 διεργασίες

Σχήμα 6.10: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (DE-TXY, διάφοροι χώροι επαναλήψεων, 16 και 12 διεργασίες, συστοιχία twins)*



(α) συνολικός χρόνος εκτέλεσης

(β) ανάλυση χρόνου εκτέλεσης

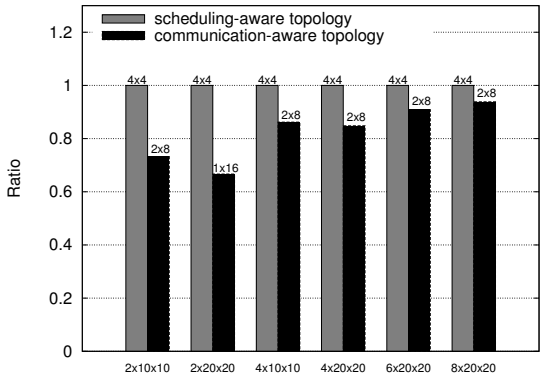
Σχήμα 6.11: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (DE-TXY, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχία xenons)*

Συμπεραίνουμε συνεπώς ότι για τον αλγόριθμο της διάχυσης η τεχνική επιλογής κατάλληλης τοπολογίας απεικόνισης για την ελαχιστοποίηση του όγκου επικοινωνίας είναι ιδιαίτερα επιθυμητή, κυρίως για τους περισσότερο ασύμμετρους χώρους επαναλήψεων. Παρόμοια είναι τα αποτελέσματα και για τη συστοιχία των xenons με 16 διεργασίες, όπως άλλωστε προκύπτει και από την εποπτεία του σχήματος 6.11.

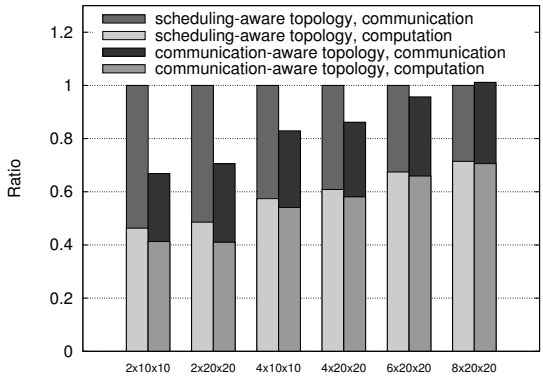
6.3.4 Adv2D

Πεδίο εφαρμογής	*16 διεργασίες*
$2 \times 10 \times 10$	2×8
$2 \times 20 \times 20$	1×16
$4 \times 10 \times 10$	2×8
$4 \times 20 \times 20$	2×8
$6 \times 20 \times 20$	2×8
$8 \times 20 \times 20$	2×8

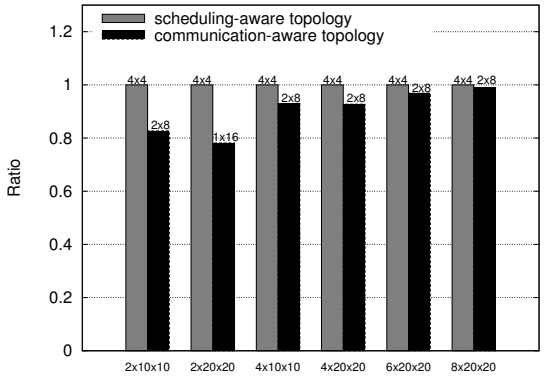
Πίνακας 6.5: *Προτεινόμενες τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα Adv2D και συνολικό πλήθος διεργασιών 16*



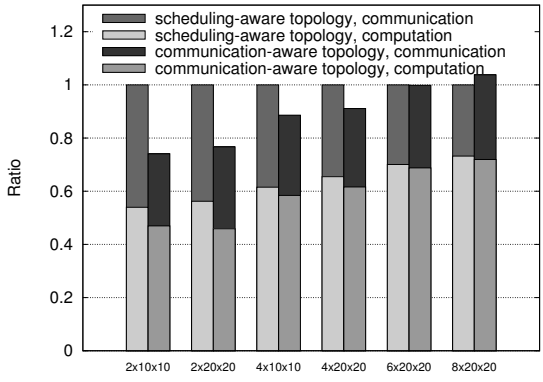
(α) συνολικός χρόνος εκτέλεσης, συστοιχία twins



(β) ανάλυση χρόνου εκτέλεσης, συστοιχία twins



(γ) συνολικός χρόνος εκτέλεσης, συστοιχία xenons



(δ) ανάλυση χρόνου εκτέλεσης, συστοιχία xenons

Σχήμα 6.12: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (Adv2D, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχίες twins και xenons)*

Το μετροπρόγραμμα Adv2D αποτελεί μια ολοκληρωμένη παράλληλη μέθοδο επίλυσης της διακριτοποιημένης εξίσωσης μεταφοράς σε δισδιάστατη επιφάνεια. Για την αξιολόγηση της επίδοσης επιλέξαμε τους χώρους του πίνακα 6.5. Όλοι οι χώροι είναι της μορφής $X \times Y \times T$, όπου $X \times Y$ το φυσικό πεδίο εφαρμογής και T ο πραγματικός χρόνος παρακολούθησης του φαινομένου. Θα πρέπει να τονιστεί ότι οι τιμές αυτές αντιστοιχούν σε *πραγματικά* πεδία εφαρμογής (application domains) και *πραγματικούς* χρόνους, και όχι σε *αλγοριθμικούς* χώρους επαναλήψεων. Το πλέγμα που σαρώνεται αλγοριθμικά κατά τον υπολογισμό των τιμών καθορίζεται σε δεύτερη φάση μέσω της επιλογής των παραμέτρων διακριτοποίησης Δx, Δy και Δt. Επιλέγοντας αυθαίρετα $\Delta x = \Delta y = 0.1$ για ομοιόμορφο χωρικό πλέγμα, λόγω της (6.1) θεωρούμε για το κβάντο χρόνου την τιμή $\Delta t = 0.001$, ώστε να διασφαλιστεί η ευστάθεια της μεθόδου επίλυσης βάσει της συνθήκης Courant.

Το σχήμα 6.12 απεικονίζει τους χρόνους εκτέλεσης για το μετροπρόγραμμα Adv2D. Σε όλες τις περιπτώσεις γίνεται σύγκριση με την 4×4 τοπολογία διεργασιών, ενώ καταγράφονται μετρήσεις τόσο για τη συστοιχία twins όσο και για εκείνη των xenons. Για κάθε συστοιχία και χώρο επαναλήψεων παρουσιάζεται και η ποιοτική ανάλυση του συνολικού χρόνου εκτέλεσης σε χρόνο υπολογισμού και χρόνο επικοινωνίας, ώστε να καταδειχθεί ότι το συγκριτικό πλεονέκτημα της εκάστοτε προτεινόμενης τοπολογίας μπορεί πράγματι να αποδοθεί σε μείωση του συνολικού χρόνου επικοινωνίας. Όπως σε όλα τα προηγούμενα μετροπρογράμματα, έτσι και στην περίπτωση του μετροπρογράμματος Adv2D παρατηρήθηκαν σημαντικές βελτιώσεις της επίδοσης του παράλληλου προγράμματος κατά την υιοθέτηση της τοπολογίας ελάχιστης επικοινωνίας, που κυμαίνονται ανάλογα με το χώρο επαναλήψεων μεταξύ 7-34% για τη συστοιχία των twins και 1-22% για τη συστοιχία των xenons με το ταχύτερο δίκτυο διασύνδεσης.

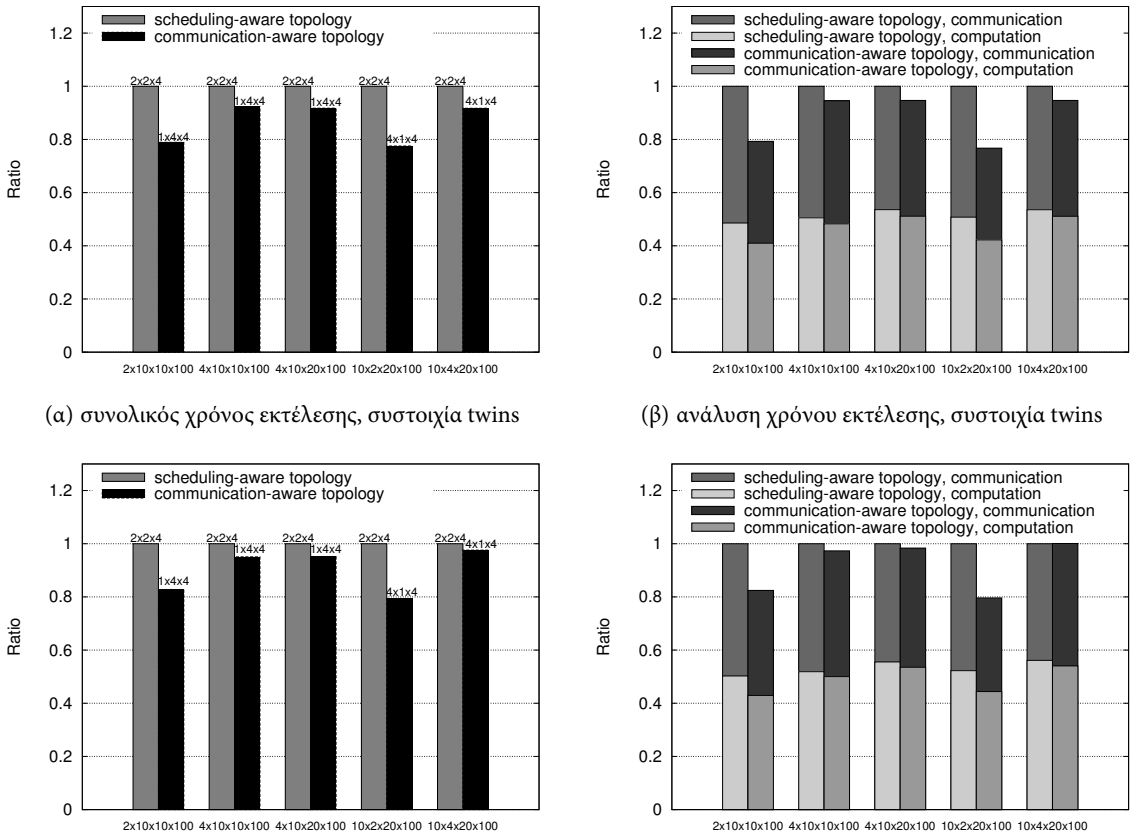
6.3.5 Adv3D

Ειδικά για τη βελτιστοποίηση που αφορά στην επιλογή κατάλληλης τοπολογίας διεργασιών υλοποιήθηκε και αξιολογήθηκε επιπλέον ο διακριτοποιημένος αλγόριθμος επίλυσης της εξίσωσης μεταφοράς σε τρισδιάστατο χωρίο $X \times Y \times Z$. Η σκοπιμότητα μιας τέτοιας επιλογής έγκειται στην ιδιομορφία της προτεινόμενης βελτιστοποίησης, που από μαθηματικής πλευράς αναλύει το δεδομένο πλήθος των διαθέσιμων διεργασιών σε ένα κατάλληλο γινόμενο N παραγόντων, όπου $N + 1$ η διάσταση του υπό εξέταση αλγορίθμου. Έχοντας μέχρι στιγμής εξετάσει αποκλειστικά τρισδιάστατους αλγορίθμους φωλιασμένων βρόχων, περιοριστήκαμε στην επαλήθευση της επίδοσης της μεθοδολογίας για γινόμενα δύο παραγόντων, που αντιστοιχούν σε δισδιάστατη καρτεσιανή τοπολογία διεργασιών. Στην ενότητα αυτή θα εξεταστεί επιπλέον η επίδραση της προτεινόμενης βελτιστοποίησης σε ένα τετραδιάστατο αλγόριθμο φωλιασμένων βρόχων, όπου οι δυνατότητες επιλογής της τρισδιάστατης καρτεσιανής τοπολογίας διεργασιών είναι σαφώς περισσότερες.

Τα φυσικά πεδία εφαρμογής που επελέγησαν για την πειραματική αξιολόγηση, μαζί με τις προτει-

Πεδίο εφαρμογής	*16 διεργασίες*
$2 \times 10 \times 10 \times 100$	$1 \times 4 \times 4$
$4 \times 10 \times 10 \times 100$	$1 \times 4 \times 4$
$4 \times 10 \times 20 \times 100$	$1 \times 4 \times 4$
$10 \times 2 \times 20 \times 100$	$4 \times 1 \times 4$
$10 \times 4 \times 20 \times 100$	$4 \times 1 \times 4$

Πίνακας 6.6: *Προτεινόμενες τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα Adv3D και συνολικό πλήθος διεργασιών 16*



(α) συνολικός χρόνος εκτέλεσης, συστοιχία twins

(β) ανάλυση χρόνου εκτέλεσης, συστοιχία twins

(γ) συνολικός χρόνος εκτέλεσης, συστοιχία xenons

(δ) ανάλυση χρόνου εκτέλεσης, συστοιχία xenons

Σχήμα 6.13: *Σύγκριση συνήθους και προτεινόμενης τοπολογίας διεργασιών (Adv3D, διάφοροι χώροι επαναλήψεων, 16 διεργασίες, συστοιχίες twins και xenons)*

νόμενες τοπολογίες σε κάθε περίπτωση παρέχονται στον πίνακα 6.6. Σε όλες τις περιπτώσεις έγινε σύγκριση με τη $2 \times 2 \times 4$ τοπολογία, που αφενός ελαχιστοποιεί την αρχική καθυστέρηση διάδοσης για 16 διεργασίες και αφετέρου εναρμονίζεται με τα εξεταζόμενα πεδία εφαρμογής, για τα οποία ισχύει

$Z \geq X, Y$. Όλοι οι παρεχόμενοι χώροι είναι της μορφής $X \times Y \times Z \times T$ και επελέγησαν με την υπόθεση της παρακολούθησης του φαινομένου σε ορθογώνιο χωρίο $\{2, 4, 10\} \times \{2, 4, 10\} \times \{10, 20\}$ για 100 χρονικές στιγμές. Στην πράξη, εξετάστηκαν ακόμα περισσότεροι συνδυασμοί, αλλά παρουσιάζονται μόνο εκείνες οι περιπτώσεις που η προτεινόμενη τοπολογία απεικόνισης διέφερε από τη $2 \times 2 \times 4$ τοπολογία ελάχιστης καθυστέρησης διάδοσης. Για τις παραμέτρους διακριτοποίησης θεωρήσαμε αυθαίρετα ότι $\Delta x = \Delta y = \Delta z = 0.5$, οπότε βάσει της (6.3) επιλέγουμε κβάντο χρόνου $\Delta t = 0.003$ για τη διασφάλιση της ευστάθειας της μεθόδου.

Τα πειραματικά αποτελέσματα για το μετροπρόγραμμα Adv3D απεικονίζονται στο σχήμα 6.13. Συνολικά, ανάλογα με το συγκεκριμένο χώρο επαναλήψεων παρατηρήσαμε βελτίωση της επίδοσης μεταξύ 8-23% για τη συστοιχία των twins και 6-21% για τη συστοιχία των xenons. Τα αποτελέσματα αυτά είναι ιδιαίτερα σημαντικά και έχουν μεγάλη πρακτική αξία, ιδίως αν αναλογιστεί κανείς τη σχετικά αυξημένη δυσκολία στην επιλογή κατάλληλης τρισδιάστατης τοπολογίας διεργασιών από το μεγάλο πλήθος έγκυρων συνδυασμών.

6.3.6 Συμπεράσματα

Συνοψίζοντας, η αξιοποίησης της πληροφορίας του χώρου επαναλήψεων και των εξαρτήσεων δεδομένων ενός αλγορίθμου στον καθορισμό κατάλληλης τοπολογίας διεργασιών προς απεικόνιση του ισοδύναμου παράλληλου προγράμματος κρίνεται ιδιαίτερα σημαντική όταν:

- ο αλγόριθμος εμφανίζει ασυμμετρία όσον αφορά στις εξαρτήσεις δεδομένων ή/και τις διαστάσεις του χώρου επαναλήψεων
- ο αλγόριθμος επιβάλλει υψηλές απαιτήσεις επικοινωνίας σε σχέση με τις αντίστοιχες απαιτήσεις υπολογισμού, για δεδομένη υποδομή υλικού και λογισμικού

6.4 Εξισορρόπηση Φορτίου Νημάτων

Το δεύτερο σκέλος της πειραματικής διαδικασίας εξετάζει την αξιολόγηση της επίδοσης των προτεινόμενων σχημάτων εξισορρόπησης φορτίου στα υβριδικά προγραμματιστικά μοντέλα. Για το σκοπό αυτό χρησιμοποιήθηκαν τα μετροπρογράμματα ADI, DE-XYT, DE-TXY και Adv2D της ενότητας 6.3 και ελήφθησαν μετρήσεις του συνολικού χρόνου παράλληλης εκτέλεσης στις συστοιχίες twins και xenons. Στην παρούσα ενότητα θα συγκρίνουμε τα προτεινόμενα υβριδικά παράλληλα προγραμματιστικά μοντέλα, ενώ στην ενότητα 6.5 θα επιχειρήσουμε επιπλέον σύγκριση και με το μοντέλο ανταλλαγής μηνυμάτων. Η διάκριση αυτή γίνεται αφενός για να εστιάσουμε στην αποτελεσματικότητα των σχημάτων εξισορρόπησης φορτίου ως προς αυτό το υβριδικό μοντέλο καθαυτό και αφετέρου γιατί η απευθείας

σύγκριση μεταξύ μοντέλου ανταλλαγής μηνυμάτων και υβριδικών προγραμματιστικών μοντέλων, με ή χωρίς την εφαρμογή κάποιας στρατηγικής εξισορρόπησης φορτίου των νημάτων, αντανακλά σε μεγάλο βαθμό τη σχετική επίδοση των χρησιμοποιούμενων βιβλιοθηκών ανταλλαγής μηνυμάτων και πολυνηματικής επεξεργασίας. Αναφορικά με το τελευταίο σημείο, είναι προφανές ότι η διαδεδομένη χρήση της βιβλιοθήκης MPICH και η επιτυχημένη παραλληλοποίηση μέσω αυτής πληθώρας εφαρμογών σε συστοιχίες χιλιάδων επεξεργαστών αντιπαρατίθεται με τη σαφώς πιο περιορισμένη βελτιστοποίηση της υποστήριξης του προτύπου OpenMP στο μεταγλωττιστή C/C++ της Intel για την ανάδειξη της επίδοσης υβριδικών προγραμμάτων.

Κατά τη θεωρητική μοντελοποίηση της συμπεριφοράς του υφιστάμενου υλικού και λογισμικού υιοθετείται μια απλουστευμένη προσέγγιση για τον ορισμό των ποσοτήτων t_{comp} και t_{comm} της (5.2). Αναφορικά με την παράμετρο t_{comp}, θεωρούμε ότι το κόστος που σχετίζεται με τον υπολογισμό x επαναλήψεων ισούται προσεγγιστικά με x φορές επί το μέσο κόστος που απαιτείται για τον υπολογισμό μίας επανάληψης. Επίσης, το κόστος επικοινωνίας t_{comm} προσεγγίζεται ως υπέρθεση μιας σταθερής συνιστώσας αρχικοποίησης και ενός γραμμικού όρου που είναι ανάλογος προς το μέγεθος s του μηνύματος και εκφράζει τη δυνατότητα παροχής του δικτύου διασύνδεσης σε επίπεδο εφαρμογής. Τυπικά ορίζουμε

$$t_{comp}(x) = xt_{comp}(1) \tag{6.4}$$

$$t_{comm}(s) = t_{startup} + \frac{s}{B_{sustained}} \tag{6.5}$$

όπου $t_{comp}(1)$ ο μέσος χρόνος υπολογισμού μίας επανάληψης και $t_{startup}$, $B_{sustained}$ η αρχική καθυστέρηση διάδοσης και ο ρυθμός παροχής του δικτύου διασύνδεσης, αντίστοιχα. Δεδομένου ότι ως πρωταρχικό στόχο θέσαμε τη διατήρηση της απλότητας και της εφαρμοσιμότητας στη μοντελοποίηση των αρχιτεκτονικών παραμέτρων, αποφύγαμε τη συνεκτίμηση περισσότερο πολύπλοκων φαινομένων, όπως αυτά που σχετίζονται με τη συμπεριφορά της κρυφής μνήμης ή την ακριβή μοντελοποίηση της επικοινωνίας. Έτσι, παρότι είχε προηγηθεί μελέτη του πηγαίου κώδικα του MPICH με στόχο την εις βάθος κατανόηση της λειτουργίας της ADI-2 εικονικής συσκευής ch_p4, προτιμήθηκε η μη ενσωμάτωση των πορισμάτων της μελέτης αυτής στο θεωρητικό μοντέλο, ώστε να διατηρηθεί η γενικότητα της μεθόδου. Το ίδιο ισχύει και για τα φαινόμενα κρυφής μνήμης κατά τη μοντελοποίηση της ιεραρχίας μνήμης, καθώς κάτι τέτοιο θα απαιτούσε μια σχετικά πολύπλοκη ανάλυση των προσπελάσεων ανάγνωσης και εγγραφής του προγράμματος ως προς τη δεδομένη αρχιτεκτονική. Μια ακριβέστερη μοντελοποίηση των παραμέτρων αυτών θα οδηγούσε πιθανότατα σε αποδοτικότερα, αλλά και περισσότερο σύνθετα, σχήματα εξισορρόπησης φορτίου.

Μια άλλη βασική δυσκολία που αντιμετωπίσαμε ήταν η μοντελοποίηση της TCP/IP επικοινωνίας

μέσω sockets, στην οποία καταφεύγει τελικά η υλοποίηση MPICH για την ανταλλαγή μηνυμάτων μέσω του δικτύου Ethernet. Κατά τη θεωρητική εκτίμηση του κόστους επικοινωνίας υποθέσαμε διακριτές μη επικαλυπτόμενες φάσεις υπολογισμού και επικοινωνίας και σχετικά υψηλό ρυθμό παροχής του δικτύου διασύνδεσης, όπως αναλύθηκε στην ενότητα 4.5. Εντούτοις, η παραδοχή αυτή αφενός υποτιμά το κόστος επικοινωνίας για σχετικά μικρά μηνύματα, που επιβαρύνονται σημαντικά από την αρχική καθυστέρηση διάδοσης του δικτύου και επιτυγχάνουν χαμηλό ρυθμό παροχής, ενώ αφετέρου υπερτιμά το αντίστοιχο κόστος για σχετικά μεγάλα μηνύματα, για τα οποία η μονάδα DMA αναλαμβάνει σημαντικό τμήμα της επικοινωνίας, ανακουφίζοντας έτσι τον επεξεργαστή και επιτρέποντας την επικάλυψη της επικοινωνίας με ωφέλιμο υπολογισμό. Καθώς όμως ως πρωταρχική προτεραιότητα θέσαμε την ποιοτική αξιολόγηση της αποδοτικότητας και της χρηστικότητας των επιμέρους σχημάτων εξισορρόπησης φορτίου, προτιμήθηκε μια προσέγγιση απλουστευμένης θεωρητικής μοντελοποίησης.

Έτσι, στην ανάλυσή μας θεωρήσαμε τις τιμές του πίνακα 6.7 για τις παραμέτρους t_{comp}, $t_{startup}$ και $B_{sustained}$ των (6.4) και (6.5). Οι τιμές του μέσου χρόνου υπολογισμού ανά επανάληψη προσδιορίστηκαν από δοκιμαστική εκτέλεση του εκάστοτε μετροπρογράμματος για ένα χώρο επαναλήψεων που ήταν κατά πολλές τάξεις μεγέθους μικρότερος από τους πειραματικούς χώρους επαναλήψεων που θα εξεταστούν, ώστε η όλη διαδικασία να απαιτεί αμελητέο χρόνο σε σχέση με τους χρόνους εκτέλεσης των πειραμάτων. Η αρχική καθυστέρηση του δικτύου εξισώθηκε προσεγγιστικά με το χρόνο round-trip (round-trip time), όπως είθισται να γίνεται σε αυτήν την περίπτωση. Τέλος, ο ρυθμός παροχής δικτύου υπολογίστηκε με χρήση μετρήσεων ping-pong σε δοκιμαστικό πρόγραμμα MPI. Από τις υπολογισθείσες τιμές για το ρυθμό παροχής του δικτύου, ως $B_{sustained}$ θεωρήσαμε αυθαίρετα την τιμή που αντιστοιχεί σε ένα ενδιάμεσο μέγεθος μηνύματος, έστω 32 KB. Άλλωστε, το MPICH διαφοροποιεί το πρωτόκολλο επικοινωνίας για μεγέθη μηνύματος μικρότερα από 1024 bytes (short), μεταξύ 1024 και 128000 bytes (eager) και μεγαλύτερα από 128000 bytes (rendezvous, βλέπε και παράρτημα Γ). Συνεπώς, η τιμή 32 KB, παρότι αυθαίρετη, αποτελεί μια εύλογη μέση τιμή μεγέθους μηνύματος.

Παράμετρος		***Συστοιχία twins***	***Συστοιχία xenons***
Χρόνος υπολογισμού επανάληψης t_{comp} (nsec)	*ADI*	238	75
	DE-XYT	485	120
	DE-TXY	390	100
	Adv2D	270	115
Αρχική καθυστέρηση δικτύου $t_{startup}$ (μsec)		107	80
Ρυθμός παροχής δικτύου $B_{sustained}$ (MB/s)		11.92	150

Πίνακας 6.7: *Παράμετροι εξισορρόπησης φορτίου για συστοιχίες twins και xenons.*

Στο σημείο αυτό μπορούμε να συνοψίσουμε τα βασικότερα μειονεκτήματα των απλοποιήσεων που υιοθετήθηκαν κατά τη μοντελοποίηση της συμπεριφοράς του συστήματος για την εφαρμογή των τε-

χνικών εξισορρόπησης φορτίου:

- Θεωρήσαμε ότι ο χρόνος υπολογισμού που σχετίζεται με τις επαναλήψεις ενός υπερκόμβου ισούται με το μέσο χρόνο που απαιτείται για μία επανάληψη πολλαπλασιασμένο με το συνολικό πλήθος των επαναλήψεων. Εντούτοις, κάτι τέτοιο αντιβαίνει με τις αρχιτεκτονικές αρχές της ιεραρχίας μνήμης, καθώς ο χρόνος προσπέλασης δεδομένων δεν ακολουθεί το γραμμικό μοντέλο και εξαρτάται από το κατά πόσο το δεδομένο που προσπελάζουμε φυλάσσεται σε κάποιο καταχωρητή, στην κρυφή μνήμη ή στην κύρια μνήμη.

- Επιπλέον, για κάθε μετροπρόγραμμα θεωρούμε έναν ενιαίο, αντιπροσωπευτικό μέσο χρόνο υπολογισμού ανά επανάληψη, ανεξαρτήτως του χώρου επαναλήψεων που εξετάζεται στο εκάστοτε πείραμα. Η παραδοχή αυτή επίσης αγνοεί την ιεραρχία μνήμης, καθώς η τιμή του μέσου χρόνου υπολογισμού ανά επανάληψη διαφοροποιείται ανάλογα με το δοκιμαστικό χώρο επαναλήψεων που θα επιλέξουμε για τον πειραματικό προσδιορισμό της. Παρότι ενδεχομένως θα μπορούσαμε για κάθε περίπτωση πραγματικού χώρου επαναλήψεων να επιλέγουμε έναν κατάλληλο δοκιμαστικό, που σε μικρή κλίμακα να διατηρεί ποιοτικά τα μορφολογικά χαρακτηριστικά του πρώτου, επιλέξαμε να υπολογίσουμε το μέσο χρόνο σε όλες τις περιπτώσεις με τη βοήθεια ενός κοινού δοκιμαστικού χώρου επαναλήψεων, για να διατηρήσουμε κατά το δυνατόν την απλότητα της προτεινόμενης μεθοδολογίας.

- Η επιλογή της γραμμικής προσέγγισης για το κόστος επικοινωνίας υπό τη μορφή $t_{startup} + s \times t_{data}$ αποτελεί υπεραπλούστευση του τρόπου λειτουργίας της επικοινωνίας μέσω ανταλλαγής μηνυμάτων. Σε κατάλληλα μεγέθη μηνυμάτων, ο επεξεργαστής προγραμματίζει τη μηχανή Άμεσης Προσπέλασης Μνήμης (Direct Memory Access - DMA) του δικτυακού προσαρμογέα με όλες τις πληροφορίες για τη μεταφορά των δεδομένων. Τη μεταφορά αυτή μπορεί ακολούθως να διεκπεραιώσει ο προσαρμογέας, επιτρέποντας έτσι ταυτόχρονα στον επεξεργαστή να προχωρήσει στην εκτέλεση άλλων εντολών. Εντούτοις, κατά την αποστολή ενός μηνύματος, η δικτυακή στοίβα πρωτοκόλλων TCP/IP προδιαγράφει την αντιγραφή των δεδομένων επικοινωνίας από το χώρο χρήστη στο χώρο πυρήνα και τη μεταγωγή περιβάλλοντος στον τελευταίο, όπου παράλληλα με την αντιγραφή των δεδομένων σε μια νέα θέση εκτελείται επιπλέον ο υπολογισμός του TCP checksum αθροίσματος και η προσθήκη TCP/IP επικεφαλίδων. Η μηχανή DMA ενδέχεται να αναλάβει την αντιγραφή των δεδομένων από το χώρο πυρήνα στο δικτυακό προσαρμογέα για τη μετέπειτα αποστολή τους, είναι όμως σαφές ότι η διαδικασία της επικοινωνίας έχει ήδη απασχολήσει σε μεγάλο βαθμό τον επεξεργαστή, ενώ δεν είναι εύκολο να προβλεφθεί θεωρητικά κατά πόσο ο τελευταίος θα διευκολυνθεί τελικά από τη μηχανή DMA. Άλλωστε, όπως αναλύεται και στο παράρτημα Γ, η ίδια η βιβλιοθήκη MPICH αφενός διατηρεί έλεγχο ροής των μηνυμά-

των, βάσει του οποίου αναμένει επιβεβαίωση λήψης ανά συγκεκριμένο αριθμό εκκρεμών πακέτων (στην τρέχουσα υλοποίηση, `MPI_Pk_hiwater` πακέτα στον αποστολέα και `MPI_Pk_ackmark` πακέτα στον παραλήπτη), ενώ αφετέρου διαχειρίζεται τις διαδικασίες αποστολής και λήψης μέσω της διεπαφής των sockets για TCP/IP δίκτυα και τριών διαφορετικών σημασιολογικών πρωτοκόλλων (short/eager/rendezvous). Συνεκτιμώντας όλα τα παραπάνω, αποφασίσαμε να υιοθετήσουμε γραμμικό μοντέλο για το κόστος επικοινωνίας με σχετικά μεγάλο ρυθμό παροχής δικτύου, καθώς η προσέγγιση αυτή συνδυάζει τη θεωρητική απλότητα με την παραδοχή μιας κατά κανόνα εύρυθμης και αποδοτικής δικτυακής λειτουργίας.

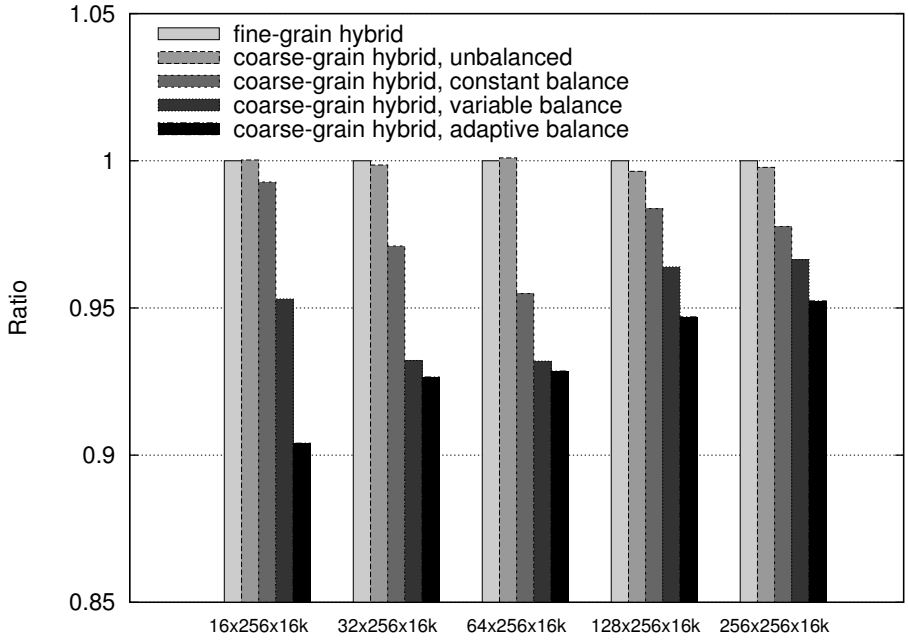
6.4.1 ADI

Τα πειραματικά αποτελέσματα της αξιολόγησης των σχημάτων εξισορρόπησης φορτίου στη συστοιχία twins για τη μέθοδο ADI και διάφορους χώρους επαναλήψεων απεικονίζονται στο σχήμα 6.14. Τα αποτελέσματα είναι κανονικοποιημένα ως προς τους χρόνους εκτέλεσης του υβριδικού μοντέλου λεπτού κόκκου για διευκόλυνση της ποσοστιαίας σύγκρισης. Επιπλέον, μετρήσεις για μεταβλητό κόκκο παραλληλισμού, ή ισοδύναμα μεταβλητό ύψος υπερκόμβου z, παρέχονται στο σχήμα 6.15 για τους ίδιους χώρους επαναλήψεων. Σε όλες τις περιπτώσεις χρησιμοποιήθηκαν 8 διεργασίες στους ισάριθμους SMP κόμβους της συστοιχίας, ενώ σε κάθε SMP κόμβο κάθε διεργασία εκκινεί 2 νήματα υπολογισμού. Για κάθε χώρο επαναλήψεων, η τοπολογία διεργασιών που εφαρμόστηκε προκύπτει με κριτήριο την ελαχιστοποίηση του όγκου των δεδομένων επικοινωνίας και αναγράφεται στον πίνακα 6.8. Συνεπώς, έχει ήδη εφαρμοστεί σε πρώτο επίπεδο η βελτιστοποίηση που αφορά στην ελαχιστοποίηση του συνολικού όγκου των δεδομένων επικοινωνίας, καθώς όπως φάνηκε στην ενότητα 6.3 η εν λόγω τεχνική μειώνει σημαντικά τους συνολικούς χρόνους εκτέλεσης. Για την τοπολογία των νημάτων εξετάστηκαν και οι δύο εναλλακτικές δυνατότητες για δύο νήματα, δηλαδή τόσο η 1×2 όσο και η 2×1 τοπολογία, και σε κάθε περίπτωση επιλέγεται εκείνη με την καλύτερη επίδοση. Η ευελιξία στην επιλογή τοπολογίας νημάτων αποτελεί ένα από τα πλεονεκτήματα του υβριδικού μοντέλου, και καθώς εξαρτάται πρωτίστως από την εκμετάλλευση της τοπικότητας αναφοράς για την αξιοποίηση της ιεραρχίας μνήμης δεν θα μας απασχολήσει στο πλαίσιο της παρούσας εργασίας.

Από την επισκόπηση των πειραματικών αποτελεσμάτων παρατηρούμε ότι τα σχήματα εξισορρόπησης φορτίου που επιτυγχάνουν την καλύτερη απόδοση είναι εκείνα της μεταβλητής και της δυναμικής εξισορρόπησης. Συνολικά, τα σχήματα αυτά αποτελούν την πιο αξιόπιστη λύση για επίτευξη ελάχιστου χρόνου εκτέλεσης με χρήση υβριδικού μοντέλου, καθώς επιτυγχάνουν σε όλες τις περιπτώσεις καλύτερη επίδοση σε σχέση τόσο με το υβριδικό μοντέλο λεπτού κόκκου όσο και με το αντίστοιχο funneled χονδρού κόκκου χωρίς εξισορρόπηση φορτίου. Έτσι, για τη συστοιχία των twins παρατηρούμε ότι το σχήμα μεταβλητής εξισορρόπησης φορτίου επιτυγχάνει μείωση του χρόνου εκτέλεσης κατά 2-8% σε

Χώρος επαναλήψεων	*Τοπολογία διεργασιών*
$16 \times 256 \times 16K$	1×8
$32 \times 256 \times 16K$	1×8
$64 \times 256 \times 16K$	1×8
$128 \times 256 \times 16K$	2×4
$256 \times 256 \times 16K$	2×4

Πίνακας 6.8: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα ADI και συνολικό πλήθος διεργασιών 8*

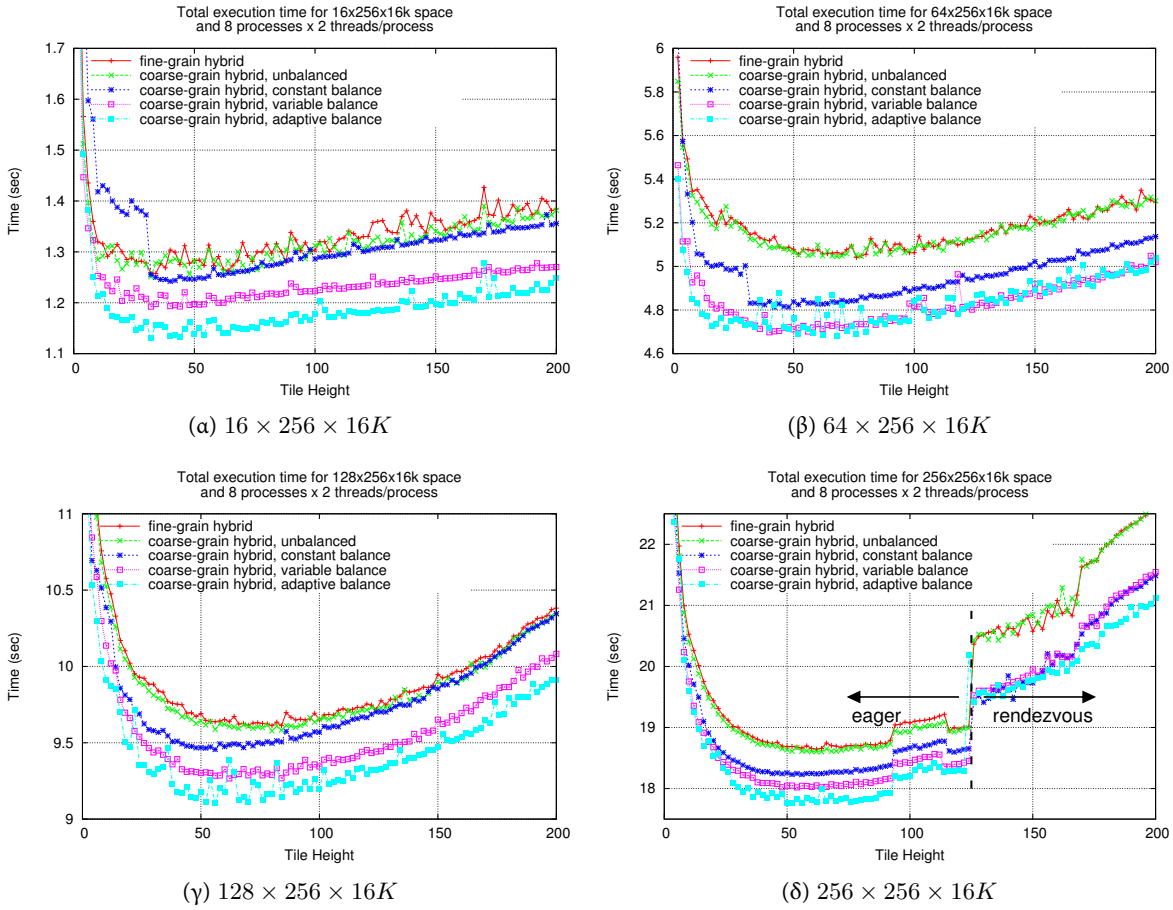


Σχήμα 6.14: *Σύγκριση υβριδικών μοντέλων (ADI, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins)*

σχέση τόσο με το μοντέλο λεπτού κόκκου όσο και με εκείνο του χονδρού κόκκου, ενώ το σχήμα δυναμικής εξισορρόπησης φορτίου επιτυγχάνει βελτίωση στο εύρος 5-10%. Το σχήμα σταθερής εξισορρόπησης φορτίου επιτυγχάνει σαφώς μικρότερη ποσοστιαία βελτίωση στο εύρος 2-5%, καθώς σε όλες τις τοπολογίες διεργασιών του πίνακα 6.8 ένα σημαντικό ποσοστό των διεργασιών είναι συνοριακές, για τις οποίες η εφαρμογή σταθερής εξισορρόπησης φορτίου υπερτιμά το κόστος επικοινωνίας που τους αναλογεί.

