MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean, Sanjay Ghemawat

Commun. ACM 51(1), 2008 and *OSDI 2004*

Google LABS

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Επισκόπηση

- Η ανάγκη:
  - χρόνος επεξεργασίας πάρα πολύ μεγάλων όγκων δεδομένων (>>1TB)
  - μαζικές παράλληλες μηχανές (εκατοντάδες ή και χιλιάδες CPUs)
  - *εύκολη χρήση*: εκτέλεση υψηλού επιπέδου εφαρμογών και αποδέσμευση προγραμματιστών από πολύπλοκες υλοποιήσεις

- Η προτεινόμενη λύση:
  - παράλληλες αναγνώσεις (reads) → επεξεργασία όγκου δεδομένων σε ομάδες (clusters) από υπολογιστές γενικής χρήσης (commodity hardware)

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

MapReduce framework

1. Ο χρήστης καθορίζει μία map συνάρτηση, η οποία επεξεργάζεται ένα ζεύγος key/value, πχ (filename, line)
2. Παράγεται ένα σύνολο από ενδιάμεσα ζεύγη key/value
3. Ο χρήστης καθορίζει μία reduce συνάρτηση η οποία συγχωνεύει όλες τις ενδιάμεσες τιμές που σχετίζονται με το ίδιο key/value

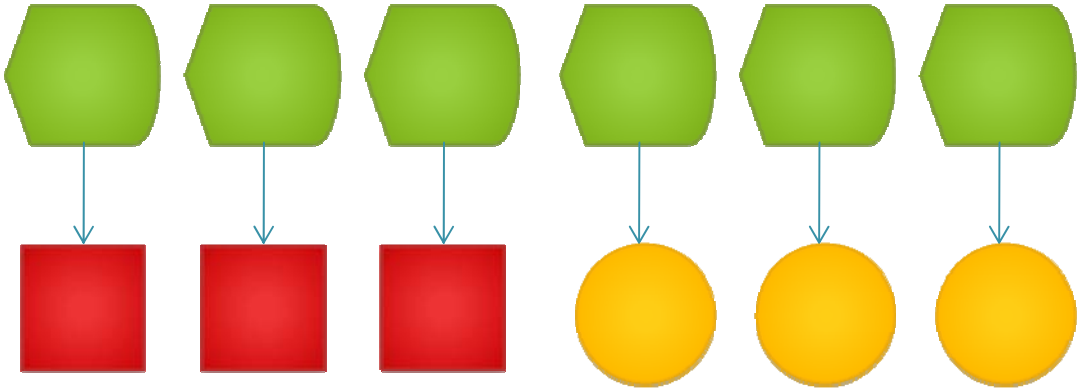
→ Προγράμματα που γράφονται με αυτό τον τρόπο εκτελούνται σε ένα μεγάλο cluster *αυτόματα* και *παράλληλα.*

Programming Model

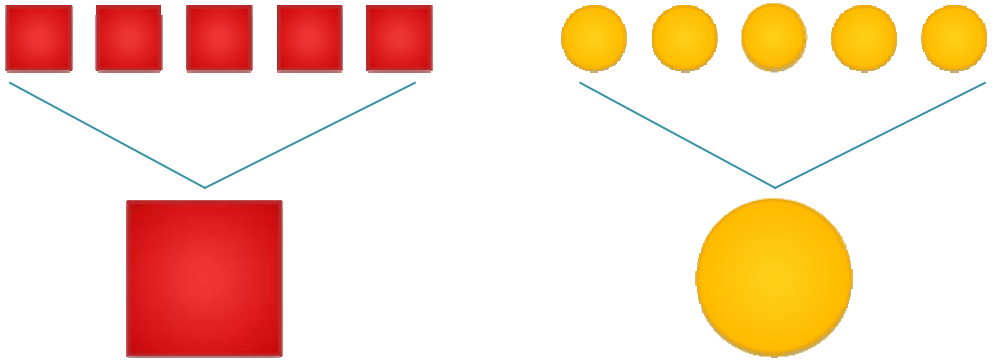
Το περιβάλλον διεπαφής των χρηστών αποτελείται από δύο συναρτήσεις:

– map (in_key, in_value) → (out_key, intermediate_value) list
– reduce (out_key, intermediate_value list) → out_value list

- **Map**



- **Reduce**



Google LABS

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

Παραδείγματα (1)

