

Google File System, HDFS, BigTable, Hbase

Κατανεμημένα Συστήματα
2016-2017

<http://www.cslab.ece.ntua.gr/courses/distrib>

Περιεχόμενα

- Εισαγωγή
- GFS
- HDFS
- BigTable
- HBase

Γενικά

- Σε ένα Cloud
 - Απαιτείται δυνατότητα αποθήκευσης και διαχείρισης μεγάλου συνόλου δεδομένων.
 - Τα δεδομένα αξιοποιούνται και παράγονται σε διαφορετικές τοποθεσίες.
 - Οι βλάβες του υλικού είναι πολύ συχνές.
 - Τα αρχεία έχουν πολύ μεγάλο μέγεθος.

Γενικά

- Λειτουργίες ενός κατανεμημένου συστήματος:
 - Ονομασία
 - Διαμοιρασμός
 - Προσωρινή αποταμίευση
 - Αντίγραφα αρχείων
- Επιπλέον πρέπει να παρέχει στους χρήστες του:
 - Συνέπεια
 - Αξιοπιστία
 - Διαθεσιμότητα
 - Κλιμακωσιμότητα
 - Διαφάνεια

GFS: Google File System

- Κλιμακώσιμο κατανεμημένο σύστημα αρχείων
- Κατάλληλο για data-intensive εφαρμογές
- Προσφέρει ανοχή σε σφάλματα, παρότι εκτελείται σε commodity υπολογιστές
- Εξυπηρετεί ένα μεγάλο σύνολο από πελάτες.

Παραδοχές

- Υψηλή συχνότητα βλαβών.
- Τα αρχεία σε τέτοιου είδους συστήματα είναι μεγάλα (πολλά GBs)
- Πρόσβαση αρχείων
 - Τα περισσότερα αρχεία τροποποιούνται με προσάρτηση(append)
 - 2 ειδών reads: μεγάλα σειριακά reads ή μικρά random reads
- Ειδικός χειρισμός για παράλληλο append
- Ταιριάζουν για data analytics
 - Συνήθως μεγάλα αρχεία.
 - Το υψηλό bandwidth προτιμάται από το χαμηλό latency.

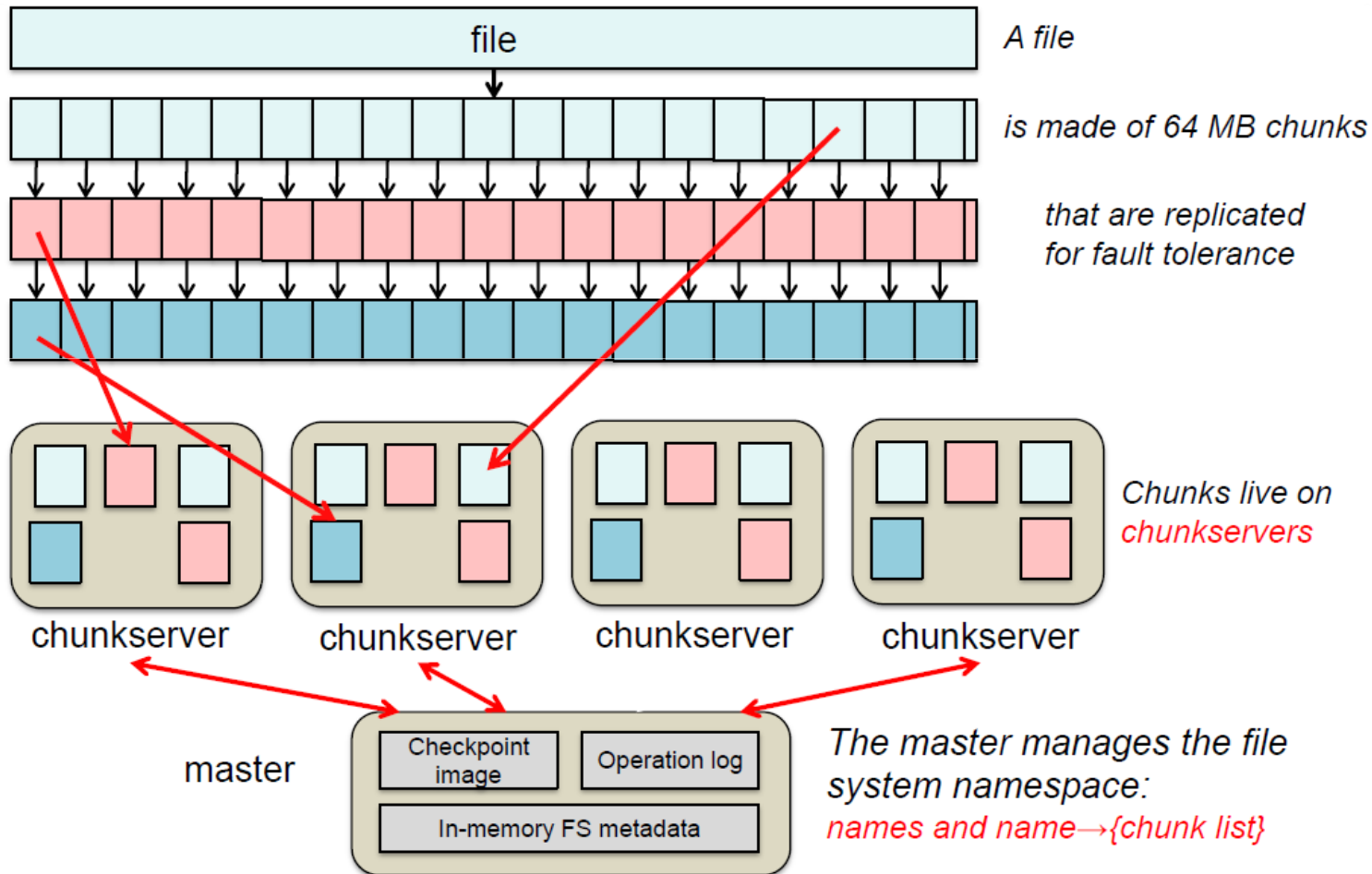
Διεπαφή συστήματος αρχείων

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

Αρχιτεκτονική

- Ένας master
- Πολλαπλοί chunkserver
- Πολλαπλοί clients
- Τα αρχεία χωρίζονται σε chunks σταθερού μεγέθους
- Δεν πραγματοποιείται caching

GFS Files

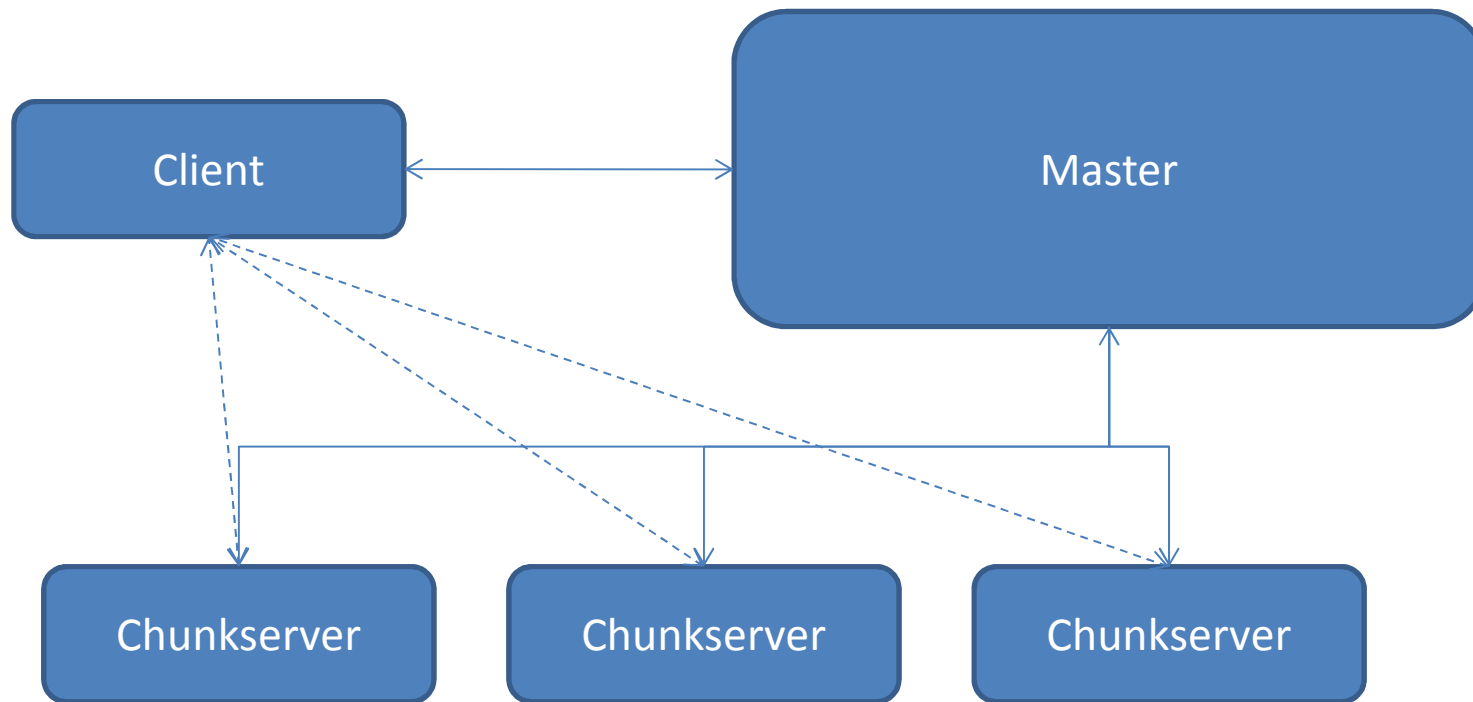


Master

- Ο master διατηρεί:
 - Το namespace
 - Την αντιστοιχία από αρχεία σε chunks
 - Τις θέσεις των chunks
- Όλα τα μεταδεδομένα βρίσκονται στη μνήμη του master
- Ο master εντοπίζει τα chunks μέσω heartbeat μηνυμάτων
- Σημαντικές αλλαγές στο GFS διατηρούνται στο operation log
- Checkpoints

- Χειρίζεται
 - Leases (locks)
 - Garbage collection
 - Chunk migration

Αρχιτεκτονική



Μέγεθος chunk

- 64 MB
- Μειωμένη ανάγκη επικοινωνίας client και master.
- Μειωμένος φόρτος δικτύου.
- Μικρότερο μέγεθος δομών δεδομένων στον master.
- Chunks από μικρά αρχεία μπορεί να αποτελέσουν hotspots.