Το απλό funneled υβριδικό μοντέλο χονδρού κόκκου αδυνατεί να υπερκεράσει την επίδοση του μοντέλου λεπτού κόκκου, αποτυγχάνοντας έτσι να μετουσιώσει τα θεωρητικά του πλεονεκτήματα σε οφέλη απόδοσης. Το γεγονός αυτό οφείλεται στη μη αποδοτική εγγενή κατανομή του συνολικού φορ-

τίου μεταξύ των νημάτων, καθώς το πρωτεύον νήμα είναι περισσότερο επιβαρυμένο σε σχέση με τα υπόλοιπα νήματα της διεργασίας και καθορίζει τελικά το χρόνο που απαιτείται για την ολοκλήρωση ενός βήματος της παράλληλης εκτέλεσης υπό τη χρονοδρομολόγηση υπερεπιπέδων.



(α) $16 \times 256 \times 16K$ (β) $64 \times 256 \times 16K$

(γ) $128 \times 256 \times 16K$ (δ) $256 \times 256 \times 16K$

Σχήμα 6.15: *Σύγκριση υβριδικών μοντέλων για μεταβλητό κόκκο παραλληλισμού (ADI, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins). Στο χώρο επαναλήψεων $256 \times 256 \times 16k$ σημειώνεται το κατώφλι μεταξύ του eager και του rendezvous πρωτοκόλλου επικοινωνίας του MPICH, που συνοδεύεται με σημαντική πτώση στην απόδοση. Σε όλους τους χώρους επαναλήψεων και για όλα τα ύψη υπερκόμβου υπερέχουν τα σχήματα μεταβλητής και δυναμικής εξισορρόπησης φορτίου.*

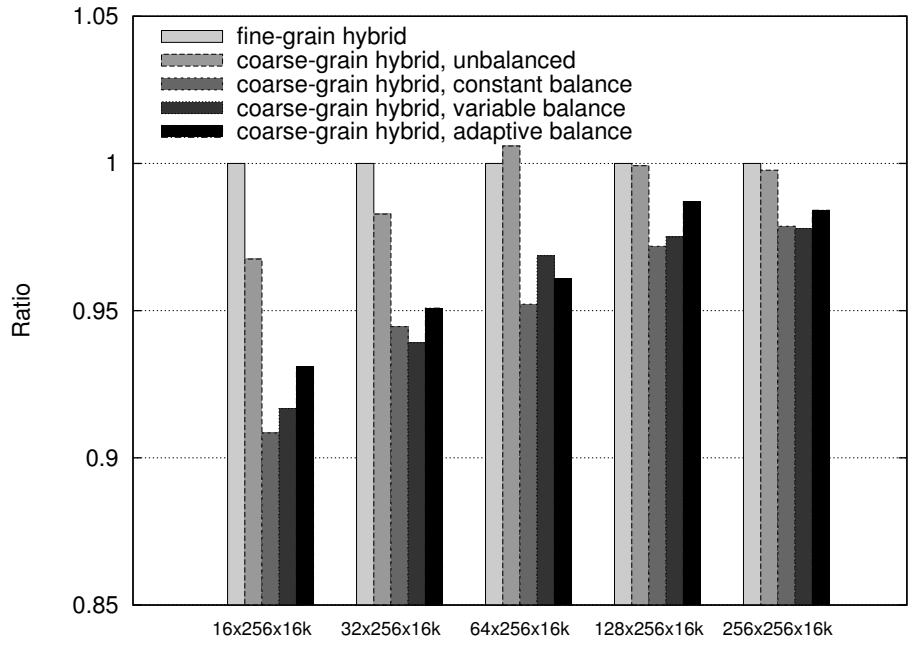
Στο σχήμα 6.15 απεικονίζονται αναλυτικές μετρήσεις του συνολικού χρόνου εκτέλεσης για όλα τα υβριδικά μοντέλα και εύρος υψών υπερκόμβου στην περιοχή 2...200. Παρατηρούμε ότι για όλο το εξεταζόμενο εύρος υψών υπερκόμβου τα σχήματα μεταβλητής και δυναμικής εξισορρόπησης φορτίου υπερτερούν, καθώς παρέχουν τους ελάχιστους χρόνους εκτέλεσης. Είναι πάντως σημαντικό να

προσθέσουμε πως σε όλες τις περιπτώσεις στους χρόνους εκτέλεσης επικρατεί το κομμάτι του υπολογισμού έναντι του τμήματος της επικοινωνίας λόγω της φύσης της εφαρμογής. Συνεπώς, τα οφέλη που αποκομίζουμε με την εφαρμογή εξισορρόπησης φορτίου είναι ιδιαίτερα σημαντικά, δεδομένου ότι προέρχονται από τεχνική που αποσκοπεί ουσιαστικά στην άμβλυνση της επιβάρυνσης του κόστους επικοινωνίας για το πρωτεύον νήμα.

Παρατηρούμε ότι στο σχήμα 6.15(δ) εμφανίζεται μια σημαντική πτώση της απόδοσης σε όλα τα υβριδικά μοντέλα για ύψη υπερκόμβου μεγαλύτερα από την τιμή $z = 125$. Το φαινόμενο αυτό παρατηρήθηκε σε όλους τους χώρους επαναλήψεων για κάποιες τιμές του z και οφείλεται στη διαφοροποίηση του πρωτοκόλλου επικοινωνίας που πραγματοποιεί στο εν λόγω κατώφλι η υλοποίηση MPICH. Πιο συγκεκριμένα, το MPICH χρησιμοποιεί ανάλογα με το μέγεθος του μηνύματος τρία διαφορετικά πρωτόκολλα επικοινωνίας για το σύνηθες ch_p4 κανάλι επικοινωνίας των TCP/IP δικτύων, ήτοι τα short, eager και rendezvous. Κάθε ένα από τα παραπάνω υιοθετεί διαφορετική προσέγγιση όσον αφορά στην ανταλλαγή μηνυμάτων, διαφοροποιώντας έτσι τις αντίστοιχες απαιτήσεις αναφορικά με το πλήθος των μεταδιδόμενων πακέτων και το πλήθος των αντιγραφών δεδομένων. Σύμφωνα με το πρωτόκολλο *eager*, η διεργασία αποστολέας στέλνει άμεσα το μήνυμα, χωρίς να προηγηθεί σχετική ειδοποίηση της διεργασίας παραλήπτη. Αντίθετα, κατά το *rendezvous* πρωτόκολλο, πριν την ανταλλαγή μηνυμάτων μεταξύ δύο διεργασιών προηγείται μια φάση *χειραψίας* (*handshake*) των διεργασιών αυτών, που διασφαλίζει την αποφυγή ενδεχόμενης ανάγκης για ενδιάμεση αποθήκευση των δεδομένων στη διεργασία παραλήπτη. Η εξ ορισμού τιμή κατωφλίου στην οποία πραγματοποιείται η μετάβαση από το eager στο rendezvous πρωτόκολλο ισούται με 128000 bytes.

Με βάση τα παραπάνω, υποθέτοντας τοπολογία διεργασιών 2×4 και χώρο επαναλήψεων $256 \times 256 \times 16K$, κάθε διεργασία αναλαμβάνει υπερκόμβους μεγέθους $128 \times 64 \times z$ πραγματικών στοιχείων διπλής ακρίβειας (=8 bytes). Για ύψος υπερκόμβου $z = 125$, η εν λόγω απεικόνιση οδηγεί σε αναγκαιότητα αποστολής μηνυμάτων μεγέθους $128 \times 1 \times 125 \times 8 = 128000$ bytes κατά μήκος της δεύτερης διάστασης του χώρου επαναλήψεων, τιμή η οποία συμπίπτει με το προαναφερθέν κατώφλι μετάβασης από το eager στο rendezvous πρωτόκολλο. Το ίδιο ισχύει και για την τιμή $z = 250$, που αντιστοιχεί στην αποστολή μηνύματος 128000 bytes κατά μήκος της πρώτης διάστασης, όπου επίσης παρατηρήθηκε σημαντική μείωση της απόδοσης του προγράμματος. Γενικά, υπάρχει ένας συμβιβασμός μεταξύ των δύο πρωτοκόλλων σε ό,τι αφορά το συγχρονισμό των διεργασιών και την ενδιάμεση αποθήκευση δεδομένων, συνεπώς κατά τη μετάβαση από το ένα πρωτόκολλο στο άλλο δεν θα πρέπει να μας εκπλήσσει μια μεταβολή στην απόδοση. Ποιοτικά, για σχετικά υψηλό αριθμό μη αναμενόμενων μηνυμάτων, το κόστος της αντιγραφής δεδομένων κατά την ενδιάμεση αποθήκευση με το eager πρωτόκολλο είναι σημαντικό, καθιστώντας το rendezvous πρωτόκολλο ως την πιο ενδεδειγμένη προσέγγιση. Αντίθετα, για σχετικά χαμηλό πλήθος μη αναμενόμενων μηνυμάτων, το κόστος του συγχρονισμού που επιβάλλει

το rendezvous πρωτόκολλο είναι απαγορευτικό, συνεπώς το eager πρωτόκολλο είναι στην περίπτωση αυτή η πιο αποδοτική λύση.



Σχήμα 6.16: *Σύγκριση υβριδικών μοντέλων (ADI, διάφοροι χώροι επαναλήψεων, συστοιχία xenons, 8 διεργασίες, 2 νήματα ανά διεργασία)*

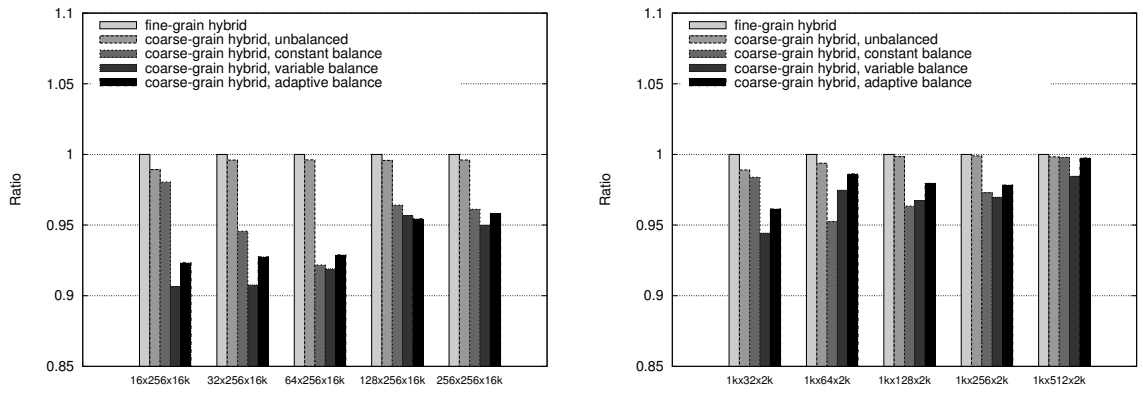
Στο σχήμα 6.16 αναπαρίστανται οι συνολικοί χρόνοι εκτέλεσης για τα διάφορα υβριδικά μοντέλα με ή χωρίς εξισορρόπηση φορτίου κατά την εκτέλεση του μετροπρογράμματος ADI στη συστοιχία xenons. Η βελτίωση στους χρόνους εκτέλεσης που επιτυγχάνεται με χρήση σταθερής εξισορρόπησης φορτίου κυμαίνεται μεταξύ 3-10%. Με χρήση μεταβλητής εξισορρόπησης φορτίου επιτυγχάνεται μείωση του χρόνου εκτέλεσης κατά 3-9%, ενώ τέλος με χρήση δυναμικής εξισορρόπησης φορτίου παρατηρείται βελτίωση της επίδοσης κατά 2-7%. Σε όλες τις περιπτώσεις, τα σχήματα εξισορρόπησης φορτίου βελτιώνουν την επίδοση του υβριδικού μοντέλου σε σχέση με τις απλές υλοποιήσεις τόσο λεπτού όσο και χονδρού κόκκου. Θα πρέπει να τονιστεί ότι το αποτέλεσμα αυτό είναι ιδιαίτερα σημαντικό, καθώς πρόκειται για ένα σαφώς ταχύτερο δίκτυο διασύνδεσης από εκείνο των twins και όλοι οι υπολογιζόμενοι συντελεστές για την εξισορρόπηση του φορτίου των νημάτων βρίσκονται κοντά στην τιμή 100%, που υποδηλώνει ισοκατανομή του υπολογιστικού φορτίου μεταξύ των νημάτων.

6.4.2 DE-XYT

Ανάλογη διαδικασία ακολουθήσαμε και για το μετροπρόγραμμα DE-XYT. Συγκεκριμένα, πραγματοποιήσαμε μετρήσεις του συνολικού χρόνου εκτέλεσης για όλα τα υβριδικά μοντέλα και τους χώρους επαναλήψεων των Συνόλων 1 και 2 (βλ. πίνακα 6.9) τόσο στη συστοιχία των twins όσο και σε εκείνη των xenons. Οι σχετικοί χρόνοι εκτέλεσης, κανονικοποιημένοι ως προς τους χρόνους του υβριδικού μοντέλου λεπτού κόκκου, απεικονίζονται στα σχήματα 6.17 για τη συστοιχία των twins και 6.18 για τη συστοιχία των xenons.

Χώρος επαναλήψεων		***Τοπολογία διεργασιών***
Σύνολο 1	$16 \times 256 \times 16K$	1×8
	$32 \times 256 \times 16K$	1×8
	$64 \times 256 \times 16K$	1×8
	$128 \times 256 \times 16K$	2×4
	$256 \times 256 \times 16K$	2×4
Σύνολο 2	$1K \times 32 \times 2K$	8×1
	$1K \times 64 \times 2K$	8×1
	$1K \times 128 \times 2K$	8×1
	$1K \times 256 \times 2K$	8×1
	$1K \times 512 \times 2K$	4×2

Πίνακας 6.9: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-XYT και συνολικό πλήθος διεργασιών 8*



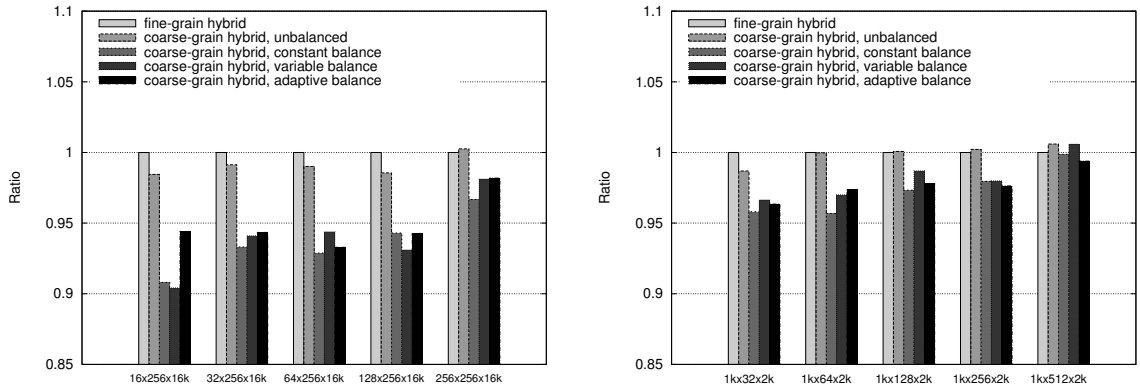
(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.17: *Σύγκριση υβριδικών μοντέλων (DE-XYT, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins)*

Όσον αφορά στη συστοιχία twins, τα πιο αποδοτικά σχήματα εξισορρόπησης φορτίου ήταν το με-

ταβλητό και το δυναμικό. Έτσι, στην περίπτωση του Συνόλου 1 το σχήμα μεταβλητής εξισορρόπησης φορτίου βελτιώνει το συνολικό χρόνο εκτέλεσης σε σχέση με τα υβριδικά μοντέλα λεπτού και χονδρού κόκκου κατά 5-10%, ενώ εκείνο της δυναμικής εξισορρόπησης φορτίου παρέχει ποσοστιαία βελτίωση της τάξης του 4-8%. Μάλιστα, τα δύο αυτά σχήματα εξισορρόπησης φορτίου υπερτερούν σε όλους τους χώρους επαναλήψεων έναντι των υπολοίπων υβριδικών υλοποιήσεων, ενώ το μοντέλο χονδρού κόκκου καταγράφει και πάλι σχεδόν ταυτόσημους χρόνους με το αντίστοιχο μοντέλο λεπτού κόκκου. Μικρότερα είναι τα οφέλη που αποκομίζουμε για την περίπτωση του Συνόλου 2, όπου το σχήμα μεταβλητής εξισορρόπησης φορτίου επιτυγχάνει βελτίωση του χρόνου εκτέλεσης κατά 2-6%, ενώ εκείνο της δυναμικής εξισορρόπησης φορτίου επιταχύνει τους χρόνους εκτέλεσης κατά 1-4%. Πάντως και εδώ σε όλες τις περιπτώσεις παρατηρούμε βελτίωση σε σχέση με τα απλά μοντέλα λεπτού και χονδρού κόκκου, ενώ αρκετά καλή είναι και η επίδοση του σχήματος σταθερής εξισορρόπησης φορτίου, που αποτιμάται σε βελτίωση 1-5%.



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1 (β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.18: *Σύγκριση υβριδικών μοντέλων (DE-XYT, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία xenons)*

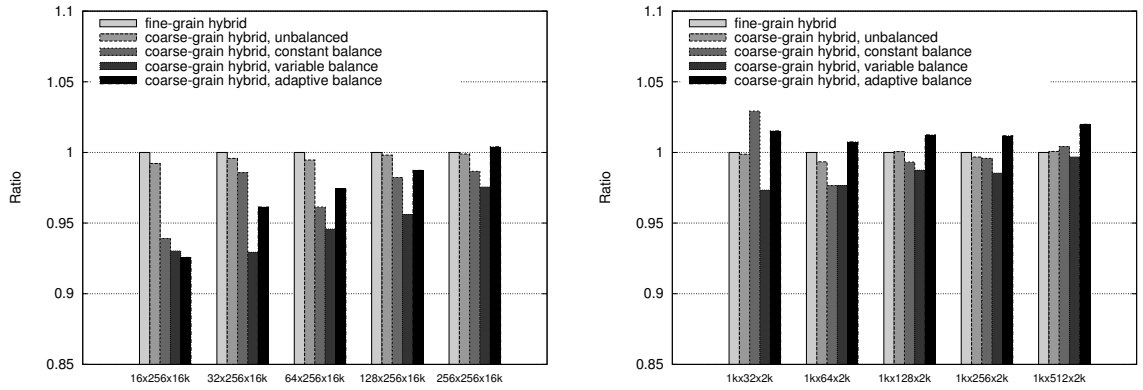
Στη συστοιχία των xenons, τα σχετικά οφέλη που αποκομίζουμε για το Σύνολο 1 κυμαίνονται μεταξύ 2-10% για την περίπτωση του σχήματος μεταβλητής εξισορρόπησης φορτίου, 4-10% για την περίπτωση της σταθερής εξισορρόπησης φορτίου και 2-7% για το σχήμα της δυναμικής εξισορρόπησης. Για την περίπτωση του Συνόλου 2 οι αντίστοιχες ποσοστιαίες βελτιώσεις είναι αισθητά μικρότερες, μέχρι 5%, καθώς γενικά τόσο το Σύνολο 2 όσο και η συστοιχία των xenons με το γρήγορο δίκτυο διασύνδεσης δεν ενδείκνυνται για την ανάδειξη των όποιων βελτιστοποιήσεων στο υβριδικό μοντέλο.

6.4.3 DE-TXY

Στον πίνακα 6.10 αναγράφονται οι προτεινόμενες τοπολογίες για την εκτέλεση του μετροπρογράμματος DE-TXY με χρήση 8 διεργασιών. Οι χρόνοι εκτέλεσης όλων των υβριδικών υλοποιήσεων του DE-TXY στη συστοιχία twins παρέχονται στο σχήμα 6.19, κανονικοποιημένοι ως προς τους χρόνους εκτέλεσης του υβριδικού μοντέλου λεπτού κόκκου. Στο σχήμα 6.20 αναπαρίστανται οι αντίστοιχοι χρόνοι εκτέλεσης για τη συστοιχία των xenons.

Χώρος επαναλήψεων		***Τοπολογία διεργασιών***
Σύνολο 1	$16 \times 256 \times 16K$	1×8
	$32 \times 256 \times 16K$	2×4
	$64 \times 256 \times 16K$	2×4
	$128 \times 256 \times 16K$	4×2
	$256 \times 256 \times 16K$	4×2
Σύνολο 2	$1K \times 32 \times 2K$	8×1
	$1K \times 64 \times 2K$	8×1
	$1K \times 128 \times 2K$	8×1
	$1K \times 256 \times 2K$	8×1
	$1K \times 512 \times 2K$	8×1

Πίνακας 6.10: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-TXY και συνολικό πλήθος διεργασιών 8*



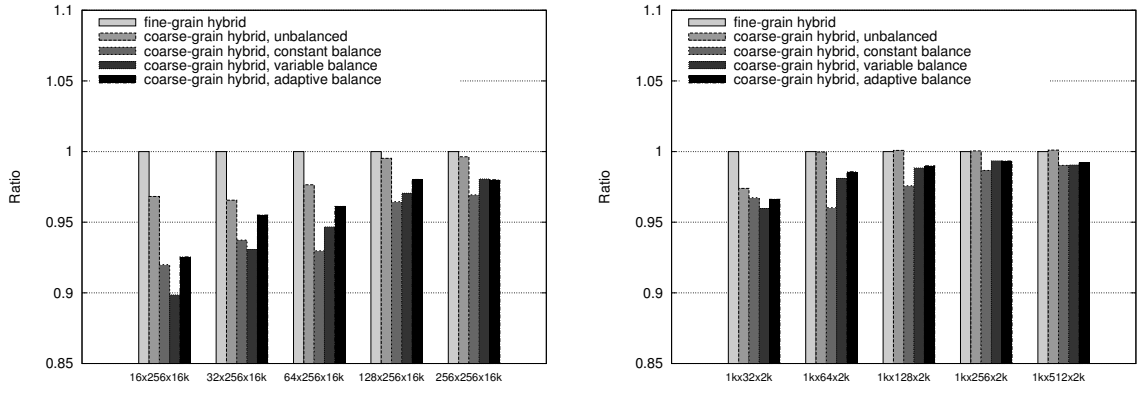
(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.19: *Σύγκριση υβριδικών μοντέλων (DE-TXY, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins)*

Παρατηρούμε ότι σε όλες τις περιπτώσεις παράλληλης εκτέλεσης στη συστοιχία των twins, τόσο για το Σύνολο 1 όσο και για το Σύνολο 2, τους μικρότερους χρόνους εκτέλεσης επιτυγχάνει το σχή-

μα μεταβλητής εξισορρόπησης φορτίου. Έτσι, στην περίπτωση του Συνόλου 1 η εφαρμογή μεταβλητής εξισορρόπησης φορτίου βελτιώνει τους χρόνους εκτέλεσης κατά 3-8%, ενώ για το Σύνολο 2 το αντίστοιχο εύρος κυμαίνεται σε μόλις 1-3%, χωρίς πάντως να καταγράφεται σε καμία περίπτωση μείωση της συνολικής επίδοσης. Εξίσου καλή είναι η επίδοση του μεταβλητού σχήματος και για τη συστοιχία των xenons, καθώς παρατηρούμε βελτίωση της επίδοσης στο εύρος 2-11% για το Σύνολο 1 και 1-4% για το Σύνολο 2.



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.20: *Σύγκριση υβριδικών μοντέλων (DE-TXY, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία xenons)*

Αντίθετα, το σχήμα δυναμικής εξισορρόπησης φορτίου δεν αποδίδει ιδιαίτερα οφέλη στην επίδοση του υβριδικού προγράμματος χονδρού κόκκου, καθώς στη συστοιχία twins για το Σύνολο 1 οι σχετικές ποσοστιαίες βελτιώσεις ανέρχονται σε 8% για το χώρο επαναλήψεων $16 \times 256 \times 16K$ και λιγότερο από 4% για όλους τους υπόλοιπους χώρους επαναλήψεων, ενώ για το Σύνολο 2 παρατηρούμε συνήθως οριακή μείωση της επίδοσης. Παρόμοια επίδοση παρατηρείται και στη συστοιχία των xenons, καθώς στην περίπτωση του Συνόλου 1 παρατηρούμε μείωση των συνολικών χρόνων εκτέλεσης κατά 2-8%, ενώ στην περίπτωση του Συνόλου 2 προκύπτει οριακή βελτίωση κατά λιγότερο από 3% σε όλες τις περιπτώσεις. Το γεγονός αυτό οφείλεται στο ότι η πρόσθετη επιβάρυνση του συγκεκριμένου σχήματος στην προκειμένη περίπτωση αναιρεί τα οφέλη που σχετίζονται με τον υπολογισμό καταλληλότερων συντελεστών για την εξισορρόπηση του φορτίου μεταξύ των νημάτων. Ιδιαίτερα στο Σύνολο 2, λόγω της ιδιομορφίας των εξεταζόμενων χώρων επαναλήψεων, κάθε νήμα αναλαμβάνει την εκτέλεση μιας σχετικά μικρής ακολουθίας υπερκόμβων, στην οποία σημαντικό μέρος καταλαμβάνει η περίοδος δειγματοληψίας των χρόνων υπολογισμού και επικοινωνίας που απαιτούνται για αναπροσαρμογή των συντελεστών.

Παρατηρούμε τέλος ότι σχετικά καλή είναι η επίδοση του σχήματος σταθερής εξισορρόπησης φορτίου, το οποίο στη συστοιχία twins αποδίδει βελτιώσεις της τάξης 2-7% για το Σύνολο 1 και οριακές διαφορές για την περίπτωση του Συνόλου 2, ενώ για τα xenons τα αντίστοιχα ποσοστά ανέρχονται σε 3-8% για το Σύνολο 1 και 1-4% για το Σύνολο 2. Επίσης, όπως και σε όλες τις προηγούμενες περιπτώσεις, τα υβριδικά μοντέλα λεπτού και χονδρού κόκκου παρέχουν σχεδόν απόλυτα ισοδύναμη επίδοση.

6.4.4 Adv2D

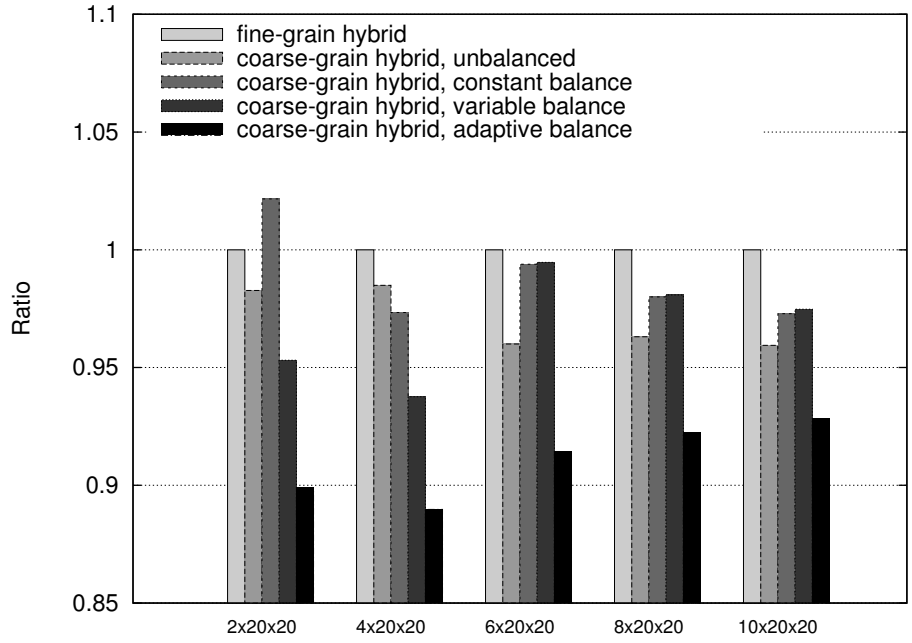
Συγκριτικές μετρήσεις επίδοσης των υβριδικών μοντέλων πραγματοποιήσαμε τέλος και για το μετροπρόγραμμα Adv2D. Έτσι, καταγράψαμε τους συνολικούς χρόνους εκτέλεσης των παράλληλων προγραμμάτων στη συστοιχία των twins για 8 διεργασίες και 2 νήματα ανά διεργασία. Η τοπολογία διεργασιών που υιοθετείται σε κάθε περίπτωση αναγράφεται στον πίνακα 6.11 και επιλέγεται βάσει του κριτηρίου ελαχιστοποίησης της επικοινωνίας. Στο σχήμα 6.21 αναπαρίστανται οι χρόνοι εκτέλεσης όλων των υβριδικών προγραμμάτων για διάφορους χώρους επαναλήψεων, κανονικοποιημένοι ως προς τους χρόνους εκτέλεσης του υβριδικού μοντέλου λεπτού κόκκου. Για τη διακριτοποίηση κάθε πεδίου εφαρμογής θεωρήσαμε και πάλι $\Delta x = \Delta y = 0.1$, οπότε βάσει της συνθήκης Courant επιλέγουμε $\Delta t = 0.001$.

Πεδίο εφαρμογής	***Τοπολογία διεργασιών***
$2 \times 20 \times 20$	1×8
$4 \times 20 \times 20$	1×8
$6 \times 20 \times 20$	2×4
$8 \times 20 \times 20$	2×4
$10 \times 20 \times 20$	2×4

Πίνακας 6.11: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα Adv2D και συνολικό πλήθος διεργασιών 8*

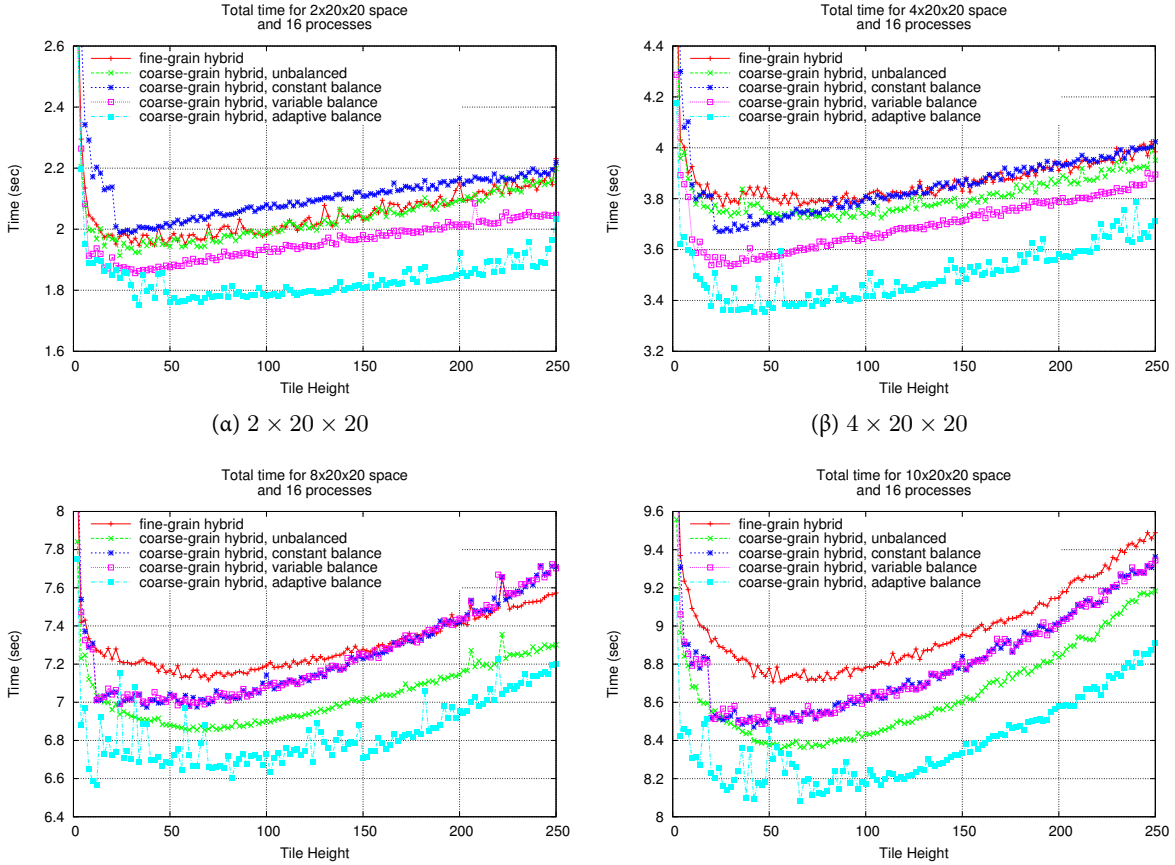
Σε όλους τους χώρους επαναλήψεων παρατηρούμε μια σημαντική βελτίωση της επίδοσης κατά την εφαρμογή σχήματος δυναμικής εξισορρόπησης φορτίου. Έτσι, σε σχέση με το υβριδικό μοντέλο λεπτού κόκκου παρατηρούμε μια μείωση των συνολικών χρόνων εκτέλεσης κατά 8-12% με χρήση δυναμικής εξισορρόπησης φορτίου και κατά 1-7% με χρήση μεταβλητής εξισορρόπησης φορτίου. Αντίθετα, τόσο η χρήση σταθερής εξισορρόπησης φορτίου όσο και το απλό μοντέλο χονδρού κόκκου αποδίδουν μέτρια αποτελέσματα.

Στο σχήμα 6.22 απεικονίζεται η διακύμανση των συνολικών χρόνων εκτέλεσης σε συνάρτηση με το ύψος του υπερκόμβου. Παρατηρούμε ότι σε όλες τις περιπτώσεις υπερισχύει αισθητά το σχήμα δυναμικής εξισορρόπησης φορτίου, με αρκετά καλές επιδόσεις του σχήματος μεταβλητής εξισορρόπησης φορτίου για τα πεδία $2 \times 20 \times 20$ και $4 \times 20 \times 20$. Τα παραπάνω εξηγούνται ως εξής: στη συγκεκριμένη



Σχήμα 6.21: *Σύγκριση υβριδικών μοντέλων (Adv2D, διάφοροι χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins)*

σειρά μετρήσεων θεωρήσαμε χώρους επαναλήψεων με αρκετά μεγάλη εσωτερική σειριακή διάσταση, εάν αναλογιστεί κανείς πως το χρονικό παράθυρο $T = 20$ σε συνδυασμό με το κβάντο χρονικής διακριτοποίησης $\Delta t = 0.001$ αντιστοιχεί σε περίπου 20000 χρονικές επαναλήψεις, που για ύψη υπερκόμβου στην περιοχή $2 \ldots 250$ ισοδυναμούν με την ανάθεση ακολουθίας $80 \ldots 10000$ υπερκόμβων σε κάθε διεργασία. Δεδομένου ότι η περίοδος δειγματοληψίας του σχήματος δυναμικής εξισορρόπησης φορτίου δεν υπερβαίνει τους πρώτους 20 υπερκόμβους, η αντίστοιχη επιβάρυνση είναι κατά κανόνα μικρή σε σχέση με τα οφέλη που αποκομίζουμε από την εξισορρόπηση του φορτίου σε χρόνο εκτέλεσης. Επίσης, πρέπει να ληφθεί υπόψη ότι οι αρχικοί συντελεστές που εφαρμόζονται στο σχήμα δυναμικής εξισορρόπησης φορτίου φαίνεται να είναι κατάλληλοι μόνο στους δύο πρώτους χώρους επαναλήψεων, καθώς αυτοί αναλογούν στους δοκιμαστικούς χώρους που χρησιμοποιήθηκαν για τη δειγματοληψία του μέσου χρόνου εκτέλεσης ανά επανάληψη. Για τους υπόλοιπους χώρους υπάρχει σαφής διαφοροποίηση του μέσου χρόνου εκτέλεσης λόγω φαινομένων ιεραρχίας μνήμης, καθιστώντας τον επαναπροσδιορισμό των συντελεστών εξισορρόπησης φορτίου κατά το χρόνο εκτέλεσης ιδιαίτερα αποδοτικό.