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
// key: a word
// values: a list of counts int result = 0;
for each v in values:
        result += ParseInt(v);
Emit(AsString(result));
```

MAP						
key	TO	BE	OR	NOT	TO	BE
value	1	1	1	1	1	1

REDUCE				
key	TO	BE	OR	NOT
value	2	2	1	1

Πηγή: Language Learning & Technology
http://llt.msu.edu/vol12num1/emerging/default.html

Παράδειγμα (2)

- Distributed Grep
  - Map → εκπέμπει μία γραμμή (line) αν ταιριάζει με ένα πρότυπο από εκείνα που παρέχονται
  - Reduce → λειτουργία ταυτοποίησης που απλά αντιγράφει τα ενδιάμεσα δεδομένα - που παρέχονται - στην έξοδο

  Παράδειγμα: `grep apple fruitlist.txt`

  Το grep θα επέστρεφε σε αυτή την περίπτωση όλες τις γραμμές του αρχείου fruitlist.txt που θα περιείχαν τουλάχιστον μια εμφάνιση της λέξης 'apple'
- Count of URL Access Frequency 
  - Map → επεξεργάζεται logs από web pages αιτήσεις και εξάγει το `<URL,1>`
  - Reduce → προσθέτει όλες τις τιμές για την ίδια διεύθυνση URL και εκπέμπει ένα ζεύγος `<URL, total count>`
- ReverseWeb-Link Graph
  - Map → εξάγει ζεύγη `<target, source>` για κάθε link σε ένα στόχο (target) URL που βρέθηκε σε μία σελίδα ονομαζόμενη πηγή (source)
  - Reduce → συνδέει αλυσιδωτά τη λίστα όλων των διευθύνσεων URL που σχετίζονται με τη συγκεκριμένη διεύθυνση URL και εκπέμπει το ζεύγος: `<target, list (source)>`

Παράδειγμα χρήσης (1)

- Επεξεργασία αρχείου Web Log για τον υπολογισμό των hits/min για χρονικό διάστημα μιας εβδομάδας.
  - *It scales! το ίδιο πρόγραμμα θα κάνει scale για μήνες, χρόνια, κλπ απλά αυξάνοντας το cluster size*

- Πιο αναλυτικά…
  - Κάθε αλγόριθμος που θα υλοποιηθεί με MapReduce πρέπει να σχεδιαστεί ώστε να έχει 2 φάσεις επεξεργασίας:
    - *Φάση **Map***
    - *Φάση **Reduce***
  - Κάθε φάση έχει ζευγάρια *(key, value)* ως είσοδο και ως έξοδο.

Παράδειγμα χρήσης (2)

- Έστω το παρακάτω εβδομαδιαίο Web log αρχείο

```
192.168.0.5 – [22/May/2009:22:07:52 +0000] “GET /HTTP/1.1” 200 1722
192.168.0.5 – [22/May/2009:22:07:52 +0000] “GET /styles/layout.css  HTTP/1.1” 200 2187
192.168.0.5 – [22/May/2009:22:08:00 +0000] “GET /projects.html  HTTP/1.1” 200 4024
```

- Φάση **Map**:
  - Για κάθε γραμμή...
    - *απομονώνει τη χρονική πληροφορία,*
    - *τη μετατρέπει σε minute-in-week slot,*
    - *επιστρέφει το ζευγάρι (<minute-in-week slot>, 1)*
  - Αποτελέσματα φάσης Map (<minute-in-week slot>, 1)
    - *(1387, 1)*
    - *(1387, 1)*
    - *(1388, 1) κ.λ.π.*
    - *... (22:07 στις 22 Μαΐου 2009 είναι το 1381στό λεπτό εκείνης της εβδομάδας)*

Παράδειγμα χρήσης (3)

- Φάση **Reduce**:
  - προσθέτει το 1 στις τιμές που εκπέμπονται από τη Map, με σκοπό την παραγωγή των συνόλων:

```
<1387, (1, 1)> -> <1387, 2>
<1388, (1)>    -> <1388, 1>
```

Για κάθε <minute-in-week slot> key και έναν iterator υπολόγισε το άθροισμα των values του key.

  - Το output δημοσιεύεται ως ένα στηλοθετημένο ξεχωριστό αρχείο. Το περιεχόμενό του μας ενημερώνει πως υπήρξαν δύο αιτήσεις στο 1387 λεπτό της εβδομάδας (minute-in-week slot), και μία στο 1388 λεπτό - που είναι σωστή.