ΣΥΝΕΠΕΙΑ

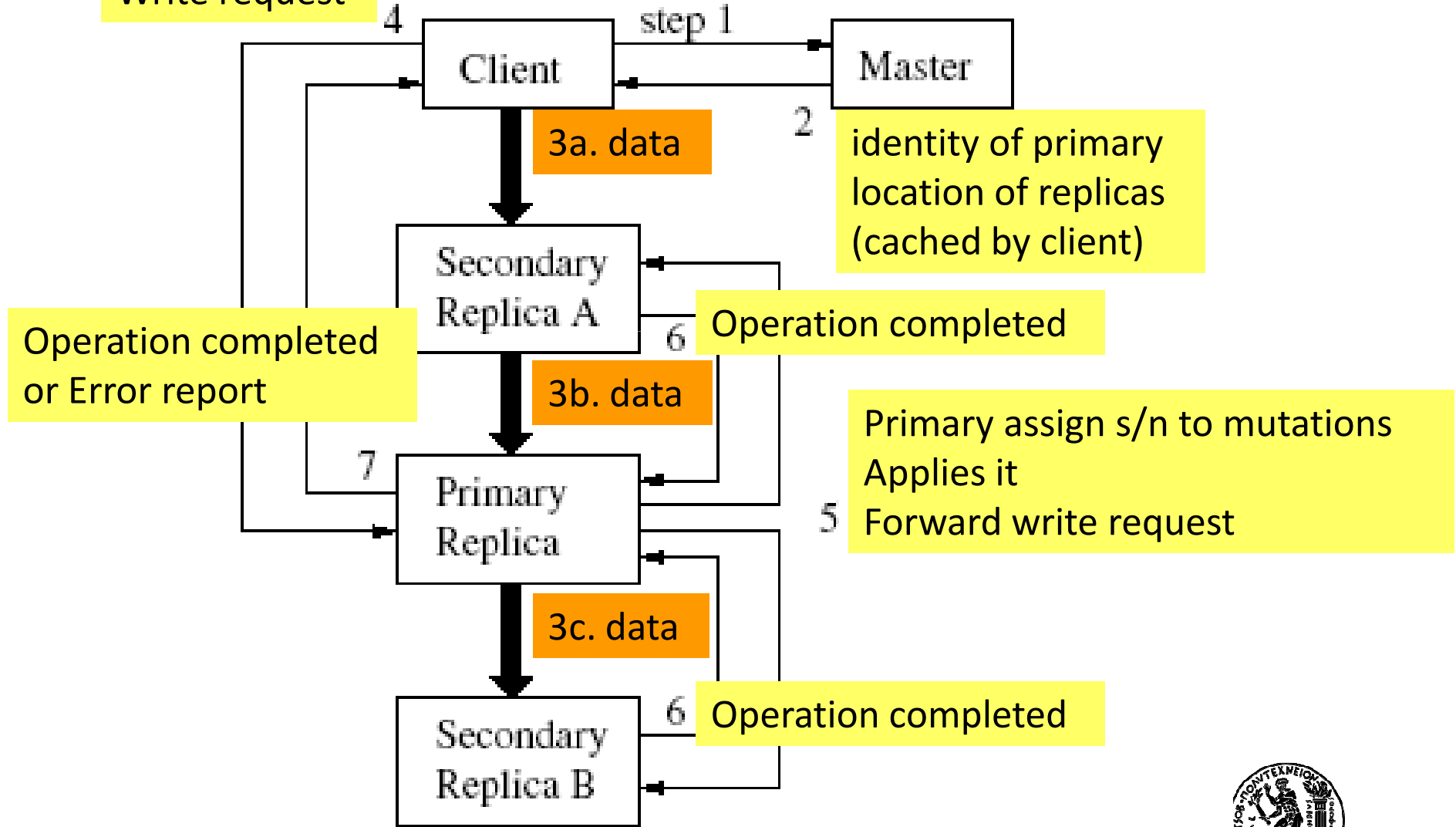
- Ένα αρχείο μπορεί να βρίσκεται σε μία από τις παρακάτω καταστάσεις:
 - Consistent
 - Defined
 - Inconsistent

Lease, mutation και ροή δεδομένων

- Mutation είναι κάθε αλλαγή στα δεδομένα ενός αρχείου και εφαρμόζεται σε όλα τα αντίγραφα του.
- Όταν σε ένα αντίγραφο του αποδοθεί ένα lease τότε θεωρείται το primary και διαχειρίζεται τα mutation στα υπόλοιπα αντίγραφα.
- Η ροή δεδομένων διαχωρίζεται από τη ροή ελέγχου.
- Τα δεδομένα διαδίδονται σειριακά από ένα chunkserver στον κοντινότερο με pipelining.

Current lease holder?

Write request



Operation completed or Error report

identity of primary location of replicas (cached by client)

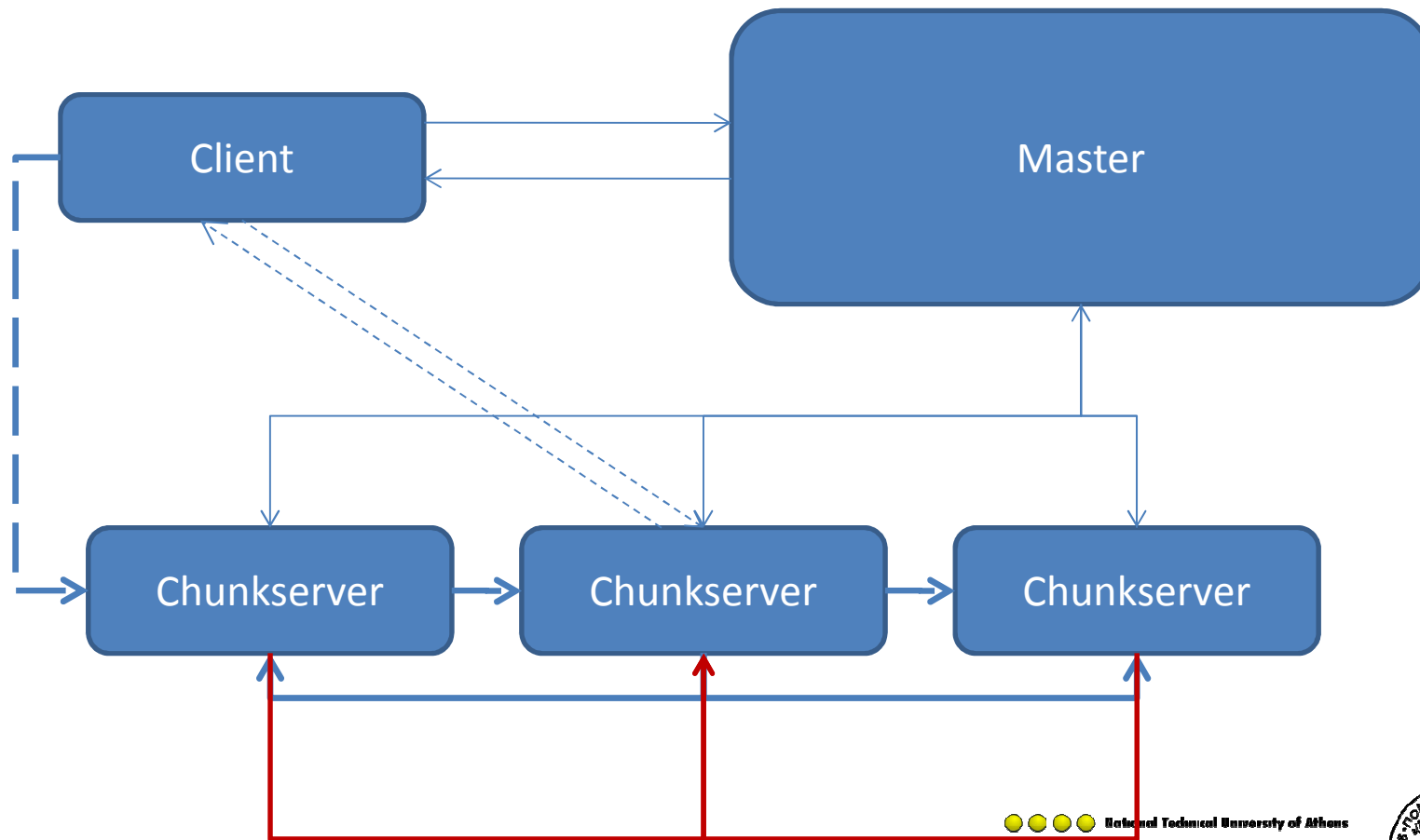
Operation completed

Primary assign s/n to mutations Applies it Forward write request

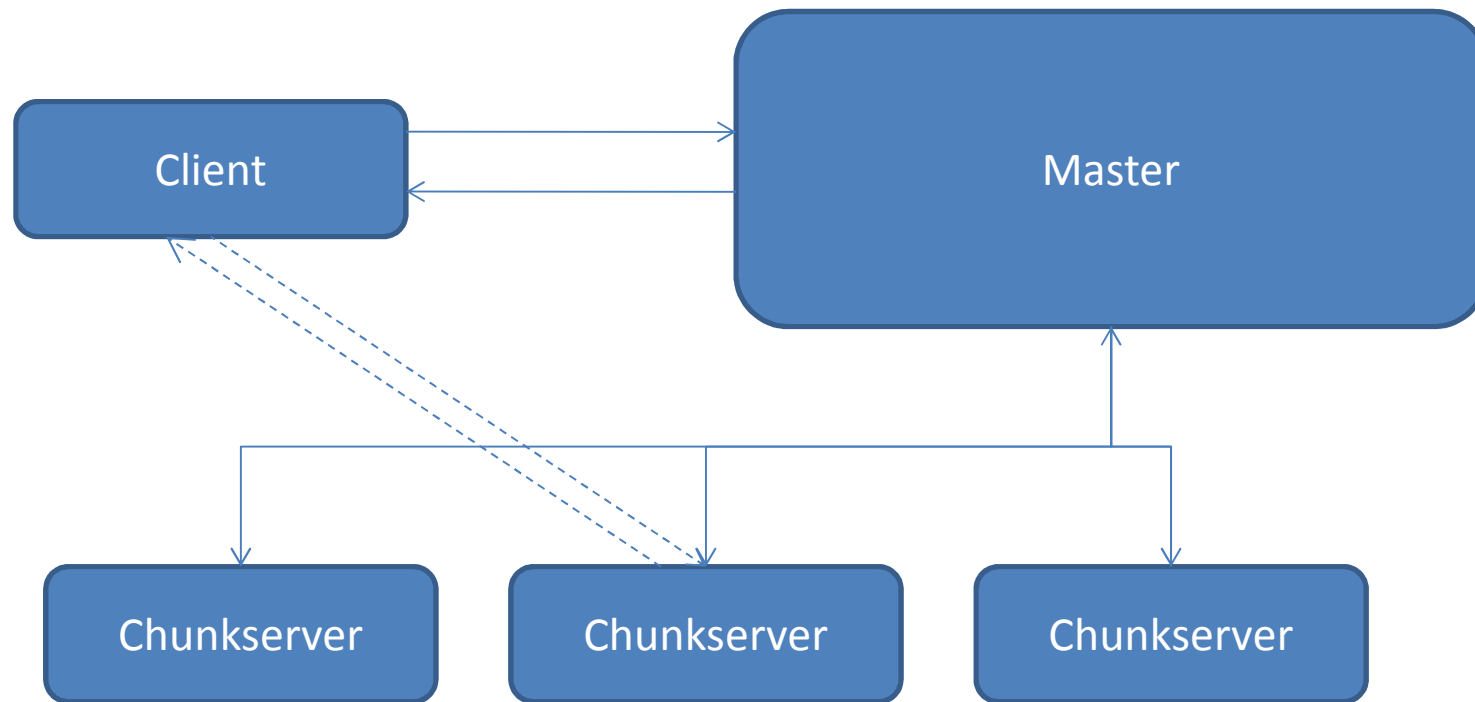
Operation completed



Lease, mutation και ροή δεδομένων (Αποθήκευση)



Lease, mutation και ροή δεδομένων (Ανάγνωση)



Διαχείριση namespace και locking

- Υπάρχει αντιστοίχιση μεταξύ ενός αρχείου και του πλήρους pathname του.
- Read lock.
- Write lock.
- Παράλληλες αλλαγές στο ίδιο directory.