(α) $2 \times 20 \times 20$ (β) $4 \times 20 \times 20$

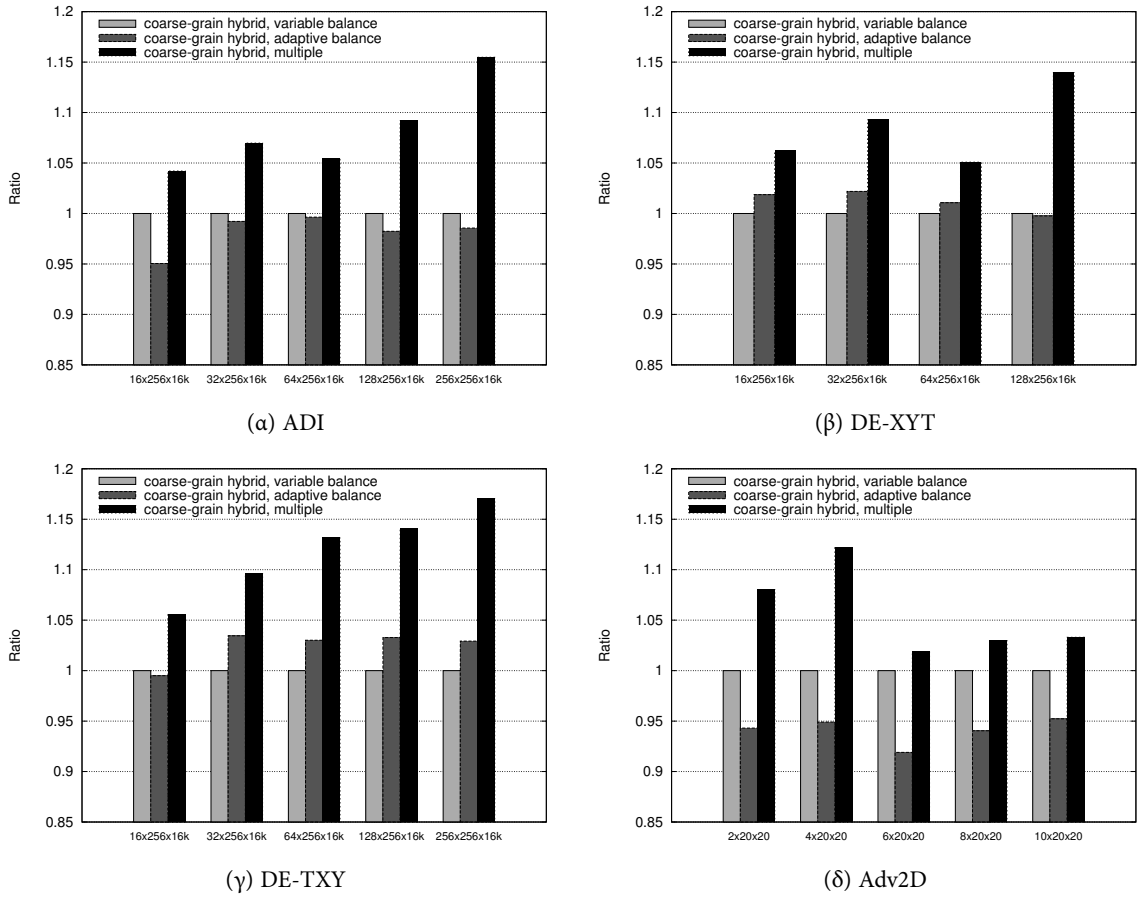
(γ) $8 \times 20 \times 20$ (δ) $10 \times 20 \times 20$

Σχήμα 6.22: *Σύγκριση υβριδικών μοντέλων για μεταβλητό κόκκο παραλληλισμού (Adv2D, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins). Σε όλους τους χώρους επαναλήψεων και για όλα τα ύψη υπερκόμβου υπερέχει το σχήμα δυναμικής εξισορρόπησης φορτίου.*

6.4.5 Αξιολόγηση Multiple Υβριδικού Μοντέλου Χονδρού Κόκκου

Για την αξιολόγηση του multiple υβριδικού μοντέλου χονδρού κόκκου χρησιμοποιήθηκε η εμπορική υλοποίηση MPI/Pro, που διασφαλίζει την απαιτούμενη πλήρη πολυνηματική υποστήριξη. Πράγματι, η υλοποίηση MPI/Pro παρέχει επίπεδο `MPI_THREAD_MULTIPLE` πολυνηματικής υποστήριξης, επιτρέποντας σε όλα τα νήματα μιας υβριδικής εφαρμογής να καλούν ρουτίνες MPI χωρίς περιορισμούς. Πραγματοποιήσαμε συγκριτικές μετρήσεις χρόνου για τα μετροπρογράμματα ADI, DE-XYT, DE-TXY και Adv2D στη συστοιχία των twins με χρήση των δύο υλοποιήσεων του υβριδικού μοντέλου χονδρού κόκκου (funneled και multiple). Για το funneled μοντέλο θεωρήσαμε τόσο μεταβλητή όσο και δυναμι-

κή εξισορρόπηση φορτίου, καθώς η αξιολόγηση των υβριδικών μοντέλων που προηγήθηκε κατέδειξε τη συγκριτική υπεροχή των δύο αυτών σχημάτων εξισορρόπησης. Τα αποτελέσματα για όλα τα μετροπρογράμματα, διάφορους χώρους επαναλήψεων και 8 διεργασίες που εκκινούν 2 νήματα ανά διεργασία συνοψίζονται στο σχήμα 6.23. Όλα τα αποτελέσματα είναι κανονικοποιημένα ως προς τους χρόνους εκτέλεσης του funneled μοντέλου υπό το σχήμα μεταβλητής εξισορρόπησης φορτίου.



(α) ADI

(β) DE-XYT

(γ) DE-TXY

(δ) Adv2D

Σχήμα 6.23: *Σύγκριση multiple υβριδικού μοντέλου χονδρού κόκκου με βελτιστοποιημένο funneled (διάφορα μετροπρογράμματα και χώροι επαναλήψεων, 8 διεργασίες, 2 νήματα ανά διεργασία, συστοιχία twins)*

Είναι φανερό ότι στο σύνολο των περιπτώσεων το multiple υβριδικό μοντέλο χονδρού κόκκου υστερεί της αντίστοιχης funneled υλοποίησης, υπό την προϋπόθεση ότι υιοθετείται κάποιο αποδοτικό σχήμα εξισορρόπησης φορτίου, όπως το μεταβλητό ή το δυναμικό. Ανάλογα με το μετροπρόγραμμα και το συγκεκριμένο χώρο επαναλήψεων, το multiple υβριδικό μοντέλο υστερεί του εξισορροπημένου funneled

κατά 2-17% για την περίπτωση μεταβλητής εξισορρόπησης φορτίου και κατά 4-18% για την περίπτωση δυναμικής εξισορρόπησης φορτίου. Το αποτέλεσμα αυτό φανερώνει ότι η διασφάλιση της πλήρους πολυνηματικής υποστήριξης συνεπάγεται και το ανάλογο κόστος, που αποτυπώνεται τελικά στη συνολική επίδοση του προγράμματος. Έτσι, ο συνδυασμός μιας αποδοτικής υλοποίησης σαν το MPICH με κατάλληλη τεχνική εξισορρόπησης φορτίου μπορεί να οδηγήσει σε αποδοτικότερη παραλληλοποίηση από ό,τι η υλοποίηση MPI/Pro με την πλήρη πολυνηματική υποστήριξη.

Το multiple υβριδικό μοντέλο χονδρού κόκκου δεν μας απασχόλησε περισσότερο στο πλαίσιο της πειραματικής αξιολόγησης, καθώς μεταξύ άλλων ο συνολικός αριθμός αδειών χρήσης του λογισμικού MPI/Pro (8) δεν μας επέτρεπε την πλήρη αξιοποίηση της αρχιτεκτονικής υποδομής των συστοιχιών 16 επεξεργαστών με ισάριθμες διεργασίες στο μοντέλο ανταλλαγής μηνυμάτων. Άλλωστε, η ίδια η φιλοσοφία ενός προτύπου που απευθύνεται πρωτίστως στην ερευνητική και ακαδημαϊκή κοινότητα, όπως το MPI, συνάδει περισσότερο με τη χρήση ανοιχτών ελεύθερα διαθέσιμων υλοποιήσεων, που επιτρέπουν την εις βάθους μελέτη της συμπεριφοράς του λογισμικού.

6.4.6 Συμπεράσματα

Συνοψίζοντας τα πειραματικά αποτελέσματα της αξιολόγησης των προτεινόμενων υβριδικών μοντέλων με ή χωρίς εξισορρόπηση φορτίου, μπορούμε να εξαγάγουμε τα ακόλουθα συμπεράσματα:

- Το funneled υβριδικό μοντέλο χονδρού κόκκου δεν είναι πάντοτε πιο αποδοτικό σε σχέση με το εναλλακτικό ισοδύναμο λεπτού κόκκου. Συχνά, η εγγενής μη αποδοτική κατανομή φορτίου του απλού υβριδικού μοντέλου χονδρού κόκκου αναιρεί τα συγκριτικά πλεονεκτήματα αυτού ως προς το αντίστοιχο μοντέλο λεπτού κόκκου (π.χ. δυνατότητα επικάλυψης υπολογισμού με επικοινωνία, αποφυγή κόστους επαναρχικοποίησης πολυνηματικών δομών κλπ.).

- Κατά την εφαρμογή εξισορρόπησης φορτίου με σταθερό συντελεστή, σε κάποιες περιπτώσεις η εξισορροπημένη υβριδική υλοποίηση χονδρού κόκκου είναι λιγότερο αποδοτική συγκριτικά με τις υπόλοιπες μη εξισορροπημένες υβριδικές υλοποιήσεις (λεπτού ή χονδρού κόκκου). Η αστοχία της συγκεκριμένης μεθόδου εξισορρόπησης φορτίου στις περιπτώσεις αυτές έγκειται αφενός στην ανακριβή θεωρητική μοντελοποίηση του υφιστάμενου συστήματος και αφετέρου στην ακαταλληλότητα της μεθόδου για τις συνοριακές διεργασίες της εικονικής τοπολογίας.

- Κατά την εφαρμογή εξισορρόπησης φορτίου με μεταβλητό συντελεστή, το υβριδικό μοντέλο χονδρού κόκκου επιτυγχάνει σε όλες τις περιπτώσεις καλύτερη απόδοση σε σχέση με το μοντέλο λεπτού κόκκου. Ανάλογα με το μετροπρόγραμμα και το χώρο επαναλήψεων, η ποσοστιαία βελτίωση κυμαίνεται κατά μέσο όρο μεταξύ 3-8%. Παρότι η μέση ποσοστιαία βελτίωση δεν είναι ιδιαίτερα υψηλή, είναι σημαντικό να επισημανθεί ότι το συγκεκριμένο σχήμα εξισορρόπησης

φορτίου επέτυχε για όλα τα μετροπρογράμματα και όλους τους χώρους επαναλήψεων μείωση του χρόνου εκτέλεσης σε σχέση με τα απλά υβριδικά μοντέλα λεπτού και χονδρού κόκκου. Τα παραπάνω βρίσκονται σε συμφωνία με τα θεωρητικώς αναμενόμενα, ότι δηλαδή υπό κατάλληλο σχήμα εξισορρόπησης φορτίου το υβριδικό σχήμα χονδρού κόκκου μπορεί να είναι πάντοτε πιο αποδοτικό σε σχέση με το αντίστοιχο μοντέλο λεπτού κόκκου. Μάλιστα, είναι σημαντικό ότι οι βελτιώσεις αυτές παρατηρήθηκαν με χρήση ενιαίου μέσου χρόνου υπολογισμού ανά επανάληψη για κάθε αλγόριθμο, ανεξάρτητα από τον εξεταζόμενο χώρο επαναλήψεων.

- Εξίσου αποδοτικό φαίνεται να είναι το σχήμα δυναμικής εξισορρόπησης φορτίου. Ανάλογα με το συγκεκριμένο αλγόριθμο και τον εκάστοτε χώρο επαναλήψεων, για τις εξεταζόμενες περιπτώσεις των μετροπρογραμμάτων πετύχαμε μια μέση βελτίωση 4-10% με χρήση δυναμικής εξισορρόπησης φορτίου. Τα οφέλη της δυναμικής εξισορρόπησης φορτίου είναι ιδιαίτερα σημαντικά σε περίπτωση που βάσει του εξεταζόμενου χώρου επαναλήψεων κάθε διεργασία αναλαμβάνει την εκτέλεση ενός επαρκούς πλήθους υπερκόμβων, σημαντικά περισσότερων από εκείνους που εκτελούνται στην περίοδο δειγματοληψίας. Αντίθετα, στις περιπτώσεις που η περίοδος δειγματοληψίας αποτελεί σημαντικό μέρος της εκτέλεσης, η επιβάρυνση του συγκεκριμένου σχήματος είναι εμφανής και μάλιστα ενδέχεται να οδηγήσει ακόμα και σε μείωση της απόδοσης.

- Η multiple υλοποίηση χονδρού κόκκου επιτρέπει μεν την υλοποίηση σχετικά απλούστερης υβριδικής παραλληλοποίησης, αλλά αφενός προϋποθέτει πλήρη πολυνηματική υποστήριξη από τη βιβλιοθήκη ανταλλαγής μηνυμάτων και αφετέρου επιτυγχάνει χαμηλότερη επίδοση σε σχέση με το εξισορροπημένο funneled μοντέλο.

- Συνολικά, οι μέθοδοι μεταβλητής και δυναμικής εξισορρόπησης φορτίου ενδείκνυνται για τη μείωση του συνολικού χρόνου εκτέλεσης του υβριδικού προγράμματος χονδρού κόκκου, ιδιαίτερα όταν οι παράμετροι μοντελοποίησης του συστήματος έχουν προσδιοριστεί με ακρίβεια για το δεδομένο μετροπρόγραμμα και χώρο επαναλήψεων.

6.5 Γενική Αξιολόγηση

Τέλος, πραγματοποιήσαμε μετρήσεις προς την κατεύθυνση συνολικής συγκριτικής αξιολόγησης τόσο του προγραμματιστικού μοντέλου ανταλλαγής μηνυμάτων όσο και των υβριδικών μοντέλων λεπτού και χονδρού κόκκου. Εφαρμόστηκαν όλες οι βελτιστοποιήσεις των προηγούμενων ενοτήτων, δηλαδή εκείνη της επιλογής τοπολογίας απεικόνισης διεργασιών ελάχιστης επικοινωνίας, καθώς και οι προτεινόμενες τεχνικές εξισορρόπησης φορτίου (σταθερή, δυναμική, μεταβλητή). Στη σύγκριση συμπεριλάβαμε και το υβριδικό μοντέλο λεπτού κόκκου, παρότι στα προηγούμενα αποδείχθηκε ότι παρέχει

πάντοτε χαμηλότερη απόδοση σε σχέση με το αντίστοιχο εξισορροπημένο υβριδικό μοντέλο χονδρού κόκκου. Λόγω όμως της δημοφιλίας του υβριδικού μοντέλου λεπτού κόκκου στη σχετική βιβλιογραφία, για λόγους πληρότητας συμπεριλαμβάνεται και στη γενική αξιολόγηση των παράλληλων προγραμματιστικών μοντέλων.

Παρότι η σύγκριση όλων των μοντέλων παραλληλοποίησης έχει πρακτική χρησιμότητα για την ανάδειξη αποδοτικών τεχνικών παράλληλου προγραμματισμού αρχιτεκτονικών κατανεμημένης μοιραζόμενης μνήμης, η συγκριτική αξιολόγηση αντικατοπτρίζει εν πολλοίς τη σχετική επίδοση των δύο βασικών προγραμματιστικών εργαλείων που χρησιμοποιήθηκαν, δηλαδή αφενός της βιβλιοθήκης MPICH και αφετέρου της υποστήριξης του OpenMP προτύπου στο μεταγλωττιστή icc της Intel. Είναι προφανές ότι τα πειραματικά αποτελέσματα εξαρτώνται άμεσα κι από την αλληλεπίδραση των δύο αυτών εργαλείων λογισμικού, δηλαδή από την αποδοτικότητα της OpenMP πολυνηματικής υποστήριξης που υλοποιεί η βιβλιοθήκη MPICH. Λόγω των παραπάνω, τα συμπεράσματα της ποσοτικής αξιολόγησης της ενότητας αυτής δεν θα πρέπει να γενικευτούν πέρα από το συγκεκριμένο συνδυασμό υλικού και λογισμικού της πειραματικής υποδομής που χρησιμοποιήθηκε. Αντίθετα, η παρούσα ενότητα αποσκοπεί στο να παρέχει μια ποιοτική διαίσθηση της συγκριτικής δυνατότητας των διάφορων παράλληλων προγραμματιστικών μοντέλων, καθώς και της επίδρασης των σχημάτων εξισορρόπησης φορτίου στη βελτίωση της σχετικής επίδοσης των υβριδικών μοντέλων συγκριτικά με το μοντέλο ανταλλαγής μηνυμάτων.

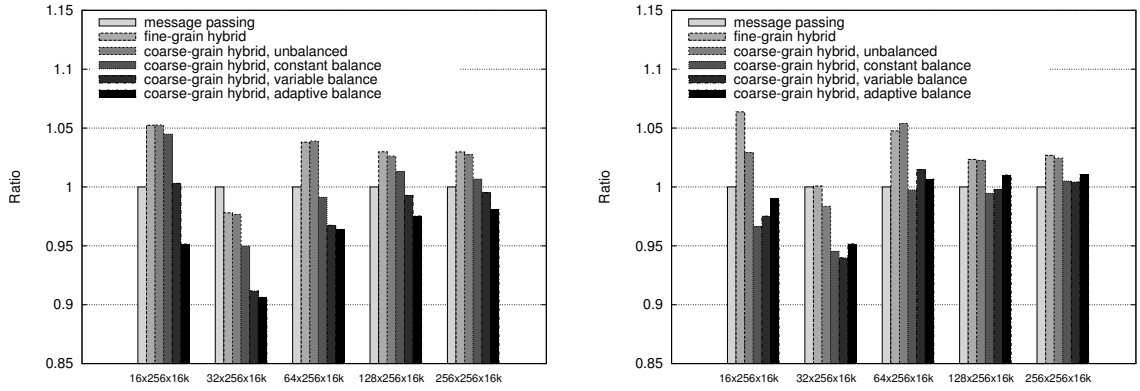
6.5.1 ADI

Οι χρόνοι εκτέλεσης του μετροπρογράμματος ADI απεικονίζονται στο σχήμα 6.24 τόσο για τη συστοιχία twins όσο και για τη συστοιχία xenons. Όλοι οι χρόνοι εκτέλεσης είναι κανονικοποιημένοι ως προς τους αντίστοιχους χρόνους του μοντέλου ανταλλαγής μηνυμάτων, ώστε να διευκολύνουν την ποσοστιαία σύγκριση της επίδοσης. Στην περίπτωση του μοντέλου ανταλλαγής μηνυμάτων χρησιμοποιήσαμε 16 διεργασίες σε καρτεσιανή δισδιάστατη τοπολογία, που επελέγη βάσει του κριτηρίου ελαχιστοποίησης της επικοινωνίας. Στο υβριδικό μοντέλο χρησιμοποιήσαμε 8 διεργασίες και 2 νήματα ανά διεργασία. Οι επιλεγμένες τοπολογίες διεργασιών που υιοθετήθηκαν σε κάθε περίπτωση αναγράφονται στον πίνακα 6.12.

Στην περίπτωση της συστοιχίας των twins παρατηρούμε ότι ενώ τα απλά υβριδικά μοντέλα λεπτού και χονδρού κόκκου υστερούν του μοντέλου ανταλλαγής μηνυμάτων σχεδόν σε όλους τους χώρους επαναλήψεων, η εφαρμογή μεταβλητής ή δυναμικής εξισορρόπησης φορτίου παρέχει σε όλες τις περιπτώσεις έστω και οριακή μείωση του συνολικού χρόνου εκτέλεσης. Έτσι, η εφαρμογή μεταβλητής εξισορρόπησης φορτίου οδηγεί σε βελτίωση του χρόνου εκτέλεσης κατά 1-9%, ενώ το αντίστοιχο σχήμα δυναμικής εξισορρόπησης φορτίου επιτυγχάνει μείωση των χρόνων εκτέλεσης κατά 2-10%. Σχετικά

Χώρος επαναλήψεων	*MPI*	*Υβριδικό*
$16 \times 256 \times 16K$	1×16	1×8
$32 \times 256 \times 16K$	2×8	1×8
$64 \times 256 \times 16K$	2×8	1×8
$128 \times 256 \times 16K$	2×8	2×4
$256 \times 256 \times 16K$	4×4	2×4

Πίνακας 6.12: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα ADI και συνολικό πλήθος διεργασιών 16 (MPI) ή 8 (υβριδικό)*



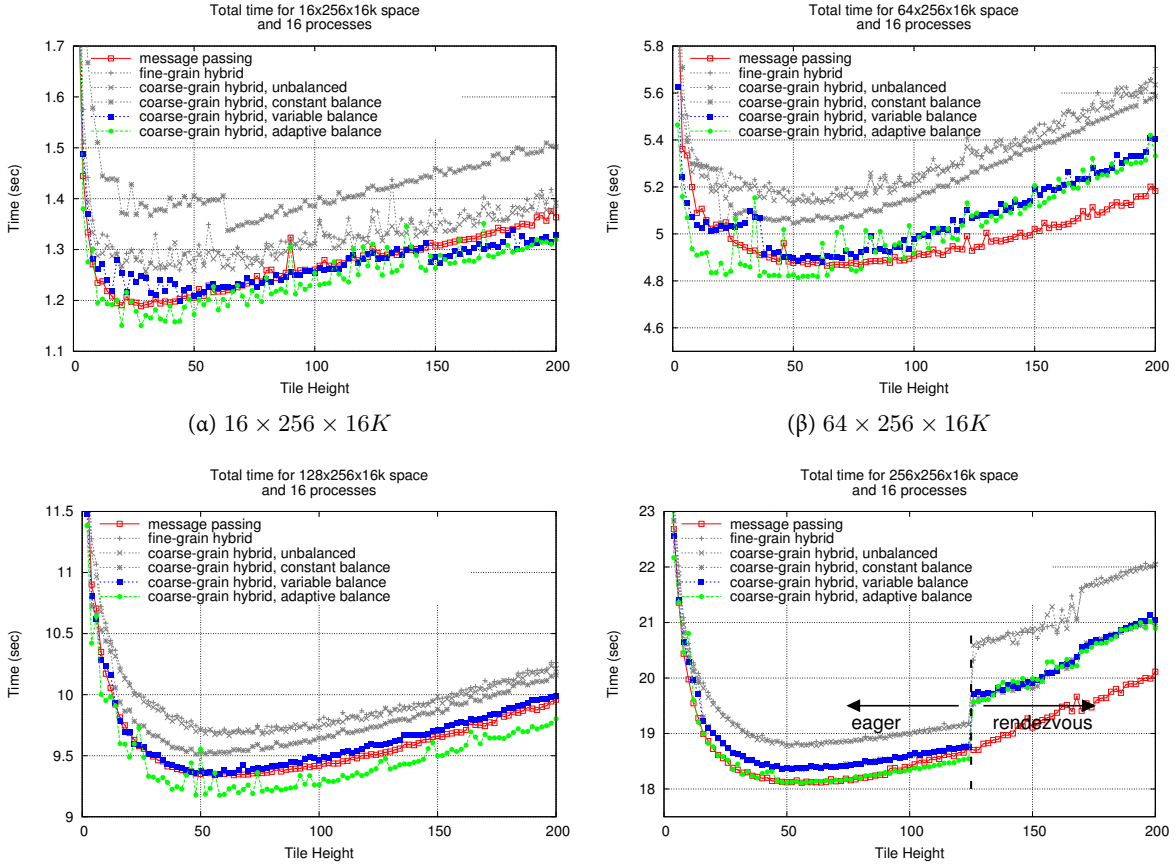
(α) συνολικός χρόνος εκτέλεσης, συστοιχία twins

(β) συνολικός χρόνος εκτέλεσης, συστοιχία xenons

Σχήμα 6.24: *Σύγκριση παράλληλων μοντέλων (ADI, διάφοροι χώροι επαναλήψεων, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα ανά διεργασία, συστοιχίες twins και xenons)*

μικρότερα είναι τα αντίστοιχα οφέλη στη συστοιχία των xenons, όπου τα δύο σχήματα εξισορρόπησης φορτίου επιτυγχάνουν μικρές βελτιώσεις μέχρι 5%, καθώς οι υπολογιζόμενοι συντελεστές εξισορρόπησης είναι κοντά στο 100% και τα σχετικά οφέλη της εξισορρόπησης του φορτίου είναι αρκετά περιορισμένα.

Στο σχήμα 6.25 αναπαρίστανται οι συνολικοί χρόνοι παράλληλης εκτέλεσης για όλα τα προγραμματιστικά μοντέλα σε συνάρτηση με το ύψος υπερκόμβου. Όλες οι παρατηρήσεις που αφορούν τους ελάχιστους χρόνους εκτέλεσης γενικεύονται και για την πλειοψηφία των τιμών του ύψους z του υπερκόμβου, συνεπώς οι τεχνικές εξισορρόπησης φορτίου παραμένουν αποτελεσματικές ακόμα και στην περίπτωση που δεν επελέγη το βέλτιστο ύψος υπερκόμβου. Δεδομένου ότι ο προσδιορισμός του βέλτιστου κόκκου παραλληλισμού είναι ένα ιδιαίτερα σύνθετο θέμα, ανοιχτό ακόμα στη διεθνή βιβλιογραφία, το παραπάνω γεγονός έχει μεγάλη πρακτική αξία και χρησιμότητα. Τα απλά υβριδικά μοντέλα λεπτού και χονδρού κόκκου επιτυγχάνουν αισθητά χειρότερη επίδοση σε σχέση με το μοντέλο ανταλ-



(α) $16 \times 256 \times 16K$ (β) $64 \times 256 \times 16K$

(γ) $128 \times 256 \times 16K$ (δ) $256 \times 256 \times 16K$

Σχήμα 6.25: *Σύγκριση παράλληλων μοντέλων για μεταβλητό κόκκο παραλληλισμού (ADI, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα ανά διεργασία, συστοιχία twins). Παρότι το απλό υβριδικό μοντέλο υστερεί σε σχέση με το παράλληλο πρόγραμμα ανταλλαγής μηνυμάτων για όλα τα ύψη υπερκόμβων, η χρήση μεταβλητής και κυρίως δυναμικής εξισορρόπησης φορτίου καθιστά το υβριδικό μοντέλο εξίσου ανταγωνιστικό και μάλιστα παρέχει τους ελάχιστους χρόνους εκτέλεσης. Στο χώρο επαναλήψεων $256 \times 256 \times 16K$ σημειώνεται το κατώφλι μεταξύ eager και rendezvous πρωτοκόλλου του MPICH για την περίπτωση των υβριδικών προγραμμάτων, στο οποίο παρατηρείται σημαντική μείωση της επίδοσης.*

λαγής μηνυμάτων, ακυρώνοντας έτσι στην πράξη τα όποια θεωρητικά πλεονεκτήματά τους για τη δεδομένη αρχιτεκτονική. Η παρατήρηση αυτή μάλιστα ισχύει για το σύνολο των χώρων επαναλήψεων, των κόκκων παραλληλισμού και των διαφορετικών μετροπρογραμμάτων που εξετάστηκαν. Το μη βελτιστοποιημένο υβριδικό μοντέλο εμφανίζει σημαντικά μειονεκτήματα σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων, αδυνατώντας έτσι να αξιοποιήσει τις δομικές του ομοιότητες με την υφιστάμενη αρχιτεκτονική της SMP συστοιχίας. Τα μειονεκτήματα αυτά εντοπίζονται κυρίως στον περιορισμό του βαθμού

παραλληλίας σύμφωνα με το νόμο του Amdahl, καθώς και στην επιβάρυνση επαναρχικοποίησης των πολυνηματικών δομών κατά την επαναλαμβανόμενη είσοδο/έξοδο προς/από τις παράλληλες περιοχές πολυνηματικής επεξεργασίας. Στον αντίποδα, τα σχήματα μεταβλητής και δυναμικής εξισορρόπησης φορτίου παρέχουν καμπύλες που είτε συμπίπτουν με εκείνη του μοντέλου ανταλλαγής μηνυμάτων είτε επιτυγχάνουν ακόμα χαμηλότερους χρόνους εκτέλεσης.

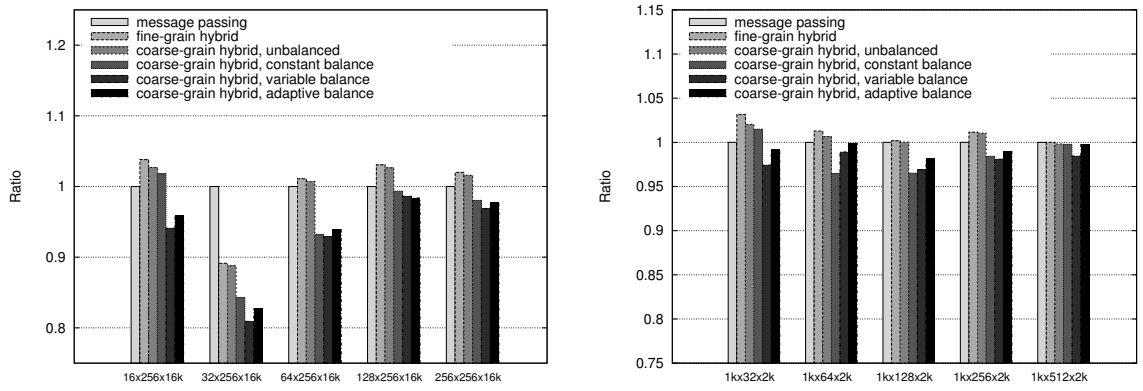
6.5.2 DE-XYT

Στην περίπτωση του μετροπρογράμματος DE-XYT οι μετρήσεις του συνολικού χρόνου εκτέλεσης για το σύνολο των παράλληλων προγραμμάτων παρέχονται στο σχήμα 6.26 για τη συστοιχία twins και στο σχήμα 6.27 για τη συστοιχία xenons. Όλοι οι χρόνοι έχουν κανονικοποιηθεί ως προς εκείνους του μοντέλου ανταλλαγής μηνυμάτων. Ο πίνακας 6.13 συνοψίζει τις τοπολογίες διεργασιών που εφαρμόζονται τόσο για την περίπτωση του MPI προγράμματος (16 διεργασίες) όσο και για τις υβριδικές υλοποιήσεις (8 διεργασίες).

Χώρος επαναλήψεων		***MPI***	***Υβριδικό***
Σύνολο 1	$16 \times 256 \times 16K$	1×16	1×8
	$32 \times 256 \times 16K$	2×8	1×8
	$64 \times 256 \times 16K$	2×8	1×8
	$128 \times 256 \times 16K$	2×8	2×4
	$256 \times 256 \times 16K$	4×4	2×4
Σύνολο 2	$1K \times 32 \times 2K$	16×1	8×1
	$1K \times 64 \times 2K$	16×1	8×1
	$1K \times 128 \times 2K$	8×2	8×1
	$1K \times 256 \times 2K$	8×2	8×1
	$1K \times 512 \times 2K$	4×4	4×2

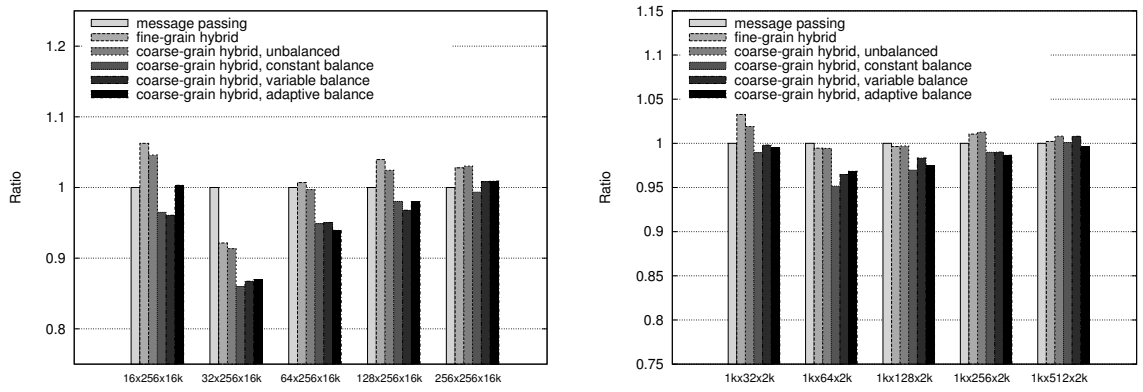
Πίνακας 6.13: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-XYT και συνολικό πλήθος διεργασιών 16 (MPI) ή 8 (υβριδικό)*

Σε σχέση με το μετροπρόγραμμα ADI, στην περίπτωση του DE-XYT παρατηρήθηκαν ακόμα μεγαλύτερα οφέλη κατά το συνδυασμό του υβριδικού μοντέλου χονδρού κόκκου με τα σχήματα μεταβλητής και δυναμικής εξισορρόπησης φορτίου. Έτσι, στην περίπτωση των χώρων επαναλήψεων του Συνόλου 1 παρατηρήθηκε μείωση του χρόνου εκτέλεσης κατά 2-19% με χρήση μεταβλητής εξισορρόπησης φορτίου, ενώ η αντίστοιχη μείωση στην περίπτωση της δυναμικής εξισορρόπησης ήταν 2-18%. Το γεγονός αυτό οφείλεται στις εγγενείς ανάγκες επικοινωνίας του συγκεκριμένου μετροπρογράμματος, καθώς το τελευταίο επιβάλλει τριπλάσιο όγκο δεδομένων επικοινωνίας σε σχέση με το ADI και κατά συνέπεια μπορεί να επωφεληθεί σημαντικά από μια αποδοτική υβριδική παράλληλη υλοποίηση. Ομοίως, στην περίπτωση της συστοιχίας των xenons, παρά το ταχύτερο δίκτυο διασύνδεσης οι αυξημένες ανάγκες



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1 (β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.26: *Σύγκριση παράλληλων μοντέλων (DE-XYT, διάφοροι χώροι επαναλήψεων, 16 διεργασίες ή 8 διεργασίες × 2 νήματα ανά διεργασία, συστοιχία twins)*



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1 (β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

Σχήμα 6.27: *Σύγκριση παράλληλων μοντέλων (DE-XYT, διάφοροι χώροι επαναλήψεων, 16 διεργασίες ή 8 διεργασίες × 2 νήματα ανά διεργασία, συστοιχία xenons)*

επικοινωνίας του μετροπρογράμματος επιτρέπουν τη βελτίωση της επίδοσης μέχρι 14% με χρήση μεταβλητής εξισορρόπησης φορτίου και μέχρι 13% με χρήση δυναμικής εξισορρόπησης.

Αντίθετα, οι βελτιώσεις δεν είναι ιδιαίτερα υψηλές στην περίπτωση του Συνόλου 2, και περιορίζονται μέχρι ενός ποσοστού 5%. Η διαφοροποίηση αυτή που παρατηρείται για τους χώρους του Συνόλου 2 σχετίζεται με την ιδιομορφία των συγκεκριμένων χώρων και ιδιαίτερα με τη σχετικά μικρή εσωτερική διάσταση σειριακής εκτέλεσης, που δεν ευνοεί την ανάδειξη του φαινομένου της σωλήνωσης.

Παρά τις αυξημένες εγγενείς ανάγκες επικοινωνίας του μετροπρογράμματος DE-XYT, παρατηρούμε ότι τα απλά υβριδικά μοντέλα λεπτού και χονδρού κόκκου αδυνατούν να επιτύχουν μικρότερους χρόνους εκτέλεσης σε σχέση με το μονολιθικό μοντέλο ανταλλαγής μηνυμάτων. Στην ουσία, αν δεν εφαρμοστούν τεχνικές εξισορρόπησης φορτίου ώστε να εξομαλυνθεί η υψηλή επιβάρυνση επικοινωνίας του πρωτεύοντος νήματος κάθε διεργασίας, το απλό μοντέλο ανταλλαγής μηνυμάτων παραμένει η πλέον αποδοτική προσέγγιση παράλληλου προγραμματισμού, ακόμα και στην περίπτωση αλγορίθμου με αυξημένες ανάγκες επικοινωνίας.

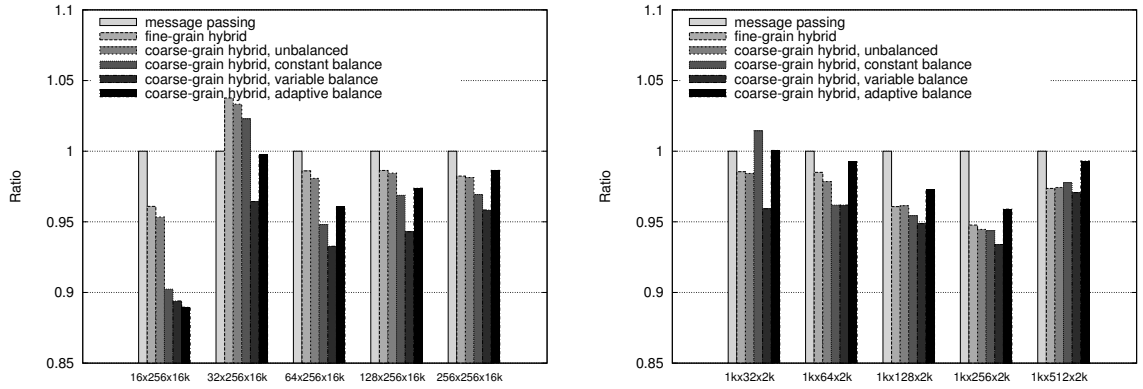
6.5.3 DE-TXY

Στην περίπτωση του μετροπρογράμματος DE-TXY εφαρμόστηκαν οι τοπολογίες του πίνακα 6.14 και τα κανονικοποιημένα αποτελέσματα για τη συστοιχία twins παρέχονται στο σχήμα 6.28. Παρατηρούμε ότι το σχήμα μεταβλητής εξισορρόπησης φορτίου παρέχει βελτιώσεις εύρους 4-11% για το Σύνολο 1 και 3-7% για το Σύνολο 2, ενώ το σχήμα δυναμικής εξισορρόπησης φορτίου είναι ελαφρώς λιγότερο αποτελεσματικό, ιδιαίτερα στο Σύνολο 2.

