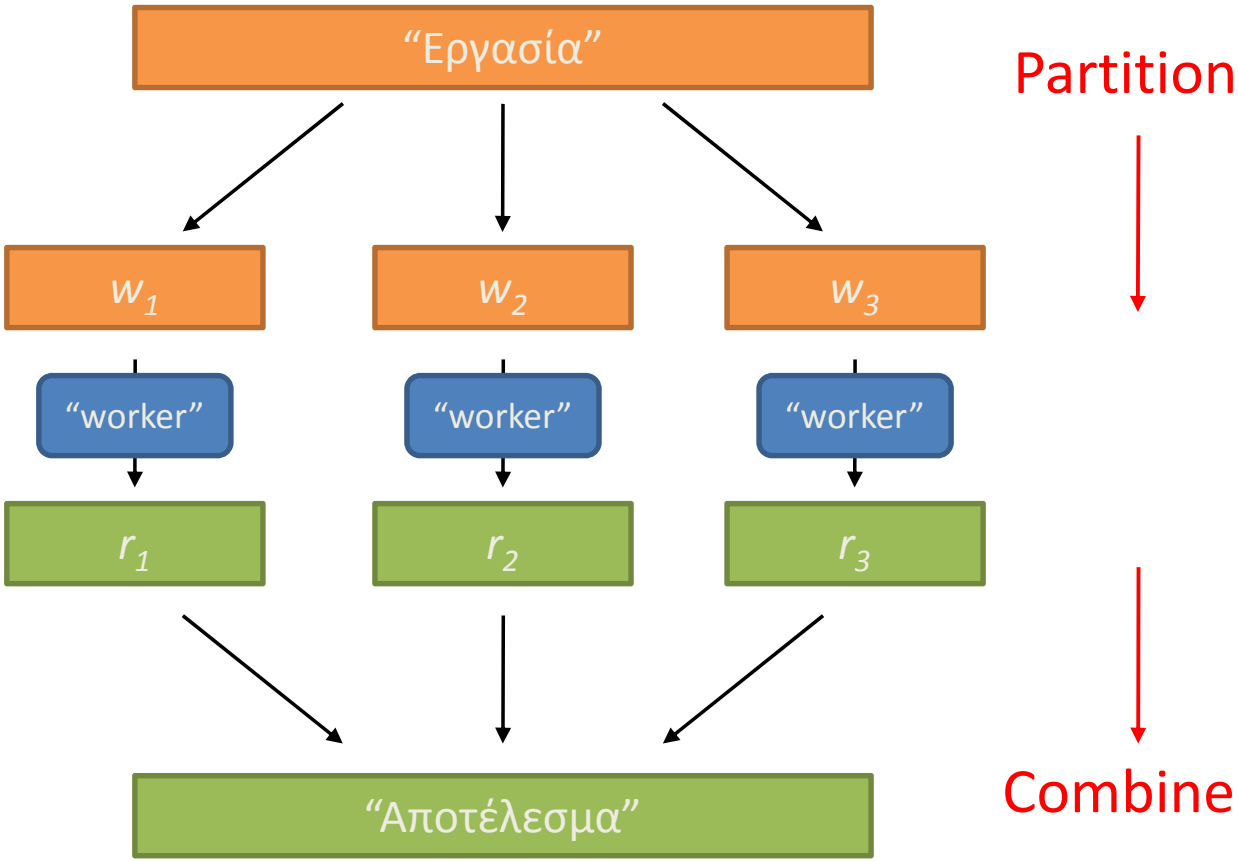
Big Data

Βάσεις Δεδομένων

2016-2017

διαίρει και βασίλευε
(divide and conquer στα ελληνικά)



Προκλήσεις παραλληλοποίησης

- Πως αναθέτουμε μονάδες εργασίας σε workers?
- Αν έχουμε περισσότερες μονάδες εργασίας από workers?
- Εάν οι workers χρειαστεί να μοιραστούν ενδιάμεσα ημιτελή δεδομένα?
- Πως συνοψίζουμε τέτοιου είδους ενδιάμεσα δεδομένα?
- Πως ξέρουμε ότι όλοι οι workers τελειώσανε?
- Τι γίνεται εάν κάποιοι workers διακοπήκανε ?

Τι το κοινό έχουν όλα αυτά τα προβλήματα?

Συγχρονισμός

- Τα προβλήματα παραλληλοποίησης προκύπτουν από:
  - Επικοινωνία μεταξύ workers
  - Πρόσβαση σε κοινόχρηστους πόρους (πχ, δεδομένα)
- Επομένως χρειαζόμαστε μηχανισμούς συγχρονισμού

Source: Ricardo Guimarães Herrmann

Διαχείριση πολλαπλών Workers

- Δύσκολο, καθώς
  - Άγνωστη σειρά εκτέλεσης των workers
  - Δεν γνωρίζουμε πότε ο ένας σταματάει τον άλλο
  - Άγνωστη η σειρά πρόσβασης σε κοινά δεδομένα
- Άρα θέλουμε:
  - Semaphores (lock, unlock)
  - Condition variables (wait, notify, broadcast)
  - Barriers
- Παρόλα αυτά, επιπλέον προβλήματα:
  - Deadlock, livelock, race conditions...
  - Dining philosophers, sleeping barbers, cigarette smokers...

Επομένως

- Η ταυτόχρονη προσπέλαση (Concurrency) είναι δύσκολη
- Πιο δύσκολη σε μεγαλύτερη κλίμακα
  - Σε επίπεδο datacenter (ή μεταξύ datacenters)
  - Όταν έχουμε αστοχίες υλικού/λογισμικού (failures)
  - Όταν έχουμε υπηρεσίες που αλληλεπιδρούν
- Αποσφαλμάτωση?
- Επομένως, στην πράξη (μέχρι πριν το MapReduce):
  - Συγκεκριμένες υλοποιήσεις του ίδιου πράγματος (custom-ιές!)
  - Γράψε την βιβλιοθήκη σου και δούλεψε με αυτή.
  - Ο προγραμματιστής παίρνει το βάρος να προγραμματίσει τα πάντα!

Τι είναι το MapReduce?