Τοποθέτηση αντιγράφων

- Μεγιστοποίηση αξιοπιστίας και διαθεσιμότητας δεδομένων.
- Μεγιστοποίηση χρήσης bandwidth.
- Ο χρήστης μπορεί να επιλέξει τον αριθμό των επιπλέον αντιγράφων (replicas).

Ισορροπία δεσμευμένων πόρων

- Νέα αντίγραφα τοποθετούνται σε κόμβους με σχετικά ελαφρύ φορτίο.
- Ίσος καταμερισμός δημιουργιών αρχείων ανά chunkserver.
- Καταμερισμός αντιγράφων.
- Ο master δίνει προτεραιότητα στην αντιγραφή.

Garbage collection

- Μετονομασία αρχείου προς διαγραφή.
- Περιοδικός έλεγχος για τη διαγραφή τους.

Stale replica detection

- Chunk version number
- Αυξάνεται με κάθε αλλαγή
- Παλιά αντίγραφα αφαιρούνται από το μηχανισμό garbage collection

Υψηλή διαθεσιμότητα

- Γρήγορη ανάρρωση από σφάλματα.
- Πολλαπλά αντίγραφα.
- Αντιγραφή της κατάστασης του master.

Ακεραιότητα δεδομένων

- Δεν είναι σωστό και πρακτικό να γίνεται έλεγχος μεταξύ των chunks
- Χρήση checksums
- Έλεγχος δεδομένων πριν μεταφερθούν
- Περιοδική εξέταση για corrupted chunks

Hadoop HDFS

- Hadoop Distributed File System (HDFS) είναι το πρωτεύον αποθηκευτικό σύστημα που χρησιμοποιείται από όλες τις εφαρμογές Hadoop.
- Το HDFS διασπάει τα 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

Πλεονεκτήματα του Hadoop

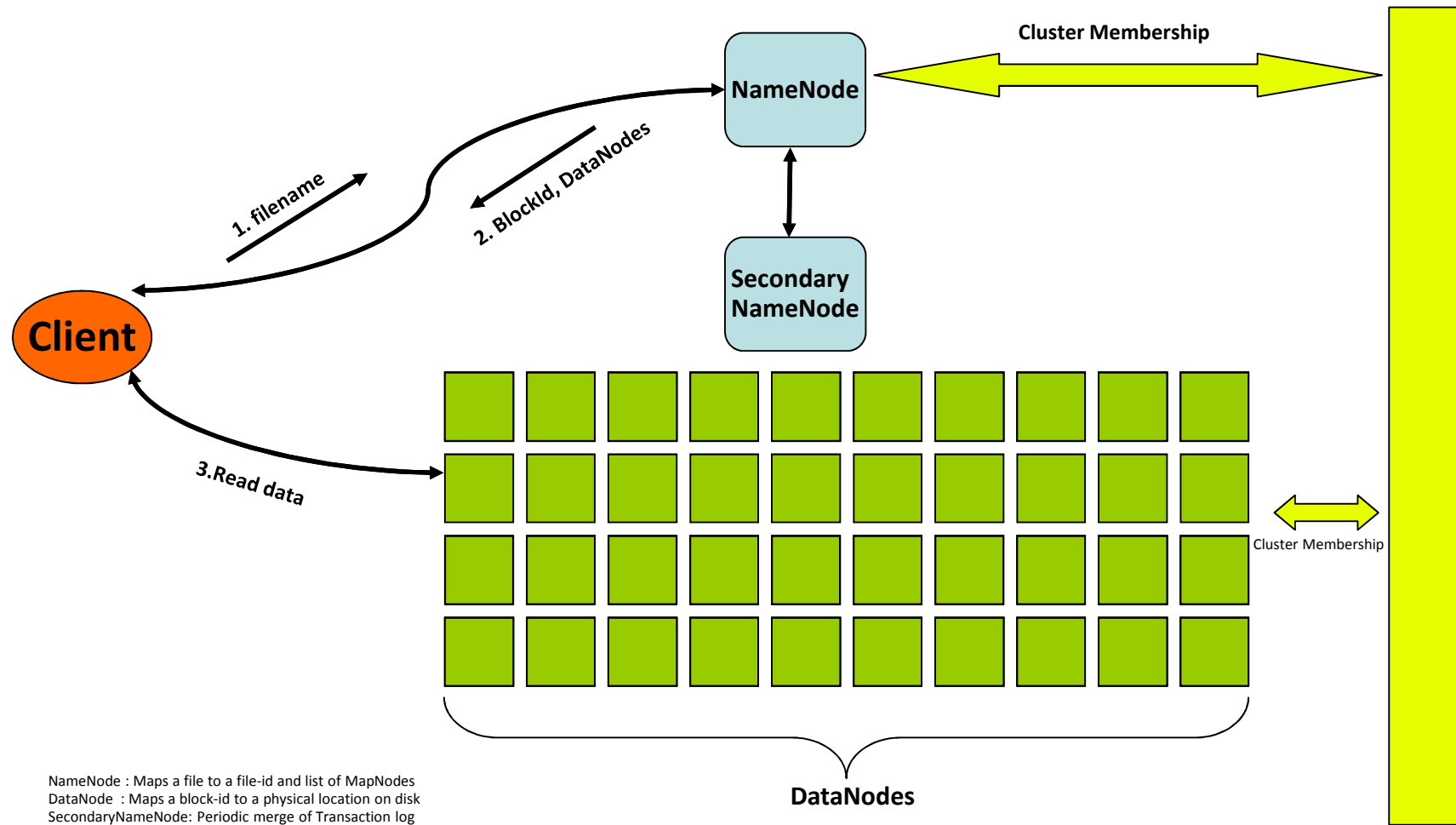
HDFS

- Κατανεμημένο αποθηκευτικό σύστημα πολύ μεγάλου μεγέθους.
 - 10.000 κόμβοι.
 - 100.000.000 αρχεία.
 - 10PB αποθηκευτικός χώρος.
- Βασίζεται σε φθηνό Hardware.
 - Κρατούνται αντίγραφα ασφαλείας των αρχείων ώστε να αντιμετωπίζονται οι βλάβες.
 - Ανίχνευση βλαβών και ανάκτηση.
- Είναι βελτιστοποιημένο για Batch processing
 - Οι τοποθεσίες των δεδομένων είναι διακριτές έτσι ώστε οι 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 προσπελούν τα δεδομένα απευθείας στους DataNodes

Η αρχιτεκτονική του HDFS



NameNode : Maps a file to a file-id and list of MapNodes
DataNode : Maps a block-id to a physical location on disk
SecondaryNameNode: Periodic merge of Transaction log