Χώρος επαναλήψεων		***MPI***	***Υβριδικό***
Σύνολο 1	$16 \times 256 \times 16K$	2×8	1×8
	$32 \times 256 \times 16K$	2×8	2×4
	$64 \times 256 \times 16K$	4×4	2×4
	$128 \times 256 \times 16K$	4×4	4×2
	$256 \times 256 \times 16K$	8×2	4×2
Σύνολο 2	$1K \times 32 \times 2K$	16×1	8×1
	$1K \times 64 \times 2K$	16×1	8×1
	$1K \times 128 \times 2K$	16×1	8×1
	$1K \times 256 \times 2K$	16×1	8×1
	$1K \times 512 \times 2K$	8×2	8×1

Πίνακας 6.14: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα DE-TXY και συνολικό πλήθος διεργασιών 16 (MPI) ή 8 (υβριδικό)*

Η βελτίωση της επίδοσης στο μετροπρόγραμμα DE-TXY είναι σημαντικά μικρότερη σε σχέση με την DE-XYT παραλλαγή, καθώς αφενός μεν στη συγκεκριμένη περίπτωση οι ανάγκες επικοινωνίας είναι μικρότερες, αφετέρου δε οι ανάγκες αυτές είναι ανομοιόμορφα κατανεμημένες στις δύο διευθύνσεις επικοινωνίας, ώστε να καθιστούν περισσότερο επιτακτική την επιλογή κατάλληλης τοπολογίας διεργασιών παρά την αποδοτική υβριδική παραλληλοποίηση.



(α) συνολικός χρόνος εκτέλεσης, Σύνολο 1

(β) συνολικός χρόνος εκτέλεσης, Σύνολο 2

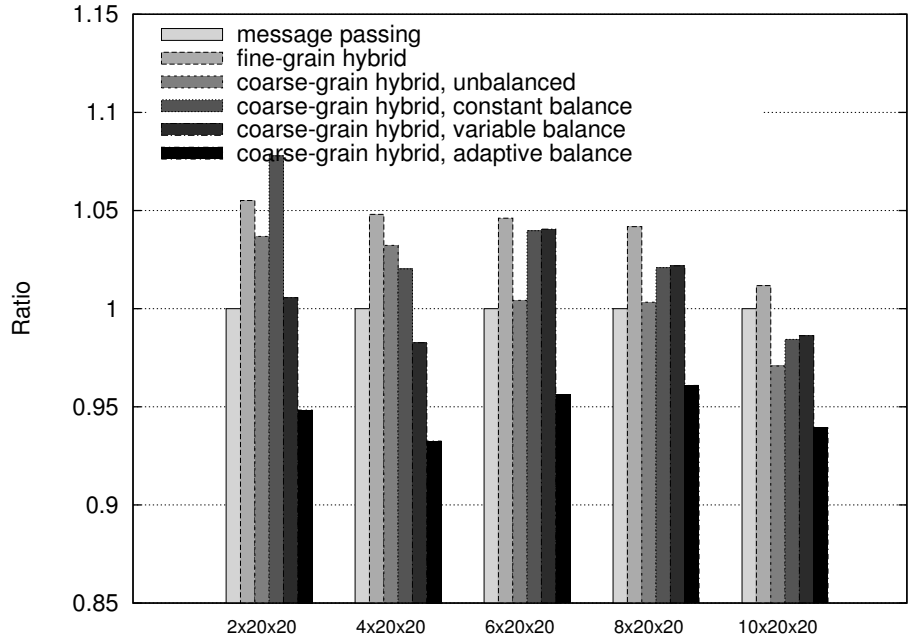
Σχήμα 6.28: *Σύγκριση παράλληλων μοντέλων (DE-TXY, διάφοροι χώροι επαναλήψεων, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα ανά διεργασία, συστοιχία twins)*

6.5.4 Adv2D

Τέλος, το σχήμα 6.29 συνοψίζει τους χρόνους εκτέλεσης όλων των παράλληλων μοντέλων για το μετροπρόγραμμα Adv2D, ενώ οι αντίστοιχες τοπολογίες διεργασιών αναγράφονται στον πίνακα 6.15. Παρατηρούμε ότι οι απλές υβριδικές υλοποιήσεις λεπτού και χονδρού κόκκου υπολείπονται σχεδόν σε όλες τις περιπτώσεις του μοντέλου ανταλλαγής μηνυμάτων. Ωστόσο, η χρήση δυναμικής εξισορρόπησης φορτίου βελτιώνει αισθητά την επίδοση του υβριδικού μοντέλου χονδρού κόκκου σε όλες τις περιπτώσεις, καθώς μάλιστα επιτρέπει τη μείωση του συνολικού χρόνου εκτέλεσης κατά 5-7% σε σχέση με το μοντέλο ανταλλαγής μηνυμάτων. Αντίθετα, το σχήμα μεταβλητής εξισορρόπησης φορτίου δεν αποφέρει ιδιαίτερα οφέλη και καταγράφει με μικρές διαφοροποιήσεις παρόμοια απόδοση με το μοντέλο ανταλλαγής μηνυμάτων.

Χώρος επαναλήψεων	***MPI***	***Υβριδικό***
$2 \times 20 \times 20$	1×16	1×8
$4 \times 20 \times 20$	2×8	1×8
$6 \times 20 \times 20$	2×8	2×4
$8 \times 20 \times 20$	2×8	2×4
$10 \times 20 \times 20$	2×8	2×4

Πίνακας 6.15: *Τοπολογίες απεικόνισης διεργασιών για μετροπρόγραμμα Adv2D και συνολικό πλήθος διεργασιών 16 (MPI) ή 8 (υβριδικό)*



Σχήμα 6.29: *Σύγκριση παράλληλων μοντέλων (Adv2D, διάφοροι χώροι επαναλήψεων, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα ανά διεργασία, συστοιχία twins)*

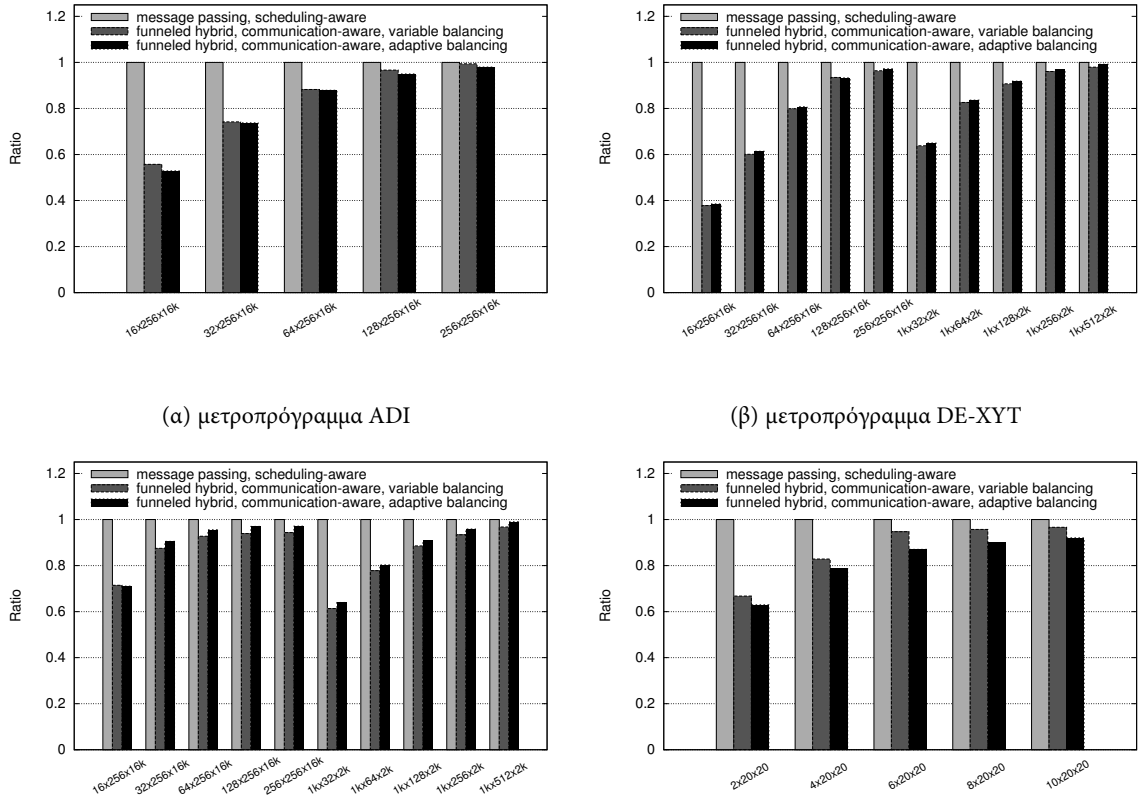
6.6 Συμβολή της Διατριβής: η Πειραματική Εκδοχή

Για λόγους οπτικοποίησης, το σχήμα 6.30 συνοψίζει την επίδραση των προτεινόμενων τεχνικών και βελτιστοποιήσεων στην επίδοση των παράλληλων προγραμματιστικών μοντέλων για βασικά μετροπρογράμματα που εξετάσαμε κατά την πειραματική διαδικασία. Έτσι, παρέχονται οι ελάχιστοι χρόνοι παράλληλης εκτέλεσης των μετροπρογραμμάτων ADI, DE-XYT, DE-TXY και Adv2D με χρήση

- του μοντέλου ανταλλαγής μηνυμάτων με επιλογή της συνήθους 4×4 τοπολογίας διεργασιών ελάχιστης καθυστέρησης διάδοσης

- του υβριδικού μοντέλου χονδρού κόκκου με εφαρμογή μεταβλητού σχήματος εξισορρόπησης του φορτίου των νημάτων και επιλογή της προτεινόμενης τοπολογίας διεργασιών ελάχιστης επικοινωνίας και

- του υβριδικού μοντέλου χονδρού κόκκου με εφαρμογή δυναμικού σχήματος εξισορρόπησης του φορτίου των νημάτων και επιλογή της προτεινόμενης τοπολογίας διεργασιών ελάχιστης επικοινωνίας



(α) μετροπρόγραμμα ADI

(β) μετροπρόγραμμα DE-XYT

(γ) μετροπρόγραμμα DE-TXY

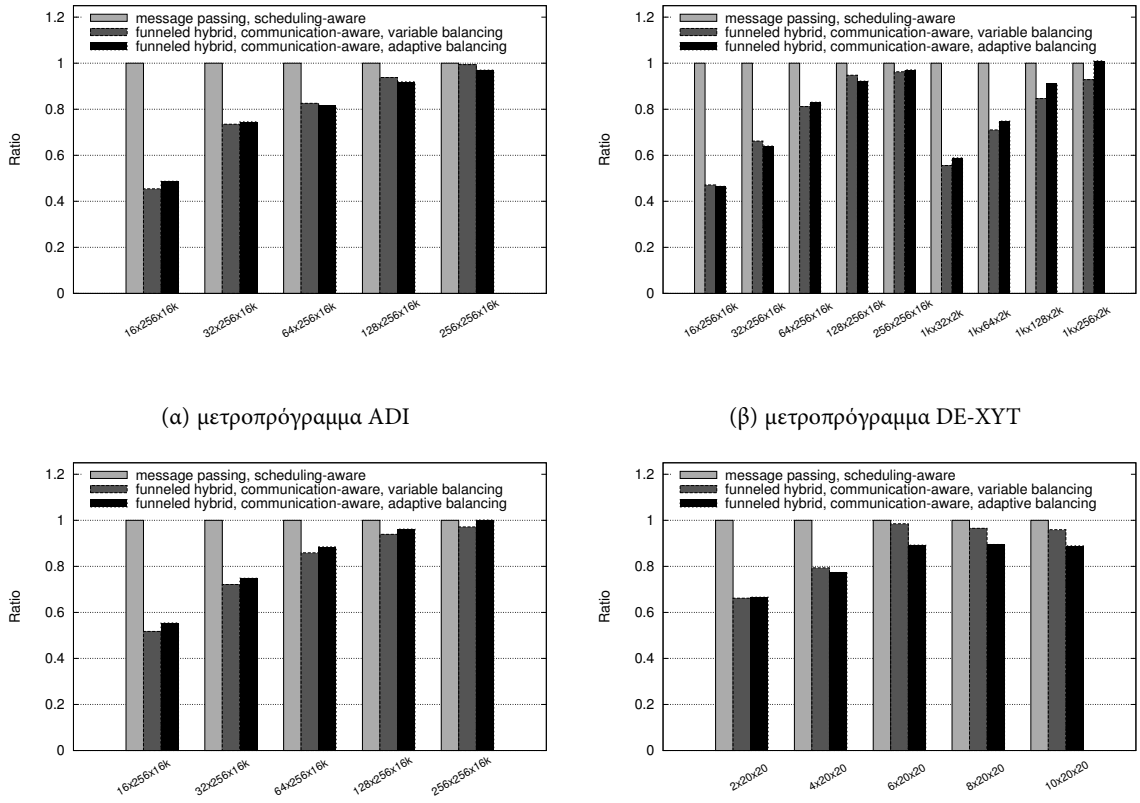
(δ) μετροπρόγραμμα Adv2D

Σχήμα 6.30: *Συμβολή της διατριβής στη βελτίωση της επίδοσης παράλληλων προγραμμάτων με χρήση των προτεινόμενων μοντέλων παραλληλοποίησης και τεχνικών βελτιστοποίησης (διάφορα μετροπρογράμματα, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα, συστοιχία twins)*

Όλοι οι χρόνοι εκτέλεσης είναι κανονικοποιημένοι ως προς τους χρόνους του μοντέλου ανταλλαγής μηνυμάτων, αναφέρονται στη συστοιχία twins και αφορούν 16 διεργασίες στην περίπτωση του μοντέλου ανταλλαγής μηνυμάτων και 8 διεργασίες $\times$ 2 νήματα ανά διεργασία στα υβριδικά μοντέλα. Σε όλες τις περιπτώσεις έχουν ληφθεί οι ελάχιστοι χρόνοι παράλληλης εκτέλεσης των μετροπρογραμμάτων, που επιτυγχάνονται για τον εκάστοτε βέλτιστο κόκκο παραλληλισμού.

Είναι εμφανές ότι οι βελτιώσεις της επίδοσης ποικίλουν από οριακή μείωση του χρόνου εκτέλεσης μέχρι ιδιαίτερα υψηλή ποσοστιαία μείωση πλέον του 60%, ανάλογα τόσο με το θεωρούμενο μετροπρόγραμμα όσο και με τον εξεταζόμενο χώρο επαναλήψεων. Η ποσοστιαία μείωση του χρόνου εκτέλεσης προέρχεται αφενός από τη μείωση του συνολικού όγκου επικοινωνίας μέσω επιλογής κατάλληλης καρ-

τεσιανής τοπολογίας απεικόνισης διεργασιών και αφετέρου από την υλοποίηση ενός αποδοτικού υβριδικού μοντέλου χονδρού κόκκου για την αξιοποίηση της υφιστάμενης αρχιτεκτονικής κατανεμημένης μοιραζόμενης μνήμης.



(α) μετροπρόγραμμα ADI

(β) μετροπρόγραμμα DE-XYT

(γ) μετροπρόγραμμα DE-TXY

(δ) μετροπρόγραμμα Adv2D

Σχήμα 6.31: *Συμβολή της διατριβής στη βελτίωση της επίδοσης παράλληλων προγραμμάτων με χρήση των προτεινόμενων μοντέλων παραλληλοποίησης και τεχνικών βελτιστοποίησης για λεπτομερή παραλληλισμό με μοναδιαίο ύψος υπερκόμβου (διάφορα μετροπρογράμματα, 16 διεργασίες ή 8 διεργασίες $\times$ 2 νήματα, συστοιχία twins)*

Μια ιδιαίτερα περιοριστική παραδοχή που υιοθετήσαμε στο αλγοριθμικό μοντέλο ήταν η απαίτηση για ομοιόμορφες μη αρνητικές εξαρτήσεις δεδομένων. Το γεγονός αυτό περιόρισε το σύνολο των υπό παραλληλοποίηση αλγορίθμων σε εκείνους που χαρακτηρίζονται από μονόδρομη ροή πληροφορίας, όπως π.χ. συμβαίνει στην εξίσωση μεταφοράς, ενώ αντίθετα οδήγησε σε αλγοριθμική απόκλιση στις περιπτώσεις που απαιτείται συνεκτίμηση της αμφίδρομης αλληλεπίδρασης, όπως π.χ. στην εξί-

σωση θερμικής διάχυσης. Όμως, οι βασικές τεχνικές βελτιστοποίησης που προτείνονται τόσο για την ελαχιστοποίηση της επικοινωνίας όσο και για την εξισορρόπηση του φορτίου των νημάτων είναι χρήσιμες και για την περίπτωση της λεπτομερούς παραλληλοποίησης που ακολουθείται σε αλγορίθμους με γενικότερα διανύσματα εξάρτησης δεδομένων τύπου ροής.

Πράγματι, αν οι εξαρτήσεις δεν επιτρέπουν την ομαδοποίηση των δεδομένων επικοινωνίας *ανά υπερκόμβο* και υπαγορεύουν την υλοποίηση λεπτομερούς παραλληλοποίησης με επικοινωνία *ανά επιφάνεια* υπολογισμού, τόσο η επιλογή κατάλληλης τοπολογίας διεργασιών απεικόνισης όσο και η αποδοτική εξισορρόπηση του φορτίου των νημάτων για κάθε επιφάνεια διατηρούν την αξία τους στη μείωση του συνολικού χρόνου εκτέλεσης. Για την ποσοτική επιβεβαίωση του ισχυρισμού αυτού παρατίθεται το σχήμα 6.31, που απεικονίζει τη βελτίωση των χρόνων εκτέλεσης στα υπό εξέταση μετροπρογράμματα για ύψος υπερκόμβου $z = 1$. Παρατηρούμε πως η ποσοστιαία μείωση του χρόνου εκτέλεσης για ύψος υπερκόμβου $z = 1$ είναι αντίστοιχη με τη μείωση που επιτυγχάνεται στον ελάχιστο χρόνο εκτέλεσης (βλ. σχήμα 6.30). Η παρατήρηση αυτή έχει ιδιαίτερη πρακτική αξία, π.χ. αν αναλογιστούμε πως η ρητή διακριτοποίηση των ΜΔΕ οδηγεί σε επαναληπτικούς αλγορίθμους φωλιασμένων βρόχων με εξαρτήσεις δεδομένων τύπου ροής, που μπορούν να παραλληλοποιηθούν με τη χρήση επικοινωνίας για την ανταλλαγή των συνοριακών δεδομένων μετά από τους υπολογισμούς που ολοκληρώνονται σε κάθε χρονική στιγμή. Η λεπτομερής αυτή παραλληλοποίηση αποτελεί εξειδίκευση της μεθόδου που παρουσιάζεται στην παρούσα διατριβή για μοναδιαίο ύψος υπερκόμβου.

ΚΕΦΑΛΑΙΟ 7

Συμπεράσματα - Προτάσεις για Μελλοντική Έρευνα

Σκοπός της παρούσας εργασίας ήταν η διερεύνηση της επίδοσης προγραμματιστικών μοντέλων για την παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων σε σύγχρονες αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Εξετάστηκαν τα δημοφιλέστερα μοντέλα, που έχουν επιτυχώς εφαρμοστεί στην πράξη για την παραλληλοποίηση σχετικών επιστημονικών εφαρμογών, όπως το μονολιθικό μοντέλο ανταλλαγής μηνυμάτων, καθώς και η υβριδική προσέγγιση παραλληλοποίησης με επιπλέον χρήση πολυνηματικής επεξεργασίας. Περιγράφησαν προγραμματιστικές υλοποιήσεις για την παραλληλοποίηση αλγορίθμων τέλεια φωλιασμένων βρόχων, που εφαρμόζουν χρονοδρομολόγηση τύπου σωλήνωσης και επιτρέπουν την επικάλυψη ωφέλιμου υπολογισμού με την απαιτούμενη επικοινωνία. Αναπτύχθηκαν τεχνικές καθορισμού κατάλληλης εικονικής τοπολογίας διεργασιών για την παράλληλη απεικόνιση ενός συγκεκριμένου αλγορίθμου, καθώς και αποδοτικές μέθοδοι εξισορρόπησης του συνολικού υπολογιστικού φορτίου των νημάτων. Οι τεχνικές αυτές αξιολογήθηκαν σε μετροπρογράμματα και αναλύθηκαν συγκριτικά τα πειραματικά αποτελέσματα.

Συμπερασματικά θα μπορούσε κανείς να πει πως η επιλογή κατάλληλης εικονικής τοπολογίας διεργασιών για την απεικόνιση του παράλληλου προγράμματος είναι πολύ σημαντική ως προς την ελαχιστοποίηση της συνολικής επιβάρυνσης της επικοινωνίας και κατ' επέκταση τη διασφάλιση βέλτιστης επίδοσης. Η επιλογή κατάλληλης τοπολογίας διεργασιών κρίνεται ως ιδιαίτερα χρήσιμη κατά την παραλληλοποίηση αλγορίθμων που

- χαρακτηρίζονται εγγενώς από υψηλές απαιτήσεις επικοινωνίας σε σχέση με τις αντίστοιχες απαιτήσεις υπολογισμού για μια δεδομένη αρχιτεκτονική υποδομή

- κατανέμουν ανομοιόμορφα τις ανάγκες επικοινωνίας ως προς τις διάφορες διευθύνσεις του χώρου, δηλαδή το κόστος επικοινωνίας που σχετίζεται με κάποια διεύθυνση του χώρου επαναλήψεων του αλγορίθμου είναι αισθητά υψηλότερο από το σχετικό κόστος που αντιστοιχεί στις υπόλοιπες διευθύνσεις του χώρου

- συνδυάζονται με ασύμμετρους χώρους επαναλήψεων, στους οποίους κάποια διάσταση είναι αρκετά μεγαλύτερη από τις υπόλοιπες

- συνδυάζουν δύο ή περισσότερα από τα παραπάνω χαρακτηριστικά

Το μοντέλο ανταλλαγής μηνυμάτων, εφοδιασμένο με ένα σχήμα επικαλυπτόμενης χρονοδρομολόγησης υπολογισμών και επικοινωνίας και υπό τον καθορισμό μιας κατάλληλης τοπολογίας διεργασιών απεικόνισης, αποδεικνύεται ιδιαίτερα αποδοτικό σε περιβάλλον συστοιχίας πολυεπεξεργαστικών στοιχείων για την παραλληλοποίηση αλγορίθμων φωλιασμένων βρόχων.

Σε ότι αφορά στα υβριδικά μοντέλα παραλληλοποίησης, ο συνήθης προσαυξητικός παραλληλισμός λεπτού κόκκου που προτιμάται, αποτελεί μεν μια αρκετά απλή και εφαρμόσιμη προσέγγιση για την υλοποίηση υβριδικού παραλληλισμού, αλλά κρίνεται σχετικά περιοριστικός όσον αφορά στις δυνατότητες επίδοσης που συνεπάγεται για αλγορίθμους φωλιασμένων βρόχων. Συγκεκριμένα, καθώς ο υπολογισμός και η επικοινωνία διαχωρίζονται σε μη επικαλυπτόμενα τμήματα, επιτυγχάνεται χαμηλός βαθμός παραλληλίας βάσει του νόμου του Amdahl, καθιστώντας το υβριδικό μοντέλο λεπτού κόκκου λιγότερο αποδοτικό σε σχέση με το μονολιθικό μοντέλο ανταλλαγής μηνυμάτων. Η υλοποίηση SPMD υβριδικού παραλληλισμού χονδρού κόκκου δεν βελτιώνει την επίδοση, παρά την πρόσθετη προγραμματιστική πολυπλοκότητα που συνεπάγεται. Στη συνήθη περίπτωση περιορισμένης πολυνηματικής υποστήριξης από τη μεριά της βιβλιοθήκης ανταλλαγής μηνυμάτων, το funneled υβριδικό μοντέλο χονδρού κόκκου επιφορτίζει το πρωτεύον νήμα περισσότερο από τα υπόλοιπα, καθιστώντας έτσι επιτακτική την εφαρμογή κάποιου σχήματος εξισορρόπησης φορτίου. Ακόμα κι αν υποστηρίζεται πλήρως η πολυνηματική επεξεργασία κατά την υβριδική παραλληλοποίηση (multiple μοντέλο χονδρού κόκκου), η διασφάλιση της ορθής πρόσβασης των νημάτων στις μοιραζόμενες δομές επικοινωνίας επιφέρει επιβάρυνση που σε πολλές περιπτώσεις δεν είναι αμελητέα. Αντίθετα, ο συνδυασμός του funneled υβριδικού μοντέλου χονδρού κόκκου με την εφαρμογή του σχήματος μεταβλητής ή δυναμικής εξισορρόπησης φορτίου αποδεικνύεται στην πράξη η πιο αποτελεσματική υβριδική προσέγγιση, καθώς επιτυγχάνει το μικρότερο χρόνο εκτέλεσης, αυξάνοντας το συνολικό βαθμό παραλληλίας του προγράμματος και κατά το δυνατόν ισοκατανέμοντας το συνολικό φορτίο μεταξύ των διαθέσιμων φορέων παράλληλης επεξεργασίας. Για την εφαρμογή εξισορρόπησης φορτίου μεταξύ των νημάτων η παρούσα διατριβή κατέδειξε τα εξής:

- Η εφαρμογή αποδοτικής στατικής εξισορρόπησης φορτίου των νημάτων με χρήση του μεταβλητού σχήματος είναι εφικτή, εφόσον εκτιμηθούν και μοντελοποιηθούν θεωρητικά τα σχετικά

κόστη υπολογισμού και επικοινωνίας έστω με ένα απλό γραμμικό μοντέλο, αλλά με χρήση ενός αντιπροσωπευτικού δοκιμαστικού χώρου επαναλήψεων σε σχέση με το πραγματικό πεδίο εφαρμογής του αλγορίθμου.

- Η εφαρμογή ενός κοινού συντελεστή εξισορρόπησης φορτίου για όλες τις διεργασίες, αδιακρίτως της θέσης τους στην καρτεσιανή τοπολογία απεικόνισης, παρέχει εν γένει χειρότερη επίδοση σε σχέση με το μεταβλητό σχήμα εξισορρόπησης, καθώς υπερτιμά το κόστος επικοινωνίας για συνοριακές διεργασίες.

- Η εφαρμογή δυναμικής εξισορρόπησης φορτίου των νημάτων μπορεί να είναι ιδιαίτερα αποδοτική εφόσον κάθε διεργασία αναλάβει την εκτέλεση ενός σημαντικού πλήθους υπερκόμβων, ώστε η περίοδος δειγματοληψίας των χρόνων και ο επαναπροσδιορισμός των συντελεστών εξισορρόπησης να μην επιφέρουν σημαντική επιβάρυνση σε σχέση με τα οφέλη που θα αποφέρει η εφαρμογή των νέων συντελεστών.

Συνολικά, η απευθείας σύγκριση της υβριδικής παραλληλοποίησης με το μονολιθικό μοντέλο ανταλλαγής μηνυμάτων είναι σχετικά πολύπλοκη διαδικασία, καθώς εμπεριέχει πολλές μεταβλητές παραμέτρους που αφορούν στο υφιστάμενο υλικό και λογισμικό. Η διαδεδομένη χρήση του μονολιθικού μοντέλου και η σχεδόν καθολική αποδοχή του από την επιστημονική και εμπορική κοινότητα έχουν οδηγήσει σε πληθώρα υψηλά βελτιστοποιημένων βιβλιοθηκών ανταλλαγής μηνυμάτων, τόσο ελεύθερης διανομής όσο και κλειστού τύπου. Για παράδειγμα, η βιβλιοθήκη MPICH έχει αφενός χρησιμοποιηθεί επιτυχώς στην παραλληλοποίηση πολλών πραγματικών εφαρμογών μεγάλης κλίμακας και αφετέρου αποτελέσει αφετηριακό σημείο για τη σχεδίαση νέων MPI υλοποιήσεων για πληθώρα αρχιτεκτονικών και δικτυακών υποδομών. Από την άλλη μεριά, το πρότυπο OpenMP μέχρι πρόσφατα υποστηριζόταν σχεδόν αποκλειστικά από τον εμπορικό μεταγλωττιστή της Intel για x86 αρχιτεκτονικές, ενώ μόλις κατά την ολοκλήρωση της διατριβής ανακοινώθηκε η πρώιμη υποστήριξη για OpenMP στον ευρύτατα διαδεδομένο GNU μεταγλωττιστή (gcc). Επίσης, παρότι το ενδεχόμενο πολυνηματικής υποστήριξης έχει προβλεφθεί θεωρητικά στο πρότυπο MPI-2, δεν υπάρχουν μέχρι αυτή τη στιγμή ελεύθερες υλοποιήσεις που να παρέχουν το υψηλότερο επίπεδο `MPI_THREAD_MULTIPLE` της πολυνηματικής επεξεργασίας. Ωστόσο, θα μπορούσε κανείς να ισχυριστεί με ασφάλεια πως το υβριδικό μοντέλο χονδρού κόκκου με επιλογή κατάλληλης τοπολογίας διεργασιών και εφαρμογή αποδοτικής εξισορρόπησης του φορτίου των νημάτων σε κάθε περίπτωση είναι εξίσου αποδοτικό με το απλό μοντέλο ανταλλαγής μηνυμάτων και σε πολλές μάλιστα περιπτώσεις παρέχει σημαντικές βελτιώσεις όσον αφορά στο συνολικό χρόνο εκτέλεσης. Η ευελιξία που προσφέρει η διεπίπεδη δομή του υβριδικού μοντέλου οδηγεί σε πληθώρα πλεονεκτημάτων, που μπορούν να υπερκεράσουν τα βασικά προτερήματα της μεταφερσιμότητας και της γενικότητας του μοντέλου ανταλλαγής μηνυμάτων.

Είναι σίγουρο πως η Παράλληλη Επεξεργασία θα συνεχίσει να απασχολεί ζωηρά την επιστημονική κοινότητα ανεξαρτήτως των τεχνολογικών και αρχιτεκτονικών εξελίξεων, κυρίως γιατί βασίζεται στις διαχρονικές αρχές της επεκτασιμότητας και της διασύνδεσης εμπορικών συστημάτων. Ενδεικτική προς την κατεύθυνση αυτή είναι η στροφή που παρατηρείται με τη μελέτη, διερεύνηση και ανάλυση εξειδικευμένων υπολογιστικά απαιτητικών εφαρμογών από ποικίλα επιστημονικά πεδία, καθώς και με την ανάπτυξη ισχυρών εργαλείων και βιβλιοθηκών για παράλληλους αλγορίθμους. Θεωρούμε ότι η μοντελοποίηση και διερεύνηση των φωλιασμένων βρόχων συνετέλεσαν καθοριστικά στη συστηματική και μεθοδική ανάπτυξη βασικών τεχνικών παραλληλοποίησης, αλλά εν μέρει μετεξελίχθηκαν σε αυτοσκοπό της σχετικής ερευνητικής δραστηριότητας, οδηγώντας σε πολυσύνθετη μαθηματική μοντελοποίηση που ελάχιστα παρακολουθεί τις ιδιομορφίες και τη φύση των πραγματικών εφαρμογών υψηλών επιδόσεων. Όλες οι τεχνικές παραλληλοποίησης και οι περαιτέρω βελτιστοποιήσεις, που έχουν κατά καιρούς καταγραφεί στη σύγχρονη βιβλιογραφία, καθώς και οι προτεινόμενες μεθοδολογίες της παρούσας διατριβής μπορούν να ωφελήσουν ουσιαστικά την επιστημονική κοινότητα μόνο στο βαθμό που βρίσκουν εφαρμογή στην βελτίωση της επίδοσης πραγματικών αλγορίθμων και ολοκληρωμένων εφαρμογών.

Βάσει των παραπάνω, θεωρούμε πως μελλοντικά η σχετική έρευνα στο χώρο της Παράλληλης Επεξεργασίας γενικότερα και των τεχνικών βελτιστοποίησης παράλληλων προγραμματιστικών μοντέλων ειδικότερα θα μπορούσε να ασχοληθεί με τα ακόλουθα ζητήματα:

- Εστίαση σε *συγκεκριμένες επιστημονικές εφαρμογές*, που μπορούν να χρησιμεύσουν ως αφετηριακό σημείο για την αξιοποίηση υπαρχουσών τεχνικών παραλληλοποίησης, την ανάδειξη εξειδικευμένων προβλημάτων της συγκεκριμένης εφαρμογής, καθώς και ενδεχομένως την ανάπτυξη νέων σχετικών βελτιστοποιήσεων και προγραμματιστικών μεθόδων. Οι φωλιασμένοι βρόχοι παρουσιάζουν μια ακολουθία χαρακτηριστικών, όπως είναι η κανονικότητα της προγραμματιστικής δομής, η σχετικά ομοιόμορφη κατανομή του υπολογιστικού φορτίου ανά επανάληψη, η έμφαση σε ζητήματα επικοινωνίας και συγχρονισμού έναντι άλλων βασικών ζητημάτων της Παράλληλης Επεξεργασίας (π.χ. αρχική κατανομή των δεδομένων εισόδου, αποθήκευση των δεδομένων εξόδου, περιορισμός της επιβάρυνσης εισόδου/εξόδου, δυναμική διαμόρφωση του πλήθους των διεργασιών σε χρόνο εκτέλεσης κ.ά.) που δεν αντιπροσωπεύουν κατ' ανάγκη το σύνολο των υπολογιστικά απαιτητικών εφαρμογών. Πιστεύουμε ότι απαιτείται συνεργασία σε διεπιστημονικό επίπεδο, ώστε να εξεταστούν τόσο ζητήματα βελτιστοποιήσεων που άπτονται της συμπεριφοράς του υλικού και του λογισμικού των παράλληλων συστημάτων όσο όμως και βασικά θέματα αλγοριθμικής φύσεως, όπως η θεωρητική αξιολόγηση της πολυπλοκότητας και της ευστάθειας μιας μεθόδου επίλυσης, η κατανόηση της φυσικής σημασίας ενός προβλήματος, η επιλογή βέλτιστου συνόλου φυσικών παραμέτρων ή/και αρχικών τιμών για συγκεκριμένο συνδυασμό αλγορίθμου και δεδομένων εισόδου, η εστίαση της προγραμματιστικής προσπάθειας σε συγκεκριμένα πεδία

επιστημονικού ενδιαφέροντος κ.ο.κ.

- Η *υβριδική ή γενικότερα πολυεπίπεδη παραλληλοποίηση* είναι σίγουρο ότι έχει μελλοντικά πολλά να προσφέρει, ιδιαίτερα καθώς παρατηρείται μια τάση προς τις ιεραρχικές αρχιτεκτονικές υψηλών επιδόσεων. Στο πλαίσιο της παρούσας διατριβής έγινε απόπειρα να αξιοποιηθούν κυρίως τα πλεονεκτήματα των υβριδικών μοντέλων που αφορούν στην ελαχιστοποίηση του κόστους επικοινωνίας, ιδίως σε αρχιτεκτονικές κατανεμημένης μοιραζόμενης μνήμης. Εντούτοις, θα μπορούσε κανείς να εκμεταλλευτεί πολλά ακόμη προτερήματα της υβριδικής παραλληλοποίησης, όπως αυτά περιγράφονται και στην εργασία [SB01]. Ενδεικτικά αναφέρουμε τις δυνατότητες για εύκολη δυναμική εξισορρόπηση φορτίου π.χ. για τη διεκπεραίωση μη κανονικών υπολογισμών, την περίπτωση εφαρμογών που ευνοούνται εγγενώς από παραλληλισμό λεπτού κόκκου, την αποφυγή ενδεχόμενων περιορισμών που σχετίζονται με το πλήθος των διεργασιών ή την επεκτασιμότητα της βιβλιοθήκης ανταλλαγής μηνυμάτων, τον περιορισμό των αντιγράφων δεδομένων με αξιοποίηση του κοινού χώρου διευθύνσεων κ.ά.

- Τα σύγχρονα δίκτυα διασύνδεσης (π.χ. Infiniband, Myrinet, Gigabit Ethernet) εξελίσσονται συνεχώς και επιτυγχάνουν ολοένα και χαμηλότερες τιμές αρχικής καθυστέρησης διάδοσης, προσεγγίζοντας μάλιστα το μέσο χρόνο προσπέλασης της τοπικής μνήμης. Απόρροια του γεγονότος αυτού είναι μεταξύ άλλων και η προτυποποίηση νέων λειτουργιών στις βιβλιοθήκες ανταλλαγής μηνυμάτων, όπως οι *λειτουργίες μονόπλευρης επικοινωνίας* (*RMA*) του MPI-2 προτύπου. Η εφαρμογή των λειτουργιών αυτών στην παραλληλοποίηση πραγματικών αλγορίθμων βρίσκεται ακόμα σε πρώιμο στάδιο, και αναμένεται να αποτελέσει αντικείμενο μελλοντικής έρευνας.

- Ομοίως, η διασφάλιση *πλήρους πολυνηματικής υποστήριξης* (επίπεδο `MPI_THREAD_MULTIPLE` του MPI προτύπου) τελεί στην παρούσα φάση υπό ανάπτυξη σε κάποιες περιπτώσεις δημοφιλών βιβλιοθηκών ανταλλαγής μηνυμάτων ανοιχτού κώδικα, όπως η Open MPI, και διατίθεται μόνο από λίγες κλειστές βιβλιοθήκες εμπορικού τύπου, όπως η MPI/Pro. Ένα ζήτημα που εξετάστηκε μερικώς στην παρούσα εργασία και χρήζει περαιτέρω διερεύνησης και κυρίως πειραματικής αξιολόγησης είναι κατά πόσο η ευχέρεια που παρέχει η πλήρης πολυνηματική υποστήριξη στην κλήση λειτουργιών ανταλλαγής μηνυμάτων από όλα τα νήματα μπορεί να αποφέρει μικρότερους χρόνους εκτέλεσης από ένα funneled μοντέλο χονδρού κόκκου, που χρησιμοποιεί κάποιο σχήμα αποδοτικής εξισορρόπησης φορτίου και παράλληλα αποφεύγει την πρόσθετη επιβάρυνση που σχετίζεται με τη διασφάλιση πλήρους πολυνηματικής υποστήριξης (π.χ. κλειδώματα κατά την πρόσβαση σε κοινές δομές επικοινωνίας ή σε συγκεκριμένα τμήματα του κώδικα ανταλλαγής μηνυμάτων).

- Δύο περαιτέρω νέα στοιχεία που προτυποποιούνται στο MPI-2 και υποστηρίζονται σε πρώιμο

στάδιο από νέες υλοποιήσεις είναι η δυνατότητα *δυναμικής διαχείρισης των διεργασιών* (*dynamic process management*) καθώς και η *παράλληλη είσοδος και έξοδος* (*parallel I/O*). Οι δυνατότητες αυτές θα μπορούσαν να χρησιμεύσουν ιδίως σε εφαρμογές με μεταβλητό φορτίο, που δεν μπορεί να προβλεφθεί με ακρίβεια κατά το χρόνο μεταγλώττισης, καθώς και σε εφαρμογές των οποίων η επίδοση περιορίζεται σημαντικά από λειτουργίες εισόδου/εξόδου.