```
1387   2
1388   1
```

Πηγή: Amazon Web Services
http://developer.amazonwebservices.com/connect/entry.jspa?externalID=873

Google LABS

Παραδείγματα (3)

- Term-Vector per Host
  - Map → ένα διάνυσμα όρου συνοψίζει τις πιο σημαντικές λέξεις που εμφανίζονται σε ένα έγγραφο ή ένα σύνολο εγγράφων ως μία λίστα από `<word, frequency>` ζεύγη. Η map() εκπέμπει ένα ζεύγος `<hostname, term vector>` σε κάθε input document (όπου το hostname εξάγεται από το URL του document)
  - Reduce → περνάει από όλα τα pre-document διανύσματα όρων για ένα δοθέντα host. Προσθέτει αυτά τα διανύσματα όρων μαζί, απομακρύνει τους σπάνιους όρους, και έπειτα εκπέμπει ένα τελικό ζεύγος `<hostname, term vector>`
- Inverted Index
  - Map → αναλύει κάθε document και εκπέμπει μία ακολουθία από `<word, document ID>`
  - Reduce → δοθείσας μιας λέξης δέχεται όλα τα ζεύγη, ταξινομεί τα αντίστοιχα IDs των documents και εκπέμπει ένα ζεύγος `<word, list (document ID)>`
- Distributed Sort
  - Map → εξάγει το key από κάθε εγγραφή και εκπέμπει ένα `<key, record>` ζεύγος
  - Reduce → εκπέμπει όλα τα ζεύγη που δεν έχουν αλλάξει

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

Εφαρμογή

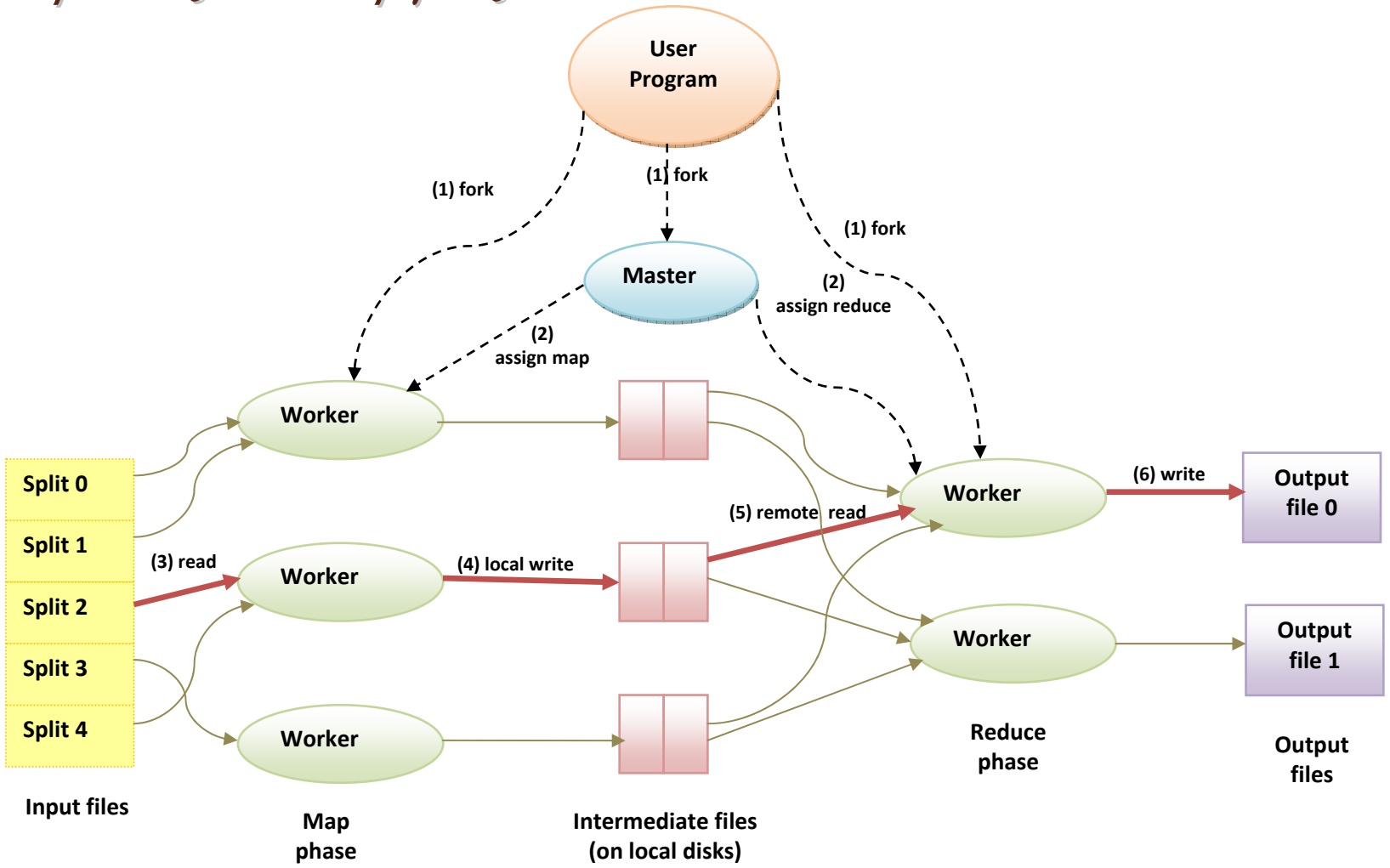
Τεχνικά χαρακτηριστικά συστήματος (Google)

- Μεγάλα clusters από commodity PCs συνδεδεμένα μαζί μέσω Ethernet
  - Μηχανές→ dual-processor x86 επεξεργαστές που τρέχουν Linux, με 2-4 GB μνήμη ανά μηχανή.
  - Χρήση διαφόρων networking hardware– τυπικά ή 100 Mbit/sec ή 1 Gbit/sec σε επίπεδο μηχανής, αλλά κατά μέσο όρο πολύ λιγότερο το συνολικό διχοτομημένο bandwidth.
  - Ένας cluster αποτελείται από εκατοντάδες ή χιλιάδες μηχανές, και γι αυτό η αποτυχίες μηχανών είναι συνήθης.