NameNode DataNode

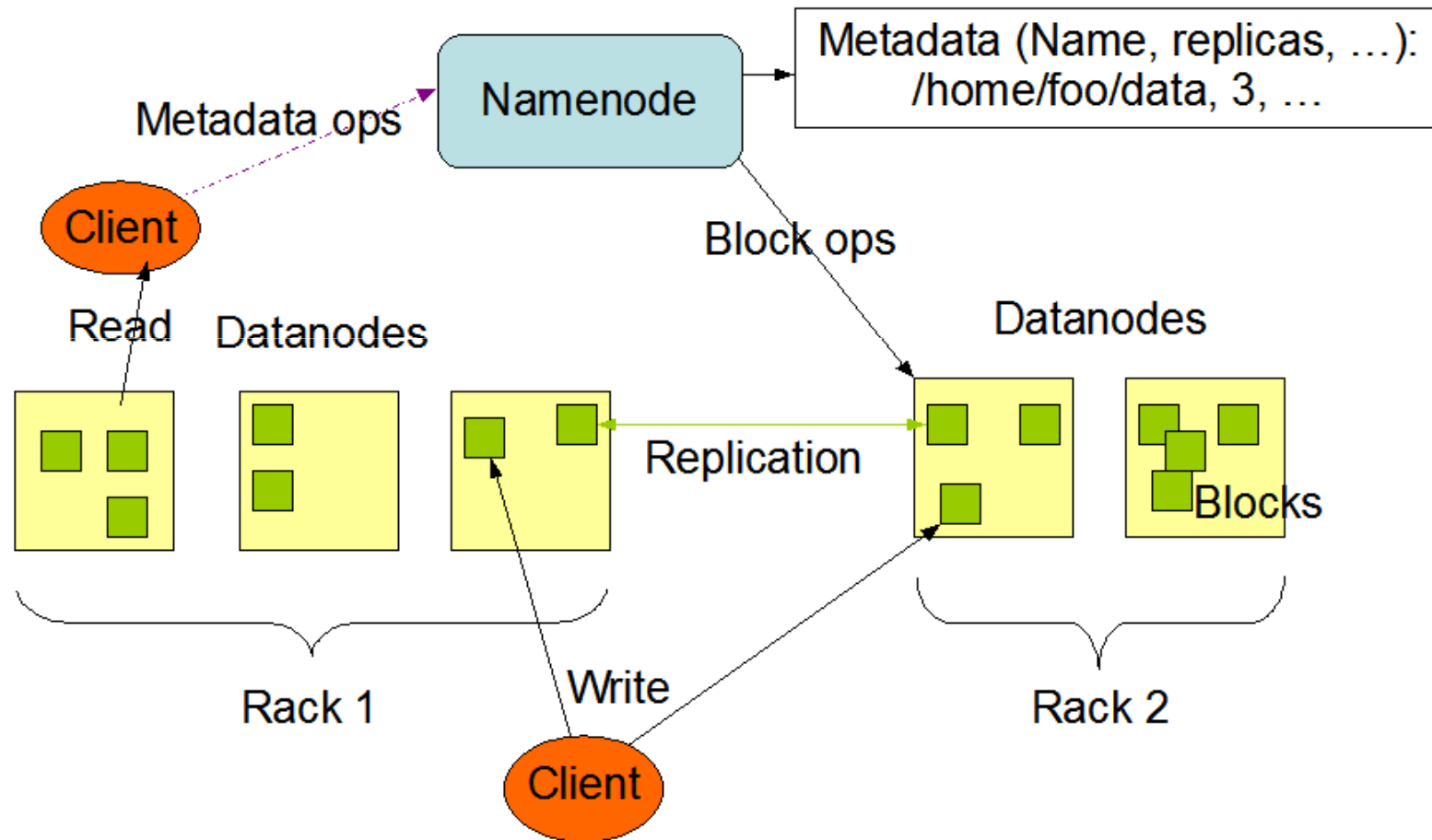
NameNode

- **Metadata στην μνήμη**
 - Όλα τα μεταδεδομένα φυλάσσονται στην κύρια μνήμη RAM
 - Δεν χρησιμοποιείται paging στα μεταδεδομένα
- **Είδη μεταδεδομένων**
 - Η λίστα των αρχείων
 - Η λίστα των Blocks για κάθε αρχείο
 - Η λίστα των DataNodes που περιέχουν το κάθε block
 - Ιδιότητες αρχείων, πχ ώρα δημιουργίας, αριθμός αντιγράφων κλπ
- **Καταγραφή συμβάντων**
 - Καταγράφονται δημιουργίες αρχείων, διαγραφές αρχείων κλπ

DataNode

- **Εξυπηρετητής Block**
 - Τα δεδομένα αποθηκεύονται στο τοπικό σύστημα αρχείων (π.χ. ext3)
 - Αποθηκεύονται μεταδεδομένα του κάθε block (π.χ. CRC)
 - Μεταφέρει δεδομένα και μεταδεδομένα στους Clients.
- **Αναφορά Block**
 - Περιοδικά στέλνει μια αναφορά με όλα τα υπάρχοντα blocks στον NameNode
- **Διευκολύνει το Pipelining των δεδομένων**
 - Προωθεί δεδομένα σε άλλους κόμβους

Εγγραφή – Ανάγνωση στο HDFS



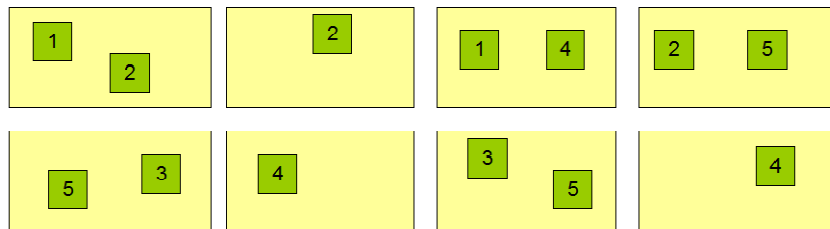
Write Data Pipelining

- Ο Client λαμβάνει μια λίστα από DataNodes στους οποίους θα δημιουργηθούν τα αντίγραφα του block
- Ο Client γράφει το block στον πρώτο DataNode
- Ο Πρώτος DataNode προωθεί τα δεδομένα στον επόμενο DataNode του με Pipeline
- Όταν όλα τα δεδομένα έχουν γραφτεί ο Client συνεχίζει την εγγραφή του επόμενου block του αρχείου

NameNode αντίγραφα Blocks

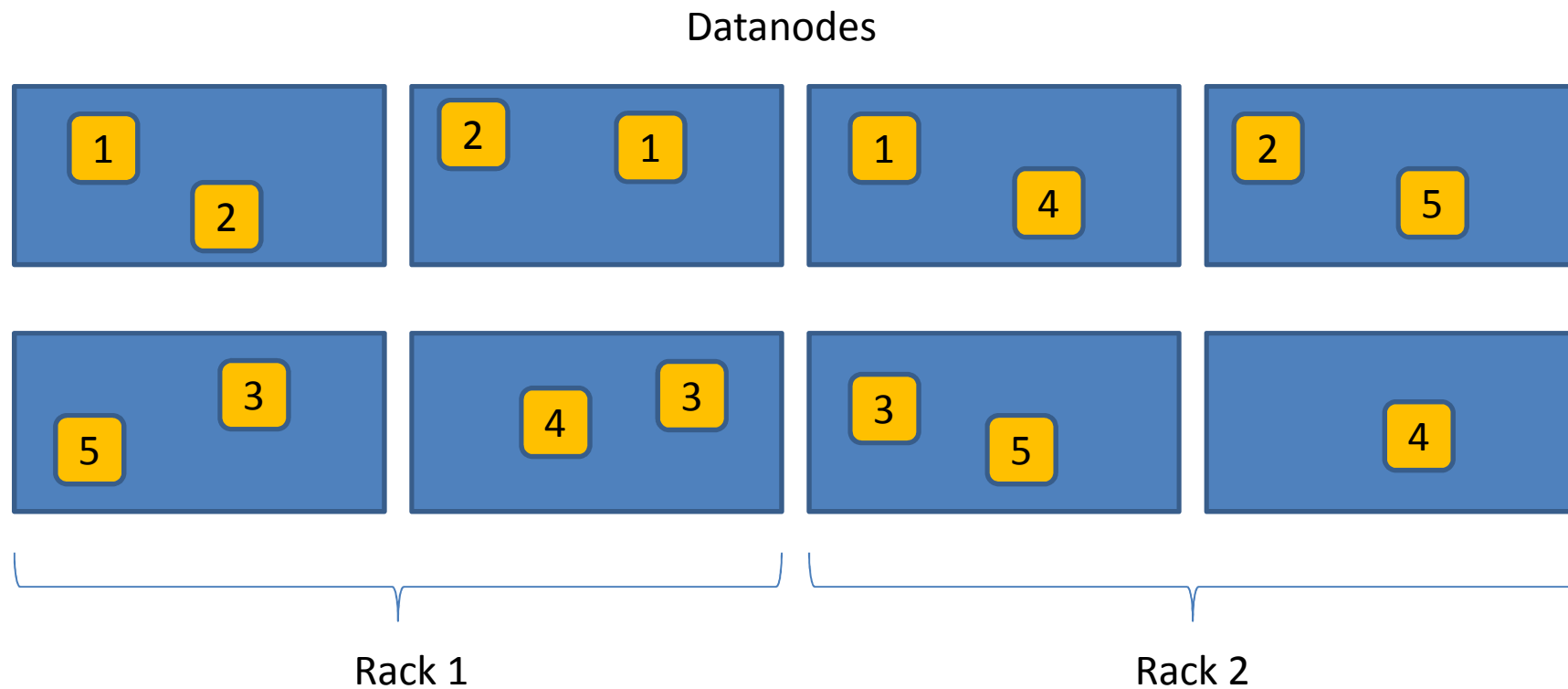
NameNode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {1,2,4}, ...

Datanodes



- Η στρατηγική που ακολουθείται
 - Ένα αντίγραφο στον τοπικό κόμβο.
 - Δεύτερο αντίγραφο στο ίδιο rack
 - Τρίτο αντίγραφο σε απομακρυσμένο rack
 - Επιπλέον αντίγραφα τοποθετούνται σε τυχαίους κόμβους
- Οι Clients διαβάζουν από το πλησιέστερο αντίγραφο

Replication



Η ορθότητα των δεδομένων

- **Η χρήση Checksums για την επικύρωση δεδομένων**
 - χρήση CRC32
- **Κατά την δημιουργία αρχείων**
 - ο Client υπολογίζει το checksum κάθε 512 bytes
 - οι DataNodes αποθηκεύουν τα checksums
- **Κατά την πρόσβαση των αρχείων**
 - ο Client λαμβάνει τα δεδομένα και το checksum από τον DataNode
 - εάν η επικύρωση αποτύχει τότε ο Client δοκιμάζει άλλο κόμβο

Βλάβη στον NameNode

- **A single point of failure**
- **Η καταγραφή των συναλλαγών αποθηκεύεται σε πολλαπλούς καταλόγους**
 - Έναν κατάλογο στο τοπικό σύστημα αρχείων
 - Έναν κατάλογο σε απομακρυσμένο σύστημα αρχείων

Rebalancer

- Σε περίπτωση που κάποιος datanode βρεθεί με πλεόνασμα αντιγράφων τότε το φορτίο μοιράζεται αυτόματα.
- Σκοπός: όλοι οι δίσκοι των DataNodes να έχουν το ίδιο ποσοστό δεδομένων
 - Συνήθως τρέχει όταν νέοι DataNodes προστίθενται στο σύστημα.
 - ο Cluster παραμένει λειτουργικός όταν εκτελείται ο Rebalancer.
 - ο Rebalancer τίθεται σε αναμονή όταν υπάρχει μεγάλη κίνηση στο δίκτυο.
 - Είναι ένα εργαλείο Command line.

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 καταναμημένο 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
- OSDI'06

Χαρακτηριστικά

- Μεγάλο εύρος εφαρμογών
 - Batch processing εφαρμογές
 - Εφαρμογές 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 $\langle x, y \rangle$
- Κάθε κελί περιέχει ένα σύνολο bytes

(row,column,time) \longrightarrow Value

Γραμμές (rows)

- Το κλειδί αποτελείται από ένα αλφαριθμητικό
- Οι ενεργειες πάνω σε μία γραμμή είναι ατομικές
- Λεξικογραφική ταξινόμηση με βάση τα κλειδιά
- Όλος ο πίνακας αποτελείται από (δισ/τρис/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

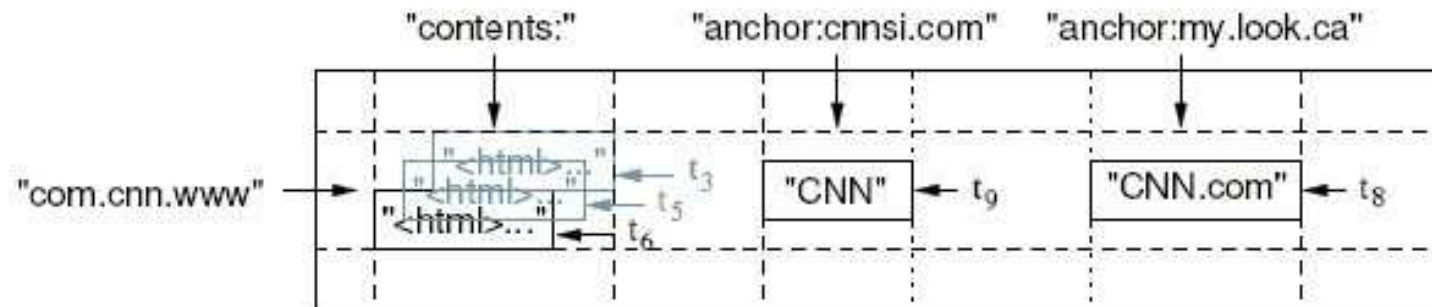
Στήλες (columns)

- 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 X εκδόσεων ή
 - Όλες τις εκδόσεις των τελευταίων X 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?