- Η επιλογή κατάλληλης καρτεσιανής τοπολογίας διεργασιών θα μπορούσε να ενσωματωθεί στη βιβλιοθήκη ανταλλαγής μηνυμάτων (π.χ. κλήση `MPI_Cart_create` του MPI) εφόσον προσκομίζεται επιπλέον πληροφορία για τη σημασιολογία του αλγορίθμου, όπως η μορφή του χώρου επαναλήψεων ή οι αλγοριθμικές εξαρτήσεις δεδομένων που οδηγούν σε επικοινωνία. Η δουλειά που έχει καταγραφεί στη διεθνή βιβλιογραφία προς την κατεύθυνση αυτή, μέρος της οποίας έχει ενσωματωθεί σε υλοποιήσεις της βιβλιοθήκης MPI, αφορά στον καθορισμό καρτεσιανής τοπολογίας που στοχεύει να αντιστοιχίσει διεργασίες που επικοινωνούν για ανταλλαγή δεδομένων σε κόμβους της υφιστάμενης αρχιτεκτονικής που επιτυγχάνουν συγκριτικά ταχύτερη αμοιβαία πρόσβαση, με δεδομένο όμως το πλήθος των διεργασιών σε κάθε διάσταση της τοπολογίας. Θα ήταν ίσως ενδιαφέρον να εξεταστεί κατά πόσο είναι χρήσιμος ο αυτόματος προσδιορισμός μιας προτεινόμενης τοπολογίας από τη βιβλιοθήκη ανταλλαγής μηνυμάτων, που να εφαρμόζει τις τεχνικές της παρούσας διατριβής για την ελαχιστοποίηση του όγκου των δεδομένων επικοινωνίας.

- Τέλος, ένα μεγάλο ανοιχτό ερευνητικό ζήτημα παραμένει ο κατά το δυνατόν αυτόματος προσδιορισμός του βέλτιστου κόκκου παραλληλισμού για μια δεδομένη αρχιτεκτονική υποδομή και ένα συγκεκριμένο αλγόριθμο φωλιασμένων βρόχων. Στο πλαίσιο της παρούσας εργασίας θεωρήσαμε τη μία από τις διαστάσεις του υπερκόμβου (ύψος z) ως ελεύθερη παράμετρο, επιτρέποντας έτσι τον πειραματικό καθορισμό του βέλτιστου κόκκου παραλληλισμού. Θα ήταν σίγουρα εξαιρετικά χρήσιμο αν η βέλτιστη τιμή σε κάθε περίπτωση μπορούσε να προκύψει από την ακριβή συνεκτίμηση της πληθώρας των παραμέτρων που επηρεάζουν τη συνολική επίδοση του παράλληλου προγράμματος (π.χ. σχετικό κόστος υπολογισμού-επικοινωνίας, τρόπος προσπέλασης δεδομένων στην ιεραρχία μνήμης, επιβαρύνσεις υλικού και λογισμικού κ.ά.).

ΠΑΡΑΡΤΗΜΑ A

Το Πρότυπο MPI

Το Message Passing Interface (MPI) αποτελεί μια προσπάθεια για καταγραφή, οργάνωση και προτυποποίηση των κυριότερων λειτουργιών που θα πρέπει να υποστηρίζει ένα προγραμματιστικό περιβάλλον ανταλλαγής μηνυμάτων, με άξονες τη λειτουργικότητα, τη φορητότητα και την επίτευξη υψηλής επίδοσης σε επίπεδο εφαρμογής. Η προσπάθεια αυτή ξεκίνησε ουσιαστικά το 1992 από μια ομάδα συντελεστών προερχόμενων από τους χώρους της ακαδημαϊκής έρευνας, των υπολογιστών και της βιομηχανίας. Σε γενικές γραμμές, ο στόχος της προσπάθειας αυτής ήταν η ανάπτυξη ενός εύχρηστου προγραμματιστικού περιβάλλοντος για τη συγγραφή προγραμμάτων ανταλλαγής μηνυμάτων, που θα γινόταν ευρύτερα αποδεκτό από την ενδιαφερόμενη κοινότητα. Τον Ιούνιο του 1994 συντάχθηκε η έκδοση 1.0 του προτύπου MPI, η οποία μετεξελίχθηκε σε 1.1 τον επόμενο χρόνο και σε 1.2 λίγο αργότερα, χωρίς πάντως σημαντικές διαφορές σε σχέση με την αρχική έκδοση 1.0. Σημαντική εξέλιξη υπήρξε το 1997, όταν ολοκληρώθηκε η καταγραφή της έκδοσης 2.0 του προτύπου, που επιπλέον προδιαγράφει και νέους τύπους λειτουργιών σε σχέση με την αρχική προσέγγιση. Στη σημερινή πραγματικότητα, το MPI έχει επικρατήσει ως η πλέον διαδεδομένη και ευρύτερα αποδεκτή βιβλιοθήκη ανταλλαγής μηνυμάτων και χρησιμοποιείται κατά κόρον στο χώρο της παράλληλης επεξεργασίας.

Βάσει του προτύπου MPI-1.2 δομική μονάδα του παράλληλου προγράμματος είναι η *διεργασία* (*process*). Οι διεργασίες οργανώνονται σε ομάδες (groups), στο πλαίσιο των οποίων αποδίδεται ως μοναδικό αναγνωριστικό σε κάθε διεργασία ένας ακέραιος *βαθμός* (*rank*). Χρησιμοποιώντας το βαθμό της κάθε διεργασία μπορεί να διαφοροποιήσει τη ροή εκτέλεσής της, επενεργώντας πάνω σε διαφορετικό υποσύνολο των δεδομένων ή/και εκτελώντας διαφορετικούς υπολογισμούς. Οι λειτουργίες

επικοινωνίας ομαδοποιούνται σε δύο βασικές κατηγορίες: από σημείο σε σημείο (point-to-point) και συλλογικής επικοινωνίας (collective communication). Για κάθε κατηγορία, ορίζεται η σημασιολογία εναλλακτικών δυνατοτήτων αποστολής και λήψης, π.χ. σύγχρονη, ασύγχρονη κλπ. Ουσιαστικά, προδιαγράφοντας πληθώρα δυνατοτήτων και χαρακτηριστικών επικοινωνίας σε αφαιρετικό επίπεδο, το MPI αφενός διασφαλίζει φορητότητα (μεταφερσιμότητα) σε πολλαπλές αρχιτεκτονικές και αφετέρου παρέχει στον προγραμματιστή μιας υλοποίησης του MPI τη δυνατότητα υποστήριξης προηγμένων αρχιτεκτονικών χαρακτηριστικών. Επιπλέον, το πρότυπο MPI-2 προδιαγράφει υποστήριξη για δυναμική διαχείριση διεργασιών (dynamic process management), απομακρυσμένη προσπέλαση στη μνήμη (remote memory access) και παράλληλη είσοδο/έξοδο (parallel I/O).

Θα πρέπει να τονιστεί πως το MPI δεν είναι μια συγκεκριμένη *υλοποίηση* ενός περιβάλλοντος ανταλλαγής μηνυμάτων, αλλά ένα *πρότυπο* που ορίζει τις σχετικές λειτουργίες που θα πρέπει να υποστηρίζει μια τέτοια βιβλιοθήκη και προσδιορίζει τη σημασιολογία αυτών. Έτσι, πολλοί κατασκευαστές λογισμικού παρέχουν ελεύθερες ή κλειστού τύπου (εμπορικές) υλοποιήσεις της βιβλιοθήκης MPI, που συμμορφώνονται στο πρότυπο του MPI και προορίζονται για πληθώρα παράλληλων αρχιτεκτονικών. Ενδεικτικά μπορεί κανείς να παραθέσει τις ακόλουθες υλοποιήσεις:

- **MPICH** [`http://www-unix.mcs.anl.gov/mpi/mpich/`] Η πλέον δημοφιλής υλοποίηση ανοιχτού κώδικα του MPI. Υποστηρίζει μια πληθώρα υφιστάμενων παράλληλων αρχιτεκτονικών, ενώ σε αφαιρετικό επίπεδο ορίζει τη συσκευή ADI (*Abstract Device Interface*, *Αφαιρετική Διεπαφή Συσκευής*), που επιτρέπει με δομημένο τρόπο την υλοποίηση κώδικα για υποστήριξη μιας νέας αρχιτεκτονικής δικτύου. Η συσκευή ADI (μετέπειτα ADI-2 και ADI-3) αποτελεί το βασικό λόγο για τον οποίο η υλοποίηση MPICH έχει χρησιμεύσει ως βάση για την ανάπτυξη πολλών άλλων υλοποιήσεων του MPI. Υποστηρίζει το πρότυπο MPI-1.2.

- **MPICH2** [`http://www-unix.mcs.anl.gov/mpi/mpich2/`] Αποτελεί προσπάθεια για αναθεώρηση και βελτίωση της υλοποίησης MPICH, με βασικό άξονα την υποστήριξη του προτύπου MPI-2 του MPI. Δεν βασίζεται στην αρχιτεκτονική του καναλιού Chameleon ή στη σχετικά παλιά βιβλιοθήκη P4 για την υλοποίηση της επικοινωνίας (όπως το MPICH), αλλά αντίθετα διαχωρίζει τη διαχείριση των διεργασιών με το σκέλος της επικοινωνίας. Για το πρώτο χρησιμοποιεί την αρχιτεκτονική δαίμονα του mpd, ενώ για το δεύτερο εφαρμόζει την αρχιτεκτονική αφαιρετικής διεπαφής συσκευής ADI-3.

- **MPICH-GM** [`http://www.myri.com/scs/`] Το MPICH-GM παρέχει υποστήριξη για το δίκτυο διασύνδεσης Myrinet και στην παρούσα φάση υποστηρίζει το πρότυπο MPI-1.2. Βασίζεται στην υλοποίηση MPICH, ενώ χρησιμοποιεί τη βιβλιοθήκη GM για την υλοποίηση των λειτουργιών επικοινωνίας στο Myrinet. Είναι υλοποίηση ανοιχτού κώδικα.

- **LAM/MPI** [http://www.lam-mpi.org/] Άλλη μια δημοφιλής υλοποίηση ανοιχτού κώδικα MPI, που παρέχει πλήρη υποστήριξη για το πρότυπο MPI-1.2 και μερική υποστήριξη για το MPI-2. Παρέχει κάλυψη για πληθώρα δικτύων διασύνδεσης (π.χ. Myrinet, TCP/IP, Infiniband, μοιραζόμενη μνήμη κ.ά.), ενώ υιοθετεί μια δομημένη (modular) προσέγγιση, που διευκολύνει την επέκταση της παρεχόμενης υποστήριξης σε διαφορετικούς τύπους αρχιτεκτονικών και παρέχει ευελιξία επιλογών σε χρόνο εκτέλεσης.

- **LA-MPI** [http://public.lanl.gov/lampi/] Μια ακόμη υλοποίηση ανοιχτού κώδικα, που παρέχει υποστήριξη για το πρότυπο MPI-1.2 και πληθώρα αρχιτεκτονικών (π.χ. Intel IA32, Intel IA64, AMD Opteron, PowerPC, Alpha, MIPS) και δικτύων διασύνδεσης (μοιραζόμενη μνήμη, Ethernet, Myrinet, QSNet, InfiniBand κ.ά.). Παράλληλα με την επίτευξη υψηλής απόδοσης δίνει ιδιαίτερη έμφαση στο ζήτημα της ανοχής σφαλμάτων (fault-tolerance).

- **SCI-MPICH** [http://www.lfbs.rwth-aachen.de/users/joachim/SCI-MPICH/] Υλοποίηση ανοιχτού τύπου (MPI-1.2), που βασίζεται στην MPICH και παρέχει υποστήριξη για το δίκτυο διασύνδεσης SCI μέσω των βιβλιοθηκών IRM και SISCI.

- **Open MPI** [http://www.open-mpi.org/] Αποτελεί μια προσπάθεια για ενοποίηση και ενσωμάτωση των πλεονεκτημάτων άλλων υλοποιήσεων του MPI (π.χ. FT-MPI, LA-MPI, LAM/MPI και PACX-MPI) προς την κατεύθυνση μιας υλοποίησης ανοιχτού κώδικα δομημένης αρχιτεκτονικής, που να παρέχει πλήρη υποστήριξη για το πρότυπο MPI-2.

- **MPI/Pro** [http://www.mpi-softtech.com/] Αποτελεί εμπορική (κλειστού τύπου) υλοποίηση. Διαθέτει προηγμένα χαρακτηριστικά, όπως πλήρη υποστήριξη του MPI-2, πολυνηματική υποστήριξη, πραγματική επικάλυψη υπολογισμού και επικοινωνίας κ.ά.

ΠΑΡΑΡΤΗΜΑ B

Το Πρότυπο OpenMP

Το OpenMP (Open Multi Processing) αποτελεί πρότυπο που ορίζει ένα σύνολο οδηγιών μεταγλωττιστή, συναρτήσεων βιβλιοθήκης και μεταβλητών συστήματος για παραλληλισμό μοιραζόμενης μνήμης σε Fortran, C και C++. Πρόκειται ουσιαστικά για μια διεπαφή προγραμματισμού εφαρμογής (Application Program Interface - API) που επιτρέπει την ανάπτυξη φορητών πολυνηματικών εφαρμογών για αρχιτεκτονικές μοιραζόμενης μνήμης. Ιστορικά, το πρώτο πρότυπο του OpenMP (1.0) εκδόθηκε τον Οκτώβριο του 1997 για Fortran και ένα χρόνο μετά για C/C++. Το Νοέμβριο του 1999 προτάθηκε η έκδοση 1.1 του προτύπου για Fortran, ενώ ακολούθησε η έκδοση 2.0 του προτύπου, που δημοσιεύτηκε το Νοέμβριο του 2000 για Fortran και το Μάρτιο του 2002 για C/C++. Η τρέχουσα έκδοση (2.5) δημοσιεύτηκε το Μάιο του 2005 και μεταξύ άλλων ενοποιεί τα πρότυπα για τις γλώσσες C/C++ και Fortran.

Το OpenMP προσφέρεται κυρίως για την εύκολη πολυνηματική παραλληλοποίηση εφαρμογών που βασίζονται σε πίνακες (array-based applications). Ο αρχικός σειριακός αλγόριθμος παραλληλοποιείται μέσω της κατανομής και απεικόνισης της υπολογιστικής ροής σε νήματα εκτέλεσης, τα οποία διατηρούν τόσο *ιδιωτικά* (*private*) όσο και *μοιραζόμενα* (*shared*) δεδομένα. Η διάκριση αυτή στα δεδομένα των νημάτων αντανακλά την ύπαρξη αναγκαιότητας ή μη για την επικοινωνία μεταξύ των νημάτων και οδηγεί αναπόφευκτα σε ανάγκη συγχρονισμού για τη διασφάλιση της σημασιολογικά ορθής ακολουθίας πρόσβασης στα μοιραζόμενα δεδομένα. Συγχρονισμός απαιτείται επίσης σε περιπτώσεις όπου υλοποιούνται λειτουργίες εισόδου/εξόδου σε πολυνηματικό επίπεδο.

Η βασική αρχή προγραμματισμού της διεπαφής OpenMP ακολουθεί το μοντέλο *διακλάδωσης-*

ένωσης (*fork-join*): αρχικά εκκινείται μοναδικό νήμα εκτέλεσης, που χαρακτηρίζεται και ως πρωτεύον νήμα (master thread). Όταν το πρωτεύον νήμα συναντήσει κάποια παράλληλη περιοχή, η ροή εκτέλεσης διακλαδώνεται σε πολλαπλά νήματα, ένα εκ των οποίων παραμένει και το αρχικό πρωτεύον νήμα. Κατά την έξοδο από την παράλληλη περιοχή τα νήματα συνενώνονται και μόνο το πρωτεύον συνεχίζει την εκτέλεση του προγράμματος. Τα νήματα επικοινωνούν με μοιραζόμενες μεταβλητές, ενώ όπου δεν παρουσιάζεται ανάγκη επικοινωνίας χρησιμοποιούνται ιδιωτικές μεταβλητές. Η ύπαρξη μοιραζόμενων μεταβλητών συνεπάγεται κατά κανόνα ανεπιθύμητες συνθήκες ανταγωνισμού μεταξύ των νημάτων (race conditions), για την αποφυγή των οποίων απαιτείται η υλοποίηση κατάλληλου συγχρονισμού (synchronization). Δεδομένου ότι ο συγχρονισμός αυτός αφενός είναι «ακριβός» σε υπολογιστική πολυπλοκότητα ή/και κατανάλωση διαθέσιμων πόρων, αφετέρου περιορίζει το βαθμό παραλληλίας, η αποδοτικότητα του παραλληλοποιημένου προγράμματος βρίσκεται σε άμεση αντιστοιχία με την ελαχιστοποίηση του χρησιμοποιούμενου συγχρονισμού.

Σε επίπεδο διεπαφής, το OpenMP παρέχει την απαιτούμενη προγραμματιστική ευκολία για την κατά το δυνατόν άμεση πολυνηματική παραλληλοποίηση ενός υπάρχοντος κώδικα Fortran ή C/C++. Έτσι, αντίθετα με άλλες προσεγγίσεις πολυνηματικής επεξεργασίας, το πρότυπο OpenMP διευκολύνει τον επαυξητικό παραλληλισμό ενός υπάρχοντος σειριακού κώδικα μέσω της προσθήκης κατάλληλων οδηγιών προς το μεταγλωττιστή, χωρίς να απαιτείται σημαντική αναμόρφωση του αρχικού κώδικα. Μάλιστα, ένα πρόγραμμα σε OpenMP μπορεί να εκτελεστεί και σειριακά (μονονηματικό περιβάλλον) σε περίπτωση που δεν υπάρχει η απαιτούμενη πολυνηματική υποστήριξη είτε από το μεταγλωττιστή, είτε από την υφιστάμενη αρχιτεκτονική, υπό την προϋπόθεση βέβαια πως έχει ληφθεί η αντίστοιχη μέριμνα σε σημασιολογικό επίπεδο από τον προγραμματιστή.

Ο πυρήνας του OpenMP είναι οι οδηγίες προς τον μεταγλωττιστή, που σύμφωνα με το πρότυπο ομαδοποιούνται στις ακόλουθες κατηγορίες:

1. *Παράλληλες περιοχές (parallel regions)*

   Αποτελούν τη βασική οδηγία δημιουργίας παράλληλων περιοχών πολυνηματικής επεξεργασίας. Δημιουργούν ομάδα νημάτων, ο αριθμός των οποίων καθορίζεται είτε δυναμικά είτε στατικά. Σε C/C++ η αντίστοιχη οδηγία είναι η `parallel`.

2. *Κατανομή εργασίας (work sharing constructs)*

   Επιτρέπουν την κατανομή εργασίας μεταξύ των διαθέσιμων νημάτων. Θα πρέπει να τονιστεί πως οι οδηγίες αυτές δεν δημιουργούν νέα νήματα, αλλά μοιράζουν τον όγκο υπολογισμού στα υπάρχοντα, που έχουν δημιουργηθεί πρωτύτερα με κάποια οδηγία `parallel`. Σε C/C++ προβλέπονται οι οδηγίες `for` (κατανομή επαναλήψεων εντολής for με στατικό, δυναμικό ή εκθετικό

τρόπο), `sections` (ορισμός ανεξάρτητων περιοχών που κατανέμονται μεταξύ των νημάτων) και `single` (τμήμα κώδικα μονονηματικής εκτέλεσης).

3. *Παράλληλες περιοχές κατανομής εργασίας (parallel work sharing constructs)*

   Αποτελούν ένα συνδυασμό των δύο παραπάνω περιπτώσεων, δηλαδή επιτρέπουν τη δημιουργία ομάδας νημάτων, στην οποία ταυτόχρονα κατανέμεται εργασία. Σε C/C++ ορίζονται οι οδηγίες `parallel for` και `parallel sections`.

4. *Σειριακή εκτέλεση - συγχρονισμός (master and synchronization directives)*

   Ορίζεται ομάδα λειτουργιών, που επιτρέπει αφενός την επίτευξη συγχρονισμού μεταξύ των διεργασιών και αφετέρου την υλοποίηση σειριακής εκτέλεσης σε περιοχές πολυνηματικής επεξεργασίας. Έτσι, σε C/C++ η οδηγία `master` ορίζει τμήμα κώδικα που εκτελείται μόνο από το πρωτεύον νήμα της ομάδας. Η `critical` οριοθετεί μια περιοχή, στην οποία επιτρέπεται σε κάθε χρονική στιγμή η πρόσβαση το πολύ σε ένα νήμα της ομάδας. Η οδηγία `barrier` συγχρονίζει όλα τα υπάρχοντα νήματα της ομάδας, ενώ η `atomic` ενημερώνει ατομικά μια περιοχή μνήμης. Η οδηγία `flush` υποδεικνύει τόσο στους μεταγλωττιστές να αποκαταστήσουν τις τιμές των καταχωρητών στη μνήμη (λειτουργία volatile της C) όσο και στο υλικό να αποκαταστήσει τους απομονωτές εγγραφής στη μνήμη. Τέλος, η λειτουργία `ordered` επιβάλλει σειριακή εκτέλεση ενός βρόχου for που περιέχεται σε μια παράλληλη περιοχή πολυνηματικής επεξεργασίας.

5. *Περιβάλλον δεδομένων (data environment)*

   Περιλαμβάνει μία οδηγία και περισσότερες προτάσεις (clauses) για τον έλεγχο του περιβάλλοντος δεδομένων σε παράλληλες περιοχές. Έτσι, σε C/C++ ορίζεται η οδηγία `threadprivate`, καθώς και οι προτάσεις `private`, `firstprivate`, `lastprivate`, `shared`, `default`, `reduction`, `copyin` και `copyprivate`. Από τα παραπάνω, ως πιο θεμελιώδεις μπορούν να θεωρηθούν οι προτάσεις `private` και `shared`, που επιτρέπουν τη δήλωση μοιραζόμενων και ιδιωτικών μεταβλητών, αντίστοιχα.

Η βιβλιοθήκη χρόνου εκτέλεσης που περιγράφεται από το OpenMP ορίζει λειτουργίες που διαχειρίζονται το περιβάλλον εκτέλεσης (π.χ. καθορισμός πλήθους νημάτων, απόδοση μοναδικού αναγνωριστικού σε κάθε νήμα, καθορισμός δυναμικής ή στατικής επιλογής του πλήθους των νημάτων), προδιαγράφουν συγχρονισμό με κλειδώματα (παρέχεται υποστήριξη και για φωλιασμένα κλειδώματα), καθώς και περιγράφουν λειτουργίες χρονομέτρησης. Τέλος, οι μεταβλητές συστήματος αναφέρονται στον προσδιορισμό της κατανομής των επαναλήψεων κατά το χρόνο εκτέλεσης (στατική, δυναμική, εκθετική μείωση), τον καθορισμό δυναμικής ή στατικής επιλογής του πλήθους των νημάτων κατά το

χρόνο εκτέλεσης, καθώς και στον ορισμό του μέγιστου αριθμού νημάτων του παράλληλου προγράμματος.

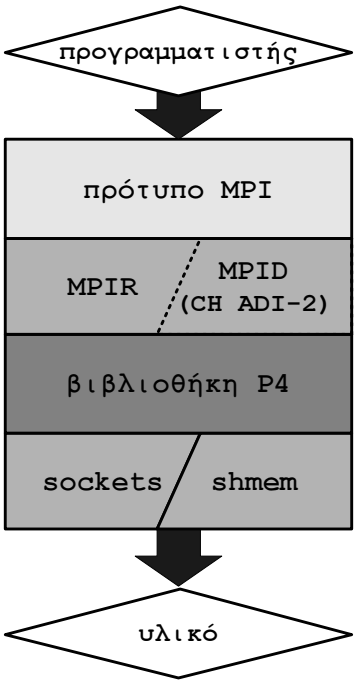
Εκτενέστερη παρουσίαση του προτύπου OpenMP υπάρχει στα [Boa05, DM98]. Το πρότυπο OpenMP εμφανίζει στην παρούσα φάση αρκετά σημαντικούς περιορισμούς στην επίδοση, όπως η μη υποστήριξη φωλιασμένου παραλληλισμού και οι περιορισμένες δυνατότητες εξισορρόπησης φορτίου μεταξύ των νημάτων. Αρκετά ενδιαφέρουσες προεκτάσεις του προτύπου αναφορικά με τα παραπάνω ζητήματα έχουν προταθεί στο μεταγλωττιστή NanosCompiler [GAM+00]. Για τις ανάγκες της παρούσας εργασίας χρησιμοποιήθηκε ο μεταγλωττιστής C/C++ της Intel (icc), που μπορεί να αναζητηθεί στη δικτυακή διεύθυνση `http://www.intel.com/software/products/compilers/clin/` και στην παρούσα φάση παρέχει υποστήριξη για το πρότυπο 2.5 του OpenMP. Οι εκδόσεις του icc μεταγλωττιστή που χρησιμοποιήθηκαν ήταν οι 8.1 και 9.

ΠΑΡΑΡΤΗΜΑ Γ

Σχεδίαση Υλοποίησης MPICH

Η υλοποίηση MPICH αποτελεί ενδεχομένως τη δημοφιλέστερη υλοποίηση του προτύπου MPI, καθώς μεταξύ άλλων έχει χρησιμεύσει ως βάση για την προσαρμογή του περιβάλλοντος MPI σε πολλά προηγμένα δίκτυα διασύνδεσης (π.χ. MPICH-GM για το δίκτυο Myrinet ή SCI-MPICH για το δίκτυο SCI). Βασικό προτέρημα της βιβλιοθήκης MPICH είναι ότι αποτελεί ελεύθερο λογισμικό ανοιχτού κώδικα, συνεπώς η στρωματική αρχιτεκτονική της επιτρέπει τόσο τη μελέτη του κώδικα για κατανόηση της λειτουργίας και βελτιστοποίηση της επίδοσης όσο και την επέκταση της υλοποίησης, όπου αυτή κρίνεται αναγκαία. Στην ενότητα αυτή θα δώσουμε μια εποπτική εικόνα της δομής του κώδικα της βιβλιοθήκης MPICH (έκδοση 1.2.5.2), καθώς θεωρούμε ότι η έστω μερική κατανόηση της λειτουργίας του υφιστάμενου λογισμικού ανταλλαγής μηνυμάτων μπορεί να συντελέσει στον αποδοτικότερο παράλληλο προγραμματισμό αρχιτεκτονικών κατανεμημένης μνήμης. Αντίστοιχα προς τις απαιτήσεις της παρούσας διατριβής, θα αρκεστούμε στην αναφορά σε ρουτίνες επικοινωνίας από σημείο προς σημείο (point-to-point communication), καθώς και στη γενική αρχικοποίηση και διαχείριση του καναλιού επικοινωνίας. Αντίθετα, δεν θα επεκταθούμε σε άλλες δυνατότητες και χαρακτηριστικά του MPI, όπως οι ρουτίνες συλλογικής επικοινωνίας, οι ρουτίνες συγχρονισμού, η συλλογή στατιστικών στοιχείων κ.ά.

Η βιβλιοθήκη MPICH ακολουθεί μια ιεραρχική δομή, στο ανώτερο επίπεδο της οποίας υλοποιούνται οι λειτουργίες του προτύπου 1.2 του MPI (σχήμα Γ.1). Στην ουσία, κατά την κλήση μιας ρουτίνας επικοινωνίας του MPI ο έλεγχος μεταβιβάζεται άμεσα σε αντίστοιχες ρουτίνες του περιβάλλοντος εκτέλεσης MPIR ή τελικά της αφαιρετικής διεπαφής συσκευής ADI-2 (Abstract Device Interface) του MPID, που αντιστοιχεί στο κανάλι CH του MPICH. Τελικά, το κανάλι CH θα προχωρήσει σε κλήση



Σχήμα Γ.1: *Στρωματική σχεδίαση της επικοινωνίας από σημείο προς σημείο στο MPICH*

ρουτινών της βιβλιοθήκης P4 (Portable Programs for Parallel Processors), που σε περιβάλλοντα κατανεμημένης μοιραζόμενης μνήμης θα οδηγήσουν τελικά στη χρήση είτε της διεπαφής sockets για ανταλλαγή μηνυμάτων πάνω από το δίκτυο διασύνδεσης, είτε μοιραζόμενης μνήμης SYS V για την περίπτωση επικοινωνίας στο εσωτερικό ενός κόμβου.

Γ.1 Βασικές Δομές

Βασική δομή του MPICH είναι η δομή `MPID_Protocol`, που περιλαμβάνει τις διάφορες μεθόδους επικοινωνίας για κάθε σημασιολογικό **πρωτόκολλο** του MPICH. Όπως θα δούμε, ανάλογα με το μέγεθος του μηνύματος το MPICH διαφοροποιεί τη διαδικασία αποστολής δεδομένων σε short, eager ή rendezvous, υιοθετώντας διαφορετικές προσεγγίσεις σε ζητήματα συγχρονισμού αποστολέα-παραλήπτη και ενδιάμεσης αποθήκευσης δεδομένων. Δείκτες προς τις συναρτήσεις επικοινωνίας ορίζονται στη δομή `MPID_Protocol`, που περιγράφεται στο αρχείο mpid/ch2/dev.h:

```
typedef struct _MPID_Protocol MPID_Protocol;
struct _MPID_Protocol {
    int (*send)        (void *, int, int, int, int, int, MPID_Msgrep_t );
```

```
    int (*recv)        (MPIR_RHANDLE *, int, void *);
    int (*isend)       (void *, int, int, int, int, int,
                        MPID_Msgrep_t, MPIR_SHANDLE *);
    int (*wait_send)   (MPIR_SHANDLE *);
    int (*push_send)   (MPIR_SHANDLE *);
    int (*cancel_send) (MPIR_SHANDLE *);
    int (*irecv)       (MPIR_RHANDLE *, int, void *);
    int (*wait_recv)   (MPIR_RHANDLE *, MPI_Status *);
    int (*push_recv)   (MPIR_RHANDLE *);
    int (*cancel_recv) (MPIR_RHANDLE *);
    int (*unex)        (MPIR_RHANDLE *, int, void *);
    int (*do_ack)      (void *, int);
    void (*delete)     (MPID_Protocol *);
};
```

Για παράδειγμα, οι `send` και `isend` είναι δείκτες προς τις συναρτήσεις απλής και ασύγχρονης αποστολής, αντίστοιχα. Ομοίως, οι `recv` και `irecv` αφορούν περιπτώσεις κανονικής ή ασύγχρονης προειδοποιημένης λήψης, ενώ η `unex` αντιμετωπίζει την απρόσμενη λήψη δεδομένων.

Στο ίδιο αρχείο περιγράφεται και η δομή `MPID_Device`, που ορίζει μια **συσκευή επικοινωνίας ADI-2**:

```
typedef struct _MPID_Device MPID_Device;
struct _MPID_Device {
    int           long_len, vlong_len;
    MPID_Protocol *short_msg, *long_msg, *vlong_msg;
    MPID_Protocol *eager, *rndv;
    int           *grank_to_devlrank;
    int           (*check_device) (MPID_Device*, MPID_BLOCKING_TYPE);
    int           (*terminate)    (MPID_Device *);
    int           (*abort)        (struct MPIR_COMMUNICATOR *, int, char *);
    struct _MPID_Device *next;
};
```

Η δομή αυτή διατηρεί δείκτες στα διαφορετικά πρωτόκολλα που υποστηρίζει η συσκευή (δείκτες σε δομές τύπου `MPID_Protocol`), ενώ ιδιαίτερα σημαντική είναι η ρουτίνα `check_device`, που ορίζει το βασικό τρόπο προσπέλασης της συσκευής για έλεγχο πακέτων και θα περιγραφεί στην ενότητα Γ.6. Η

δομή MPID_DevSet

```
typedef struct {
    int         ndev;
    MPID_Device **dev;
    int         ndev_list;
    MPID_Device *dev_list;
    MPI_Request req_pending;
} MPID_DevSet;
```

ορίζει το σύνολο των εικονικών συσκευών ADI-2 του MPICH, ενώ για τον ίδιο σκοπό ορίζεται η καθολική μεταβλητή MPID_devset στο αρχείο mpid/ch2/adi2init.c. Θα πρέπει να τονιστεί πως παρά την παραμετρικότητα της αρχιτεκτονικής σχεδίασης, το MPICH είναι ουσιαστικά άρρηκτα συνδεδεμένο με το κανάλι CH, που αποτελεί πρακτικά και τη μοναδική συσκευή ADI-2 που χρησιμοποιείται.

Τέλος, στο αρχείο mpid/ch2/req.h ορίζονται οι δομές MPIR_SHANDLE και MPIR_RHANDLE, που συσχετίζονται με μία δεδομένη λειτουργία αποστολής ή λήψης, αντίστοιχα. Για παράδειγμα, η πρώτη ορίζεται ως

```
typedef struct _MPIR_SHANDLE MPIR_SHANDLE;
struct _MPIR_SHANDLE {
    MPIR_OPTYPE  handle_type;
    MPIR_COOKIE
    int          is_complete;
    int          self_index;
    int          ref_count;
    int          is_cancelled;
    int          cancel_complete;
    int          partner;
    int          errval;
    MPI_Comm     comm;
    MPI_Status   s;

    int           is_non_blocking;
    void          *start;
    int           bytes_as_contig;
    ASYNCSendId_t sid;
```

```
    MPID_RNDV_T   recv_handle;

    int (*test)   (MPIR_SHANDLE *);
    int (*push)   (MPIR_SHANDLE *);
    int (*wait)   (MPIR_SHANDLE *);
    int (*cancel) (MPIR_SHANDLE *);
    int (*finish) (MPIR_SHANDLE *);
};
```

ενώ ανάλογα ορίζεται και η δομή `MPIR_RHANDLE`.

Από τα παραπάνω θα μπορούσε κανείς να ξεχωρίσει τα πεδία `start` και `bytes_as_contig`, που διατηρούν το περιεχόμενο και το μήκος του μηνύματος, σε περίπτωση π.χ. που η ολοκλήρωση μιας λειτουργίας επικοινωνίας μετατίθεται χρονικά στο μέλλον, όπως συμβαίνει κατά την rendezvous αποστολή ενός μηνύματος. Επίσης, η `push` χρησιμοποιείται κυρίως στις απροσδόκητες λήψεις μηνυμάτων, για να περιγράψει την ολοκλήρωση της επικοινωνίας εφόσον αποσαφηνιστεί το περιεχόμενο αυτής. Αντίστοιχα, οι λειτουργίες `test` και `wait` προδιαγράφουν τον τρόπο με τον οποίο θα πρέπει να ελέγξουμε ή ακόμα και να διασφαλίσουμε την ολοκλήρωση της επικοινωνίας που σχετίζεται με κάποιο handle, ενώ η `cancel` αναφέρεται στις λειτουργίες που απαιτούνται για την ακύρωση μιας εκκρεμούς επικοινωνίας.

Γ.2 Έλεγχος Ροής

Ο έλεγχος ροής στο MPICH διασφαλίζεται σε δύο επίπεδα, ήτοι αφενός στην παρακολούθηση του πλήθους των εκκρεμών πακέτων και αφετέρου στην καταγραφή των διαθέσιμων πόρων μνήμης, ώστε οι τελευταίοι να μην εξαντληθούν κατά την ενδιάμεση αποθήκευση δεδομένων επικοινωνίας. Αναφορικά με την **παρακολούθηση των εκκρεμών πακέτων επικοινωνίας**, στο αρχείο mpid/ch2/chpackflow.h ορίζονται οι σταθερές

```
#define MPI_Pk_ackmark 25
#define MPI_Pk_hiwater 40
```

Η σταθερά `MPI_Pk_ackmark` αφορά τη διεργασία-παραλήπτη και υποδηλώνει πως για κάθε 25 πακέτα που λαμβάνει μια διεργασία B από μια διεργασία A, η B θα πρέπει να στείλει επιβεβαίωση λήψης στην A ένα ειδικό μήνυμα ελέγχου ACK. Αντίστοιχα, η σταθερά `MPI_Pk_hiwater` αφορά τη διεργασία-αποστολέα και υπονοεί πως για κάθε 40 πακέτα που αποστέλλονται από μια διεργασία A προς μια διεργασία B, η A θα πρέπει να περιμένει επιβεβαίωση λήψης από τη B πριν συνεχίσει την αποστολή πακέτων προς αυτή. Παρατηρούμε ότι ο έλεγχος ροής πακέτων αφορά ζεύγη διεργασιών, έτσι π.χ. μια διεργασία A μπορεί να έχει αποστείλει μέχρι `MPI_Pk_hiwater` ανεπιβεβαίωτα πακέτα προς μια διεργασία B

και ταυτόχρονα να συνεχίσει να στέλνει πακέτα προς μια τρίτη διεργασία Γ. Ο έλεγχος ροής πακέτων πραγματοποιείται τακτικά σε πολλά σημεία του κώδικα, π.χ. πριν την αποστολή ενός νέου πακέτου. Έτσι, το τμήμα κώδικα

```
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
```

επαναλαμβάνεται σε πολλά σημεία λήψης πακέτων και αποστέλλει επιβεβαίωση (`MPID_SendProtoAck`) εφόσον έχουν ληφθεί 25 ανεπιβεβαίωτα πακέτα (`MPID_PACKET_RCVD_GET` αληθής), ενώ αυξάνει κατά ένα το πλήθος των πακέτων που έχουν ληφθεί (`MPID_PACKET_ADD_RCVD`). Οι βασικές μακροεντολές του παραπάνω κώδικα ορίζονται στο αρχείο mpid/ch2/chpackflow.h:

```
#define MPID_PACKET_RCVD_GET(partner) \
    (MPID_pack_info.pack_rcvd[partner] + 1 == MPI_Pk_ackmark)

#define MPID_PACKET_ADD_RCVD(me, partner) \
    MPID_pack_info.pack_rcvd[partner] += 1
```

όπου η `MPID_PACKET_RCVD_GET` ελέγχει κατά πόσο με το νέο εισερχόμενο πακέτο το πλήθος των ανεπιβεβαίωτων πακέτων φτάνει το κατώφλι `MPI_Pk_ackmark`, ενώ η `MPID_PACKET_ADD_RCVD` ενημερώνει το πλήθος των ληφθέντων πακέτων. Αντίστοιχα, ο κώδικας

```
while (!MPID_PACKET_CHECK_OK(dest))
    MPID_DeviceCheck( MPID_BLOCKING );
MPID_PACKET_ADD_SENT(MPID_MyWorldRank, dest);
```

πραγματοποιεί έλεγχο στον αποστολέα για το κατώφλι `MPI_Pk_hiwater`, περιμένει επιβεβαίωση λήψης (`MPID_DeviceCheck`) εφόσον έχουν καταγραφεί 40 εκκρεμή πακέτα (`MPID_PACKET_CHECK_OK` ψευδής) και ενημερώνει το πλήθος των πακέτων που έχουν αποσταλεί (`MPID_PACKET_ADD_SENT`).