- Ένα προγραμματιστικό μοντέλο
- Ένα προγραμματιστικό πλαίσιο
- Για την ανάπτυξη εφαρμογών οι οποίες
  - επεξεργάζονται γρήγορα και παράλληλα τεράστιες ποσότητες δεδομένων
  - Σε συστοιχίες (clusters) υπολογιστών
- Closed-source υλοποίηση Google
  - Scientific papers του ’03 και ’04 που το περιγράφουν
- Hadoop: opensource υλοποίηση των αλγορίθμων που περιγράφονται στα paper
  - http://hadoop.apache.org/

Hadoop HDFS

- Hadoop Distributed File System (HDFS) είναι το πρωτεύον αποθηκευτικό σύστημα που χρησιμοποιείται από όλες τις εφαρμογές Hadoop.
- Το HDFS διασπάει το δεδομένα σε blocks και δημιουργεί αντίγραφα τους σε διαφορετικούς υπολογιστικούς κόμβους για να επιτύχει αξιόπιστους και υπερβολικά γρήγορους υπολογισμούς .
- Φροντίζει για τα αντίγραφα και την τοπικότητα των δεδομένων
- Ξεκίνησε σαν open source υλοποίηση του GFS (Google File System)

Πλεονεκτήματα του Hadoop HDFS

- Κατανεμημένο αποθηκευτικό σύστημα πολύ μεγάλου μεγέθους.
  - 10.000 κόμβοι.
  - 100.000.000 αρχεία.
  - 10PB αποθηκευτικός χώρος.
- Βασίζεται σε φθηνό Hardware.
  - Αντίγραφα ασφαλείας των αρχείων ώστε να αντιμετωπίζονται οι βλάβες.
  - Ανίχνευση βλαβών και ανάκτηση.
- Είναι βελτιστοποιημένο για Batch processing
  - Υπολογισμοί να μεταφέρονται εκεί που βρίσκονται τα δεδομένα
  - Παρέχει πολύ υψηλό συνολικό εύρος ζώνης
- Ο χώρος αποθήκευσης μπορεί να βρίσκεται σε ετερογενή λειτουργικά συστήματα.

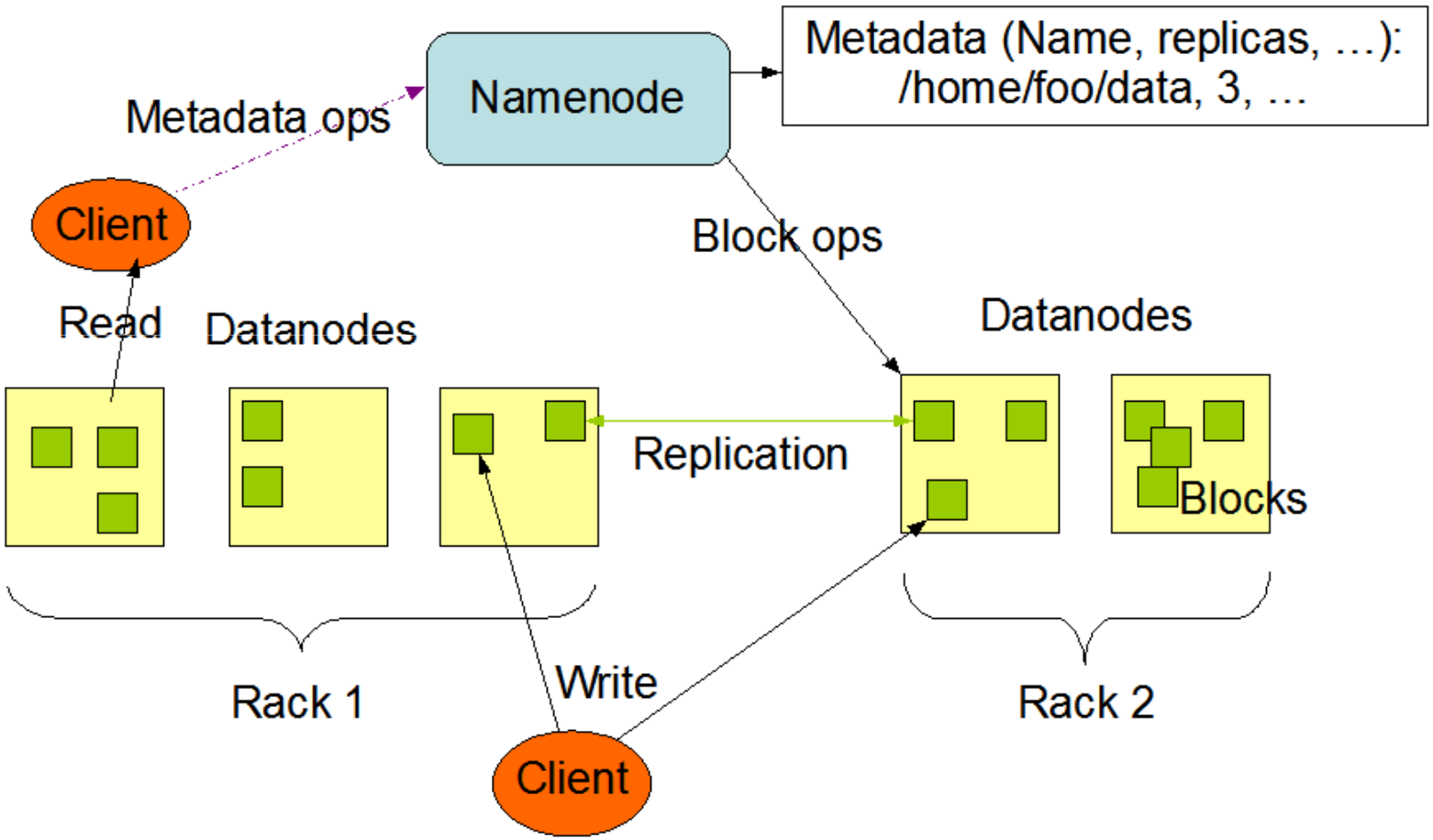
Βασικές αρχές του HDFS

- Ο χώρος των αρχείων είναι ενιαίος για όλο το cluster
- Επιβλέπει την συνέπεια των δεδομένων
  - Βασίζεται στο μοντέλο Write-once-read-many
  - Υποστηρίζεται στα αρχεία μόνο η διαδικασία append
- Τα αρχεία διασπώνται σε blocks
  - Τυπικό μέγεθος block 128 MB.
  - Κάθε block αντιγράφεται σε πολλαπλούς κόμβους δεδομένων (DataNodes).
  - Τα δεδομένα δεν γράφονται απευθείας στο δίσκο. Πρώτα αποθηκεύονται σε buffer.
- Βασίζεται σε έξυπνους πελάτες (Clients).
  - Οι Clients μπορούν να βρουν την τοποθεσία των blocks
  - Οι Client προσπελαύνουν τα δεδομένα απευθείας στους DataNodes

Διεπαφή συστήματος αρχείων

- create/delete
- open/close
- read/write
- Snapshot
- record append

Αρχιτεκτονική



Τι δεν κάνει το HDFS

- Transactional data? (e.g. concurrent reads and writes to the same data)
  - Εδώ το HDFS θα χρειαστεί να αποθηκεύει τα δεδομένα ένα file κάθε εγγραφή.
- Structured data? (e.g. record oriented views, columns)
  - Τα metadata είναι μόνο σε μορφή καταλόγων και ονομάτων αρχείων
- Relational data? (e.g. indexes)
  - Δεν υποστηρίζει αναζητήσεις.
- Ότι δεν κάνει το HDFS το κάνει η HBase (BigTable)...