API 1/2

- **βασικές** λειτουργίες βάσεων Δεδομένων:
- Put(row_key, column_key, timestamp, value): βάλε μια τιμή σε ένα κελί.
- Get(row_key) : επέστρεψε όλα τα κελιά για μια γραμμή
- Get(row_key, column_key, timestamp): επέστρεψε ένα συγκεκριμένο κελί
- Scan(start_row_key, end_row_key): επέστρεψε όλα τα κλειδιά μεταξύ start_key και end_key

API 2/2

- Δεν υποστηρίζει joins!!! (κάντο με MapReduce εάν θες)
- Δεν υποστηρίζει `get(column_key)` σκέτο: θα πρέπει να ξέρεις το `row_key`
- No multi-row transactions
- Atomic single-row writes
- Optional atomic single-row reads

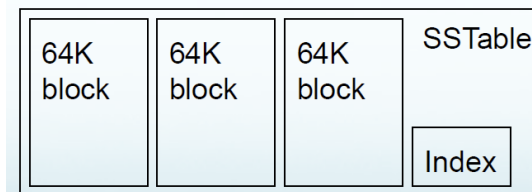
Αρχιτεκτονική

- Αποτελείται από:
 - Την βιβλιοθήκη client
 - Ένα master server
 - Πολλούς tablet servers
- Στηρίζεται πάνω στα:
 - Google filesystem
 - SSTable
 - Chubby

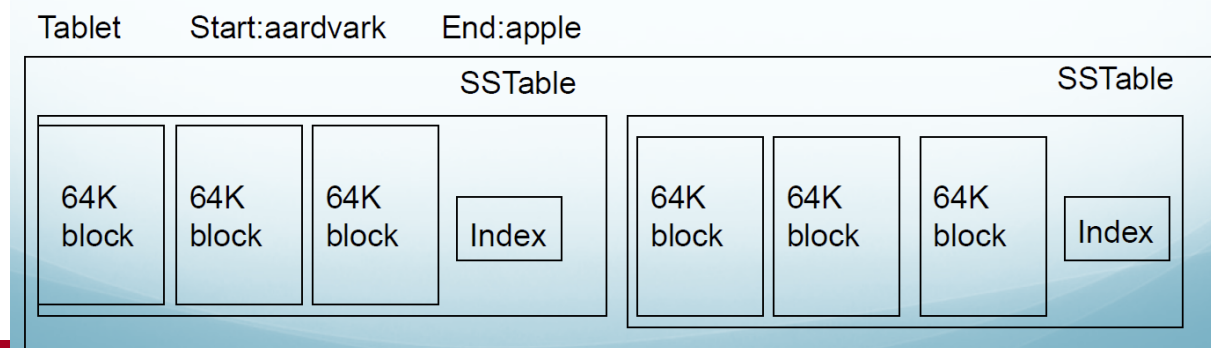
Tablets

- Το ευρος των τιμών χωρίζεται σε tablets
- Tablet: Ένα «ορθογώνιο κομμάτι» του πίνακα που περιέχει όλες τις γραμμές και στήλες μεταξύ δυο τιμών start και end.
- Αποτελείται από πολλά SSTables

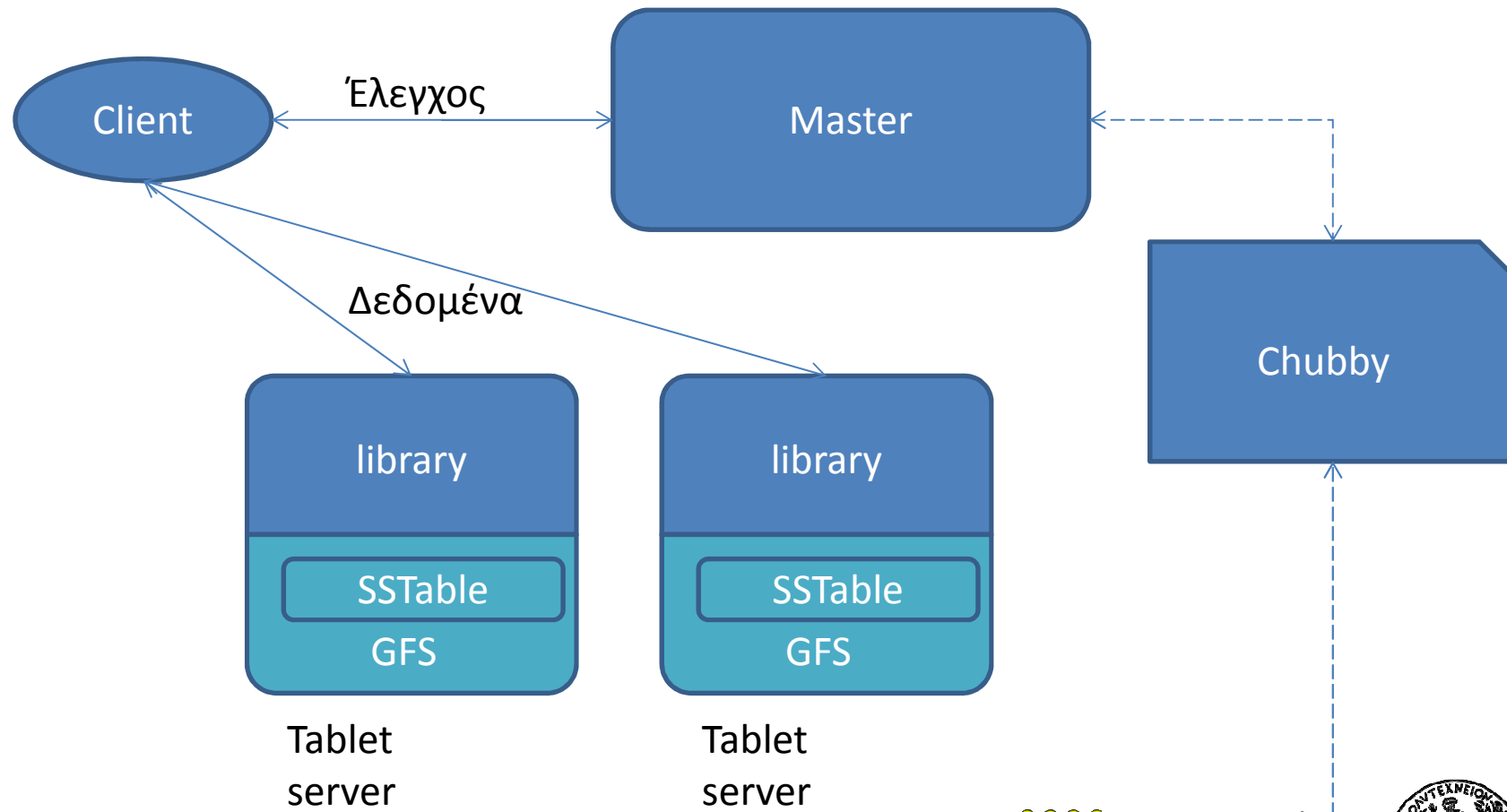
SSTable:



Tablet:



Αρχιτεκτονική



Master

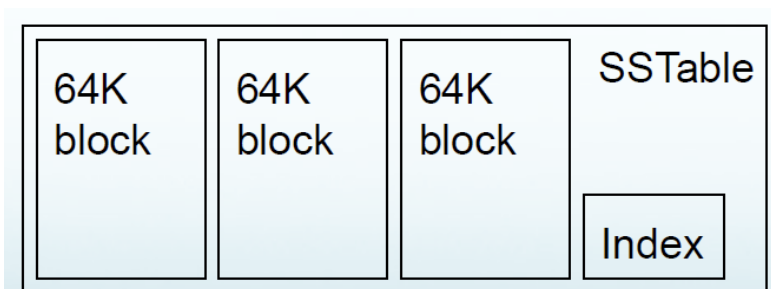
- Ανάθεση tablet σε κάποιον tablet server
- Εντοπισμός νέων tablet servers
- Εξισσόροπηση φόρτου
- Συλλογή σκουπιδιών από το GFS
- Διαχείριση schema
- Δεν περνάνε δεδομένα από αυτόν
- Τρέχει μαζί με τον Master του GFS.

Tablet server

- Διαχείριση ενός συνόλου tablets
 - Από δεκάδες μέχρι μερικές χιλιάδες
- Εξυπηρέτηση αιτήσεων ανάγνωσης και εγγραφής
- Διαίρεση των tablets που έχουν μεγαλώσει υπερβολικά (διαδικασία compaction)
 - Αρχικά υπάρχει ένα μόνο tablet ανά πίνακα το οποίο διαιρείται όταν γίνει περίπου 100-200 MB
- Μπορούν να προστεθούν/αφαιρεθούν δυναμικά

SSTable

- Format αρχείου της Google
- Μέγεθος τάξης MB (πχ 128MB)
- Μπορεί να είναι και όλο στην RAM
- Ταξινομημένα δεδομένα
- Αντιστοιχεί κλειδιά σε τιμές
- Αποτελείται από blocks τάξης KB (πχ 64KB)
 - Ένα «ειδικό» block περιέχει index για γρήγορη αναζήτηση
 - Σε περίπτωση που το SSTable είναι στον δίσκο, με το index block σε μια αναζήτηση βρίσκεται το block με 2 disk accesses.