Σε δεύτερο επίπεδο, διατηρείται **έλεγχος για τις απαιτήσεις μνήμης** που σχετίζονται με την αποθήκευση δεδομένων επικοινωνίας. Ο έλεγχος αυτός αφορά μόνο στο eager πρωτόκολλο επικοινωνίας, καθώς στην περίπτωση αυτή όπως θα δούμε ο παραλήπτης δεν συγχρονίζεται με τον αποστολέα κατά την αποστολή ενός μηνύματος από τον τελευταίο, και ενδέχεται έτσι να χρειαστεί η *ενδιάμεση αποθήκευση* του μηνύματος στον παραλήπτη μέχρι να κληθεί η αντίστοιχη ρουτίνα λήψης, που θα αναλάβει να αντιγράψει τα δεδομένα στον καθοριζόμενο από το χρήστη χώρο μνήμης. Έτσι, στο αρχείο mpid/ch2/chflow.c ορίζεται το κατώφλι μνήμης `mem_thresh`

```
if (mem_thresh <= 0) mem_thresh = MPID_FLOW_BASE_THRESH;
```

ενώ στο αρχείο mpid/ch2/flow.h ορίζεται η σταθερά MPID_FLOW_BASE_THRESH

```
#define MPID_FLOW_BASE_THRESH 1048576
```

Ο έλεγχος των διαθέσιμων πόρων μνήμης γίνεται ως εξής: κατά την αποστολή ενός πακέτου με το eager πρωτόκολλο, ο αποστολέας ελέγχει αρχικά αν υπάρχει διαθέσιμη μνήμη στον παραλήπτη. Στην περίπτωση που αυτό δεν συμβαίνει, ο αποστολέας αναβάλλει την αποστολή, π.χ. ζητώντας άδεια αποστολής (πακέτο MPID_PKT_OK_TO_SEND) με επιλογή του rendezvous πρωτοκόλλου ή εκτελώντας ένα βρόχο ελέγχου του καναλιού με τη λειτουργία CheckDevice για να ενημερωθεί από τον παραλήπτη μόλις υπάρξει διαθέσιμος χώρος μνήμης. Αντίστοιχα, ο παραλήπτης ενημερώνει τους διαθέσιμους πόρους αποθήκευσης κάθε φορά που λαμβάνεται ένα (αναμενόμενο ή απροσδόκητο) eager μήνυμα, μειώνοντας τη χρησιμοποιημένη μνήμη ισόποσα προς το μέγεθος του μηνύματος που ελήφθη. Στην περίπτωση που κατά τη διαδικασία αυτή η διαθέσιμη μνήμη υπερβεί την τιμή κατωφλίου, ο παραλήπτης ενημερώνει τον αποστολέα ότι είναι έτοιμος να δεχθεί περισσότερα eager μηνύματα. Για παράδειγμα, οι μακροεντολές του αρχείου mpid/ch2/flow.h

```
#define MPID_FLOW_MEM_OK(size,partner) \
   (MPID_flow_info[partner].mem_use < MPID_flow_info[partner].mem_thresh)
#define MPID_FLOW_MEM_SEND(size,partner) \
   MPID_flow_info[partner].mem_use += size
```

σε συνδυασμό με την ακολουθία ελέγχου

```
while (!MPID_FLOW_MEM_OK(len,dest))
    MPID_DeviceCheck( MPID_BLOCKING );
MPID_FLOW_MEM_SEND(len,dest);
```

της βασικής ρουτίνας ασύγχρονης eager αποστολής MPID_CH_Eagerb_isend του καναλιού CH υλοποιούν στον αποστολέα τον έλεγχο για διαθέσιμη μνήμη στη διεργασία-παραλήπτη. Παρατηρούμε ότι και εδώ ο έλεγχος ροής αφορά ζεύγη διεργασιών που επικοινωνούν, και ουσιαστικά επιτρέπει σε μια διεργασία A να στείλει μέχρι 1 MB δεδομένα με το eager πρωτόκολλο σε μια δεύτερη διεργασία B πριν η A αναγκαστεί να περιμένει την ολοκλήρωση των αντίστοιχων λήψεων στη B.

Γ.3 Αρχικοποίηση Καναλιού CH(ameleon)

Το κανάλι CH αποτελεί τη βασική συσκευή ADI-2 του MPICH και αρχικοποιείται κατά την κλήση της ρουτίνας MPI_Init του MPI. Η ρουτίνα αυτή ορίζεται στο αρχείο src/env/init.c και μεταβιβάζει

τον έλεγχο στη ρουτίνα MPIR_Init του αρχείου src/env/initutil.c, η οποία με τη σειρά της θα καλέσει την MPID_Init του αρχείου mpid/ch2/adi2init.c. Στην τελευταία αρχικοποιείται η καθολική μεταβλητή MPID_devset, που δηλώνεται στο αρχείο mpid/ch2/adi2init.c ως

```
MPID_DevSet *MPID_devset = 0;
```

και αναφέρεται στο σύνολο των συσκευών ADI-2 που διαχειρίζεται το MPICH, ουσιαστικά δηλαδή το κανάλι CH. Έτσι, στη ρουτίνα MPID_Init γίνεται αρχικοποίηση του MPID_devset από το τμήμα κώδικα

```
MPID_Device *dev;
MPID_devset->dev_list = 0;
dev = MPID_CH_InitMsgPass( argc, argv, MPID_Short_len, -1 );
np = MPID_MyWorldSize;
MPID_devset->ndev = 1;
for (i = 0; i<np; i++)
    MPID_devset->dev[i] = dev;
MPID_devset->ndev_list = 1;
MPID_devset->dev_list = dev;
```

Αξίζει να σταθούμε στη ρουτίνα MPID_CH_InitMsgPass, καθώς εκεί γίνεται αρχικοποίηση των τριών πρωτοκόλλων επικοινωνίας (short/eager/rendezvous), του ελέγχου ροής, καθώς και άλλων βασικών λειτουργιών του καναλιού CH. Η MPID_CH_InitMsgPass ορίζεται στο αρχείο mpid/ch2/chinit.c και τα βασικότερα σημεία της έχουν ως εξής:

```
MPID_Device *dev;


if (short_len < 0) short_len = MPID_PKT_MAX_DATA_SIZE;
if (long_len < 0)  long_len  = 128000;
dev->long_len     = short_len;
dev->vlong_len    = long_len;
dev->short_msg    = MPID_CH_Short_setup();
dev->long_msg     = MPID_CH_Eagerb_setup();
dev->vlong_msg    = MPID_CH_Rndvb_setup();
dev->eager        = dev->long_msg;
dev->rndv         = dev->vlong_msg;
dev->check_device = MPID_CH_Check_incoming;
```

```
dev->terminate    = MPID_CH_End;
dev->abort        = MPID_CH_Abort;
dev->next         = 0;
PIiInit( argc, argv );

#ifdef MPID_FLOW_CONTROL
    MPID_FlowSetup( buf_thresh, mem_thresh );
#endif
#ifdef MPID_PACK_CONTROL
    MPID_PacketFlowSetup( );
#endif
return dev;
```

Η συνάρτηση επιστρέφει ένα δείκτη dev σε δομή τύπου MPID_Device. Το πεδίο long_len της δομής ορίζει το κατώφλι μεταξύ short και eager πρωτοκόλλου και εξισώνεται με την τιμή της συμβολικής σταθεράς MPID_PKT_MAX_DATA_SIZE. Για συστοιχίες υπολογιστών, στην εν λόγω σταθερά αποδίδεται τιμή ίση με 1024 bytes (αρχείο mpid/ch2/packets.h). Ομοίως, το πεδίο vlong_len της επιστρεφόμενης δομής αρχικοποιείται στην τιμή 128000 bytes και αντιστοιχεί στο κατώφλι μεταξύ eager και rendezvous πρωτοκόλλου. Τα πεδία short_msg, long_msg και vlong_msg αποτελούν δείκτες σε δομές τύπου MPID_Protocol και υλοποιούν ουσιαστικά τη σημασιολογία των τριών διαφορετικών πρωτοκόλλων επικοινωνίας του MPICH. Τα πρωτόκολλα αυτά αρχικοποιούνται μέσω των συναρτήσεων MPID_CH_Short_setup (πρωτόκολλο short, αρχείο mpid/ch2/chshort.c), MPID_CH_Eagerb_setup (πρωτόκολλο eager, αρχείο mpid/ch2/chbeager.c) και MPID_CH_Rndvb_setup (πρωτόκολλο rendezvous, αρχείο mpid/ch2/chbrndv.c). Το πεδίο check_device του dev αποτελεί δείκτη προς τη συνάρτηση MPID_CH_Check_incoming, που χρησιμοποιείται για τη βασική λειτουργία ελέγχου κίνησης στο κανάλι CH. Τέλος, οι συναρτήσεις MPID_FlowSetup και MPID_PacketFlowSetup αναλαμβάνουν την αρχικοποίηση του ελέγχου ροής, όπως αυτός περιγράφηκε στην ενότητα Γ.2.

Γ.4 Ασύγχρονη Αποστολή

Η βασική ρουτίνα ασύγχρονης αποστολής του MPI είναι η MPI_Isend, που άλλωστε χρησιμοποιήσαμε και στους αλγοριθμικούς συμβολισμούς της παρούσας διατριβής. Σημασιολογικά, η έξοδος από την MPI_Isend δεν διασφαλίζει την ολοκλήρωση της διαδικασίας αποστολής, συνεπώς ο σχετικός απομονωτής δεδομένων δεν μπορεί να επαναχρησιμοποιηθεί πριν διασφαλιστεί η επιτυχής περάτωση της αποστολής με κάποια κλήση MPI_Test ή MPI_Wait. Η MPI_Isend ορίζεται στο αρχείο src/pt2pt/isend.c,

και αφού πραγματοποιήσει απλές μετατροπές τύπων και αρχικοποιήσει ένα handle τύπου `MPI_Request` για την αποστολή μεταβιβάζει τον έλεγχο στη συνάρτηση `MPID_IsendDatatype`, που ορίζεται στο αρχείο mpid/ch2/adi2hsend.c. Η `MPID_IsendDatatype` υπολογίζει από το χρησιμοποιούμενο τύπο δεδομένων (`MPI_Datatype`) και το πλήθος των στοιχείων του απομονωτή το συνολικό αριθμό bytes που θα σταλούν και καλεί τη συνάρτηση `MPID_IsendContig` του αρχείου mpid/ch2/adi2send.c. Στην τελευταία γίνεται επιλογή πρωτοκόλλου επικοινωνίας με τη βοήθεια του κώδικα

```
request->shandle.finish = 0;
if (len < dev->long_len)
    fcn = dev->short_msg->isend;
else if (len < dev->vlong_len && MPID_FLOW_MEM_OK(len,dest_grank))
    fcn = dev->long_msg->isend;
else
    fcn = dev->vlong_msg->isend;
*error_code = (*(fcn))( buf, len, src_lrank, tag, context_id, dest_grank,
                        msgrep, (MPIR_SHANDLE *)request );
```

Στον παραπάνω κώδικα, `len` είναι το μήκος του μηνύματος προς αποστολή, που μεταβιβάζεται ως παράμετρος στην `MPID_IsendContig`. Αν το μέγεθος αυτό είναι μικρότερο από 1024 bytes καλείται η σχετική συνάρτηση `dev->short_msg->isend` για short αποστολή. Αν το μήνυμα προς αποστολή έχει μήκος μεταξύ 1024 και 128000 bytes και επιπλέον υπάρχει διαθέσιμος χώρος μνήμης για ενδεχόμενη ενδιάμεση αποθήκευση δεδομένων στον παραλήπτη (μακροεντολή `MPID_FLOW_MEM_OK` αληθής) καλείται η `dev->long_msg->isend` για eager αποστολή. Τέλος, αν το μήνυμα υπερβαίνει τα 128000 bytes έχουμε rendezvous αποστολή (`dev->vlong_msg->isend`).

Τα πεδία `short_msg->isend`, `long_msg->isend` και `vlong_msg->isend` της συσκευής ADI-2 δεικτοδοτούν τις βασικές συναρτήσεις ασύγχρονης αποστολής του MPICH. Κάθε μία από τις συναρτήσεις του επιπέδου MPID ονοματίζεται στο MPICH σύμφωνα με τη γενική μορφή

```
MPID_<ΣΥΣΚΕΥΗ>_<ΠΡΩΤΟΚΟΛΛΟ><Β ΓΙΑ ΣΥΓΧΡΟΝΗ ΛΕΙΤΟΥΡΓΙΑ>_<ΣΥΝΑΡΤΗΣΗ>
```

που φανερώνει τη συσκευή επικοινωνίας, το πρωτόκολλο που ακολουθείται, το αν η συνάρτηση αυτή είναι σύγχρονη ή ασύγχρονη, καθώς και το περιγραφικό όνομα της ρουτίνας. Για παράδειγμα, η συνάρτηση `MPID_CH_Rndvb_send` αφορά σύγχρονη απλή αποστολή με χρήση του καναλιού CH και του rendezvous πρωτοκόλλου.

Γ.4.1 Short Αποστολή

Στην περίπτωση **short αποστολής** καλείται η συνάρτηση `MPID_CH_Eagerb_isend_short`, στην οποία δείχνει το πεδίο `short_msg->isend` της συσκευής ADI-2. Τα βασικά σημεία της συνάρτησης έχουν ως εξής (αρχείο mpid/ch2/chshort.c):

```
MPID_PKT_SHORT_T pkt;
pkt_len        = sizeof(MPID_PKT_SHORT_T) - MPID_PKT_MAX_DATA_SIZE;
pkt.mode       = MPID_PKT_SHORT;
pkt.context_id = context_id;
pkt.lrank      = src_lrank;
pkt.to         = dest;
pkt.seqnum     = len + pkt_len;
pkt.src        = MPID_MyWorldRank;
pkt.tag        = tag;
pkt.len        = len;
shandle->is_complete = 1;
MEMCPY( pkt.buffer, buf, len );
MPID_SendControlBlock( &pkt, len + pkt_len, dest );
return MPI_SUCCESS;
```

Ουσιαστικά, δημιουργείται ένα μήνυμα-επικεφαλίδα `pkt` τύπου `MPID_PKT_SHORT_T` μήκους `pkt_len`. Το μήνυμα του χρήστη αντιγράφεται από τον απομονωτή `buf` της `MPI_Isend` στον αντίστοιχο απομονωτή `pkt.buffer` του `pkt`, και τελικά το συνολικό μήνυμα μήκους `len+pkt_len` αποστέλλεται στον παραλήπτη μέσω της μακροεντολής `MPID_SendControlBlock`, που όπως θα δούμε χρησιμεύει για την αποστολή μηνυμάτων ελέγχου.

Γ.4.2 Eager Αποστολή

Κατά την **eager αποστολή** πραγματοποιείται κλήση της ρουτίνας `MPID_CH_Eagerb_isend` του αρχείου mpid/ch2/chbeager.c, αφού σε αυτή δείχνει το πεδίο `long_msg->isend` της ADI-2 συσκευής:

```
MPID_PKT_LONG_T  pkt;
while (!MPID_FLOW_MEM_OK(len,dest))
    MPID_DeviceCheck( MPID_BLOCKING );
MPID_FLOW_MEM_SEND(len,dest);
while (!MPID_PACKET_CHECK_OK(dest))
```

```
    MPID_DeviceCheck( MPID_BLOCKING );
MPID_PACKET_ADD_SENT(MPID_MyWorldRank, dest);
pkt.mode       = MPID_PKT_LONG;
pkt_len        = sizeof(MPID_PKT_LONG_T);
pkt.context_id = context_id;
pkt.lrank      = src_lrank;
pkt.to         = dest;
pkt.seqnum     = pkt_len + len;
pkt.src        = MPID_MyWorldRank;
pkt.tag        = tag;
pkt.len        = len;
MPID_SendControlBlock( &pkt, pkt_len, dest );
MPID_SendChannel( buf, len, dest );
shandle->is_complete = 1;
if (shandle->finish)
    (shandle->finish)( shandle );
return MPI_SUCCESS;
```

Αρχικά πραγματοποιείται έλεγχος ροής, τόσο για την ύπαρξη διαθέσιμου χώρου μνήμης στη διεργασία παραλήπτη (μακροεντολή `MPID_FLOW_MEM_OK` για έλεγχο, μακροεντολή `MPID_FLOW_MEM_SEND` για ενημέρωση) όσο και για το αν με το παρόν συμπληρώνονται 40 ανεπιβεβαίωτα απεσταλμένα πακέτα, οπότε και ο αποστολέας θα πρέπει να περιμένει σχετική επιβεβαίωση λήψης (`MPID_PACKET_CHECK_OK` για έλεγχο του κατωφλίου `MPI_Pk_hiwater`, `MPID_PACKET_ADD_SENT` για ενημέρωση του πλήθους των πακέτων που έχουν αποσταλεί). Στη συνέχεια δημιουργείται ένα πακέτο-επικεφαλίδα `pkt` τύπου `MPID_PKT_LONG_T`, στο οποίο καταγράφονται βασικές πληροφορίες που σχετίζονται με το προς αποστολή μήνυμα (π.χ. βαθμός της διεργασίας αποστολέα, ετικέτα και μήκος του μηνύματος κ.ο.κ.). Η επικεφαλίδα `pkt` αποστέλλεται στον παραλήπτη μέσω της μακροεντολής `MPID_SendControlBlock`, ενώ ακολουθεί αποστολή των δεδομένων του μηνύματος που περιέχονται στον απομονωτή `buf` της `MPI_Isend` μέσω της μακροεντολής `MPID_SendChannel`. Η διαδικασία της ασύγχρονης αποστολής θεωρείται ότι ολοκληρώθηκε επιτυχώς (`shandle->is_complete = 1`), και κατά την έξοδο από την `MPID_CH_Eagerb_isend` επιστρέφεται η τιμή επιτυχίας `MPI_SUCCESS`.

Γ.4.3 Rendezvous Αποστολή

Στην περίπτωση της **rendezvous αποστολής** πραγματοποιείται κλήση της ρουτίνας που δεικτοδοτείται από το πεδίο `vlong_msg->isend` της συσκευής ADI-2. Η ρουτίνα αυτή είναι η `MPID_CH_Rndvb_isend`,

όπως καθορίζεται από τη συνάρτηση MPID_CH_Rndvb_setup κατά την αρχικοποίηση του καναλιού CH μέσω της συνάρτησης MPID_CH_InitMsgPass. Η MPID_CH_Rndvb_isend ορίζεται στο αρχείο mpid/ch2/chbrndv.c:

```
MPID_PKT_REQUEST_SEND_T  pkt;
while (!MPID_PACKET_CHECK_OK(dest))
    MPID_DeviceCheck( MPID_BLOCKING );
MPID_PACKET_ADD_SENT(MPID_MyWorldRank, dest );
pkt.mode       = MPID_PKT_REQUEST_SEND;
pkt.context_id = context_id;
pkt.lrank      = src_lrank;
pkt.to         = dest;
pkt.src        = MPID_MyWorldRank;
pkt.seqnum     = sizeof(MPID_PKT_REQUEST_SEND_T);
pkt.tag        = tag;
pkt.len        = len;
shandle->is_complete     = 0;
shandle->start           = buf;
shandle->bytes_as_contig = len;
shandle->wait            = MPID_WaitForCompleteSend;
shandle->test            = 0;
shandle->finish          = 0;
shandle->partner         = dest;
MPID_n_pending++;
MPID_SendControlBlock( &pkt, sizeof(pkt), dest );
return MPI_SUCCESS;
```

Η συνάρτηση αυτή διαφέρει από την eager περίπτωση στο ότι αφενός δεν πραγματοποιεί έλεγχο διαθέσιμης μνήμης στον παραλήπτη (γίνεται μόνο έλεγχος του κατωφλίου MPI_Pk_hiwater για εκκρεμή πακέτα), ενώ αφετέρου αποστέλλει μόνο το μήνυμα-επικεφαλίδα pkt τύπου MPID_PKT_REQUEST_SEND_T με χρήση της MPID_SendControlBlock. Επίσης, ορίζεται και μια συνάρτηση wait στο handle αποστολής του μηνύματος (MPID_WaitForCompleteSend), η οποία θα κληθεί μελλοντικά για ολοκλήρωση της διαδικασίας αποστολής. Οι παραπάνω διαφορές αποδίδονται στο ότι κατά το rendezvous πρωτόκολλο πραγματοποιείται *χειραψία* (*handshake*) μεταξύ του αποστολέα και του παραλήπτη, για να αποφευχθεί η επιβάρυνση λόγω ενδεχόμενης ενδιάμεσης αποθήκευσης του μεγάλου μηνύματος που αποστέλλεται. Έτσι, αρχικά ο αποστολέας ζητά άδεια από τον παραλήπτη για τη μετάδοση του μηνύμα-

τος. Ο τελευταίος μπορεί να επιτρέψει την αποστολή όταν πληροφορηθεί τον απομονωτή προορισμού του μηνύματος π.χ. από μια συνάρτηση MPI_Recv ή MPI_Irecv, ώστε να μη χρειαστούν ενδιάμεσες αντιγραφές των δεδομένων επικοινωνίας. Τελικά, αφού λάβει τη σχετική άδεια, ο αποστολέας θα καλέσει τη ρουτίνα που δεικτοδοτείται από το πεδίο wait του handle αποστολής για την ολοκλήρωση της διαδικασίας αποστολής με τη μετάδοση των πραγματικών δεδομένων.

Γ.4.4 P4 Βιβλιοθήκη Επικοινωνίας

Όλες οι παραπάνω περιπτώσεις βλέπουμε πως καταλήγουν σε αποστολή μηνυμάτων-επικεφαλίδας μέσω της MPID_SendControlBlock ή/και αποστολή των δεδομένων επικοινωνίας μέσω της κλήσης της MPID_SendChannel. Αμφότερες μακροεντολές ορίζονται στο αρχείο mpid/ch2/channel.h και οδηγούν σε κλήση της συνάρτησης PIbsend, με διαφορετική εσωτερική ετικέτα αποστολής MPID_PT2PT_TAG. Συγκεκριμένα, τα μηνύματα ελέγχου χρησιμοποιούν την ετικέτα αποστολής 0, ενώ τα μηνύματα δεδομένων έχουν πάντα μη μηδενική ετικέτα αποστολής (αρχείο mpid/ch2/packets.h):

```
/* μηνύματα ελέγχου */
#define MPID_PT2PT_TAG 0
/* μηνύματα δεδομένων */
#define MPID_PT2PT2_TAG(src) (1+(src))
```

Η ρουτίνα PIbsend ορίζεται στη βιβλιοθήκη P4, που χρησιμοποιεί το MPICH για την υλοποίηση της επικοινωνίας σε χαμηλό επίπεδο. Από τη συνάρτηση PIbsend (αρχείο mpid/ch_p4/chdef.h) καλείται η p4_sendx (αρχείο mpid/ch_p4/p4/lib/p4_sr.h) και τελικά η send_message, που ορίζεται στο αρχείο mpid/ch_p4/p4/lib/p4_tsr.c:

```
conntype = p4_local->conntab[to].type;
switch (conntype){
    case CONN_SHMEM:
        tmsg = get_tmsg(type, from, to, msg, len, data_type,
                        ack_req, p4_buff_ind);
        shmem_send(tmsg);
        break;
    case CONN_REMOTE_EST:
        if (data_type == P4NOX || p4_local->conntab[to].same_data_rep)
            socket_send(type, from, to, msg, len, data_type, ack_req);
        else
            xdr_send(type, from, to, msg, len, data_type, ack_req);
```

```
        break;
```

Παρατηρούμε ότι αναλόγως με το πώς έχει οριστεί η σύνδεση με τη διεργασία παραλήπτη `to` προσδιορίζεται ο τύπος σύνδεσης `conntype`, και είτε αξιοποιείται η SYS V μοιραζόμενη μνήμη μέσω της `shmem_send`, είτε χρησιμοποιείται επικοινωνία μέσω sockets για απομακρυσμένους κόμβους (συγκεκριμένα, `socket_send` στην περίπτωση κοινής αναπαράστασης δεδομένων σε αποστολέα και παραλήπτη, `xdr_send` σε περίπτωση που χρειάζονται μετατροπές π.χ. για κόμβους διαφορετικής αρχιτεκτονικής). Παρατηρούμε ότι ακόμα και στην περίπτωση που επιτυγχάνεται η χρήση μοιραζόμενης μνήμης με κατάλληλη περιγραφή των κόμβων στο αρχείο `machinefile` κατά την εκτέλεση του προγράμματος, δεν αποφεύγεται η αντιγραφή από τον απομονωτή του χρήστη (`msg`) σε ειδικό χώρο μνήμης που έχει δεσμευθεί μέσω shmem (`tmsg`). Η παρατήρηση αυτή μπορεί να ερμηνεύσει τη διαφορά στην επίδοση κατά τη χρήση του συνδυασμού MPICH+shmem σε σχέση με αμιγώς πολυνηματικό προγραμματισμό (π.χ. OpenMP), όπως άλλωστε καταδείχθηκε και εποπτικά στο σχήμα 2.7(β) του κεφαλαίου 2.

Συνάρτηση	***short_msg->isend***	***long_msg->isend***	***vlong_msg->isend***
Πρωτόκολλο	short	eager	rendezvous
Επικεφαλίδα	`MPID_PKT_SHORT`	`MPID_PKT_LONG`	`MPID_PKT_REQUEST_SEND`
`shandle->is_complete`	1	1	0
Μέγεθος μηνυμάτων	`len` < 1024	$1024 \leq$ `len` < 128000	`len` ≥ 128000
Πλήθος μηνυμάτων	1 με `ControlBlock`	1 με `ControlBlock` + 1 με `Channel`	2 με `ControlBlock` + 1 με `Channel`
Αντιγραφές (αποστολέας)	1	0	1
Αντιγραφές (παραλήπτης)	2-3	2-3	1
Έλεγχος ροής (πακέτα)	Όχι	Ναι	Ναι
Έλεγχος ροής (μνήμη)	Όχι	Ναι	Όχι
Συγχρονισμός	Όχι	Όχι	Ναι

Πίνακας Γ.1: *Σύνοψη διαφορών σε ρουτίνες ασύγχρονης αποστολής για τα τρία πρωτόκολλα της συσκευής CH του MPICH*

Για πληρότητα, ο πίνακας Γ.1 συνοψίζει τις κυριότερες διαφορές μεταξύ των τριών πρωτοκόλλων στην περίπτωση της ασύγχρονης αποστολής δεδομένων. Θα πρέπει να τονιστεί ότι ως αντιγραφές καταμετρώνται όλες οι ενδιάμεσες αποθηκεύσεις των δεδομένων επικοινωνίας μέχρι τις πρωτογενείς κλήσεις του λειτουργικού συστήματος, π.χ. των `read` και `write` για την περίπτωση των sockets. Ουσιαστικά δηλαδή, μας ενδιαφέρουν μόνο οι αντιγραφές που αφορούν στην υλοποίηση MPICH, και όχι εκείνες που σχετίζονται με το λειτουργικό σύστημα ή το δικτυακό πρωτόκολλο.

Γ.5 Ολοκλήρωση Αποστολής

Η ολοκλήρωση μιας λειτουργίας ασύγχρονης αποστολής MPI_Isend πραγματοποιείται με κλήση της ρουτίνας MPI_Wait. Η MPI_Wait διασφαλίζει τη δυνατότητα επαναχρησιμοποίησης του απομονωτή επικοινωνίας που χρησιμοποιήθηκε στην αντίστοιχη MPI_Isend, δηλαδή βεβαιώνει ότι όλα τα δεδομένα επικοινωνίας θα έχουν μεταφερθεί από τον απομονωτή χρήστη σε ειδικό χώρο μνήμης για αποστολή προς τη διεργασία παραλήπτη. Η συνάρτηση MPI_Wait ορίζεται στο αρχείο src/pt2pt/wait.c και επιστρέφει άμεσα κλήση στην MPI_Waitall. Η επιλογή αυτή ενδεχομένως επιβαρύνει με μια αχρείαστη επιπλέον κλήση συνάρτησης, αλλά έχει προφανώς γίνει για λόγους απλότητας και δομής του κώδικα. Η MPI_Waitall ορίζεται στο αρχείο src/pt2pt/waitall.c για την περίπτωση της ασύγχρονης αποστολής ως εξής:

```
for (i = 0; i < count; i++){
    request = array_of_requests[i];
    if (!request)
        continue;
    if ( request->handle_type == MPIR_SEND ) {
        if (MPID_SendRequestCancelled(request)) {
            if (array_of_statuses) {
                array_of_statuses[i].MPI_TAG   = MPIR_MSG_CANCELLED;
                array_of_statuses[i].MPI_ERROR = MPI_SUCCESS;
            }
        }
    else {
        rc = MPI_SUCCESS;
        MPID_SendComplete( request, &rc );
        if (rc) {
            if (array_of_statuses)
                MPIR_Set_Status_error_array( array_of_requests, count,
                                             i, rc, array_of_statuses );
            mpi_errno = MPI_ERR_IN_STATUS;
            MPIR_RETURN(MPIR_COMM_WORLD, mpi_errno, myname );
        }
        MPID_SendFree( array_of_requests[i] );
        array_of_requests[i]    = 0;
```

```
    }
}
```

Ο πίνακας `array_of_requests` περιέχει τα request handles όλων των εκκρεμών λειτουργιών επικοινωνίας που θέλουμε να ολοκληρώσουμε και περνιέται ως παράμετρος στη συνάρτηση `MPI_Waitall`. Ο κώδικας είναι σχετικά απλός, καθώς διασχίζει όλα τα request handles και εξετάζει τον τύπο της επικοινωνίας στην οποία αναφέρονται. Αν έχουμε εκκρεμή ασύγχρονη αποστολή (`request->handle_type == MPIR_SEND`) εξετάζεται αρχικά το ενδεχόμενο να έχει προηγηθεί ακύρωση της εν λόγω επικοινωνίας, και ενημερώνεται κατάλληλα η αντίστοιχη θέση στον πίνακα `array_of_statuses`. Αν αυτό δεν συμβαίνει, καλείται η ρουτίνα `MPID_SendComplete` για την ολοκλήρωση της επικοινωνίας που σχετίζεται με το τρέχον request. Στη συνέχεια εξετάζεται αν επήλθε επιτυχής τερματισμός ή αντίθετα εμφανίστηκε κάποιο σφάλμα, και σε όλες τις περιπτώσεις ενημερώνεται κατάλληλα η αντίστοιχη θέση στον πίνακα `array_of_statuses`. Επιπλέον, στην περίπτωση επιτυχούς τερματισμού της επικοινωνίας απελευθερώνονται οι πόροι που είχαν δεσμευθεί και σχετίζονται με το υπό εξέταση request handle.

Η συνάρτηση `MPID_SendComplete` ορίζεται στο αρχείο mpid/ch2/adi2send.c ως εξής:

```
MPIR_SHANDLE *shandle = &request->shandle;
while (!shandle->is_complete) {
    if (shandle->wait)
        *error_code = (*shandle->wait)( shandle );
    else {
        MPID_Device *dev;
        if (MPID_devset->ndev_list == 1) {
            dev = MPID_devset->dev_list;
            if (!shandle->is_complete)
                (*dev->check_device)( dev, MPID_BLOCKING );
        }
        else {
            ...
        }
    }
}
```

Παρατηρούμε ότι το πεδίο `shandle->is_complete` θα είναι μηδενικό μόνο στην περίπτωση του rendezvous πρωτοκόλλου, συνεπώς για τις περιπτώσεις short και eager αποστολής δεν πραγματοποιείται είσοδος στο βρόχο `while`. Αντίθετα, κατά το rendezvous πρωτόκολλο καλείται η `(*shandle->wait)`,

που όπως είδαμε έχει αντιστοιχιστεί στη συνάρτηση MPID_WaitForCompleteSend του πηγαίου αρχείου mpid/ch2/adi2init.c και επιτελεί κατά βάση την ακόλουθη λειτουργία:

```
while (!request->is_complete)
    MPID_DeviceCheck( MPID_BLOCKING );
```

Έτσι, μέχρι να σημειωθεί το πεδίο shandle->is_complete ως αληθές, καλείται σε έναν ατέρμονα βρόχο η MPID_DeviceCheck για έλεγχο εισερχόμενων πακέτων στο κανάλι CH, ώστε να ληφθεί το επιθυμητό μήνυμα ελέγχου MPID_PKT_OK_TO_SEND από τη διεργασία παραλήπτη και να αποσταλούν τα δεδομένα επικοινωνίας. Το εσωτερικό else της MPID_SendComplete αναφέρεται στην περίπτωση διαχείρισης πολλαπλών συσκευών ADI-2 και έχει συνεπώς μόνο θεωρητική αξία για την παρούσα υλοποίηση MPICH, καθώς η τελευταία βασίζεται αποκλειστικά στη μοναδική συσκευή CH.

Γ.6 Λήψη

Το MPICH διατηρεί δύο ουρές για τη λήψη μηνυμάτων και την αποθήκευση των σχετικών handles, την ουρά για τα *αναμενόμενα εισερχόμενα μηνύματα* (*posted receives queue*) καθώς και την ουρά με τα *απροσδόκητα εισερχόμενα μηνύματα* (*unexpected receives queue*). Οι δύο ουρές αποτελούν πεδία της καθολικής μεταβλητής MPID_recvs, με το πεδίο MPID_recvs.posted να αντιστοιχεί στην ουρά των αναμενόμενων μηνυμάτων και το πεδίο MPID_recvs.unexpected να προδιαγράφει την ουρά των απροσδόκητων μηνυμάτων. Η αναγκαιότητα τήρησης δύο διαφορετικών ουρών για τα ληφθέντα μηνύματα έγκειται στο ότι ένα μήνυμα μπορεί να ληφθεί από τη συσκευή ADI-2 πριν ή αφού κληθεί η αντίστοιχη συνάρτηση λήψης του MPI. Στην πρώτη περίπτωση το μήνυμα θα τοποθετηθεί στην ουρά απροσδόκητων μηνυμάτων, ενώ στη δεύτερη θα συνδυαστεί με κάποιο handle της ουράς αναμενόμενων μηνυμάτων.

Αναλυτικότερα, όταν λαμβάνεται κάποιο πακέτο από μία διεργασία, αυτή εξετάζει πρώτα την ουρά MPID_recvs.posted αναζητώντας ένα σχετικό handle λήψης, που περιγράφει τον απομονωτή προορισμού του πακέτου και έχει δημιουργηθεί νωρίτερα από την εκτέλεση αντίστοιχης MPI ρουτίνας λήψης (π.χ. MPI_Recv ή MPI_Irecv). Στην περίπτωση που βρεθεί κάποιο τέτοιο handle, εφόσον δηλαδή έχει προηγηθεί η κλήση ρουτίνας λήψης που να καθορίζει τον απομονωτή προορισμού του πακέτου, αντιγράφεται το μήνυμα στον απομονωτή αυτό, η διεύθυνση του οποίου φυλάσσεται στο πεδίο buf του handle λήψης. Αν αντίθετα δεν βρεθεί ένα handle στην ουρά MPID_recvs.posted, τότε δημιουργείται ένα νέο handle στην ουρά MPID_recvs.unexpected και αποθηκεύονται διάφορες πληροφορίες για το ληφθέν μήνυμα. Έτσι, όταν κληθεί κάποια ρουτίνα λήψης του MPI εξετάζεται αρχικά η ουρά MPID_recvs.unexpected για να διευκρινιστεί αν έχει ήδη ληφθεί το αντίστοιχο μήνυμα από τη συσκευή ADI-2. Αν αυτό συμβαίνει, τότε το handle του μηνύματος αφαιρείται από την ουρά απροσδό-

κητων μηνυμάτων και γίνεται επεξεργασία του μηνύματος. Αν αντίθετα δεν βρεθεί καταχώρηση στην ουρά MPID_recvs.unexpected, τότε δημιουργείται μια περιγραφή του μηνύματος σε ένα νέο handle που αποθηκεύεται στην ουρά MPID_recvs.posted.

Το λεπτό σημείο στην κατανόηση της λειτουργίας λήψης του MPICH είναι ο σαφής διαχωρισμός μεταξύ της *εκτέλεσης μιας ρουτίνας λήψης* κατά τη ροή του προγράμματος MPI και της *πραγματικής λήψης ενός πακέτου* από τη συσκευή ADI-2. Στην πρώτη περίπτωση συναντάται κατά την εκτέλεση του προγράμματος μια MPI συνάρτηση λήψης, π.χ. MPI_Recv ή MPI_Irecv, και ενημερώνεται το κανάλι CH με την περιγραφή μιας συγκεκριμένης διαδικασίας επικοινωνίας, π.χ. για το βαθμό της διεργασίας αποστολέα από την οποία περιμένουμε το μήνυμα, καθώς και για την ετικέτα, το μέγεθος και τον απομονωτή προορισμού του μηνύματος. Το ίδιο το μήνυμα ενδέχεται να έχει ήδη ληφθεί από τη συσκευή ADI-2 ή να ληφθεί μελλοντικά, όπως επίσης και ανάλογα με τη σημασιολογία της συνάρτησης λήψης που επιλέγουμε ενδέχεται είτε να αναστείλουμε την εκτέλεση περιμένοντας για τη λήψη του μηνύματος (MPI_Recv) είτε να συνεχίσουμε τη ροή του προγράμματος και να ελέγξουμε τη λήψη σε κάποια μελλοντική στιγμή (MPI_Irecv). Αυτή καθεαυτή η λήψη πακέτων από τη συσκευή ADI-2 συντονίζεται από τη ρουτίνα MPID_DeviceCheck, που καθότι υλοποιεί τη λήψη σε χαμηλό επίπεδο δεν αποτελεί μέρος του προτύπου MPI, αλλά του στρώματος MPID.