BigTable

- Το Bigtable αποτελεί ένα κατανεμημένο σύστημα αποθήκευσης για τη διαχείριση μεγάλης ποσότητας ημι-δομημένων δεδομένων και προσανατολισμένο στην κλιμακωσιμότητα (scalability)
- Χρησιμοποιείται από την Google
  - Analytics, Google Earth, web indexing, κλπ
- Κλειστού κώδικα -> HBase
- OSDI'06

Χαρακτηριστικά

- Μεγάλο εύρος εφαρμογών
  - Batch processing εφαρμογές
  - Εφαμογές χαμηλής καθυστέρησης για χρήστες
- Κλιμακωσιμότητα
- Υψηλή απόδοση
- Υψηλή διαθεσιμότητα
- Δυνατότητα χρήσης σε συνδυασμό με MapReduce
- Εκτελείται σε μέσου κόστους υλικό

Μοντέλο δεδομένων

- Είναι ένας αραιός, κατανεμημένος, πολυδιάστατος πίνακας
- Διευθυνσιοδοτείται από:
  - Κλειδί γραμμής
  - Κλειδί στήλης
  - Χρονοσφραγίδα
- Κάτι σαν συντεταγμένες <x,y>
- Κάθε κελί περιέχει ένα σύνολο bytes

(row,column,time) → Value

Γραμμές (rows)

- Το κλειδί απότελείται από ένα αλφαριθμητικό
- Οι ενεργειες πάνω σε μία γραμμή είναι ατομικές
- Λεξικογραφική ταξινόμηση με βάση τα κλειδιά
- Όλος ο πίνακας αποτελείται από (δισ/τρισ/κλπ)εκατομμύρια λεξικογραφικά ταξινομημένες γραμμές.
- Προσοχή: το row key είναι το **μόνο** πεδίο που γίνεται indexed στον BigTable
  - Αναζήτηση σε όλα τα άλλα πεδία γίνεται με full table scan

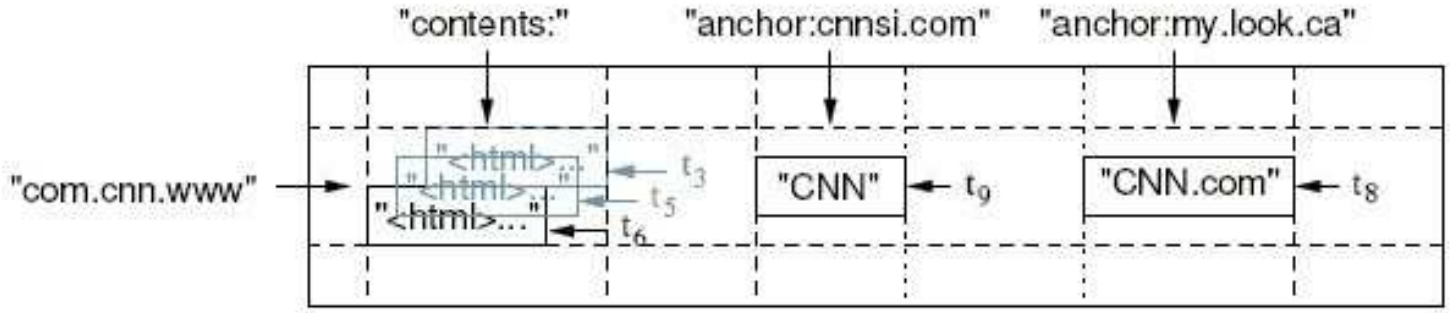
Στήλες (columns)

- Ομαδοποίηση σε column families. Σπάσιμο σε column families ανάλογα το application
- Μικρός αριθμός από column families (πχ ~100)
- Άπειρος αριθμός από columns
- Μορφή family:qualifier
- Ο έλεγχος πρόσβασης γίνεται με βάση τα column families

Χρονοσφραγίδες (timestamps)

- Πολλαπλές εκδόσεις των ίδιων δεδομένων
- Πραγματικός χρόνος ή
- Καθορισμένος από το χρήστη
- Οι πιο πρόσφατες εκδόσεις είναι ευκολότερα προσβάσιμες
- Ρύθμιση για την διατήρηση των
  - Τελευταίων Χ εκδόσεων **ή**
  - Όλες τις εκδόσεις των τελευταίων Χ εβδομάδων

Παράδειγμα



- rowkey: URL
  - Γιατί είναι ανάποδα γραμμένο?
  - Π.χ. gr.ntua.www
    gr.ntua.cslab.www
    gr.ntua.dblab.www
- Column families
  - Contents: Χωρίς column id. Το value είναι τα html contents (πολλές εκδόσεις)
  - Anchor: Έχει column id το url του link. Value είναι το κείμενο του link.
- Ερώτηση: πως μπορώ να βρω όλες τις στήλες των οποίων το όνομα είναι cnnsi.com?

Αρχιτεκτονική HDFS/MapReduce

- Αρχιτεκτονική Master/Slave
  - Ένας κεντρικός JobTracker διαχειρίζεται πολλαπλούς TaskTrackers