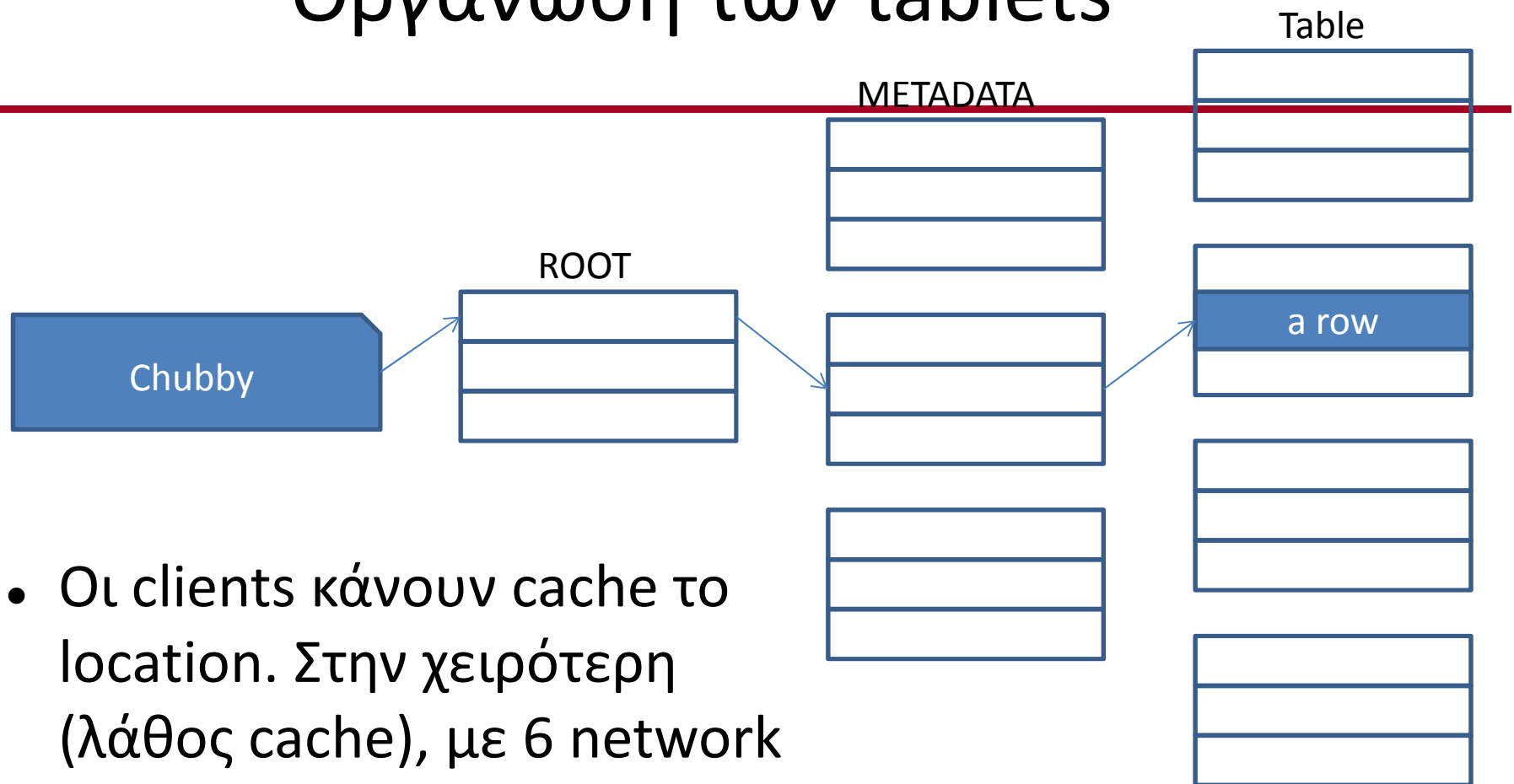
Chubby

- Κατανεμημένο locking service
 - Quorum από περιττό αριθμό servers
 - Paxos - like Algorithm (επίλυση διαφωνιών)
- Υψηλή διαθεσιμότητα
- Χρησιμοποιείται για:
 - υπηρεσίες lock (atomic transactions)
 - Εξασφάλιση λειτουργίας master
 - Αποθήκευση σημαντικών πληροφοριών(π.χ. schema)
 - Ανακάλυψη νέων tablet servers
 - Πληροφορίες ελέγχου πρόσβασης

Οργάνωση των tablets

- Ιεραρχία τριών επιπέδων για την αποθήκευση της πληροφορίας
- Ένα αρχείο στο chubby περιέχει την τοποθεσία του root tablet
- Το root tablet περιέχει πληροφορίες για το που βρίσκονται τα tablets ενός ειδικού πίνακα METADATA (1st METADATA) και δεν διαιρείται ποτέ
- Ο METADATA πίνακας περιέχει τις πληροφορίες για όλα τα υπόλοιπα tablets που περιέχουν τα δεδομένα του πίνακα

Οργάνωση των tablets

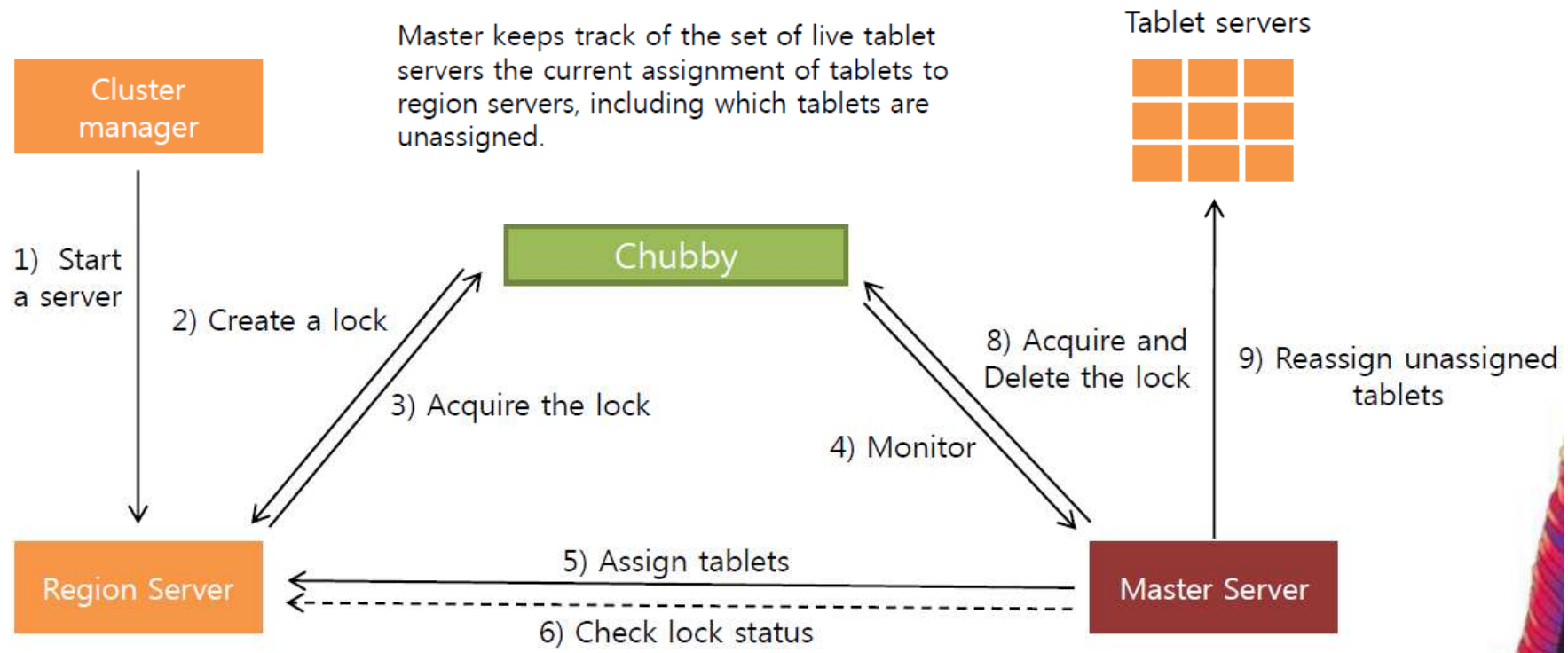


- Οι clients κάνουν cache το location. Στην χειρότερη (λάθος cache), με 6 network msgs βρίσκουν το location.

Ανάθεση των tablets

- Κάθε tablet ανατίθεται σε ένα tablet server
- Ο master είναι υπεύθυνος για την ανάθεση τους
 - Ελεγχει την κατάσταση κάθε server περιοδικά
- Το Chubby χρησιμοποιείται για την παρακολούθηση των tablet servers
- Κατά την έναρξη ενός Master
 - Lock στο Chubby, ls στο dir, επικοινωνία με κάθε live server για να βρει ποια tablets είναι ήδη assigned, διάβασμα του METADATA

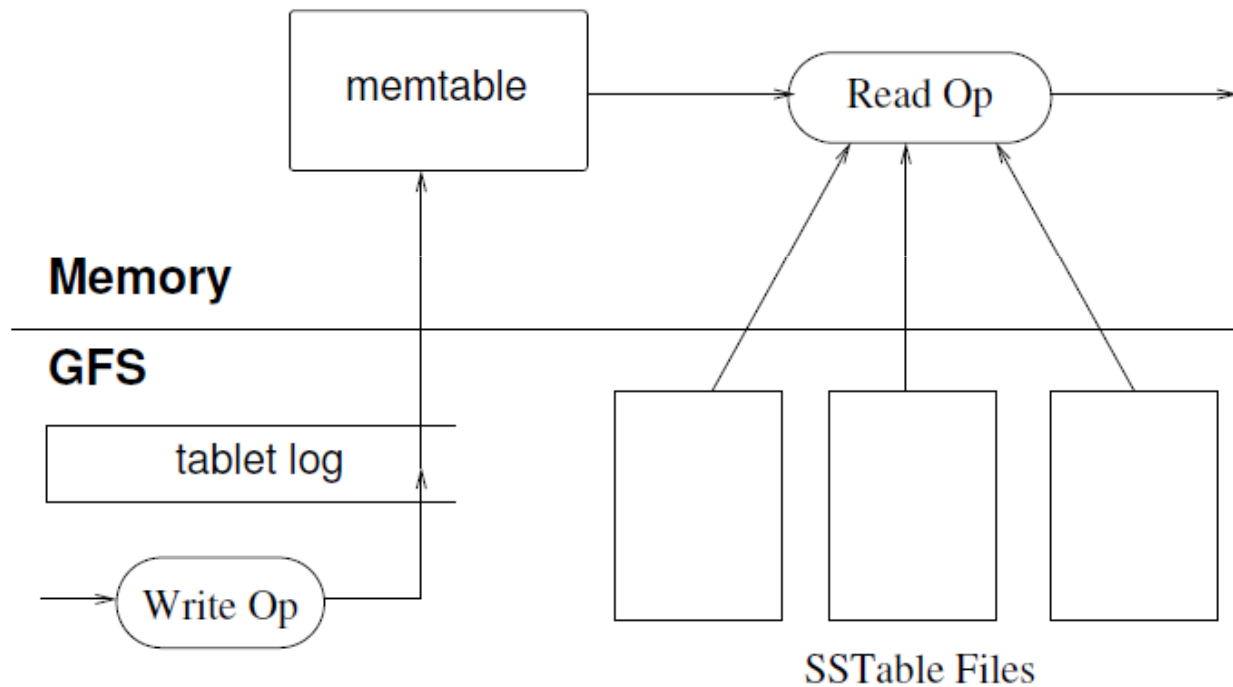
Ξεκίνηση Master



Μοίρασμα των tablets

- Χρησιμοποιείται το GFS για την μόνιμη αποθήκευση των δεδομένων ενός tablet
- Ένα commit log χρησιμοποιείται για τις ενημερώσεις
- Οι πιο πρόσφατες διατηρούνται σε ένα memtable στη μνήμη RAM
- Όταν ένα tablet ανακτάται τότε οι πληροφορίες του συγχωνεύονται με αυτές του memTable
 - Κάνει recovery από το log (WAL)