Γ.6.1 MPI_Irecv

Η συνάρτηση ασύγχρονης λήψης MPI_Irecv ορίζεται στο αρχείο src/pt2pt/irecv.c, όπου αφού αρχικοποιήσει μια καινούρια αίτηση λήψης (request) και κάνει ορισμένες μετατροπές τύπων καλεί σχετικά άμεσα την MPID_IrecvDatatype του αρχείου mpid/ch2/adi2hrecv.c. Η τελευταία υπολογίζει σε bytes το μέγεθος του μηνύματος που επιθυμούμε να λάβουμε, αξιοποιώντας την πληροφορία του τύπου και του πλήθους των δεδομένων, και καλεί την MPID_IrecvContig του αρχείου mpid/ch2/adi2recv.c. Συνοπτικά, η ρουτίνα MPID_IrecvContig έχει ως εξής:

```
MPIR_RHANDLE *dmpi_unexpected, *rhandle = &request->rhandle;
rhandle->len         = maxlen;
rhandle->buf         = buf;
rhandle->is_complete = 0;
rhandle->wait        = 0;
rhandle->test        = 0;
rhandle->finish      = 0;
MPID_Search_unexpected_queue_and_post( src_lrank, tag, context_id,
                                       rhandle, &dmpi_unexpected );
if (dmpi_unexpected)
```

```
    *error_code = (*dmpi_unexpected->push)( rhandle, dmpi_unexpected );
```

ενώ η καλούμενη MPID_Search_unexpected_queue_and_post ορίζεται στο αρχείο πηγαίου κώδικα mpid/util/queue.c και συνοπτικά υλοποιείται ως ακολούθως:

```
MPID_THREAD_DS_LOCK(&MPID_recvs)
MPID_Search_unexpected_queue( src_lrank, tag, context_id, 1, rhandleptr );
if (*rhandleptr) {
    MPID_THREAD_DS_UNLOCK(&MPID_recvs)
    return;
}
MPID_Enqueue( &MPID_recvs.posted, src_lrank, tag, context_id, request );
MPID_THREAD_DS_UNLOCK(&MPID_recvs)
```

Από τις παραπάνω συναρτήσεις συνάγονται τα εξής για τη λειτουργία της MPID_IrecvContig: καταρχάς αρχικοποιούνται βασικά πεδία του handle λήψης (rhandle). Στη συνέχεια, με βάση τα στοιχεία του μηνύματος (αποστολέας, ετικέτα, μέγεθος) γίνεται αναζήτηση στη σειρά των μη αναμενόμενων μηνυμάτων MPID_recvs.unexpected μέσω της ρουτίνας MPID_Search_unexpected_queue για να διαπιστωθεί αν το μήνυμα έχει ήδη ληφθεί από τη συσκευή ADI-2. Αν αυτό συμβαίνει, η συνάρτηση MPID_Search_unexpected_queue επιστρέφει ένα δείκτη dmpi_unexpected προς το ζητούμενο handle. Σε αντίθετη περίπτωση, αν δηλαδή το μήνυμα δεν έχει ληφθεί, τότε το handle λήψης εφοδιάζεται με τα στοιχεία του μηνύματος και τοποθετείται στην ουρά των αναμενόμενων μηνυμάτων MPID_recvs.posted με χρήση της ρουτίνας MPID_Enqueue. Παρατηρούμε ότι όλες οι λειτουργίες επεξεργασίας των ουρών αναμενόμενων και απροσδόκητων μηνυμάτων περικλείονται σε λειτουργίες κλειδώματος (MPID_THREAD_DS_LOCK-MPID_THREAD_DS_UNLOCK, ουσιαστικά λειτουργίες mutex της βιβλιοθήκης Pthreads), για να μην προκύψει λανθασμένη διπλή επεξεργασία του ίδιου μηνύματος σε πολυνηματικό περιβάλλον. Παρόλα αυτά, η υλοποίηση MPICH δεν παρέχει πλήρη πολυνηματική υποστήριξη, καθιστώντας τήν ταυτόχρονη εκτέλεση ρουτινών επικοινωνίας MPI απαγορευτική.

Τέλος, παρατηρούμε ότι στην περίπτωση που βρέθηκε σχετικό με το μήνυμα handle στην ουρά απροσδόκητων μηνυμάτων και άρα επεστράφη μη μηδενική τιμή στο δείκτη dmpi_unexpected, καλείται η συνάρτηση που δεικτοδοτείται από το πεδίο push του handle λήψης. Η κλήση της τελευταίας μας επιτρέπει να διαχειριστούμε σωστά την περίπτωση που η πραγματική λήψη από τη συσκευή ADI-2 έχει προηγηθεί χρονικά της κλήσης της MPI_Irecv, όπως θα δούμε και στις ακόλουθες ενότητες.

Γ.6.2 MPID_DeviceCheck

Η συνάρτηση MPID_DeviceCheck του αρχείου mpid/ch2/adi2init.c αποτελεί τη βασική ρουτίνα ελέγχου εισερχομένων μηνυμάτων της συσκευής ADI-2. Η ρουτίνα αυτή καλείται οποτεδήποτε θέλουμε να εξετάσουμε την άφιξη νέου μηνύματος, είτε πρόκειται για μήνυμα ελέγχου, είτε για μήνυμα δεδομένων. Η ρουτίνα καλεί τελικά την MPID_CH_Check_incoming, που ορίζεται στο πηγαίο αρχείο mpid/ch2/chchkdev.c και ξεκινάει ως εξής:

```
if (is_blocking == MPID_NOTBLOCKING)
    if (!MPID_PKT_CHECK()) return -1;
MPID_PKT_WAIT();
MPID_PKT_UNPACK( &pkt, sizeof(MPID_PKT_HEAD_T), from_grank );
```

Αρχικά λαμβάνεται ένα εισερχόμενο πακέτο ελέγχου στη συσκευή ADI-2, είτε με σύγχρονο (μακροεντολή MPID_PKT_WAIT) είτε με ασύγχρονο τρόπο (MPID_PKT_CHECK+MPID_PKT_WAIT). Η μακροεντολή MPID_PKT_WAIT μεταβιβάζει τελικά τον έλεγχο στη συνάρτηση PIbrecv (mpid/ch_p4/chdef.h), που αποτελεί τη βασική συνάρτηση λήψης πακέτου και καλεί τη ρουτίνα p4_recv της P4 βιβλιοθήκης. Μάλιστα, η συγκεκριμένη ακολουθία συναρτήσεων περιλαμβάνει και μία επιπλέον ενδιάμεση αντιγραφή των δεδομένων που λαμβάνονται, καθώς η p4_recv χρησιμοποιεί δικό της εσωτερικό απομονωτή ενώ η PIbrecv επιστρέφει τα δεδομένα στον απομονωτή pkt. Η μακροεντολή MPID_PKT_UNPACK αποσκοπεί στην εξαγωγή της επικεφαλίδας του πακέτου, η οποία αποθηκεύεται στη μεταβλητή pkt. Στη συνέχεια, ακολουθεί το εξής τμήμα κώδικα:

```
if (MPID_PKT_IS_MSG(pkt.head.mode)) {
    MPID_Msg_arrived( pkt.head.lrank, pkt.head.tag, pkt.head.context_id,
                      &rhandle, &is_posted );
    if (!is_posted) {
        if (pkt.head.mode == MPID_PKT_REQUEST_SEND)
            rhandle->send_id = pkt.request_pkt.send_id;
        else if (pkt.head.mode == MPID_PKT_SHORT)
            rhandle->send_id = pkt.short_pkt.send_id;
        else if (pkt.head.mode == MPID_PKT_LONG)
            rhandle->send_id = pkt.long_pkt.send_id;
    }
    if (is_posted) {
        switch (pkt.head.mode) {
            case MPID_PKT_SHORT:
```

```
                err = (*dev->short_msg->recv)( rhandle, from_grank, &pkt );
                break;
            case MPID_PKT_REQUEST_SEND:
                err = (*dev->rndv->irecv)( rhandle, from_grank, &pkt );
                break;
            case MPID_PKT_LONG:
                err = (*dev->eager->irecv)( rhandle, from_grank, &pkt );
                break;
        }
    }
    else {
        switch (pkt.head.mode) {
            case MPID_PKT_SHORT:
                err = (*dev->short_msg->unex)( rhandle, from_grank, &pkt );
                break;
            case MPID_PKT_REQUEST_SEND:
                err = (*dev->rndv->unex)( rhandle, from_grank, &pkt );
                break;
            case MPID_PKT_LONG:
                err = (*dev->eager->unex)( rhandle, from_grank, &pkt );
                break;
        }
    }
}
```

Χρησιμοποιώντας το πεδίο `head.mode` της επικεφαλίδας `pkt` διευκρινίζεται αν πρόκειται για μήνυμα ελέγχου ή μήνυμα δεδομένων. Αν η επιστρεφόμενη τιμή της μακροεντολής `MPID_PKT_IS_MSG` είναι αληθής πρόκειται για μήνυμα δεδομένων, και συγκεκριμένα είτε για μήνυμα τύπου `MPID_PKT_SHORT` (short πρωτόκολλο) είτε για μήνυμα τύπου `MPID_PKT_LONG` (eager πρωτόκολλο) είτε για μήνυμα τύπου `MPID_PKT_REQUEST_SEND` (rendezvous πρωτόκολλο). Η ρουτίνα `MPID_Msg_arrived` (πηγαίο αρχείο mpid/util/queue.c) αναζητά σχετικό με το μήνυμα handle στην ουρά `MPID_recvs.posted`. Αν το βρει, θέτει σε αληθή τιμή τη μεταβλητή `is_posted` και επιστρέφει το handle στη μεταβλητή `rhandle`. Αν αντίθετα δεν βρεθεί σχετική καταχώρηση στην ουρά `MPID_recvs.posted`, τότε δημιουργείται μία νέα στην ουρά `MPID_recvs.unexpected` και επιστρέφεται στην `rhandle`, ενώ η `is_posted` ανατίθεται σε μηδενική τιμή. Αν πρόκειται για αναμενόμενο μήνυμα, καλείται η κατάλληλη ρουτίνα λήψης της

συσκευής ADI-2 (short_msg->recv, eager->irecv ή rndv->irecv). Αν αντίθετα πρόκειται για απροσδόκητο μήνυμα, τότε αφενός καταγράφεται το αναγνωριστικό αποστολής στο rhandle, αφετέρου καλείται η αντίστοιχη ρουτίνα λήψης του CH (short_msg->unex, eager->unex ή rndv->unex). Ο κώδικας της MPID_CH_Check_incoming ολοκληρώνεται με το τμήμα

```
else {
    switch (pkt.head.mode) {
        case MPID_PKT_OK_TO_SEND:
            err = (*dev->rndv->do_ack)( &pkt, from_grank );
            break;
        case MPID_PKT_ANTI_SEND:
            MPID_SendCancelOkPacket( &pkt, from_grank );
            break;
        case MPID_PKT_ANTI_SEND_OK:
            MPID_RecvCancelOkPacket( &pkt, from_grank );
            break;
#ifdef MPID_FLOW_CONTROL
        case MPID_PKT_FLOW:
            MPID_RecvFlowPacket( &pkt, from_grank );
            break;
#endif
#ifdef MPID_PACK_CONTROL
        case MPID_PKT_PROTO_ACK:
        case MPID_PKT_ACK_PROTO:
            MPID_RecvProtoAck( &pkt, from_grank );
            break;
#endif
}
```

το οποίο αναφέρεται στο ενδεχόμενο το ληφθέν μήνυμα να είναι μήνυμα ελέγχου.

Γ.6.3 Αναμενόμενη Short Λήψη

Η περίπτωση της **αναμενόμενης short λήψης** αντιμετωπίζεται από την MPID_CH_Eagerb_recv_short, που δεικτοδοτεί η short_msg->recv. Η συνάρτηση αυτή ορίζεται στο αρχείο mpid/ch2/chshort.c και συνοπτικά καθορίζει τα εξής:

```
MPID_PKT_SHORT_T *pkt = (MPID_PKT_SHORT_T *)in_pkt;
int          msglen;
msglen               = pkt->len;
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
rhandle->s.MPI_TAG    = pkt->tag;
rhandle->s.MPI_SOURCE = pkt->lrank;
MEMCPY( rhandle->buf, pkt->buffer, msglen );
rhandle->s.count      = msglen;
if (rhandle->finish)
    (rhandle->finish)( rhandle );
rhandle->is_complete = 1;
```

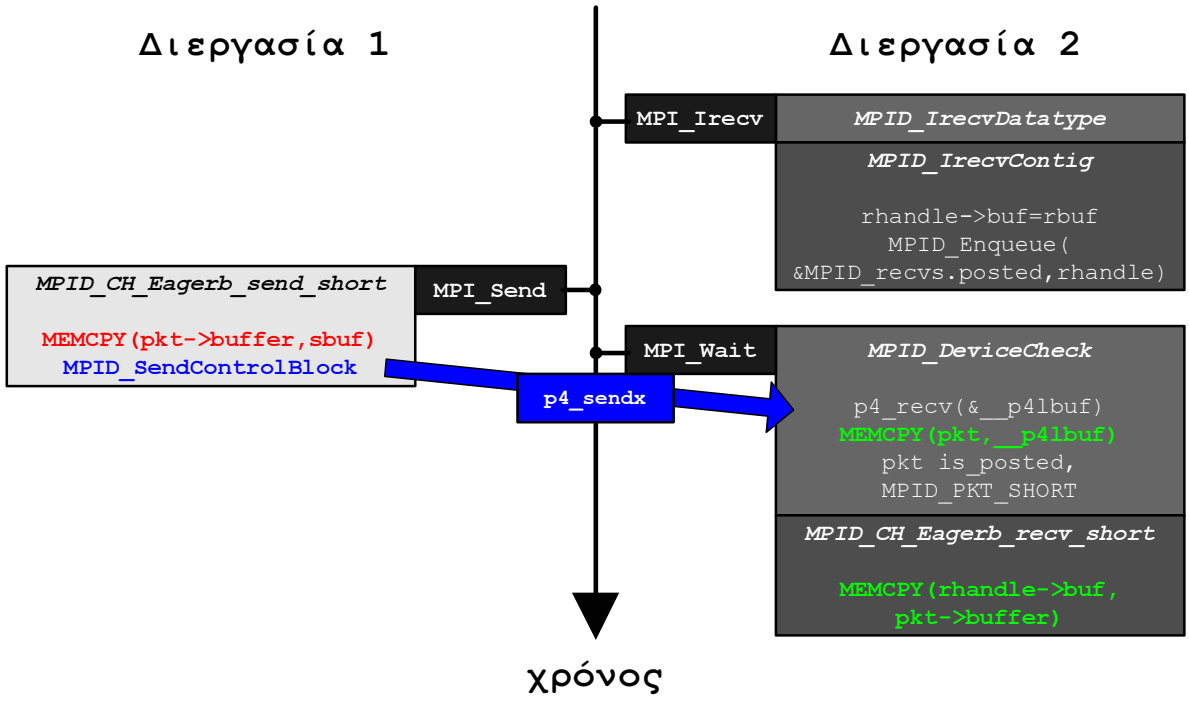
Έτσι, αφού πρώτα γίνει έλεγχος ροής ληφθέντων πακέτων για το κατώφλι `MPI_Pk_ackmark` (μακροεντολή `MPID_PACKET_RCVD_GET`), αντιγράφεται με χρήση της μακροεντολής `MEMCPY` το μήνυμα από το πεδίο `buffer` της επικεφαλίδας `pkt` στο χώρο μνήμης που έχει καθοριστεί πρωτύτερα π.χ. από κλήση της `MPI_Irecv` και η διεύθυνση του οποίου παρέχεται από το πεδίο `buf` του handle λήψης `rhandle`. Έτσι, ακόμα και στην περίπτωση της αναμενόμενης λήψης μηνύματος με το short πρωτόκολλο, παρότι αποστέλλεται μοναδικό μήνυμα δεν αποφεύγουμε την αντιγραφή δεδομένων στον παραλήπτη.

Για να κατανοήσουμε καλύτερα τη διαδικασία της λήψης στις διαφορετικές εκδοχές της και για τα τρία πρωτόκολλα επικοινωνίας του MPICH θα εξετάσουμε την αλληλουχία συμβάντων κατά την ανταλλαγή ενός μηνύματος μεταξύ δύο διεργασιών. Ας υποθέσουμε λοιπόν ότι η διεργασία 1 θέλει να στείλει στη διεργασία 2 ένα μήνυμα. Έστω ότι το μήνυμα βρίσκεται σε κάποιο απομονωτή `sbuf` στη διεργασία αποστολέα, και επιθυμούμε να αποθηκευτεί στον πίνακα `rbuf` στη διεργασία παραλήπτη. Μια αρκετά συνήθης προσέγγιση είναι να επιλεγεί η απλή ρουτίνα αποστολής `MPI_Send` για τη μετάδοση του μηνύματος και να χρησιμοποιηθεί ένας συνδυασμός της ασύγχρονης `MPI_Irecv` με τη ρουτίνα ελέγχου `MPI_Wait` για τη λήψη, προς αποφυγή του συγχρονισμού ή ακόμα και του αδιεξόδου που ενδέχεται να προκαλέσει η απρόσεκτη χρήση της σύγχρονης `MPI_Recv`. Σε MPI θα μπορούσαμε να υλοποιήσουμε την επικοινωνία αυτή ως εξής:

```
int myrank, tag = 13, *sbuf, *rbuf;
MPI_Request request;
MPI_Status status;
```

```
MPI_Comm_rank( MPI_COMM_WORLD, &myrank );

if (myrank == 1)
    MPI_Send( sbuf, sizeof(sbuf)/sizeof(sbuf[0]), MPI_INT, 2,
              tag, MPI_COMM_WORLD );
else if (myrank == 2) {
    MPI_Irecv( rbuf, sizeof(rbuf)/sizeof(rbuf[0]), MPI_INT, 1,
              tag, MPI_COMM_WORLD, &request );
    MPI_Wait( &request, &status );
}
```



Σχήμα Γ.2: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του short πρωτοκόλλου του MPICH (περίπτωση αναμενόμενης λήψης)*

Στο σχήμα Γ.2 απεικονίζεται η αποστολή του μηνύματος `sbuf` από τη διεργασία 1 και η αντίστοιχη λήψη στον `rbuf` από τη διεργασία 2. Αρχικά το μήνυμα αντιγράφεται στη διεργασία 1 στο πακέτο επικεφαλίδα που θα αποσταλεί προς τη 2, σύμφωνα με τη λογική του short πρωτοκόλλου. Η διεργασία 2

προλαβαίνει να εκτελέσει την αντίστοιχη MPI_Irecv πριν ληφθεί το μήνυμα, με συνέπεια τη δημιουργία handle περιγραφής του μηνύματος στην ουρά αναμενόμενων μηνυμάτων MPID_recvs.posted. Εν συνεχεία εκτελείται η MPI_Wait, που λαμβάνει το μήνυμα επικεφαλίδα και αντιλαμβάνεται από το πεδίο head.mode αυτού ότι πρόκειται για το short πρωτόκολλο. Μέσω αναζήτησης στην ουρά MPID_recvs.posted προκύπτει ότι το μήνυμα είναι αναμενόμενο (is_posted αληθής). Έτσι, καλείται η MPID_CH_Eagerb_recv_short για να πραγματοποιήσει την αντιγραφή του μηνύματος από το πεδίο buffer του πακέτου επικεφαλίδας στο χώρο μνήμης που είχε οριστεί από το χρήστη κατά την κλήση της MPI_Irecv.

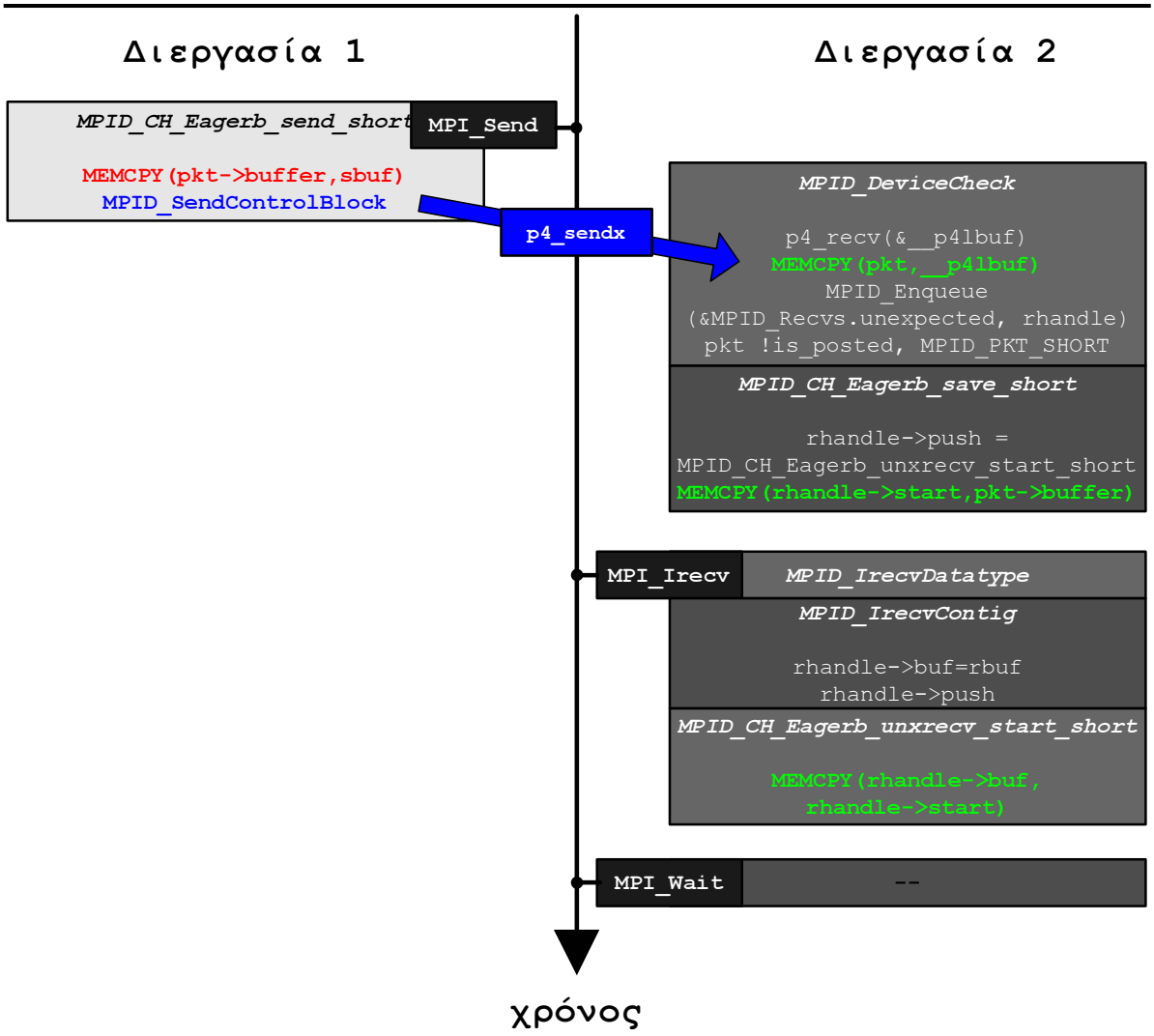
Γ.6.4 Μη Αναμενόμενη Short Λήψη

Η **μη αναμενόμενη short λήψη** αντιμετωπίζεται από την MPID_CH_Eagerb_save_short, που δεικτοδοτεί η short_msg->unex. Η συνάρτηση αυτή ορίζεται στο αρχείο mpid/ch2/chshort.c και συνοπτικά έχει ως εξής:

```
MPID_PKT_SHORT_T   *pkt = (MPID_PKT_SHORT_T *)in_pkt;
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
rhandle->s.MPI_TAG    = pkt->tag;
rhandle->s.MPI_SOURCE = pkt->lrank;
rhandle->s.MPI_ERROR  = 0;
rhandle->from         = from;
rhandle->partner      = pkt->to;
rhandle->s.count      = pkt->len;
rhandle->start    = (void *)MALLOC( pkt->len );
MEMCPY( rhandle->start, pkt->buffer, pkt->len );
rhandle->push = MPID_CH_Eagerb_unxrecv_start_short;
```

Η βασική διαφορά της από την περίπτωση της αναμενόμενης λήψης έγκειται αφενός στην αντιγραφή του μηνύματος στο πεδίο start του handle λήψης αντί για το πεδίο buf, και αφετέρου στον καθορισμό συνάρτησης push, της MPID_CH_Eagerb_unxrecv_start_short. Η τελευταία συνάρτηση είναι αυτή που αναλαμβάνει να πραγματοποιήσει την αντιγραφή των δεδομένων από το πεδίο start του handle στον απομονωτή του χρήστη, μόλις αυτός γίνει γνωστός π.χ. μέσω της MPI_Irecv. Για το λόγο αυτό,

η MPID_CH_Eagerb_unxrecv_start_short καλείται στην MPI_Irecv αφού αρχικοποιηθεί το πεδίο buf του handle λήψης.



Σχήμα Γ.3: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του short πρωτοκόλλου του MPICH (περίπτωση μη αναμενόμενης λήψης)*

Για παράδειγμα, στο απλό σενάριο επικοινωνίας μεταξύ δύο διεργασιών που έχουμε υποθέσει (σχήμα Γ.3), η διαφορά σε σχέση με την αναμενόμενη λήψη είναι ότι η κλήση της MPID_DeviceCheck που λαμβάνει το μήνυμα προηγείται χρονικά της MPI_Irecv. Έτσι, δημιουργείται handle σχετικό με το μήνυμα που τοποθετείται στην ουρά MPID_recvs.unexpected των απροσδόκητων μηνυμάτων και το μήνυμα αντιγράφεται στο πεδίο start του handle αυτού. Κατά την κλήση της MPI_Irecv γνωστοποιείται ο απομονωτής προορισμού rbuf και αφού διαπιστωθεί ότι το μήνυμα έχει ήδη ληφθεί απροσ-

δόκητα, γίνεται κλήση της MPID_CH_Eagerb_unxrecv_start_short για την τελική αντιγραφή των δεδομένων στο σωστό προορισμό τους.

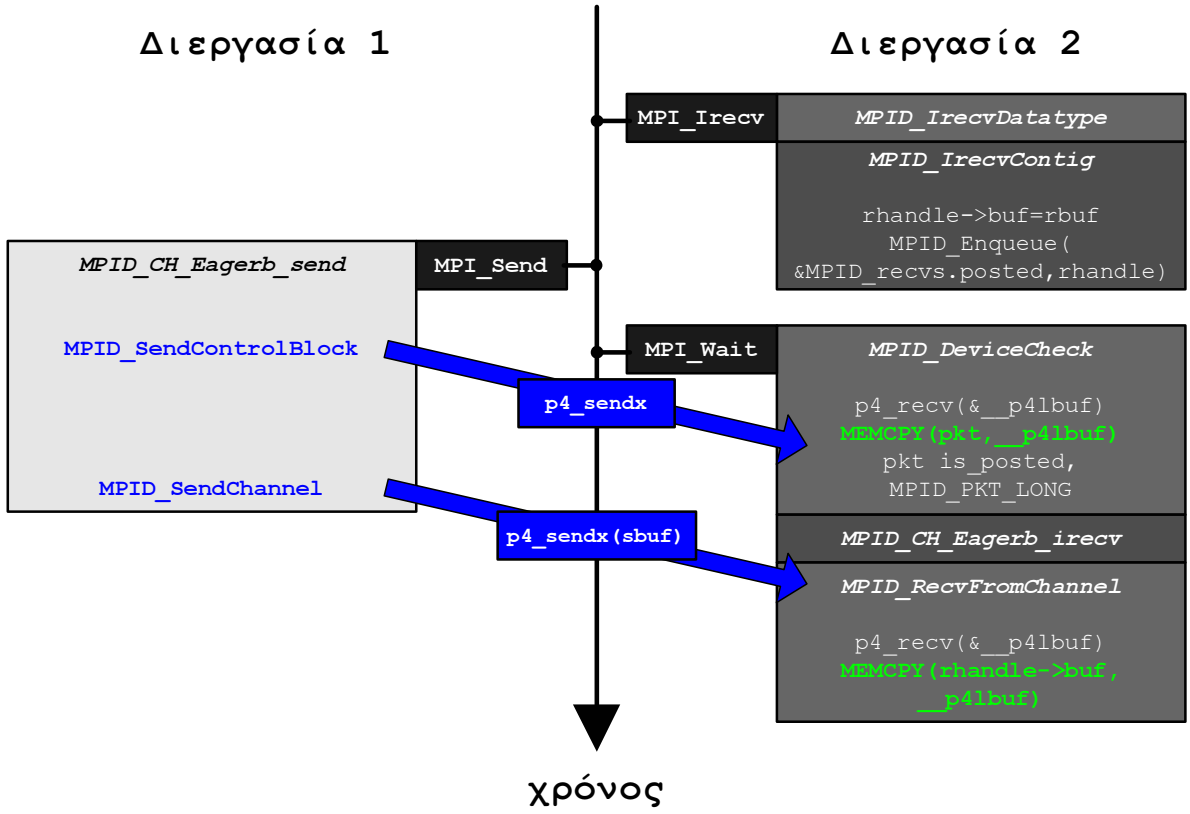
Γ.6.5 Αναμενόμενη Eager Λήψη

Η **αναμενόμενη eager λήψη** πραγματοποιείται από τη συνάρτηση MPID_CH_Eagerb_irecv του αρχείου mpid/ch2/chbeager.c:

```
MPID_PKT_LONG_T *pkt = (MPID_PKT_LONG_T *)in_pkt;
int    msglen = pkt->len;
#ifdef MPID_FLOW_CONTROL
    MPID_FLOW_MEM_GET(pkt,from);
    MPID_FLOW_MEM_READ(msglen,from);
    MPID_FLOW_MEM_RECV(msglen,from);
#endif
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
rhandle->s.count      = msglen;
rhandle->s.MPI_TAG    = pkt->tag;
rhandle->s.MPI_SOURCE = pkt->lrank;
MPID_RecvFromChannel( rhandle->buf, msglen, from );
if (rhandle->finish)
    (rhandle->finish)( rhandle );
rhandle->wait        = 0;
rhandle->test        = 0;
rhandle->push        = 0;
rhandle->is_complete = 1;
```

Αρχικά πραγματοποιείται έλεγχος ροής τόσο για τη διαθέσιμη μνήμη όσο και για τα ανεπιβεβαίωτα πακέτα. Στη συνέχεια, και δεδομένου ότι έχει προηγηθεί ο καθορισμός του απομονωτή λήψης από κλήση σχετικής MPI ρουτίνας, καλείται η μακροεντολή MPID_RecvFromChannel για τη λήψη των δεδομένων από τη συσκευή ADI-2. Η μακροεντολή αυτή αποτελεί τη δυαδική της μακροεντολής αποστολής MPID_SendChannel, και μεταβιβάζει τελικά τον έλεγχο στη βιβλιοθήκη P4. Είναι πάντως σημαντικό

να επισημάνουμε ότι η MPID_RecvFromChannel επιφέρει μια πρόσθετη αντιγραφή δεδομένων, καθώς η βιβλιοθήκη P4 χρησιμοποιεί δικό της εσωτερικό απομονωτή για τη λήψη ενός μηνύματος από το κανάλι επικοινωνίας.



Σχήμα Γ.4: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του eager πρωτοκόλλου του MPICH (περίπτωση αναμενόμενης λήψης)*

Για παράδειγμα, στην περίπτωση της ανταλλαγής μηνύματος μεταξύ δύο διεργασιών (σχήμα Γ.4), η διεργασία 2 προλαβαίνει να εκτελέσει την MPI_Irecv πριν ληφθεί το μήνυμα από το κανάλι ADI-2. Δημιουργείται έτσι το handle λήψης rhandle με τα στοιχεία του μηνύματος και τοποθετείται στην ουρά MPID_recvs.posted των αναμενόμενων μηνυμάτων. Η διεργασία 1 αποστέλλει δύο διαφορετικά μηνύματα, την επικεφαλίδα μέσω της MPID_SendControlBlock και τα δεδομένα μέσω της μακροεντολής MPID_SendChannel. Κατά τη λήψη της επικεφαλίδας μέσω της MPI_Wait, η διεργασία 2 αντιλαμβάνεται ότι πρόκειται για eager μήνυμα, του οποίου ο απομονωτής προορισμού έχει ήδη γνωστοποιηθεί στο αντίστοιχο rhandle->buf της ουράς αναμενόμενων μηνυμάτων. Τελικά, θα κληθεί η MPID_RecvFromChannel για να λάβει τα δεδομένα από το κανάλι CH με τη βοήθεια της P4 βιβλιοθήκης.

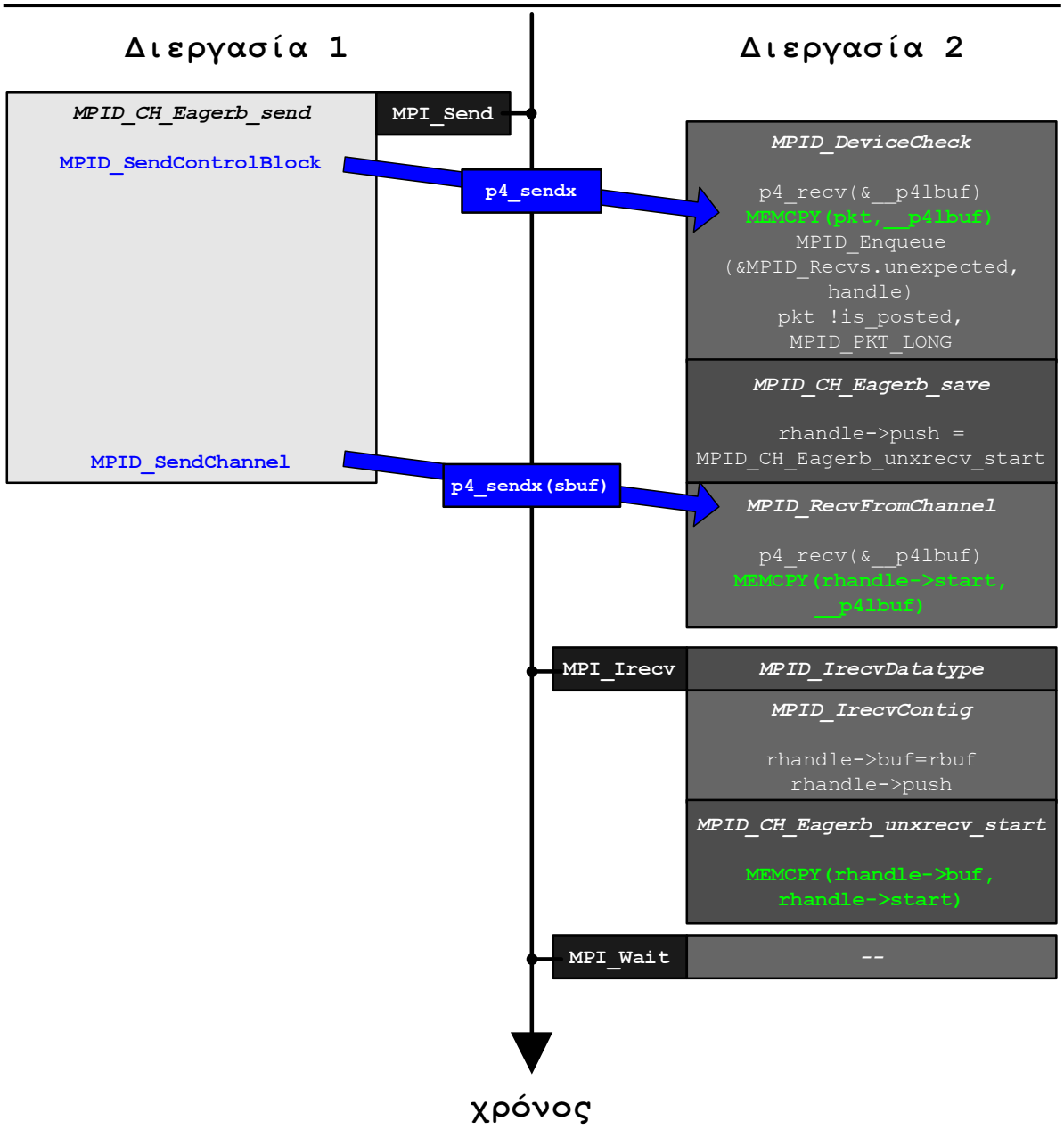
Γ.6.6 Μη Αναμενόμενη Eager Λήψη

Η **μη αναμενόμενη eager λήψη** υλοποιείται από τη ρουτίνα `MPID_CH_Eagerb_save`, που ορίζεται στο αρχείο mpid/ch2/chbeager.c:

```
MPID_PKT_T *pkt = (MPID_PKT_T *)in_pkt;
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->head.src))
        MPID_SendProtoAck(pkt->head.to, pkt->head.src);
    MPID_PACKET_ADD_RCVD(pkt->head.to, pkt->head.src);
#endif
rhandle->s.MPI_TAG    = pkt->head.tag;
rhandle->s.MPI_SOURCE = pkt->head.lrank;
rhandle->partner      = pkt->head.to;
rhandle->s.count      = pkt->head.len;
rhandle->from         = from;
rhandle->is_complete  = 1;
if (pkt->head.len > 0) {
    rhandle->start    = (void *)MALLOC( pkt->head.len );
    rhandle->is_complete  = 1;
    #ifdef MPID_FLOW_CONTROL
        MPID_FLOW_MEM_READ(pkt->head.len,from);
    #endif
    MPID_RecvFromChannel( rhandle->start, pkt->head.len, from );
}
rhandle->push = MPID_CH_Eagerb_unxrecv_start;
```

Η διαφορά με την αναμενόμενη περίπτωση έγκειται στο ότι το μήνυμα λαμβάνεται στον απομονωτή `rhandle->start` του handle λήψης, ενώ επίσης ορίζεται συνάρτηση push για να ολοκληρωθεί η επικοινωνία κατά την κλήση της αντίστοιχης MPI ρουτίνας λήψης.

Έτσι, στο υποθετικό σενάριο επικοινωνίας που εξετάζουμε (σχήμα Γ.5), η διεργασία 2 λαμβάνει το eager μήνυμα μέσω της `DeviceCheck` πριν κληθεί η `MPI_Irecv`. Κάτι τέτοιο θα μπορούσε να συμβεί αν π.χ. η διεργασία 2 έλεγχε το κανάλι μέχρι να λάβει μήνυμα από μια τρίτη διεργασία, και κατά τη λειτουργία αυτή λάμβανε πρόωρα το μήνυμα της διεργασίας 1. Κατά τα γνωστά, η διεργασία 2 τοποθετεί ένα σχετικό handle λήψης `rhandle` στην ουρά απροσδόκητων μηνυμάτων, και το μήνυμα αποθηκεύεται προσωρινά στο πεδίο `rhandle->start`. Όταν κληθεί η `MPI_Irecv` γνωστοποιείται ο απομονωτής



Σχήμα Γ.5: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του eager πρωτοκόλλου του MPICH (περίπτωση μη αναμενόμενης λήψης)*

προορισμού `rbuf`, και καλείται η συνάρτηση `MPID_CH_Eagerb_unxrecv_start` για την απαιτούμενη αντιγραφή του μηνύματος.