  - Αποθήκευση που παρέχεται από χαμηλού κόστους IDE δίσκους συνδεδεμένοι απευθείας με τα μηχανήματα. Ένα κατανεμημένο σύστημα αρχείων που αναπτύχθηκε εσωτερικά χρησιμοποιήθηκε για τη διαχείριση των αποθηκευμένων δεδομένων σε αυτούς τους δίσκους. Το σύστημα χρησιμοποιεί αντίγραφα αρχείων για να παρέχει διαθεσιμότητα και αξιοπιστία στην κορυφή του αναξιόπιστου υλικού.
  - Οι χρήστες υποβάλουν εργασίας σε ένα σύστημα προγραμματισμού. Κάθε εργασία αποτελείται από ένα σύνολο καθηκόντων (tasks), και αντιστοιχίζεται με το χρονοδιάγραμμα για το σύνολο των διαθέσιμων μηχανών εντός ενός συμπλέγματος (cluster).

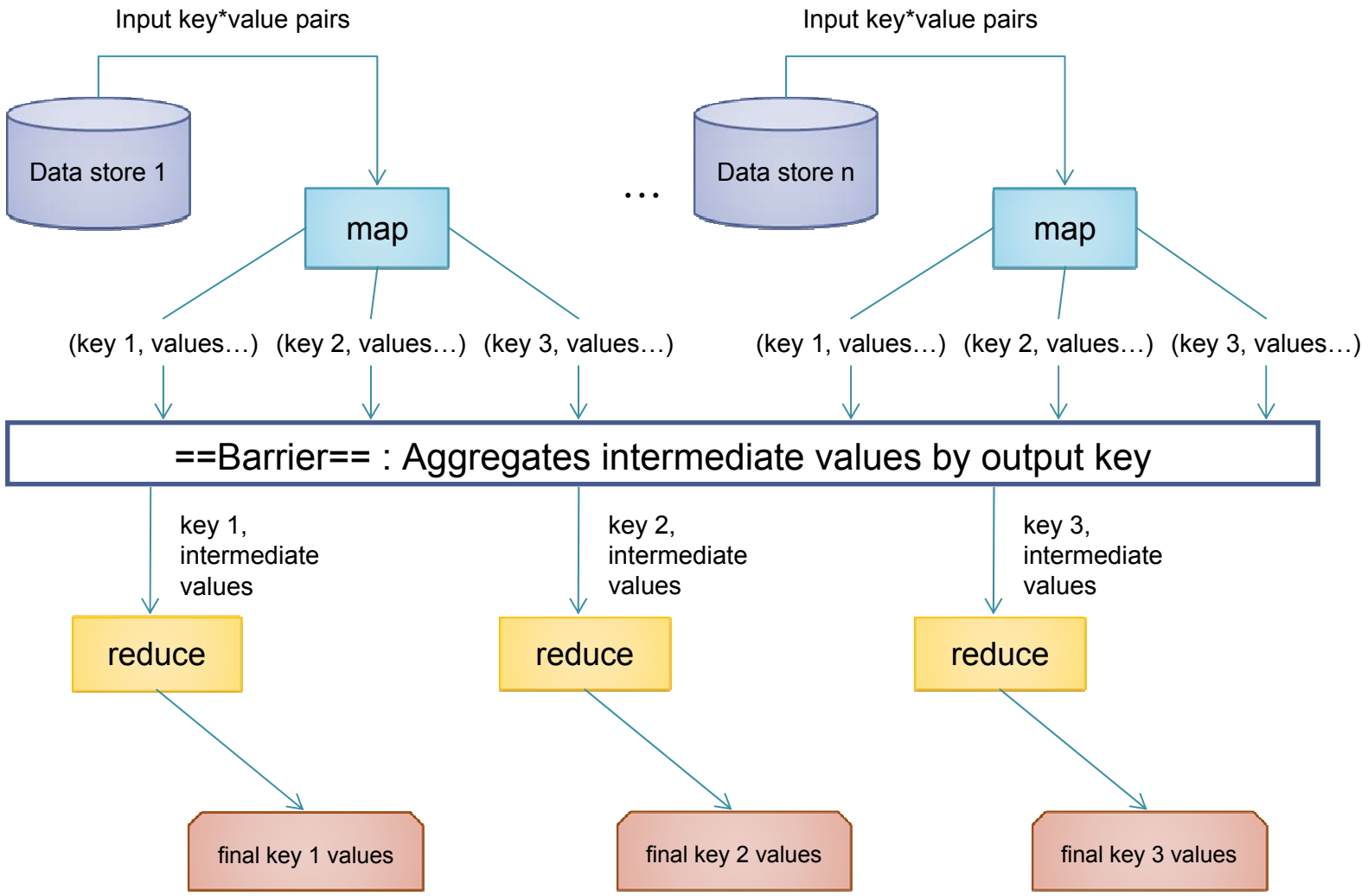
Google LABS

Εφαρμογή

Τρόπος λειτουργίας

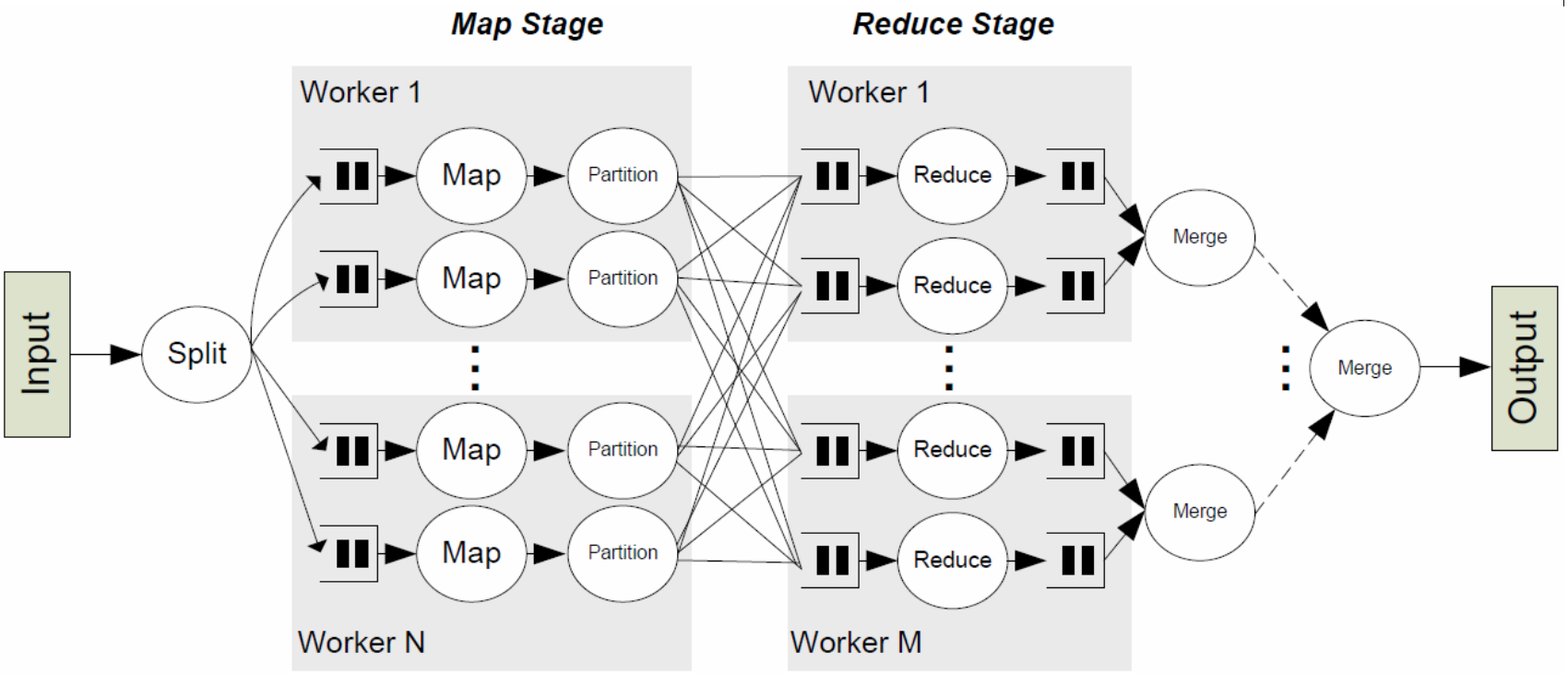


Εφαρμογή
Τρόπος λειτουργίας (2)



Πηγή: Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, Ion Stoica. "Improving MapReduce Performance in Heterogeneous Environments", *USENIX Association* 8th USENIX Symposium on Operating Systems Design and Implementation 2008.

Εφαρμογή
Τρόπος λειτουργίας (3)



Πηγή: Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, Christos Kozyrakis. "Evaluating MapReduce for Multi-core and Multiprocessor Systems", *Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA)*, Phoenix, AZ, February 2007

Εφαρμογή

Ανοχή σφαλμάτων - Αποτυχία εργάτη (worker)

- Ο αφέντης επικοινωνεί περιοδικά με κάθε εργάτη
- Αν ο εργάτης δε του απαντήσει μέσα σε συγκεκριμένο χρόνο θεωρεί ότι ο εργάτης απέτυχε

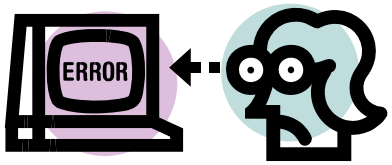
1. Ο εργάτης προτού αποτύχει είχε ολοκληρώσει μία map εργασία
   → η εργασία ακυρώνεται και επαναλαμβάνεται από κάποιον άλλο εργάτη (τα αποτελέσματά της βρίσκονταν αποθηκευμένα στον τοπικό δίσκο του εργάτη που απέτυχε)
2. Ο εργάτης προτού αποτύχει είχε ολοκληρώσει την reduce εργασία