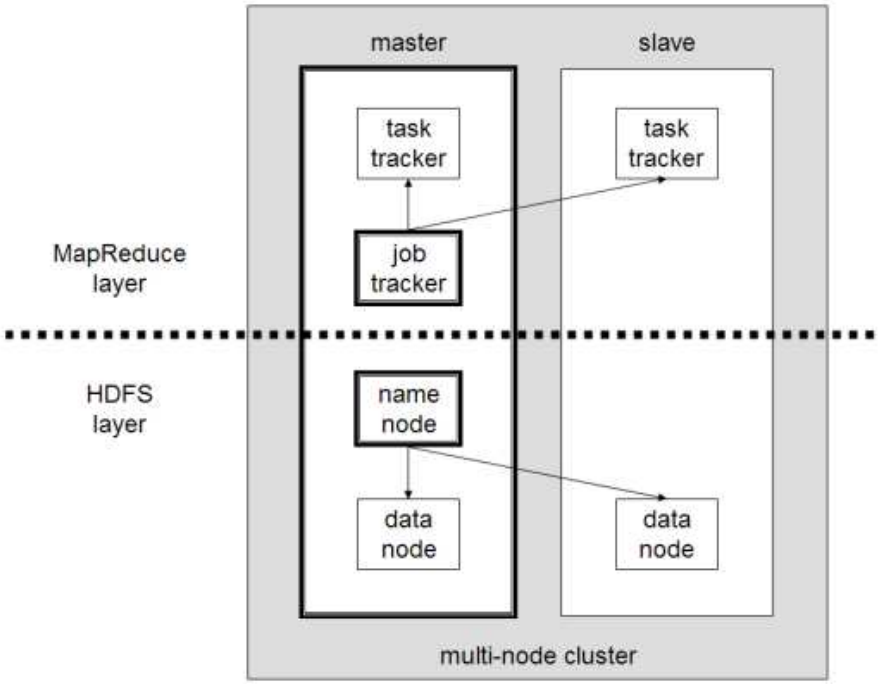
-NameNode και JobTracker

τρέχουν στον master

-DataNode και TaskTracker

τρέχουν στους slaves

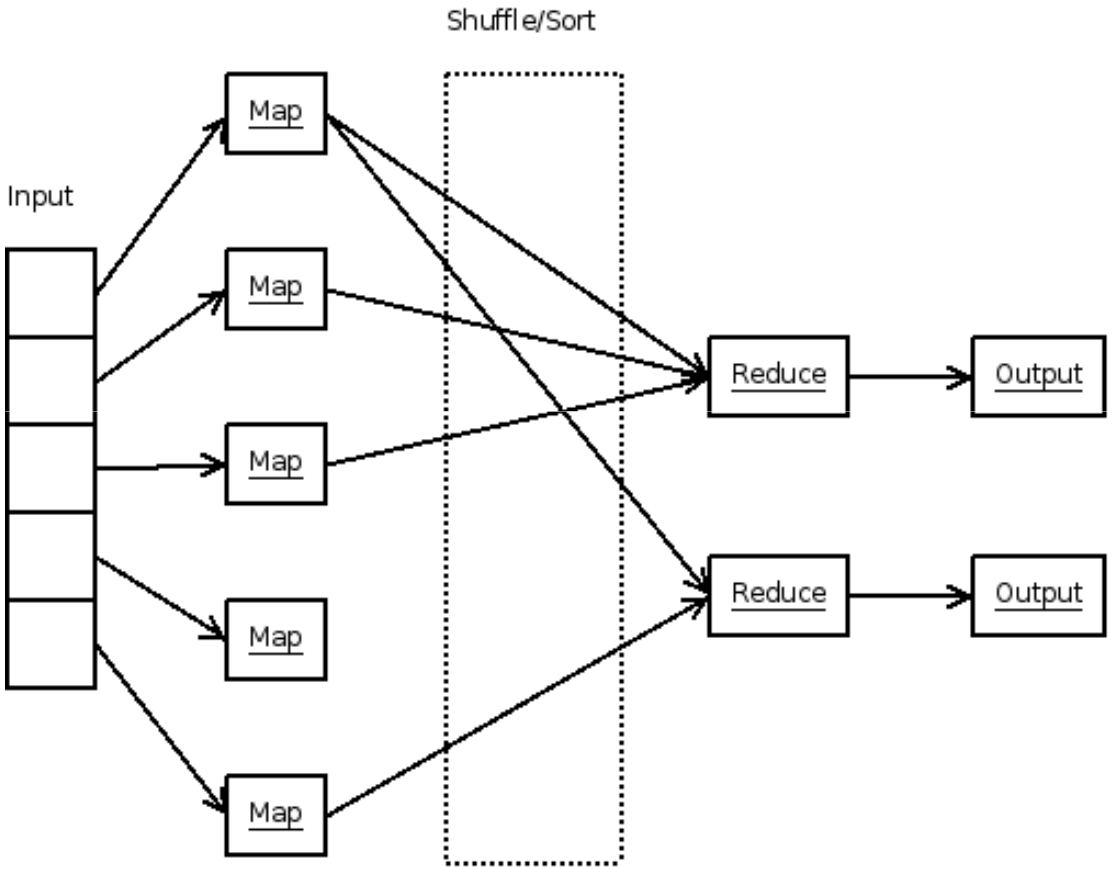
- Data locality



MapReduce

- Το πρόβλημα “σπάει” σε 2 φάσεις, την Map και την Reduce
- **Map**: Μη αλληλο-επικαλυπτόμενα κομμάτια από δεδομένα εισόδου (εγγραφές <key,value>) ανατίθενται σε διαφορετικές διεργασίες (mappers) οι οποίες βγάζουν ένα σετ από ενδιάμεσα <key,value> αποτελέσματα
- **Reduce**: Τα δεδομένα της Map φάσης τροφοδοτούνται σε ένα συνήθως μικρότερο αριθμό διεργασιών (reducers) οι οποίες “συνοψίζουν” τα αποτελέσματα εισόδου σε μικρότερο αριθμό <key,value> εγγραφών

MapReduce



Πότε είναι χρήσιμο?

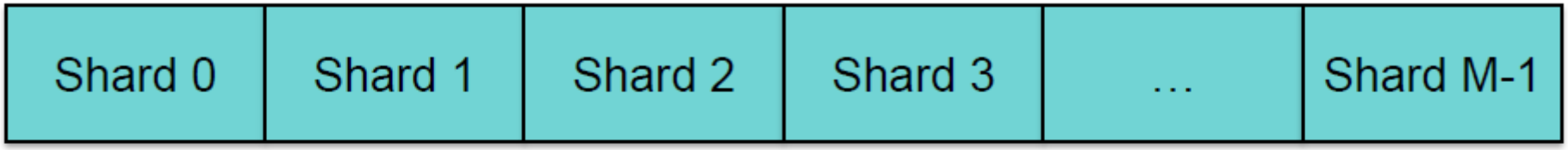
- Καλή επιλογή για:
  - Δεικτοδότηση/ανάλυση log αρχείων
  - Ταξινόμηση μεγάλου όγκου δεδομένων
  - Ανάλυση εικόνων
- Κακή επιλογή για:
  - Υπολογισμός ακολουθιών Fibonacci
  - Αντικατάσταση της MySQL

Πριν ξεκινήσουμε

- Η είσοδος ανεβαίνει στο HDFS “χωρίζεται” σε M κομμάτια, μεγέθους 128 MB
  - Κάθε κομμάτι περιέχει «ζεύγη» εγγραφών <key,value>
- Κάθε μηχάνημα που συμμετέχει στον υπολογισμό εκτελεί ένα αντίγραφο του προγράμματος σε ένα κομμάτι των δεδομένων
- Ένα από όλα τα μηχανήματα αναλαμβάνει το ρόλο του master. Αυτός αναθέτει εργασίες στα υπόλοιπα (εργάτες). Αυτές μπορεί να είναι map ή reduce εργασίες.