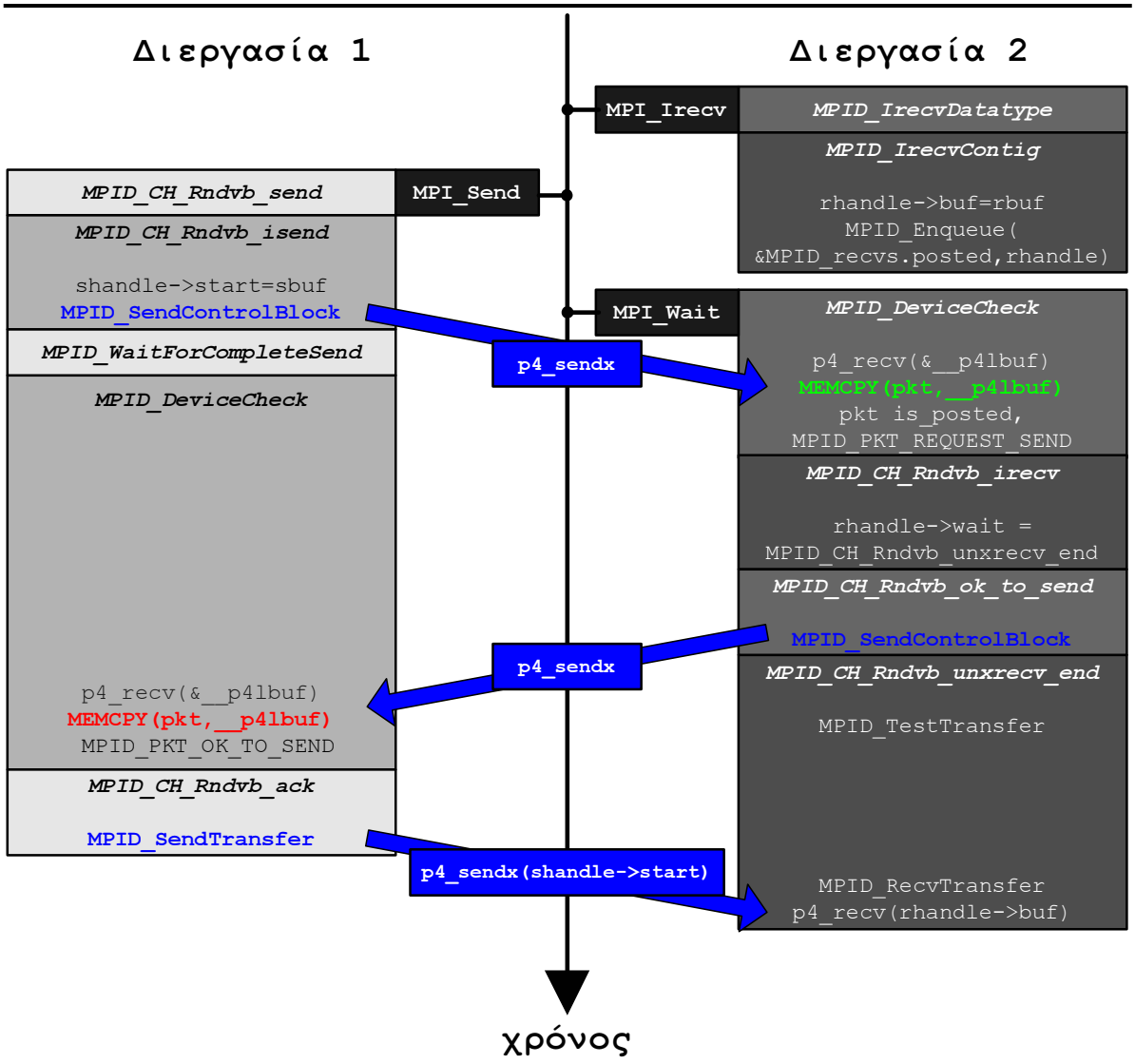
Παρατηρούμε ότι κατά την απροσδόκητη eager λήψη πραγματοποιούνται συνολικά τρεις αντιγραφές στον παραλήπτη (μακροεντολή `MEMCPY`), εκ των οποίων μάλιστα η μία αφορά ενδιάμεση αποθή-

κευση των δεδομένων του μηνύματος στον `rhandle->start`. Οι αντιγραφές αυτές μπορεί να προκαλέσουν κατασπατάληση των διαθέσιμων πόρων μνήμης του συστήματος, οδηγώντας δυνητικά ακόμα και σε αδυναμία περάτωσης της eager επικοινωνίας. Για το λόγο αυτό, το MPICH αφενός υλοποιεί έλεγχο ροής μνήμης πριν αποσταλεί ένα μήνυμα μέσω του eager πρωτοκόλλου, ενώ στην περίπτωση σχετικά μεγάλων μηνυμάτων επιλέγεται το rendezvous πρωτόκολλο για να αποφευχθεί η χρονοβόρα ενδιάμεση αντιγραφή της απροσδόκητης λήψης.

Γ.6.7 Αναμενόμενη Rendezvous Λήψη

Περισσότερο σύνθετη περίπτωση είναι αυτή της **αναμενόμενης rendezvous λήψης**. Η διαχείριση της λήψης μεταβιβάζεται στη ρουτίνα `MPID_CH_Rndvb_irecv`, που ορίζεται στο αρχείο mpid/ch2/chbrndv.c:

```
MPID_PKT_REQUEST_SEND_T *pkt = (MPID_PKT_REQUEST_SEND_T *)in_pkt;
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
rhandle->s.count      = pkt->len;
rhandle->s.MPI_TAG    = pkt->tag;
rhandle->s.MPI_SOURCE = pkt->lrank;
rhandle->s.MPI_ERROR  = err;
rhandle->from         = from;
rhandle->send_id      = pkt->send_id;
#ifdef MPID_PACK_CONTROL
    while (!MPID_PACKET_CHECK_OK(from))
        MPID_DeviceCheck( MPID_BLOCKING );
#endif
MPID_CreateRecvTransfer( 0, 0, from, &rtag );
MPID_CH_Rndvb_ok_to_send( rhandle->send_id, rtag, from );
rhandle->recv_handle = rtag;
rhandle->wait        = MPID_CH_Rndvb_unxrecv_end;
rhandle->test        = MPID_CH_Rndvb_unxrecv_test_end;
rhandle->push        = 0;
rhandle->from    = from;
rhandle->is_complete = 0;
```



Σχήμα Γ.6: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του rendezvous πρωτοκόλλου του MPICH (περίπτωση αναμενόμενης λήψης)*

Αρχικά πραγματοποιείται έλεγχος για το κατώφλι `MPI_Pk_ackmark` και αποστέλλεται μήνυμα επιβεβαίωσης, εφόσον κριθεί απαραίτητο. Εν συνεχεία ενημερώνεται το handle λήψης, το οποίο έχει ήδη δημιουργηθεί στην αναμενόμενη περίπτωση που εξετάζουμε. Τελικά, αν το επιτρέπει και ο αντίστροφος έλεγχος ροής πακέτων προς τη διεργασία αποστολέα (μακροεντολή `MPID_PACKET_CHECK_OK`), αποστέλλεται μέσω της ρουτίνας `MPID_CH_Rndvb_ok_to_send` σχετική έγκριση για την περαιτέρω συνέχιση της επικοινωνίας με τη μετάδοση των δεδομένων, οπότε και ολοκληρώνεται η φάση χειραψίας μεταξύ αποστολέα και παραλήπτη. Επιπλέον, αρχικοποιείται το πεδίο `wait` του handle λήψης, ώστε

να κληθεί η κατάλληλη συνάρτηση για την ολοκλήρωση της rendezvous επικοινωνίας. Τέλος, παρατηρούμε ότι αντίθετα με τα short και eager πρωτόκολλα, εδώ το πεδίο `is_complete` του handle λήψης αρχικοποιείται σε μηδενική τιμή, ώστε να καταδείξει τη βηματική ακολουθία που πρέπει να εκτελεστεί για την επιτυχή ολοκλήρωση της επικοινωνίας.

Για περισσότερη αποσαφήνιση του rendezvous πρωτοκόλλου, ας εξετάσουμε εποπτικά την επικοινωνία μεταξύ των δύο διεργασιών στο γνωστό σενάριο (σχήμα Γ.6). Η διεργασία 1 αποστέλλει με χρήση της `MPID_SendControlBlock` ένα πακέτο τύπου `MPID_PKT_REQUEST_SEND` στη διεργασία 2, που έχει στο μεταξύ προλάβει να καταχωρίσει την επικοινωνία στην ουρά `MPID_recvs.posted` ως αναμενόμενη. Η διεργασία 2 λαμβάνει την αίτηση χειραψίας και σε κατάλληλο χρόνο θα απαντήσει με ένα μήνυμα ελέγχου `MPID_PKT_OK_TO_SEND` για να ολοκληρώσει τη χειραψία. Μόλις η διεργασία 1 λάβει την έγκριση, θα αποστείλει τα δεδομένα μέσω της μακροεντολής `MPID_SendTransfer`, που μεταβιβάζει όπως και η `MPID_SendChannel` τον έλεγχο στη βιβλιοθήκη P4. Στο μεταξύ, η διεργασία 2 έχει εισέλθει σε έναν ατέρμονα βρόχο αναμονής των δεδομένων, τα οποία τελικά λαμβάνει μόλις γίνουν διαθέσιμα μέσω της `MPID_RecvTransfer`.

Από το παραπάνω παράδειγμα γίνεται εμφανής ο συγχρονισμός των δύο διεργασιών κατά τη χρήση του rendezvous πρωτοκόλλου. Προφανώς, το όφελος που αποκομίζουμε από το συγχρονισμό αυτό έγκειται στην αποφυγή ενδιάμεσης αποθήκευσης των δεδομένων επικοινωνίας, που μπορεί να αποδειχθεί ιδιαίτερα δαπανηρή για μεγάλα μηνύματα.

Γ.6.8 Μη Αναμενόμενη Rendezvous Λήψη

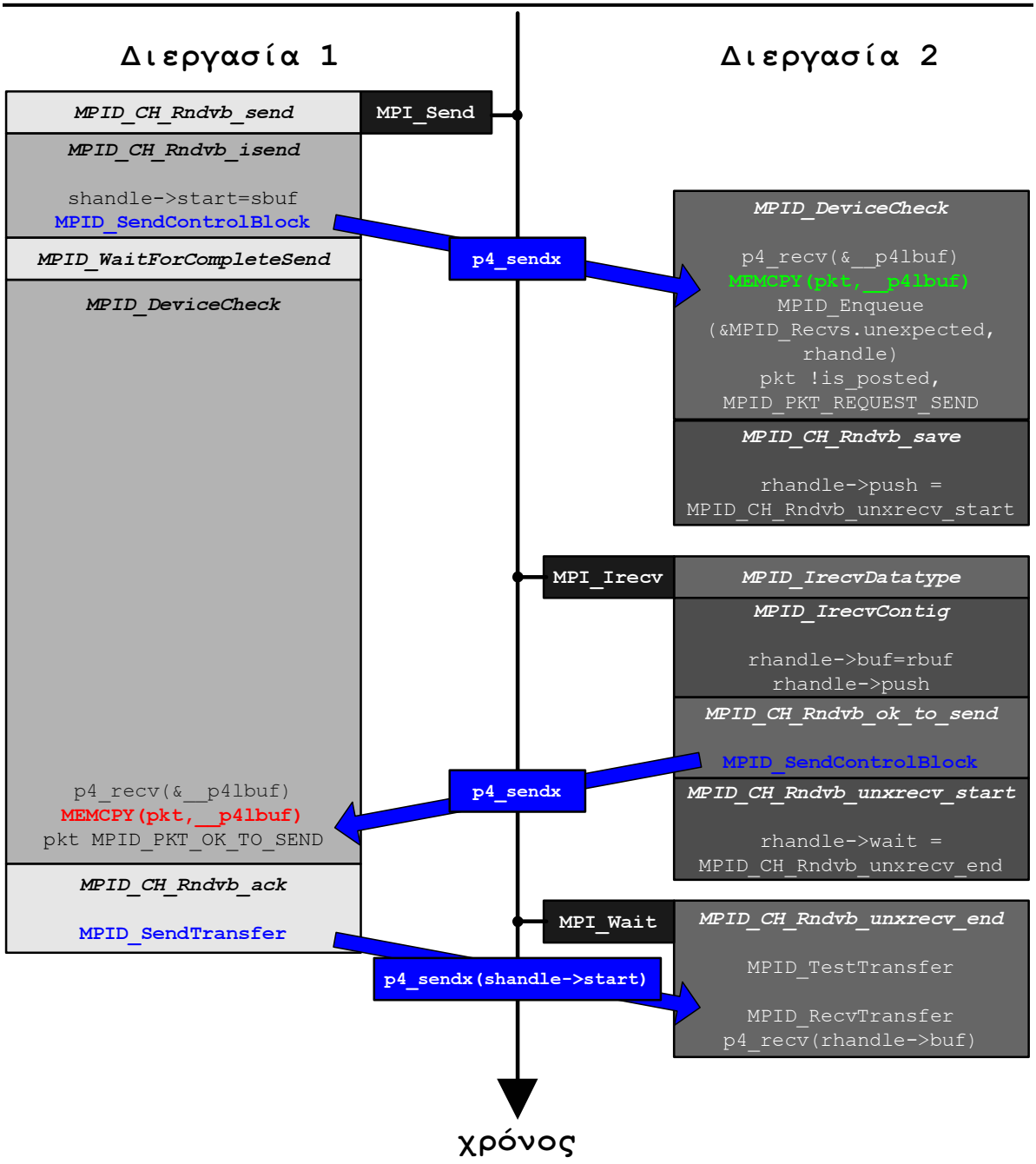
Η τελευταία περίπτωση είναι αυτή της **μη αναμενόμενης rendezvous λήψης**. Εκεί το αρχικό μήνυμα αίτησης χειραψίας λαμβάνεται από τη ρουτίνα `MPID_CH_Rndvb_save` του αρχείου mpid/ch2/chbrndv.c:

```
MPID_PKT_REQUEST_SEND_T   *pkt = (MPID_PKT_REQUEST_SEND_T *)in_pkt;
#ifdef MPID_PACK_CONTROL
    if (MPID_PACKET_RCVD_GET(pkt->src))
        MPID_SendProtoAck(pkt->to, pkt->src);
    MPID_PACKET_ADD_RCVD(pkt->to, pkt->src);
#endif
rhandle->s.MPI_TAG    = pkt->tag;
rhandle->s.MPI_SOURCE = pkt->lrank;
rhandle->s.MPI_ERROR  = 0;
rhandle->s.count      = pkt->len;
rhandle->is_complete  = 0;
rhandle->from         = from;
```

```
rhandle->partner      = pkt->to;
rhandle->send_id      = pkt->send_id;
rhandle->push = MPID_CH_Rndvb_unxrecv_start;
```

Η βασική διαφορά σε σχέση με την αναμενόμενη περίπτωση έγκειται στην αρχικοποίηση του πεδίου `push` του handle λήψης, ώστε να δεικτοδοτεί τη ρουτίνα που θα ολοκληρώσει τη φάση χειραψίας. Έτσι, όπως βλέπουμε και στο σενάριο του σχήματος Γ.7, η αρχική αίτηση `MPID_PKT_REQUEST_SEND` λαμβάνεται πρόωρα από τη ρουτίνα `MPID_DeviceCheck`. Βάσει της αίτησης αυτής αρχικοποιείται handle λήψης στην ουρά απροσδόκητων λήψεων `MPID_recvs.unexpected` και καθορίζεται το πεδίο `push` για τη μελλοντική ολοκλήρωση της επικοινωνίας. Όταν κληθεί η `MPI_Irecv` στη διεργασία 2 καθορίζεται ο απομονωτής προορισμού, αποστέλλεται έγκριση αποστολής και καλείται η λειτουργία push, που με τη σειρά της καθορίζει τη wait λειτουργία για την ολοκλήρωση της λήψης των δεδομένων. Λαμβάνοντας την έγκριση αποστολής, η διεργασία 1 μεταβαίνει στη ρουτίνα `MPID_CH_Rndvb_ack` για την τελική αποστολή των δεδομένων μέσω της μακροεντολής `MPID_SendTransfer`. Τα δεδομένα θα παραληφθούν από τη διεργασία 2 μόλις η τελευταία καλέσει τη σχετική `MPI_Wait` και θα αποθηκευτούν απευθείας στον απομονωτή που καθόρισε ο χρήστης.

Παρατηρούμε ότι ακόμα και στην περίπτωση της μη αναμενόμενης rendezvous λήψης δεν προκύπτει ενδιάμεση αποθήκευση των δεδομένων επικοινωνίας. Αντίθετα, απλώς αναστέλλεται η φάση της χειραψίας μέχρι να καθοριστεί από την `MPI_Irecv` ο απομονωτής προορισμού των δεδομένων. Φυσικά, εδώ η επιβάρυνση λόγω του συγχρονισμού των διεργασιών αναμένεται να είναι ακόμα μεγαλύτερη σε σχέση με την περίπτωση της αναμενόμενης λήψης.



Σχήμα Γ.7: *Επικοινωνία μεταξύ δύο διεργασιών κατά τη χρήση του rendezvous πρωτοκόλλου του MPICH (περίπτωση μη αναμενόμενης λήψης)*

Βιβλιογραφία

[ABRY03] R. Andonov, S. Balev, S. Rajopadhye, and N. Yanev. Optimal Semi-oblique Tiling. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):944–960, Sep 2003.

[ACN+00] R. Andonov, P. Calland, S. Niar, S. Rajopadhye, and N. Yanev. First Steps Towards Optimal Oblique Tile Sizing. In *Proceedings of the 8th International Workshop on Compilers for Parallel Computers*, pages 351–366, Aussois, France, Jan 2000.

[AGMJ04] E. Ayguadé, M. González, X. Martorell, and G. Jost. Employing Nested OpenMP for the Parallelization of Multi-zone Computational Fluid Dynamics Applications. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium 2004 (IPDPS 2004)*, Santa Fe, New Mexico, USA, Apr 2004.

[AKPT99] T. Andronikos, N. Koziris, G. Papakonstantinou, and P. Tsanakas. Optimal Scheduling for UET/UET-UCT Generalized N-Dimensional Grid Task Graphs. *Journal of Parallel and Distributed Computing*, 57(2):140–165, May 1999.

[ASTK02] M. Athanasaki, A. Sotiropoulos, G. Tsoukalas, and N. Koziris. Pipelined Scheduling of Tiled Nested Loops onto Clusters of SMPs Using Memory Mapped Network Interfaces. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Baltimore, Maryland, USA, 2002. IEEE Computer Society Press.

[Ban88] U. Banerjee. *Dependence Analysis for Supercomputing*. Kluwer Academic Publishers, 1988.

[BDH+02] R. Biswas, M. Djomehri, R. Hood, H. Jin, C. Kiris, and S. Saini. An Application-Based Performance Characterization of the Columbia Supercluster. In *Proceedings of the 2005 ACM Conference on Supercomputing*, Seattle, Washington, USA, Nov 2002.

[BDRR94] P. Boulet, A. Darte, T. Risset, and Y. Robert. (Pen)-ultimate Tiling? *INTEGRATION, The VLSI Jounal*, 17:33–51, 1994.

[BDRV99] P. Boulet, J. Dongarra, Y. Robert, and F. Vivien. Static Tiling for Heterogeneous Computing Platforms. *Journal of Parallel Computing*, 25(5):547–568, May 1999.

[BL94] R. Butler and E. Lusk. Monitors, Messages, and Clusters: The p4 Parallel Programming System. *Journal of Parallel Computing*, 20(4):547–564, 1994.

[Boa05] OpenMP Architecture Review Board. OpenMP Application Program Interface. Technical report, OpenMP Architecture Review Board, May 2005. Version 2.5.

[CDMC+05] C. Coarfă, Y. Dotsenko, J. Mellor-Crummey, F. Cantonnet, T. El-Ghazawi, A. Mohanti, Y. Yao, and D. Chavarría-Miranda. An Evaluation of Global Address Space Languages: Co-Array Fortran and Unified Parallel C. In *Proceedings of the ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming (PPoPP)*, pages 36–47, Chicago, Illinois, USA, Jun 2005. ACM Press.

[CDR97] P. Calland, J. Dongarra, and Y. Robert. Tiling with Limited Resources. In *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pages 229–238. IEEE Computer Society Press, Jul 1997.

[CDR99] P. Calland, J. Dongarra, and Y. Robert. Tiling on Systems with Communication/Computation Overlap. *Journal of Concurrency: Practice and Experience*, 11(3):139–153, 1999.

[CE00] F. Cappello and D. Etiemble. MPI versus MPI+OpenMP on IBM SP for the NAS Benchmarks. In *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (CDROM)*, page 12, Dallas, Texas, USA, 2000. IEEE Computer Society.

[CL95] M. Cierniak and W. Li. Unifying Data and Control Transformations for Distributed Shared Memory Machines. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'95)*, pages 205–217, La Jolla, California, USA, Jun 1995.

[CM95] S. Coleman and K. S. McKinley. Tile Size Selection Using Cache Organization and Data Layout. In *Proceedings of the ACM SIGPLAN 1995 Conference on Programming Language Design and Implementation*, pages 279–290, La Jolla, CA, Jun 1995.

[CM99] J. Chame and S. Moon. A Tile Selection Algorithm for Data Locality and Cache Interference. In *Proceedings of the 13th ACM International Conference on Supercomputing (ICS '99)*, pages 492–499, Rhodes, Greece, Jun 1999.

[CMT94] S. Carr, K. McKinley, and C. Tseng. Compiler Optimizations for Improving Data Locality. In *Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'94)*, pages 252–262, San Jose, California, USA, Oct 1994.

[Con05] UPC Consortium. UPC Language Specifications, V1.2. Technical Report LBNL-59208, Lawrence Berkeley National Lab, May 2005.

[CRE00] F. Cappello, O. Richard, and D. Etiemble. Investigating the Performance of Two Programming Models for Clusters of SMP PCs. In *Proceedings of the 2000 IEEE HPCA Conference*, pages 349–359. IEEE Press, Jan 2000.

[DGR05] A. Debudaj-Grabysz and R. Rabenseifner. Nesting OpenMP in MPI to Implement a Hybrid Communication Method of Parallel Simulated Annealing on a Cluster of SMP Nodes. In *Proceedings of the 12th EuroPVM/MPI Conference 2005 (EuroPVM/MPI 2005)*, pages 18–27, Sorrento, Italy, Sep 2005.

[D'H92] E. D'Hollander. Partitioning and Labeling of Loops by Unimodular Transformations. *IEEE Transactions on Parallel and Distributed Systems*, 3(4):465–476, Jul 1992.

[DK02] S. Dong and G. Karniadakis. Dual-level Parallelism for Deterministic and Stochastic CFD Problems. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Baltimore, Maryland, USA, 2002. IEEE Computer Society Press.

[DK04] S. Dong and G. Karniadakis. Dual-level Parallelism for High-order CFD Methods. *Journal of Parallel Computing*, 30(1):1–20, 2004.

[DM98] L. Dagum and R. Menon. OpenMP: An Industry-standard API for Shared-Memory Programming. *IEEE Computational Science and Engineering*, 5(1):46–55, 1998.

[DMS99] J. Dongarra, H. Meuerand, and S. Strohmaier. Top500 Supercomputer Sites. Technical Report UT-CS-99-434, Department of Computer Science, University of Tennessee, Nov 1999.

[EGCD02] T. El-Ghazawi, W. Carlson, and J. Draper. UPC Language Specifications, V1.1.1. In *2nd UPC Workshop*, Washington DC, USA, Mar 2002.

[For94] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. *International Journal of Supercomputer Applications*, 8(3/4):159–416, 1994.

[For97] High Performance Fortran Forum. HPF Language Specification, V2.0. Technical report, Jan 1997.

[GAM+00] M. González, E. Ayguadé, X. Martorell, J. Labarta, N. Navarro, and J. Oliver. NanosCompiler: Supporting Flexible Multilevel Parallelism Exploitation in OpenMP. *Journal of Concurrency: Practice and Experience*, 12(12):1205–1218, 2000.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering Computation. MIT Press, Nov 1994.

[GKKS00] W. Gropp, D. Kaushik, D. Keyes, and B. Smith. Performance Modeling and Tuning of an Unstructured Mesh CFD Application. In *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (CDROM)*, page 833, Dallas, Texas, USA, Nov 2000.

[GLT99] W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2: Advanced Features of the Message Passing Interface*. Scientific and Engineering Computation series. MIT Press, Cambridge, MA, 1999.

[GSK01] G. Goumas, A. Sotiropoulos, and N. Koziris. Minimizing Completion Time for Loop Tiling with Computation and Communication Overlapping. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)*, San Francisco, USA, Apr 2001.

[HAA+96] M. Hall, J. Anderson, S. Amarasinghe, B. Murphy, S. Liao, E. Bugnion, and M. Lam. Maximizing Multiprocessor Performance with the SUIF Compiler. *Computer*, 29(12):84–89, 1996.

[HCF97] K. Högstedt, L. Carter, and J. Ferrante. Determining the Idle Time of a Tiling. In *Proceedings of the Principles of Programming Languages (POPL'97)*, pages 319–323, Paris, France, Jan 1997.

[HCF99] K. Högstedt, L. Carter, and J. Ferrante. Selecting Tile Shape for Minimal Execution Time. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 201–211, Saint Malo, France, 1999.

[HCF03] K. Högstedt, L. Carter, and J. Ferrante. On the Parallel Execution Time of Tiled Loops. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):307–321, Mar 2003.

[HD02] Y. He and C. Ding. MPI and OpenMP Paradigms on Cluster of SMP Architectures: the Vacancy Tracking Algorithm for Multi-dimensional Array Transposition. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Baltimore, Maryland, USA, 2002. IEEE Computer Society Press.

[Hen00] D. Henty. Performance of Hybrid Message-Passing and Shared-Memory Parallelism for Discrete Element Modeling. In *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing (CDROM)*, page 10, Dallas, Texas, USA, 2000. IEEE Computer Society.

[HLCZ00] Y. Hu, H. Lu, A. Cox, and W. Zwaenepoel. OpenMP for Networks of SMPs. *Journal of Parallel and Distributed Computing*, 60(12):1512–1530, 2000.

[Hod99] E. Hodzic. *Time Optimal Tiling of Algorithms with Uniform Dependencies for Distributed Memory Parallel Computers*. PhD thesis, Santa Clara University, School of Engineering, Jun 1999.

[HS98] E. Hodzic and W. Shang. On Supernode Transformation with Minimized Total Running Time. *IEEE Transactions on Parallel and Distributed Systems*, 9(5):417–428, May 1998.

[HS99] E. Hodzic and W. Shang. On Time Optimal Supernode Shape. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pages 2019–2026, Las Vegas, USA, Jun 1999.

[HS02] E. Hodzic and W. Shang. On Time Optimal Supernode Shape. *IEEE Transactions on Parallel and Distributed Systems*, 13(12):1220–1233, Dec 2002.

[HST99] W. Hu, W. Shi, and Z. Tang. JIAJIA: An SVM System Based on a New Cache Coherence Protocol. In *Proceedings of the High-Performance Computing and Networking Europe 1999 (HPCN'99)*, pages 463–472, 1999.

[IT88] F. Irigoin and R. Triolet. Supernode Partitioning. In *Proceedings of the 15th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'85)*, pages 319–329, San Diego, California, USA, Jan 1988.

[KC03a] G. Krawezik and F. Cappello. Performance Comparison of MPI and three OpenMP Programming Styles on Shared Memory Multiprocessors. In *Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures*, pages 118–127, San Diego, California, USA, Jun 2003. ACM Press.

[KC03b] G. Krawezik and F. Cappello. Performance Comparison of MPI and three OpenMP Programming Styles on Shared Memory Multiprocessors. *Journal of Concurrency and Computation: Practice and Experience*, 2003.

[KCDZ94] P. Keleher, A. Cox, S. Dwarkadas, and W. Zwaenepoel. TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems. In *Proceedings of the 1994 Winter USENIX Conference*, Jan 1994.

[KCRB99] M. Kandemir, A. Choudhary, J. Ramanujam, and P. Banerjee. On Reducing False Sharing while Improving Locality on Shared Memory Multiprocessors. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT'99)*, pages 203–211, Newport Beach, California, USA, Oct 1999.

[KCS+98] M. Kandemir, A. Choudhary, N. Shenoy, P. Banerjee, and J. Ramanujam. A Hyperplane Based Approach for Optimizing Spatial Locality in Loop Nests. In *Proceedings of the International Conference on Supercomputing (ICS'98)*, pages 69–76, Melbourne, Australia, Jul 1998.

[KK02] G. Karniadakis and R. Kirby. *Parallel Scientific Computing in C++ and MPI : A Seamless Approach to Parallel Algorithms and their Implementation*. Cambridge University Press, 2002.

[KKH03] Y. Kee, J. Kim, and S. Ha. ParADE: An OpenMP Programming Environment for SMP Cluster Systems. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing (CDROM)*, Phoenix, Arizona, USA, Nov 2003.

[KLB02] S. Karlsson, S. Lee, and M. Brorsson. A Fully Compliant OpenMP Implementation on Software Distributed Shared Memory. In *Proceedings of the 9th International Conference on High Performance Computing (HiPC)*, pages 195–208, Bangalore, India, Dec 2002.

[KM92] K. Kennedy and K. McKinley. Optimizing for Parallelism and Data Locality. In *Proceedings of the International Conference on Supercomputing (ICS'92)*, pages 323–334, Washington, D. C., USA, Jul 1992.

[KMP+96] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott. The Omega Library Interface Guide. Technical Report UMIACS-TR-95-41, University of Maryland at College Park, College Park, Maryland, USA, Apr 1996.

[KRC99] M. Kandemir, J. Ramanujam, and A. Choudary. Improving Cache Locality by a Combination of Loop and Data Transformations. *IEEE Transactions on Parallel and Distributed Systems*, 48(2):159–167, Feb 1999.

[KSG03] N. Koziris, A. Sotiropoulos, and G. Goumas. A Pipelined Schedule to Minimize Completion Time for Loop Tiling with Computation and Communication Overlapping. *Journal of Parallel and Distributed Computing*, 63(11):1138–1151, Nov 2003.

[KW04] W. Kuchera and C. Wallace. The UPC Memory Model: Problems and Prospects. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium 2004 (IPDPS 2004)*, Santa Fe, New Mexico, Apr 2004.

[LP92] W. Li and K. Pingali. Compiler Optimizations for Cache Locality and Coherence. Technical Report TR 92-1278, Cornell University Ithaca, New York, 1992.

[LRW91] M. Lam, E. Rothberg, and M. Wolf. The Cache Performance and Optimizations of Blocked Algorithms. In *Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 63–74, Santa Clara, CA, Apr 1991.

[LTD01] R. Loft, S. Thomas, and J. Dennis. Terascale Spectral Element Dynamical Core for Atmospheric General Circulation Models. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM)*, page 18, Denver, Colorado, 2001. ACM Press.

[Maj00] A. Majumdar. Parallel Performance Study of Monte Carlo Photon Transport Code on Shared-, Distributed-, and Distributed-Shared-Memory Architectures. In *Proceedings of the 14th International Parallel Processing Symposium (IPPS 2000)*, page 93, Cancun, Mexico, May 2000.

[MCT96] K. McKinley, S. Carr, and C.-W. Tseng. Improving Data Locality with Loop Transformations. *ACM Transactions on Programming Languages and Systems*, 18(4):424–453, Jul 1996.

[MH95] J. Merlin and A. Hey. An Introduction to High Performance Fortran. *Scientific Programming*, 4(2):87–113, 1995.

[MLNA96] X. Martorell, J. Labarta, N. Navarro, and E. Ayguadé. A Library Implementation of the Nano-Threads Programming Model. In *Proceedings of the Euro-Par Conference*, pages 644–649, 1996.

[Nak03] K. Nakajima. OpenMP / MPI Hybrid vs. Flat MPI on the Earth Simulator: Parallel Iterative Solvers for Finite Element Method. In *Proceedings of the 5th International Symposium on High Performance Computing (ISHPC 2003)*, pages 486–499, Tokyo-Odaiba, Japan, Oct 2003.

[NBF96] B. Nichols, D. Buttlar, and J. Farrell. *Pthreads Programming*. O'Reilly & Associates, Sebastopol, CA, USA, 1996.

[NR98] R. Numrich and J. Reid. Co-Array Fortran for Parallel Programming. *SIGPLAN Fortran Forum*, 17(2):1–31, 1998.

[OSKO95] H. Ohta, Y. Saito, M. Kainaga, and H. Ono. Optimal Tile Size Adjustment in Compiling General DOACROSS Loop Nests. In *Proceedings of the 9th International Conference on Supercomputing (ICS'95)*, pages 270–279, Barcelona, Spain, Jul 1995.

[PMN+98] E. Polychronopoulos, X. Martorell, D. Nikolopoulos, J. Labarta, T. Papatheodorou, and N. Navarro. Kernel-Level Scheduling for the Nano-Threads Programming Model. In *Proceedings of the 12th ACM International Conference on Supercomputing (ICS '98)*, pages 337–344, Melbourne, Australia, 1998. ACM Press.

[Rab02] R. Rabenseifner. Communication Bandwidth of Parallel Programming Models on Hybrid Architectures. In *Proceedings of WOMPEI 2002, International Workshop on OpenMP: Experiences and Implementations, part of ISHPC-IV, International Symposium on High Performance Computing*, Kansai Science City, Japan, May 2002.

[Rab03] R. Rabenseifner. Hybrid Parallel Programming: Performance Problems and Chances. In *Proceedings of the Cray User Group Conference, CUG SUMMIT 2003*, Columbus, Ohio, USA, May 2003.

[RS92] J. Ramanujam and P. Sadayappan. Tiling Multidimensional Iteration Spaces for Multicomputers. *Journal of Parallel and Distributed Computing*, 16:108–120, 1992.

[RW03] R. Rabenseifner and G. Wellein. Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures. *International Journal of High Performance Computing Applications*, 17(1):49–62, 2003.

[SB01] L. Smith and M. Bull. Development of Mixed Mode MPI / OpenMP Applications. *Scientific Programming*, 9(2-3):83–98, 2001.

[SDH+97] R. Stets, S. Dwarkadas, N. Hardavellas, G. Hunt, L. Kontothanassis, S. Parthasarathy, and M. Scott. Cashmere-2L: Software Coherent Shared Memory on a Clustered Remote-Write Network. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP '97)*, pages 170–183, Saint Malo, France, 1997. ACM Press.

[SEKBL04] M. Su, I. El-Kady, D. Bader, and S. Lin. A Novel FDTD Application Featuring OpenMP-MPI Hybrid Parallelization. In *Proceedings of the International Conference on Parallel Processing (ICPP'04)*, pages 373–379, Montreal, Canada, Aug 2004.

[SF91] W. Shang and J. Fortes. Time Optimal Linear Schedules for Algorithms with Uniform Dependencies. *IEEE Transactions on Computers*, 40(6):723–742, Jun 1991.

[SL01] Y. Song and Z. Li. Impact of Tile-Size Selection for Skewed Tiling. In *Proceedings of the 5th Workshop on Interaction between Compilers and Architectures (INTERACT'01)*, Monterrey, Mexico, Jan 2001.

[SSKT99] M. Sato, S. Satoh, K. Kusano, and Y. Tanaka. Design of OpenMP Compiler for an SMP Cluster. In *Proceedings of the European Workshop on OpenMP (EWOMP '99)*, pages 32–39, Lund, Sweden, Sep 1999.

[SSOB03] H. Shan, J. Singh, L. Oliker, and R. Biswas. Message Passing and Shared Address Space Parallelism on an SMP Cluster. *Parallel Computing*, 29(2):167–186, 2003.

[Ste90] R. Stevens. *UNIX Network Programming*. Software Series. Prentice Hall, Upper Saddle River, NJ, USA, 1990.

[Taf01] J. Taft. Achieving 60 GFLOP/s on the Production CFD Code OVERFLOW-MLP. *Parallel Computing*, 27(4):521–536, 2001.

[TZ94] P. Tang and J. Zigman. Reducing Data Communication Overhead for DOACROSS Loop Nests. In *Proceedings of the 8th International Conference on Supercomputing (ICS'94)*, pages 44–53, Manchester, UK, Jul 1994.

[WHBF02] G. Wellein, G. Hager, A. Basermann, and H. Fehske. Fast Sparse Matrix-Vector Multiplication for TeraFlop/s Computers. In *Proceedings of the 5th International Conference of High Performance Computing for Computational Science - VECPAR 2002*, pages 287–301, Porto, Portugal, Jun 2002.

[WJB04] P. Wong, H. Jin, and J. Becker. Load Balancing Multi-zone Applications on a Heterogeneous Cluster with Multi-level Parallelism. In *Proceedings of the 3rd International Symposium on Parallel and Distributed Computing/International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04)*, pages 388–393, Cork, Ireland, Jul 2004.

[WL91a] M. Wolf and M. Lam. A Data Locality Optimizing Algorithm. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'91)*, Toronto, Ontario, Canada, Jun 1991.

[WL91b] M. Wolf and M. Lam. A Loop Transformation Theory and an Algorithm to Maximize Parallelism. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):452–471, Oct 1991.

[WM03] F. Wolf and B. Mohr. Automatic Performance Analysis of Hybrid MPI/OpenMP Applications. In *Proceedings of the 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing (EuroPDP 2003)*, pages 13–22, Genova, Italy, Feb 2003.

[Xue97] J. Xue. Communication-Minimal Tiling of Uniform Dependence Loops. *Journal of Parallel and Distributed Computing*, 42(1):42–59, 1997.

[XW02] J. Xue and W.Cai. Time-Minimal Tiling when Rise is Larger than Zero. *Journal of Parallel Computing*, 28(6):915–939, 2002.

[Zig94] J. Zigman. Perfectly Nested Loops: Communication and Memory Reuse. Master's thesis, Faculty of Engineering and Information Technology, Australian National University, Dec 1994.

[Αθα05] Μ. Αθανασάκη. *Παραλληλοποίηση Κώδικα Βρόχων σε Αρχιτεκτονικές μη Ομοιόμορφης Προσπέλασης Μνήμης (NUMA)*. PhD thesis, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο, Δεκέμβριος 2005.

[Γκο03] Γ. Γκούμας. *Αυτόματη Παραγωγή Παράλληλου Κώδικα για Μετασχηματισμούς Υπερκόμβων*. PhD thesis, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο, Δεκέμβριος 2003.

[Σωτ04] Α. Σωτηρόπουλος. *Αποδοτική Αξιοποίηση Σύγχρονων Δικτυακών Τεχνολογιών στην Παράλληλη Εκτέλεση Υπολογισμών σε Συστοιχίες Υπολογιστών Υψηλών Επιδόσεων*. PhD thesis, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο, Φεβρουάριος 2004.

Ευρετήριο