   → η εργασία δεν εκτελείται ξανά διότι τα αποτελέσματά της έχουν ήδη αποθηκευτεί στο global σύστημα
3. Ο εργάτης τη στιγμή που απέτυχε εκτελούσε map ή reduce εργασία
   → η εργασία ακυρώνεται και επαναλαμβάνεται από την αρχή από κάποιον άλλο εργάτη

- Οι reduce εργασίες που βρίσκονται σε εξέλιξη ενημερώνονται κατάλληλα σε περίπτωση που η map εργασία από την οποία προμηθεύονται τα δεδομένα εκτελεσθεί από την αρχή από κάποιον άλλο εργάτη

Εφαρμογή

Ανοχή σφαλμάτων - Αποτυχία αφέντη (master)

- Όλες οι υπολογιστικές διαδικασίες του MapReduce ακυρώνονται (!)
  - Η αποτυχία του κόμβου αφέντη είναι σπάνιο φαινόμενο.



Google LABS

Εφαρμογή

Ανοχή σφαλμάτων – Σημασία των αποτυχιών

- Ντετερμινιστικές map και reduce συναρτήσεις
  - ίδιο αποτέλεσμα σε σειριακή εκτέλεση του προγράμματος
  - Μη-ντετερμινιστικές map και reduce συναρτήσεις
  - το αποτέλεσμα εξαρτάται από τα δεδομένα εισόδου

Εφαρμογή

- Master Data Structures
  - Ο αφέντης διατηρεί διάφορες δομές δεδομένων. Για κάθε map και reduce εργασία αποθηκεύει την κατάσταση (*idle*, *in-process* ή *completed*) και την ταυτότητα του εργάτη.
- Locality
  - Ο αφέντης διαιρεί τις εργασίες βασιζόμενος στην τοποθεσία των δεδομένων: προσπαθεί να έχει τις map() εργασίες στην ίδια μηχανή σαν φυσικό αρχείο δεδομένων ή τουλάχιστον στο ίδιο rack (στους τοπικούς δίσκους των μηχανών που αποτελούν τον cluster)
  - Ο εισαγόμενες map () εργασίες χωρίζονται σε 64 MB μπλοκ [ίδιο μέγεθος με τα κομμάτια του Google File System]

Εφαρμογή

- Task Granularity
  - Υποδιαιρούμε τη map φάση σε M κομμάτια και την reduce σε R κομμάτια → Ο αφέντης πρέπει να κάνει *O(M+R)* προγραμματισμένες αποφάσεις και να διατηρεί *O(M*R)* καταστάσεις στη μνήμη
- Backup Tasks