Βήμα 1ο: Διαίρεση της εισόδου σε shards

- Η είσοδος διαιρείται σε M κομμάτια των 128MB

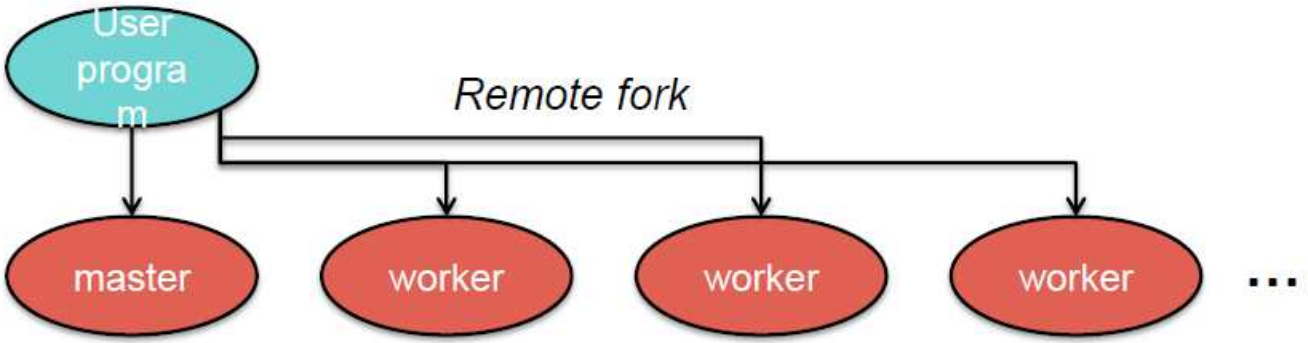


Input files

Divided into *M* shards

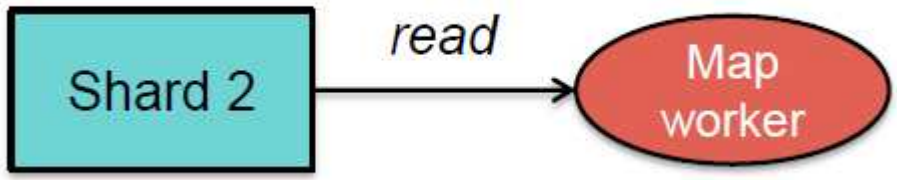
Βήμα 2ο: Fork διεργασιών

- Ξεκινά πολλά αντίγραφα του προγράμματος σε cluster υπολογιστών
  - 1 master
  - Πολλοί workers
- Στους idle workers ανατίθενται
  - Map tasks (καθένα σε ένα shard): M map tasks
  - Reduce tasks (καθένα δουλεύει σε ενδιάμεσα αποτελέσματα): R reduce tasks



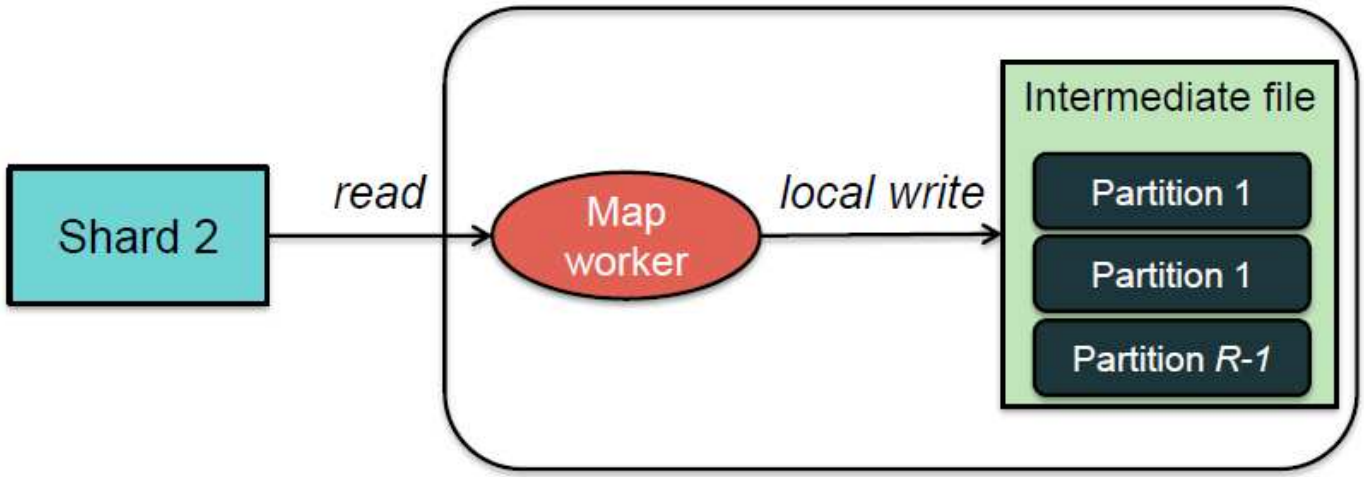
Βήμα 3ο: Map εργασία

- Για έναν εργάτη που του έχει ανατεθεί μία map εργασία
  - Διαβάζει από το HDFS το κομμάτι της εισόδου(input split) που του αντιστοιχεί, αναλύει τα ζεύγη <key, value> που προκύπτουν και τα δίνει σαν είσοδο στη map συνάρτηση.
  - Η map συνάρτηση επεξεργάζεται τα ζεύγη και παράγει ενδιάμεσα ζεύγη και τα συσσωρεύει στη μνήμη.



Βήμα 4ο:

- Τα ενδιάμεσα <key, value> που παράγει ο mapper γράφονται σε buffer στη μνήμη και αποθηκεύονται περιοδικά στον τοπικό δίσκο
  - Διαιρούνται σε R regions από μια συνάρτηση διαίρεσης (partitioning function)

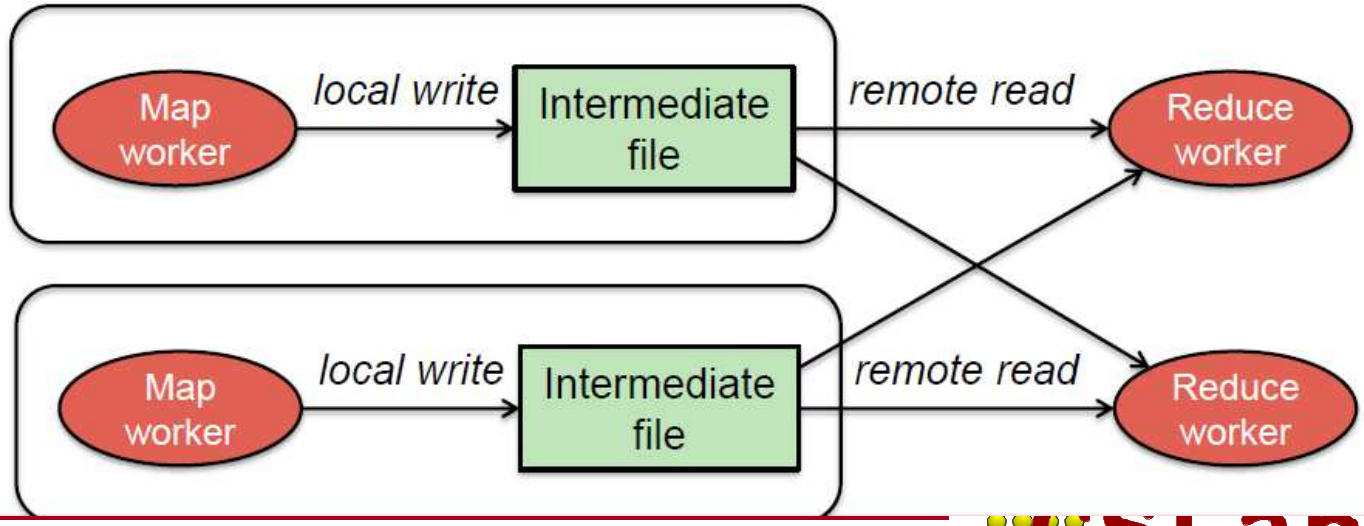


Συνάρτηση διαίρεσης

- Εκτελείται περιοδικά και αποθηκεύει τα ενδιάμεσα ζεύγη στον τοπικό δίσκο
- Χωρίζει τα κλειδιά σε R ομάδες -> αποφασίζει ποιος από τους R reducers θα επεξεργαστεί ποιο ενδιάμεσο κλειδί
  - Default: hash(key) mod R
  - User defined (π.χ. λεξικογραφικά)
- Όταν η συνάρτηση διαίρεσης ολοκληρώσει την αποθήκευση των ζευγών ενημερώνει τον master για το που βρίσκονται τα δεδομένα.
- Ο master προωθεί αυτή την πληροφορία στους εργάτες που εκτελούν reduce εργασίες (για να πάρουν το partition που τους αναλογεί από τον κάθε worker)