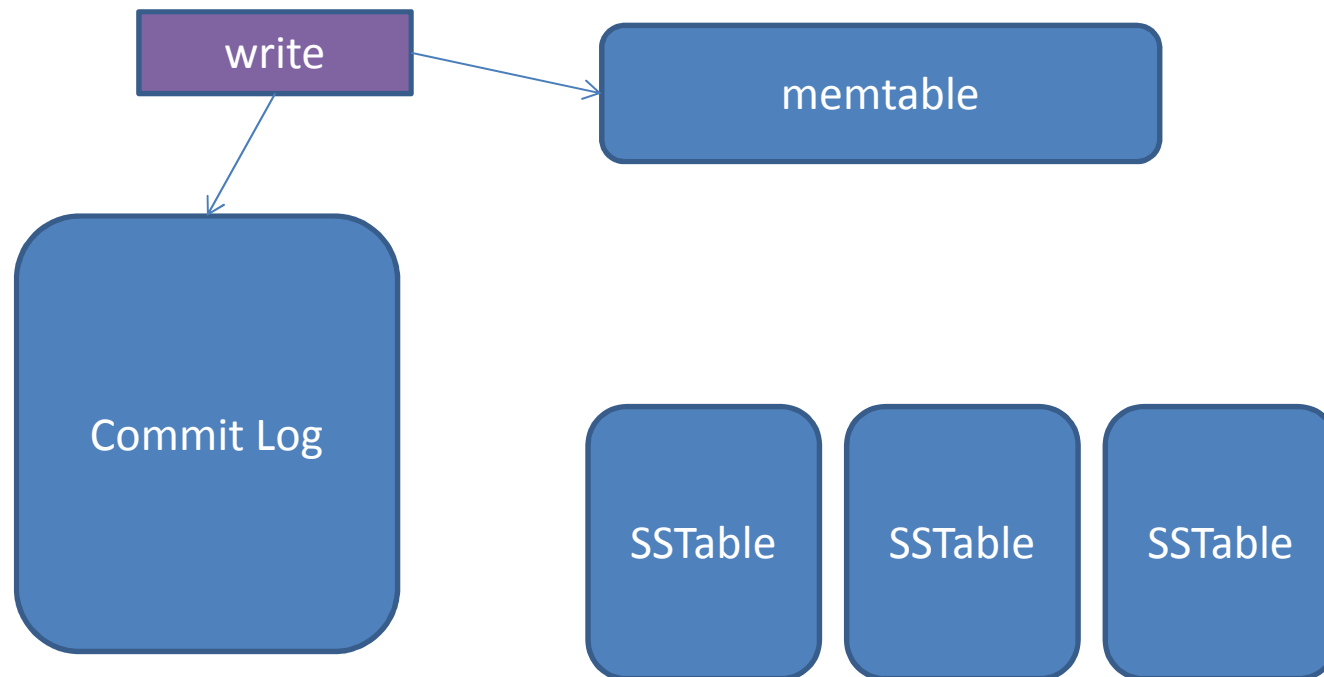
Εξυπηρέτηση αιτήσεων



Εγγραφή δεδομένων

- Έλεγχος του είδους της εγγραφής και των δικαιωμάτων του χρήστη
- Οι αλλαγές καταγράφονται στο commit log
- Το memtable ενημερώνεται όταν ολοκληρωθεί η εγγραφή

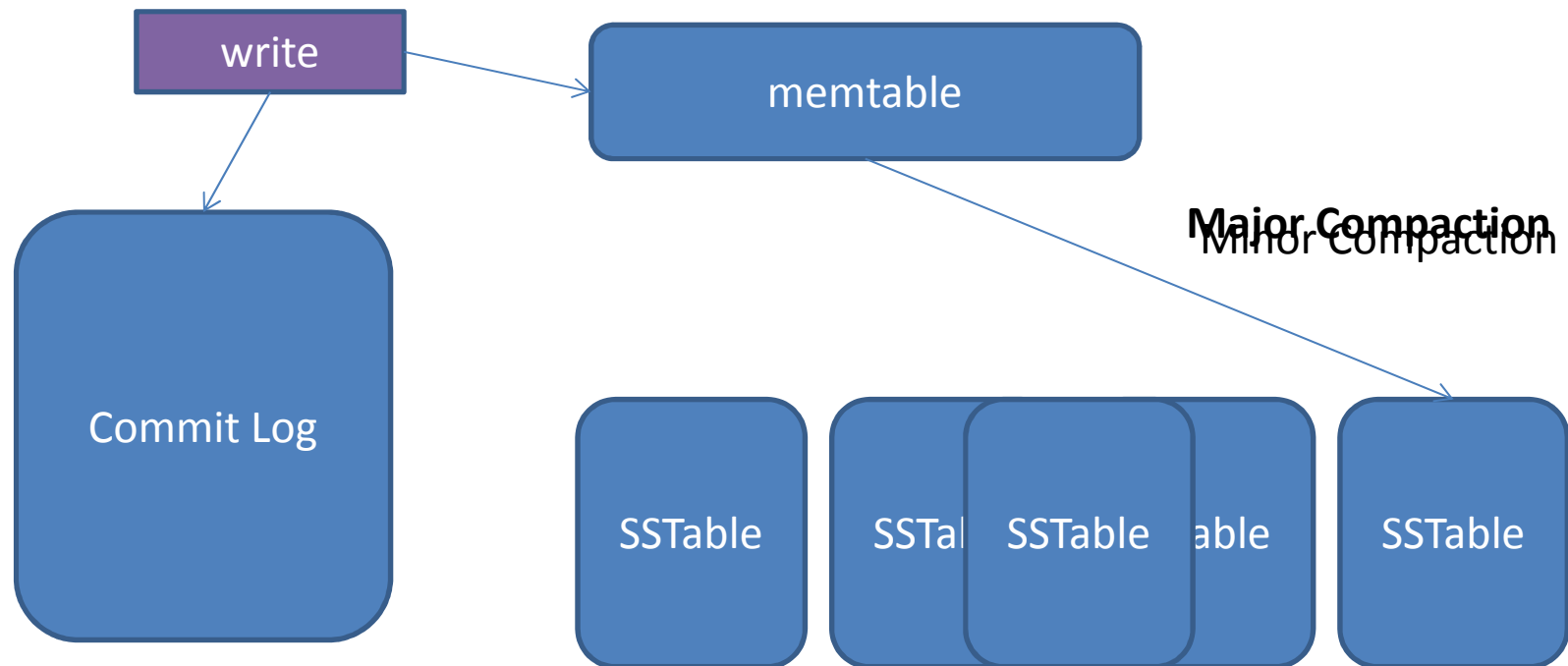
Εγγραφή δεδομένων



Συμπύκνωση (compaction)

- Όταν το memtable μεγαλώσει αρκετά:
 - Minor compaction
 - Σταματάει να χρησιμοποιείται
 - Δημιουργείται ένα νέο
 - Το αρχικό εγγράφεται ως SSTable στο GFS
- Major compaction: περιοδικά τα SSTables συγχωνεύονται
 - Αποφεύγεται η δημιουργία πολλών αρχείων από πολλά minor compactions

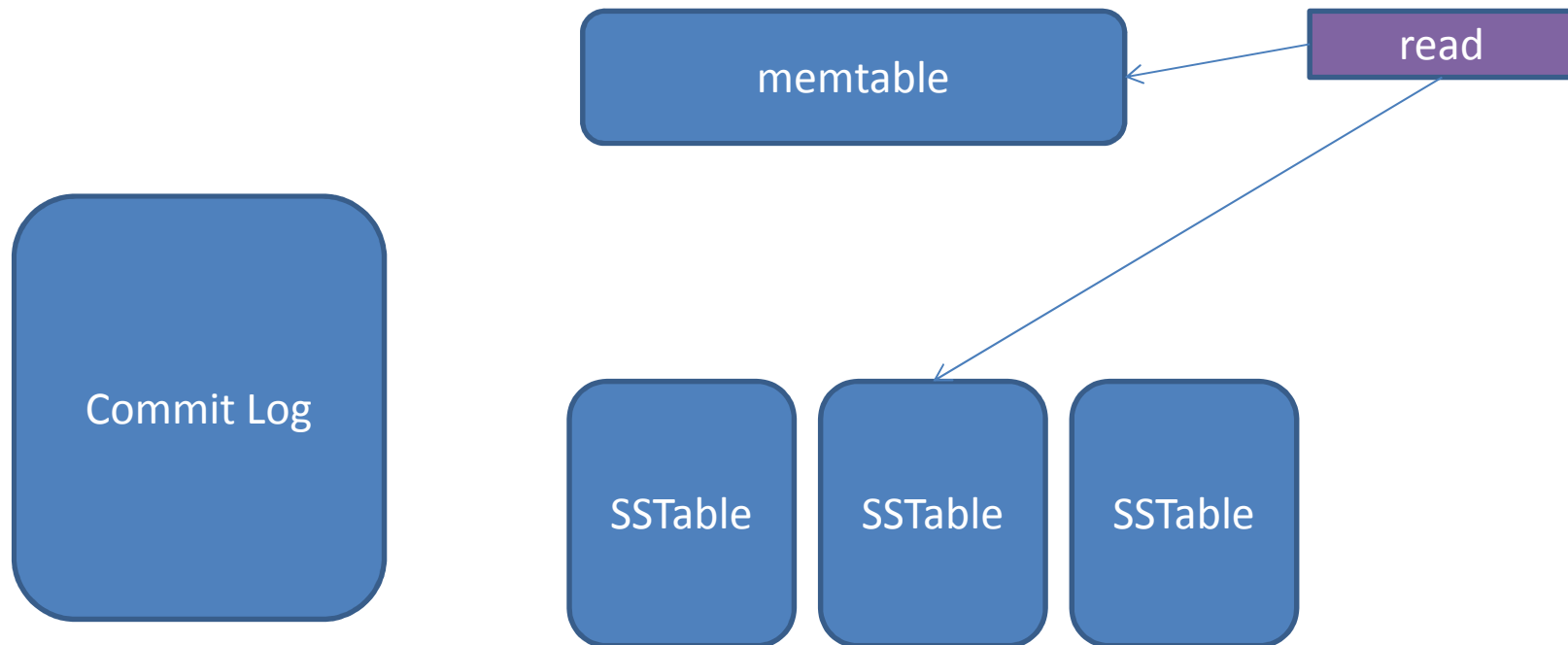
Συμπύκνωση



Ανάγνωση δεδομένων

- Έλεγχος του είδους της ανάγνωσης και των δικαιωμάτων του χρήστη
- Η ανάγνωση πραγματοποιείται σε συνδυασμένα δεδομένα από το SSTable και του memtable

Ανάγνωση δεδομένων



Βελτιώσεις

- Locality groups
 - Με χρήση τους ομαδοποιούνται τα column families
 - Ένα ξεχωριστό SSTable δημιουργείται για καθένα
 - Διευκολύνουν την διαχείριση των πινάκων

Βελτιώσεις

- Συμπύεση
 - Ο χρήστης μπορεί να καθορίσει το επίπεδο της
 - Εφαρμόζεται σε κάθε SSTable block ξεχωριστά
 - Ταχύτητα εναντίον μεγάλης συμπίεσης (φτηνοί δίσκοι)
 - Δεν απαιτείται αποσυμπύεση ολοκληρου του αρχείου
 - Λόγοι 10/1 σε σχέση με τυπικό zip 3/1 επειδή τα δεδομένα που είναι «κοντά» μοιάζουν μεταξύ τους.

Βελτιώσεις

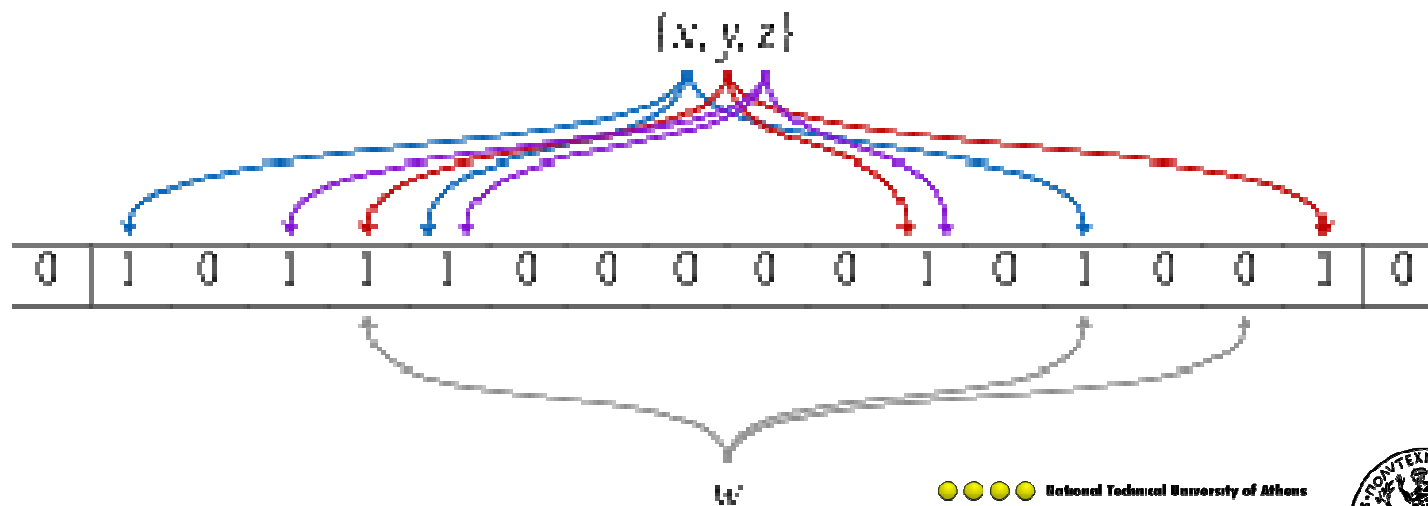
- Caching
 - Πραγματοποιείται σε δύο επίπεδα
 - Η Scan Cache διατηρεί τα ζευγη key-value(υψηλό επίπεδο)
 - Η Block Cache διατηρεί ολόκληρα blocks από το SSTable(χαμηλό επίπεδο)

Βελτιώσεις

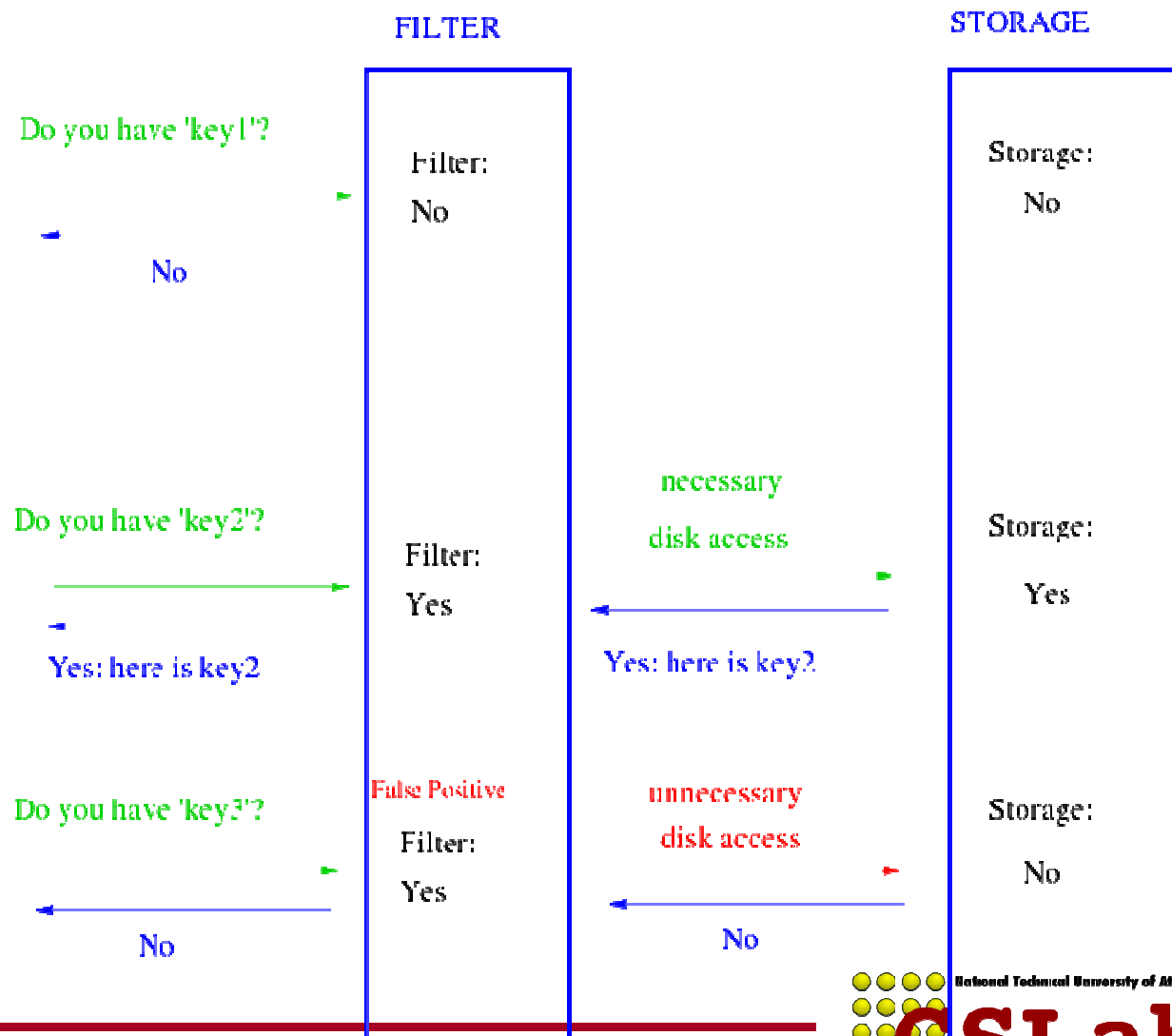
- Bloom filters
 - Ο χρήστης καθορίζει αν θα χρησιμοποιηθούν
 - Χρησιμοποιείται για να καθοριστεί αν κάποιο SSTable περιέχει δεδομένα από συγκεκριμένο row χωρίς να το ανακτήσουμε
 - Με λίγο αποθηκευτικό χώρο παραπάνω μπορούμε να αποκλείουμε αμέσως μεγάλο αριθμό sstables.
 - Πιθανά false positives: Σε αυτή την περίπτωση απλά δεν θα βρει κάτι στο sstable.
 - Ποτέ false negatives: ότι δεν βρίσκει, σίγουρα δεν υπάρχει.
 - Δημιουργούνται βάση locality group

Παράδειγμα Bloom Filter

- Array of m bits
- K hash functions
- Empty set \rightarrow all 0s
- Add element: k hash functions \rightarrow k positions set to 1



Παράδειγμα Bloom Filter



Βελτιώσεις

- Υλοποίηση του commit log
 - Υπάρχει μόνο ένα ανά tablet server (και όχι ανά tablet)
 - Αλλιώς θέλαμε πολλά ταυτόχρονα ανοιχτά αρχεία ανά server
 - Βελτιώνει την απόδοση, αλλά προκαλεί προβλήματα σε περίπτωση σφάλματος
 - Tablets γίνονται re-assign σε πολλούς νέους servers
 - Χρειάζονται όλοι να σκανάρουν όλο το log για να βρουν τα tablets που τους ενδιαφέρουν (για REDO/UNDO)
- Ταξινόμηση βάσει του key
 - Σπάσιμο του commit log σε 64MB chunks για παραλληλοποίηση του sorting
 - Όλες οι αλλαγές ενός tablet βρίσκονται κοντά: μόνο ένα disk seek και seq. read για recovery ενός tablet

HBase

- Κλώνος του Bigtable
- Ανοιχτού κώδικα
- Apache project
- «Συνοδεύει» το Hadoop
 - HDFS αντί GFS
 - Μπορεί να χρησιμοποιηθεί από το Hadoop MapReduce
- Java

Αντιστοιχίες

BigTable

- Master
- TabletServer
- SSTable
- Tablet
- Chubby
- GFS

HBase

- HMaster
- Region Server
- Hfile (περίπου)
- TableRegion
- Zookeeper
- HDFS

Ερωτήσεις?