  - "Straggler" → μηχανή που χρειάζεται ασυνήθιστα μεγάλο χρόνο για να ολοκληρώσει μία από τις λίγες τελευταίες map ή reduce εργασίες υπολογισμού
  - *Λύση*: Όταν μια λειτουργία MapReduce κοντεύει στο τέλος της, ο αφέντης προγραμματίζει backup εκτελέσεις των υπολοίπων εργασιών που βρίσκονται σε εξέλιξη. Η εργασία μαρκάρεται ως ολοκληρωμένη όταν είτε η αρχική είτε η backup εκτέλεση ολοκληρωθεί

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

Επεκτάσεις (1)

- Partition Function
  - Δυνατότητα τροποποίηση της συνάρτησης διαχωρισμού στο ενδιάμεσο key: *hash(key)modR* ☞ *hash(Hostname(urlkey))modR*
- Ordering Guarantees
  - Εύκολη δημιουργία ταξινομημένου εξερχόμενου αρχείου ανά τμήμα
- Combiner Function
  - Δυνατότητα προσδιορισμός προαιρετικής *Combiner* συνάρτησης που κάνει μερική συγχώνευση των δεδομένων πριν την αποστολή τους μέσω δικτύου
- Input and Output Types
  - Υποστήριξη για ανάγνωση δεδομένων (εισερχόμενων και εξερχόμενων) διαφόρων μορφών (format)
- Side-effects
  - Παραγωγή βοηθητικών αρχείων ως επιπρόσθετα outputs των map ή/και reduce λειτουργιών

Επεκτάσεις (2)

- Skipping Bad Records
  - παράλειψη εγγραφών που μπορεί να προκαλέσουν προβλήματα (deterministic crashes) στην πρόοδο της λειτουργίας του συστήματος
- Local Execution
  - τοπική εκτέλεση όλων των εργασιών για μία MapReduce λειτουργία για ευκολία debugging, profiling και μικρής έκτασης testing
- Status Information
  - αναφορά κατάστασης συστήματος μέσω status σελίδων που εξάγει ο αφέντης με χρήση ενός εσωτερικού HTTP server: πρόβλεψη υπολειπόμενου χρόνου εκτέλεσης και αναφορά αποτυχίας εργατών με τις αντίστοιχες map ή reduce εργασίες τους
- Counters
  - παροχή μέσω της βιβλιοθήκης MapReduce ενός μετρητή για την καταμέτρηση γεγονότων προκληθέντων από διάφορα συμβάντα