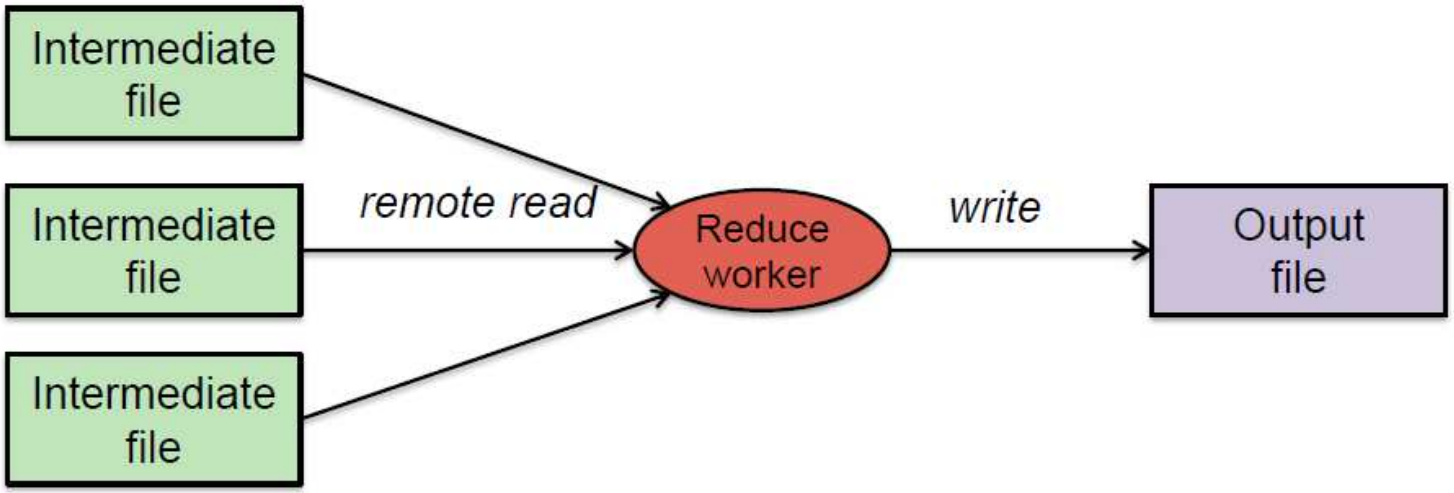
Βήμα 5ο: Reduce task - sorting

- Διαβάζει από κάθε εργάτη που έχει εκτελεσθεί τα ζεύγη που του αντιστοιχούν από τις τοποθεσίες που του υποδεικνύει ο master.
- Όταν όλα τα ενδιάμεσα ζεύγη έχουν ανακτηθεί ταξινομούνται βάση του key
- Όσα values έχουν κοινό key ομαδοποιούνται



Βήμα 6ο:

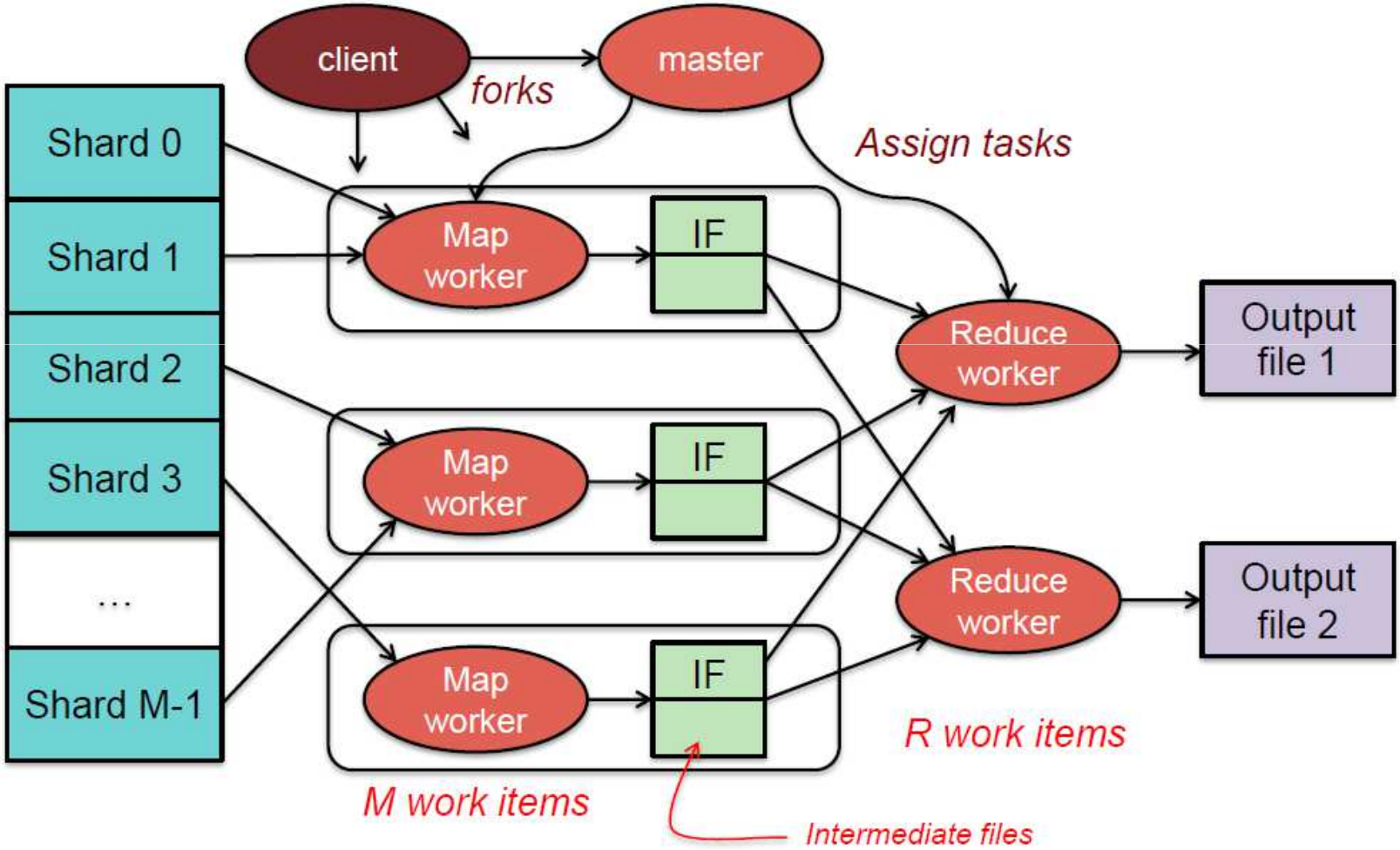
- Εκτελείται η συνάρτηση reduce με είσοδο τα ζεύγη <key, group_of_values> που προέκυψαν στην προηγούμενη φάση
- Η reduce επεξεργάζεται τα δεδομένα εισόδου και παράγει τα τελικά ζεύγη
- Τα ζεύγη εξόδου προσαρτώνται σε ένα αρχείο στο τοπικό σύστημα αρχείων.
- Όταν ολοκληρωθεί η reduce το αρχείο γίνεται διαθέσιμο στο κατανεμημένο σύστημα αρχείων