```
Counter* uppercase;
uppercase =
GetCounter("uppercase");

map(String name, String contents):
            for each word w in
contents:
            if (IsCapitalized(w)):
            uppercase-
>Increment();
            EmitIntermediate(w,
"1");
```

Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

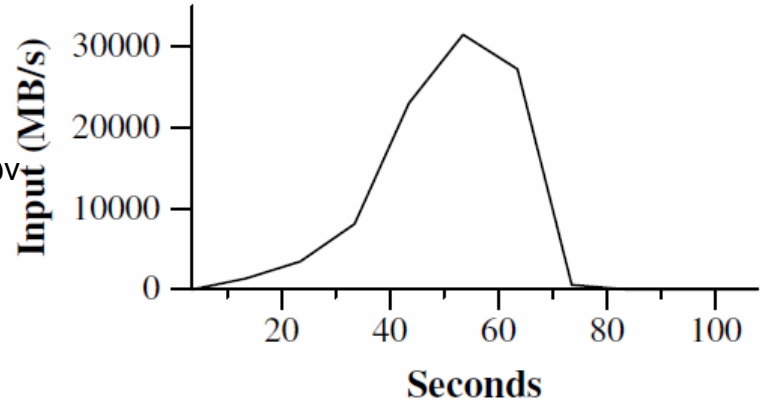
Μετρήσεις (1)

Διαμόρφωση Cluster

- 1800 Μηχανές → *Κάθε μηχανή*: Δύο 2GHz Intel Xeon επεξεργαστές με Hyper-Threading enabled, 4GB μνήμη, δύο 160GB IDE δίσκους και ένα gigabit Ethernet link
- Οι μηχανές ήταν τοποθετημένα σε δύο επίπεδα σε σχήμα δένδρου με δίκτυο σχεδόν 100-200 Gbps του συνολικού εύρους ζώνης που διατίθενται στη ρίζα.
- Όλα τα μηχανήματα ήταν στην ίδια hosting εγκατάσταση και γι'αυτό ο round-trip χρόνος μεταξύ κάθε ζεύγους μηχανών ήταν λιγότερο από 1msec.
- 4GB μνήμης, σχεδόν1-1.5GB δεσμευμένα από άλλες εργασίες που έτρεχαν στον cluster
- Εκτέλεση προγραμμάτων: χρονική στιγμή που οι περισσότεροι CPUs, δίσκοι, και δίκτυο ήταν αδρανείς

Grep

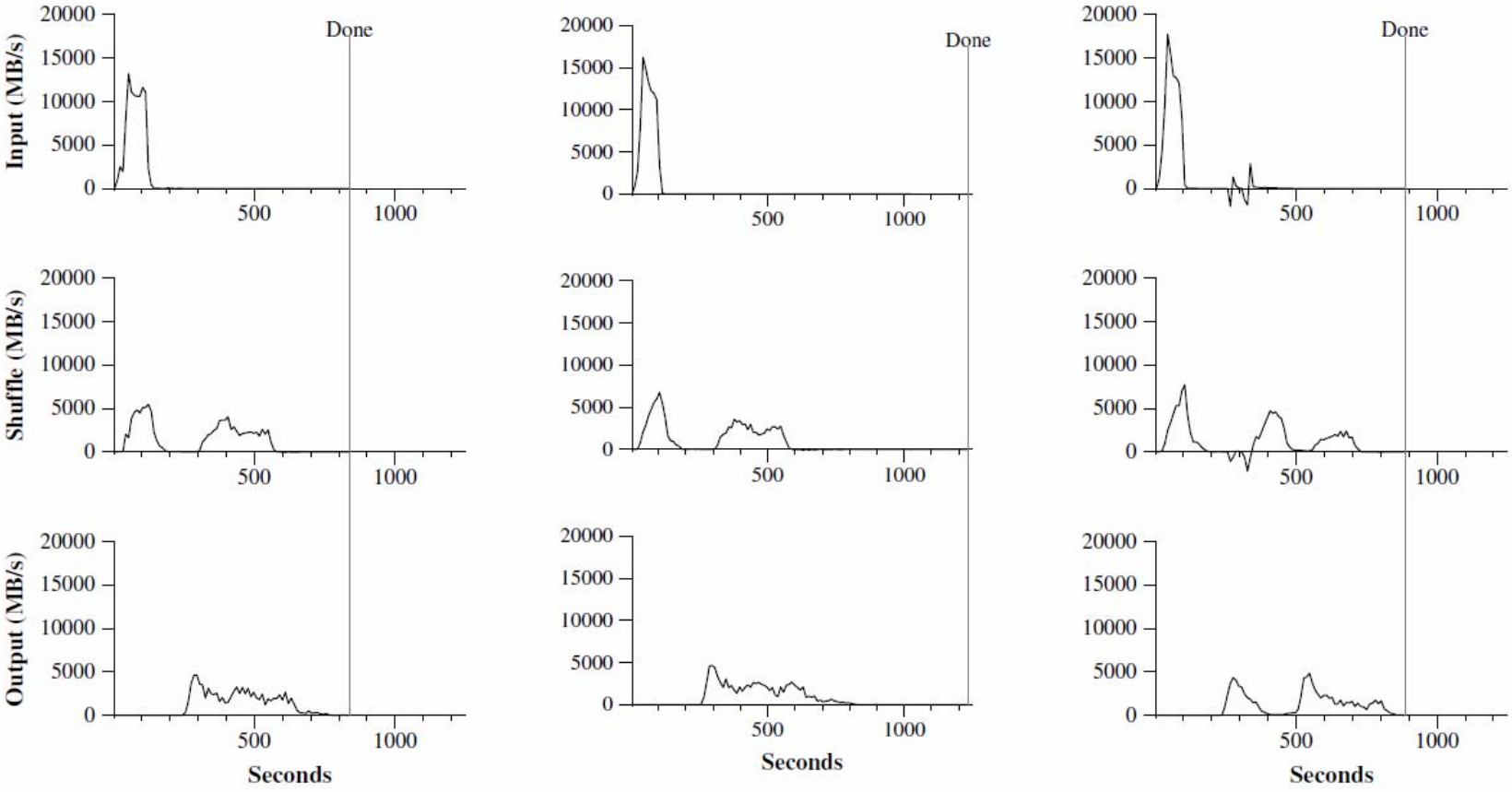
Σάρωση 10^{10} 100-byte εγγραφές
Αναζήτηση σχετικά σπάνιου 3-χαρακτήρων pattern
input → 64MB κομμάτια (M=15000)
output → σε ένα αρχείο (R=1)



Σχήμα 1: Data transfer rate over time

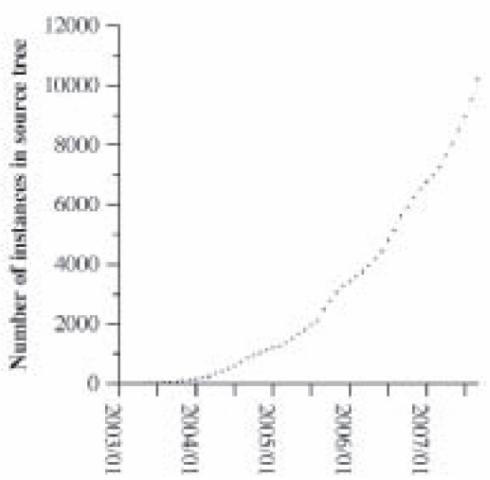
Μετρήσεις (2)

Sort

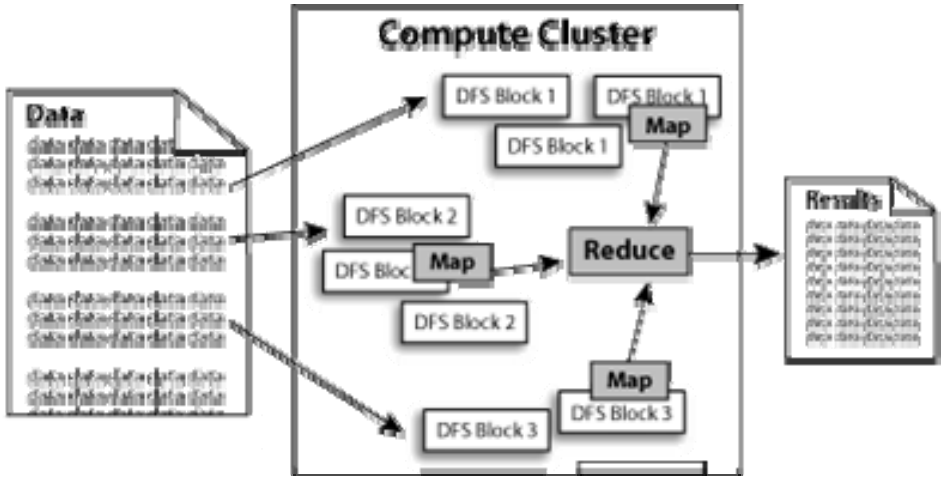


Σχήμα 2: Data transfer rate over time for different executions of the sort program

Εμπειρία



	Aug. '04	Mar. '06	Sep. '07
Number of jobs (1000s)	29	171	2,217
Avg. completion time (secs)	634	874	395
Machine years used	217	2,002	11,081
map input data (TB)	3,288	52,254	403,152
map output data (TB)	758	6,743	34,774
reduce output data (TB)	193	2,970	14,018
Avg. machines per job	157	268	394
Unique implementations			
map	395	1958	4083
reduce	269	1208	2418