Βήμα 7ο: Ολοκλήρωση εργασιών

- Όταν ένας εργάτης ολοκληρώσει την εργασία του ενημερώνει τον master.
- Όταν όλοι ενημερωσουν τον master τότε αυτός επιστρέφει τη λειτουργία στο αρχικό πρόγραμμα του χρήστη.

Η μεγάλη εικόνα



Παράδειγμα: Μέτρηση λέξεων 1/3

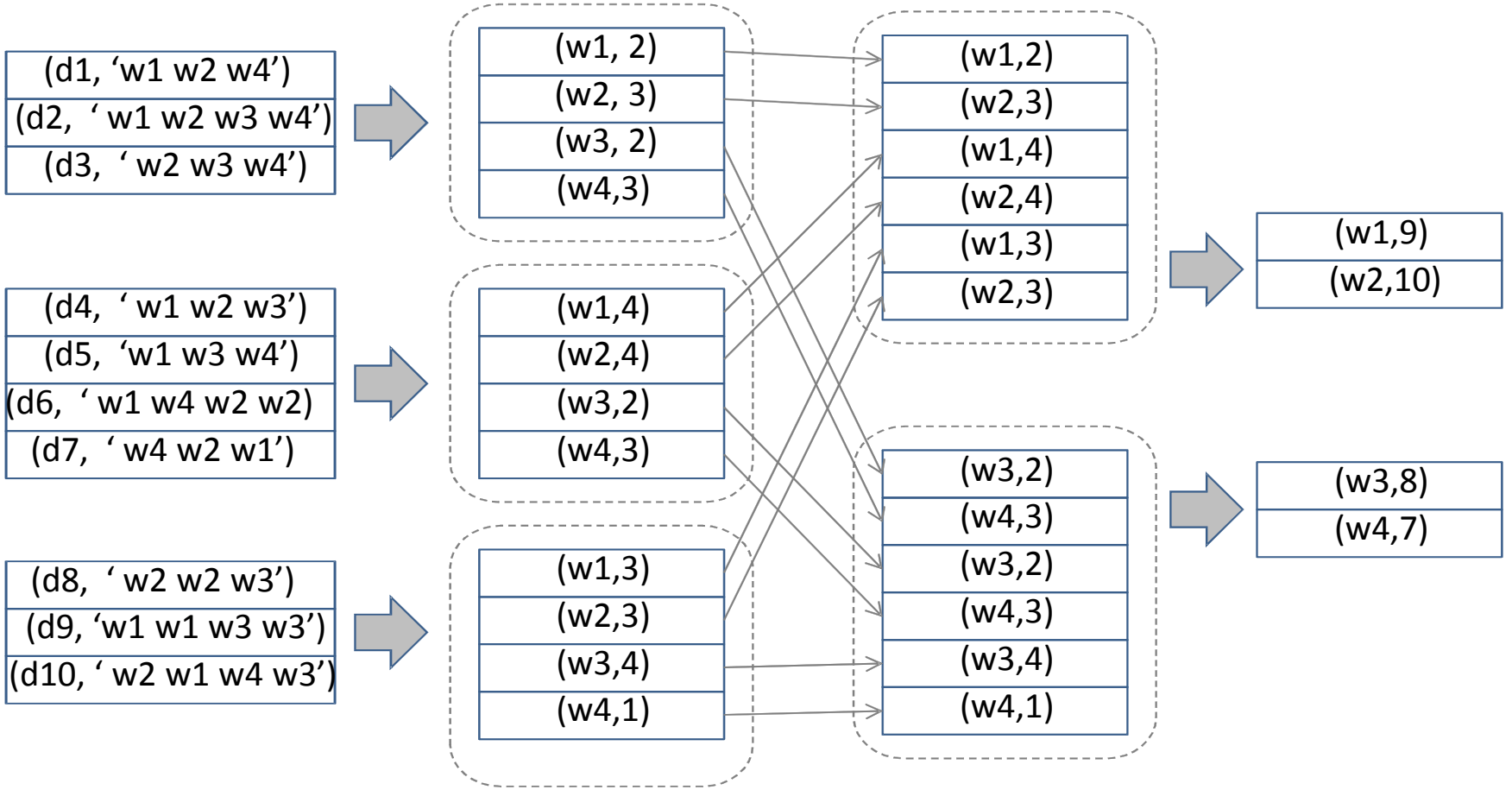
- Στόχος: μέτρηση της συχνότητας εμφάνισης λέξεων σε ένα μεγάλο σύνολο κειμένων
- Πιθανή χρήση: Εύρεση δημοφιλών url σε webserver logfiles
- Πλάνο υλοποίησης:
  - “Ανέβασμα” των κειμένων στο κατανεμημένο File System
  - Γράφω μια map και μια reduce συνάρτηση
  - Τρέχω μια MapReduce εργασία
  - Παίρνω πίσω τα αποτελέσματα

Παράδειγμα: Μέτρηση λέξεων 2/3

```
map(key, value):
// key: document name; value: text of document
    for each word w in value:
        emit(w, 1)

reduce(key, values):
// key: a word; value: an iterator over counts
        result = 0
        for each count v in values:
            result += v
        emit(result)
```


Παράδειγμα: Μέτρηση λέξεων 3/3



M=3 mappers

R=2 reducers

Ανοχή στα σφάλματα

- Ο master επικοινωνεί με τους εργάτες περιοδικά. Εάν κάποιος δεν ανταποκριθεί για ένα χρονικό διάστημα τότε αναθέτει την εργασία του σε κάποιον άλλο.
- Τα ενδιάμεσα αποτελέσματα που παράγονται από τις map και reduce εργασίες διατηρούνται σε προσωρινά αρχεία σε τοπικά συστήματα αρχείων έως ότου όλη η είσοδος να έχει υποστεί επεξεργασία. Στη συνέχεια ενημερώνεται ο master και η πληροφορία γίνεται διαθέσιμη σε όλους.