Θεματολογία

[1] • Επισκόπηση

[2] • MapReduce framework

[3] • Παραδείγματα

[4] • Εφαρμογή

[5] • Επεκτάσεις

[6] • Μετρήσεις

[7] • Συμπεράσματα

Google LABS

Συμπεράσματα

Δυνατά σημεία

- Επιτρέπεται η εκτέλεση λειτουργιών map/reduce με χρήση κατανεμημένης επεξεργασίας, δεδομένου ότι η κάθε λειτουργία map που γίνεται είναι ανεξάρτητη από άλλες, όλο το mapping μπορεί να γίνει παράλληλα.
- Μπορεί να γίνεται επεξεργασία πολύ μεγαλύτερου όγκου δεδομένων από αυτόν που ένας commodity εξυπηρετητής μπορεί να επεξεργαστεί.
- Σε περίπτωση που ένας εξυπηρετητής είναι αργός, ή αδυνατεί να επεξεργαστεί τα δεδομένα που του ανατέθηκαν, γίνεται rescheduling της εργασίας του αυτόματα.

Συμπεράσματα

Αδύνατα σημεία

1. Το framework του MapReduce θα μπορούσε να χαρακτηριστεί λιγότερο αποδοτικό σε σχέση με άλλους αλγορίθμους που είναι περισσότερο γραμμικοί (πληρώνουμε αυτό το κόστος γιατί κερδίζουμε όσον αφορά την επεξεργασία μεγάλου όγκου δεδομένων).
2. Η χρησιμότητα του συστήματος μοιάζει να εξαρτάται από ένα cluster όπως αυτό της Google, και δεν είναι σαφές το πόσο εύκολα μεταφέρσιμη είναι η βιβλιοθήκη ώστε να μπορέσει να συνυπάρξει και με άλλους τύπους multi-processor συστημάτων.
3. Όσον αφορά την αξιολόγηση του συστήματος δίνονται μετρήσεις χρόνου εκτέλεσης για πολλούς διαφορετικούς αλγόριθμους που έχουν γραφτεί χρησιμοποιώντας το MapReduce, αλλά δεν υπάρχει τιμή αναφοράς που να συγκρίνει αυτές τις μετρήσεις. Η μόνη μετρική με την οποία συγκρίνουν το MapReduce άμεσα έναντι των άλλων μεθόδων είναι η αναφορά ότι τα MapReduce προγράμματα χαρακτηρίζονται από λιγότερες γραμμές κώδικα από τα αντίστοιχα που γράφονται χωρίς αυτό.

Ερωτήσεις-Συζήτηση

Ποια η διαφορά εκτέλεσης και αποτελέσματος μεταξύ των ντετερμινιστικών και μη- ντετερμινιστικών map και reduce συναρτήσεων;

Google LABS