Τοπικότητα

- Τα δεδομένα αποθηκευονται στους δίσκους των εργατών.
- Χωρίζονται σε block (64MB συνήθως) με αντίγραφα σε άλλους εργάτες.
- Move computation near the data: Ο master προσπαθεί να εκτελέσει μία εργασία σε ένα εργάτη “κοντά” στα δεδομένα εισόδου, ώστε να μειωθεί το εύρος δικτύου που θα καταναλωθεί.

Διακριτότητα εργασιών

- Ο αριθμός των προς εκτέλεση εργασιών είναι συνήθως μεγαλύτερος από το πλήθος των διαθέσιμων εργατών
- Ένας εργάτης μπορεί να εκτελέσει περισσότερες από μία εργασίες
- Έτσι η ισορροπία φόρτου βελτιώνεται και σε περίπτωση που υπάρξει βλάβη σε έναν εργάτη υπάρχει γρηγορότερη ανάρρωση με την ανακατανομή των εργασιών του σε άλλους

Εφεδρικές εργασίες

- Μερικές εργασίες καθυστερούν την ολοκλήρωση τους και μαζί και την ολοκλήρωση της συνολικής δουλειάς
- Η λύση στο πρόβλημα είναι η δημιουργία αντιγράφων της εργασίας (speculative execution)
- Μία εργασία θεωρείται ολοκληρωμένη όταν ενημερώσει τον master αυτή ή ένα αντίγραφο της

Partitioning-combining

- Ένας χρήστης μπορεί να ορίσει μία δική του συνάρτηση διαίρεσης κατά το shuffling.
- Μία συνάρτηση combiner μπορεί να οριστεί για να επεξεργαστεί τα δεδομένα εξόδου μίας εργασίας map πριν αυτά γίνουν διαθέσιμα στους reducers. Εκτελείται από τον ίδιο εργάτη που εκτελεί τη map εργασία και συνήθως είναι παρόμοια με τη συνάρτηση reduce
- Ο τύπος των δεδομένων εισόδου και εξόδου μπορεί να καθοριστεί από το χρήστη και δεν έχει περιορισμούς του τι μορφής μπορεί να είναι.

Partitioning

- HashPartitioner: Typical “vanilla” partitioner
  - Δίκαιος, αλλά δεν διατηρεί συνολική ταξινόμηση
- TotalOrder Partitioner: διατηρεί την συνολική ταξινόμηση των ενδιάμεσων αποτελεσμάτων
  - Αρκετά άδικος σε περιπτώσεις ανομοιόμορφων κατανομών

Συγκρίνοντας RDBMS και MapReduce

	Traditional RDBMS	MapReduce
Data Size	Gigabytes (Terabytes)	Petabytes (Exabytes)
Access	Interactive and Batch	Batch
Updates	Read / Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High (ACID)	Low
Scaling	Nonlinear	Linear
DBA Ratio	1:40	1:3000

Hadoop

- Open-source υλοποίηση του MapReduce.
- Java
- HDFS
- http://hadoop.apache.org/
- http://wiki.apache.org/hadoop/
- Ποιοι το χρησιμοποιούν:
  - Yahoo!
  - Amazon
  - Facebook
  - Twitter
  - και πολλοί άλλοι...

Use cases 1/3

The New York Times

- Large Scale Image Conversions
- 100 Amazon EC2 Instances, 4TB raw TIFF data
- 11 Million PDF in 24 hours and 240$

facebook

- Internal log processing
- Reporting, analytics and machine learning
- Cluster of 1110 machines, 8800 cores and 12PB raw storage
- Open source contributors (Hive)

twitter

- Store and process tweets, logs, etc
- Open source contributors (hadoop-lzo)

Use cases 2/3

- 100.000 CPUs in 25.000 computers
- Content/Ads Optimization, Search index
- Machine learning (e.g. spam filtering)
- Open source contributors (Pig)

- Natural language search (through Powerset)
- 400 nodes in EC2, storage in S3
- Open source contributors (!) to HBase

- ElasticMapReduce service
- On demand elastic Hadoop clusters for the Cloud

Use cases 3/3

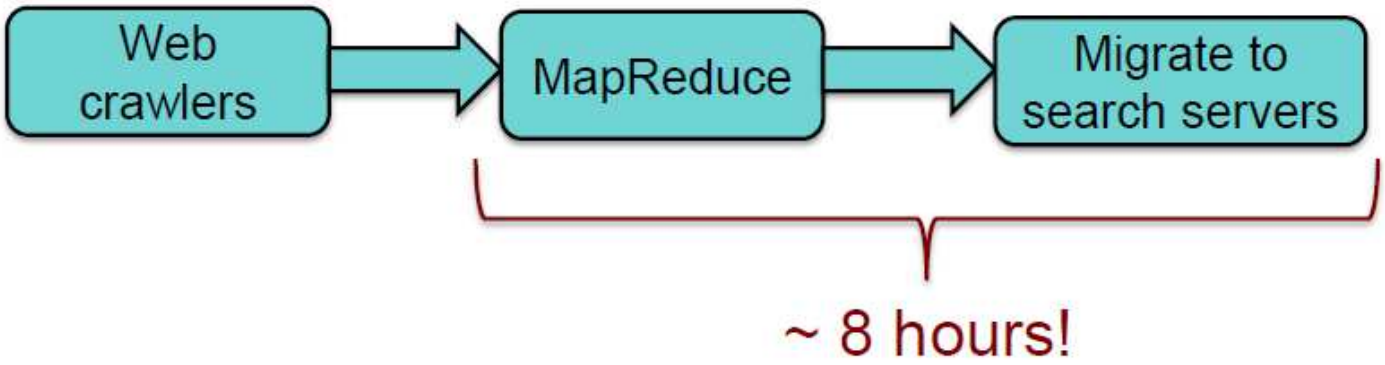
- ETL processing, statistics generation
- Advanced algorithms for behavioral analysis and targeting

- Used for discovering People you May Know, and for other apps
- 3X30 node cluster, 16GB RAM and 8TB storage

- Leading Chinese language search engine
- Search log analysis, data mining
- 300TB per week
- 10 to 500 node clusters

Δεν είναι όλα τέλεια…

- MapReduce was used to process webpage data collected by Google's crawlers.
  - It would extract the links and metadata needed to search the pages
  - Determine the site's PageRank
- The process took around eight hours.
  - Results were moved to search servers.
  - This was done continuously.



Στην πράξη

- Web has become more dynamic
  - an 8+ hour delay is a lot for some sites
- Goal: refresh certain pages within seconds
- MapReduce
  - Batch-oriented
  - Not suited for near-real-time processes
  - Cannot start a new phase until the previous has completed
    - Reduce cannot start until all Map workers have completed
  - Suffers from “stragglers” – workers that take too long (or fail)
- MapReduce is still used for many Google services
- Most data not simple files
  - B-trees, tables, SQL databases, memory-mapped key-values

Περισσότερες πληροφορίες

- Dean, Jeff and Ghemawat, Sanjay. **MapReduce: Simplified Data Processing on Large Clusters** http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf