Google File System, HDFS, BigTable, Hbase

Κατανεμημένα Συστήματα

2017-2018

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

Περιεχόμενα

- File systems – distributed File systems
- GFS
- HDFS
- BigTable
- HBase

File System

- Originally developed for centralized computer systems and desktop computers.
- Operating system facility providing a convenient programming interface to disk storage
- File systems are responsible for the organization, storage, retrieval, naming, sharing and protection of files.
- Files contain both data and attributes

Γενικά

- Σε ένα Κατανεμημένο σύστημα
  - 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.

Distributed File System

- Implement a common file system that can be shared by all autonomous computers in a distributed system

Απαιτήσεις

- Συνέπεια – consistency
- Ανοχή σε σφάλματα – fault tolerance
- Διαθεσιμότητα – high availability
- Κλιμακωσιμότητα – scalability
- Διαφάνεια – transparency
- Ταυτοχρονισμό – concurrency
- Απόδοση – efficiency

Αρχιτεκτονικές

- Fully distributed: files distributed to all sites
  - Issues: performance, implementation complexity
- Client-server Model:
  - Fileserver: dedicated sites storing files perform storage and retrieval operations
  - Client: rest of the sites use servers to access files

Λειτουργίες (1)

Λειτουργίες ενός κατανεμημένου συστήματος αρχείων:

- Ονομασία (Name Server)
  - Provides mapping (name resolution) the names supplied by clients into objects (files and directories)
  - Takes place when process attempts to access file or directory the first time.
- Διαμοιρασμός

Λειτουργίες (2)

- Προσωρινή αποθήκευση: caching
  - Improves performance
  - Caching at the client
    - When client references file at server: Copy of data brought from server to client machine
    - Subsequent accesses done locally at the client
  - Caching at the server:
    - File saved in memory to reduce subsequent access time
    - Issue: different cached copies can become inconsistent. Cache managers (at server and clients) have to provide coordination.
- Replication

GFS: Google File System

- Proprietary distributed file system
- 2003 paper: Symposium on Operating Systems Principles
- Successor: Colossus, released 2010

GFS: Google File System

- Κλιμακώσιμο κατανεμημένο σύστημα αρχείων
- Modest number of large files
- Κατάλληλο για data-intensive εφαρμογές
- Ανοχή σε σφάλματα
- Εξυπηρετεί ένα μεγάλο σύνολο από πελάτες.
  - Need semantics for concurrent appends

Παραδοχές

- Υψηλή συχνότητα βλαβών.
- Τα αρχεία σε τέτοιου είδους συστήματα είναι μεγάλα (πολλά GBs)
- Πρόσβαση αρχείων
  - Large streaming reads + small random reads
  - Many large sequential writes
  - Τα περισσότερα αρχεία τροποποιούνται με προσάρτηση(append)
  - Random access overwrites don’t need to be efficient
- Ειδικός χειρισμός για παράλληλο append
- Ταιριάζουν για data analytics
  - Συνήθως μεγάλα αρχεία.
  - Το υψηλό bandwidth προτιμάται από το χαμηλό latency.

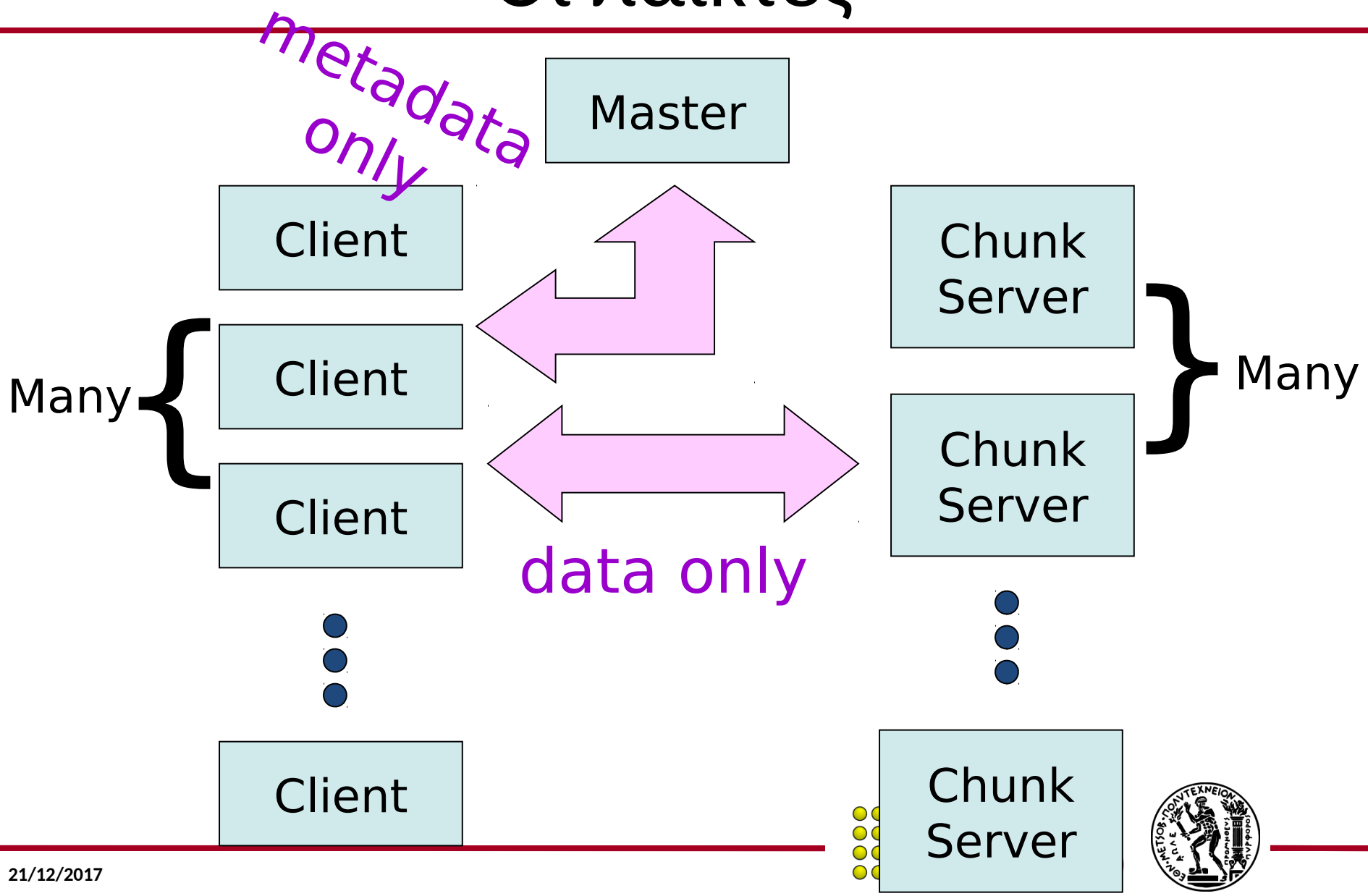
Διεπαφή συστήματος αρχείων

- create/delete
- open/close
- read/write
- Snapshot
  - Low cost
- record append
  - Atomicity with multiple concurrent writes

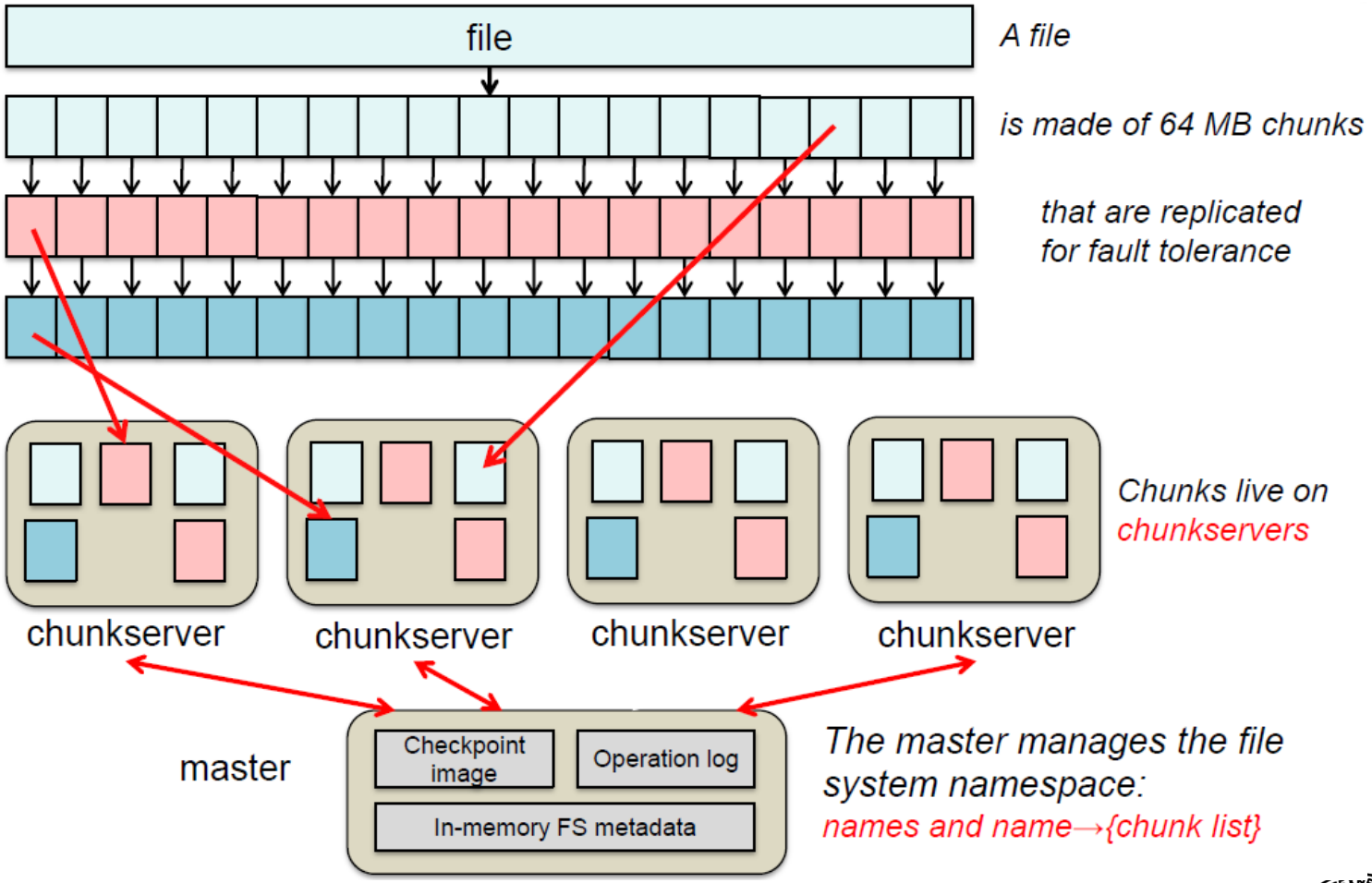
Αρχιτεκτονική

- Ένας master
- Πολλαπλοί slaves -> chunkservers
- Πολλαπλοί clients
- Τα αρχεία χωρίζονται σε chunks σταθερού μεγέθους
- Δεν πραγματοποιείται caching στα data (μόνο σε metadata)

Οι παίκτες


metadata only
Master
Client
Client
Client
Client
Many
Chunk Server
Chunk Server
Chunk Server
Many
data only

GFS Files



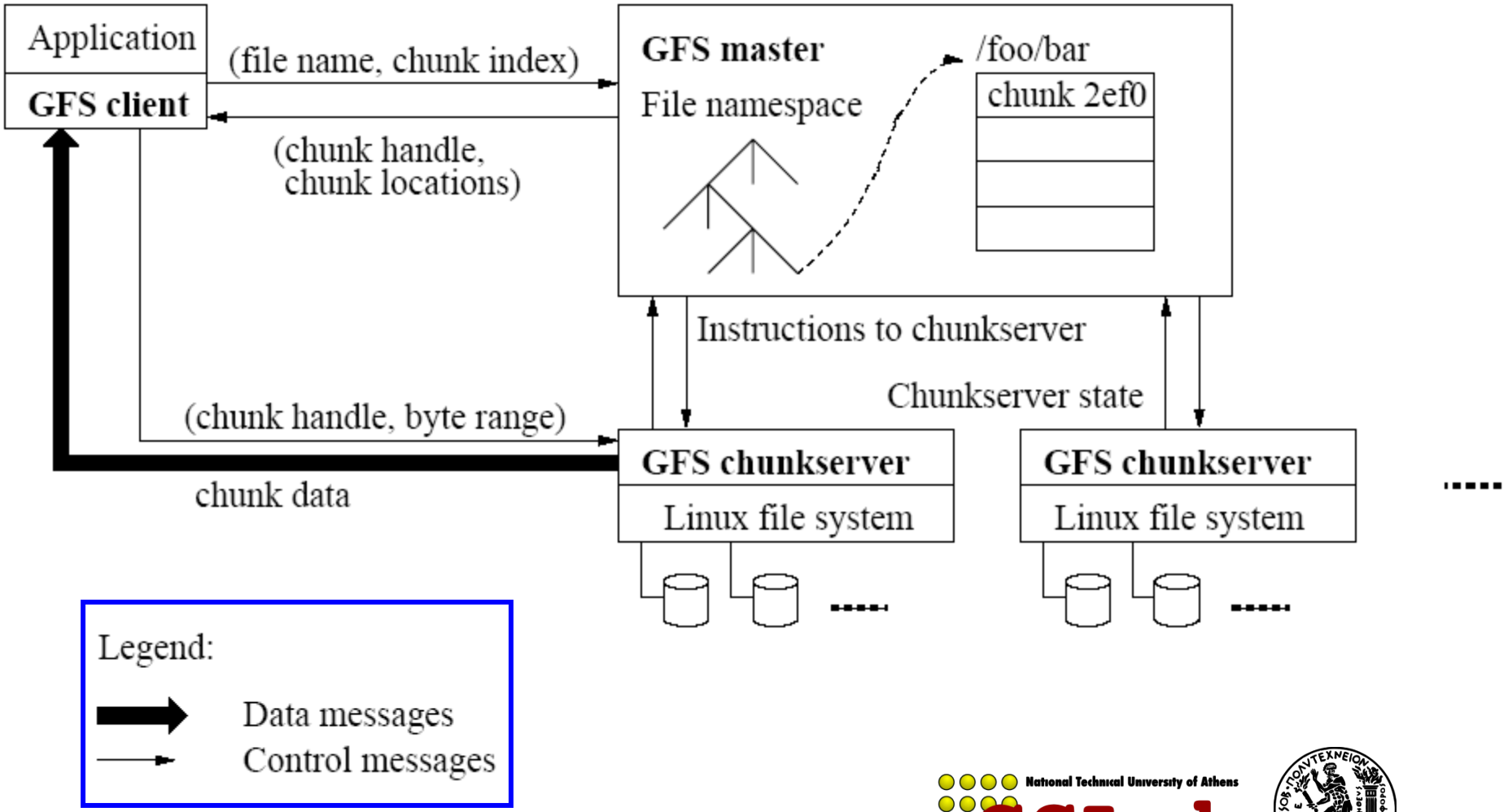
Μέγεθος chunk

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

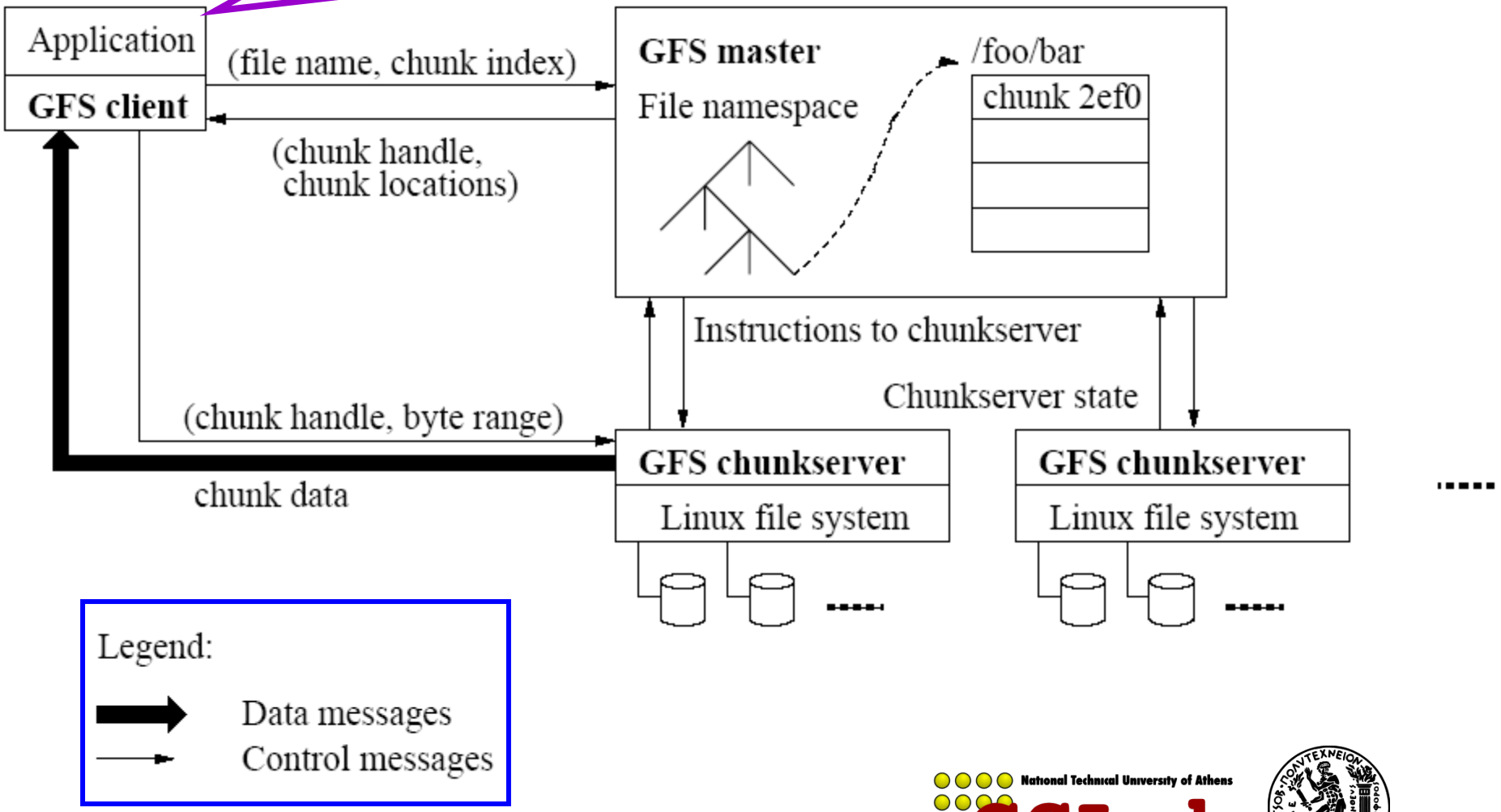
Master

- Ο master διατηρεί:
  - Το namespace
  - Την αντιστοιχία από αρχεία σε chunks
  - Τις θέσεις των chunks (και των replicas)
- 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
- Having one master -> global knowledge
  - Allows better placement / replication
  - Simplifies design

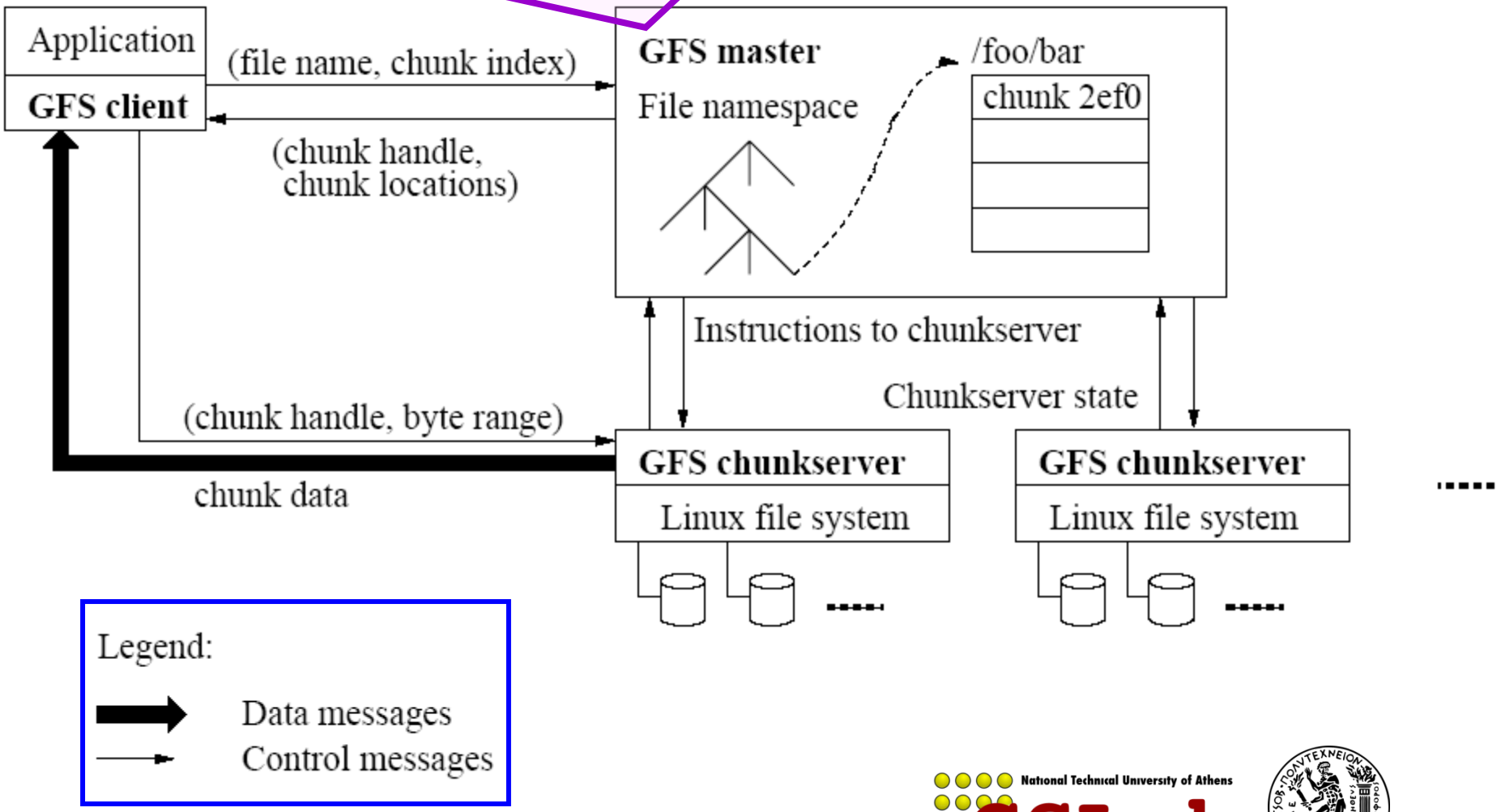
Client Read



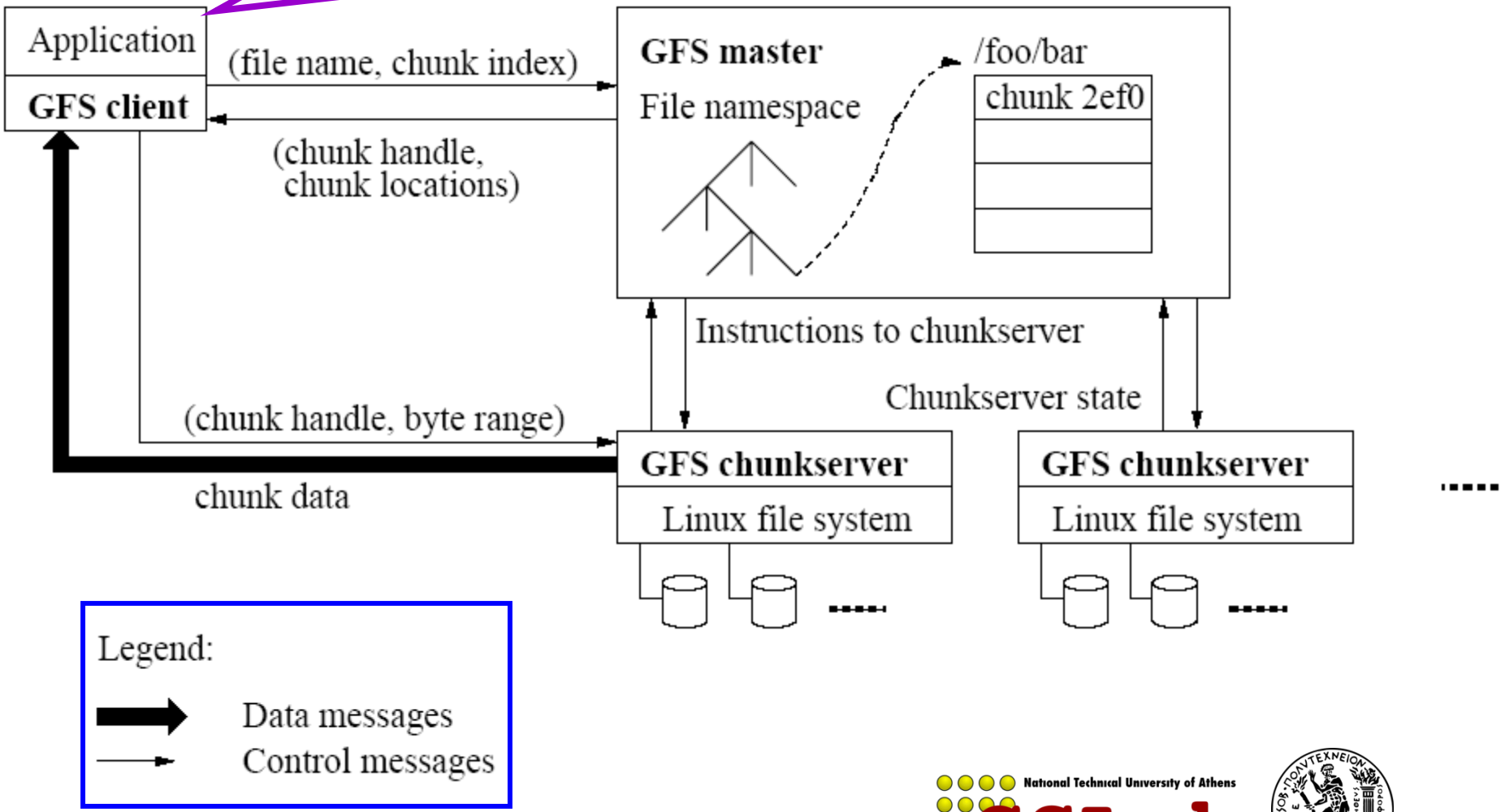
Using fixed chunk size, translate filename & byte offset to chunk index.
Send request to master

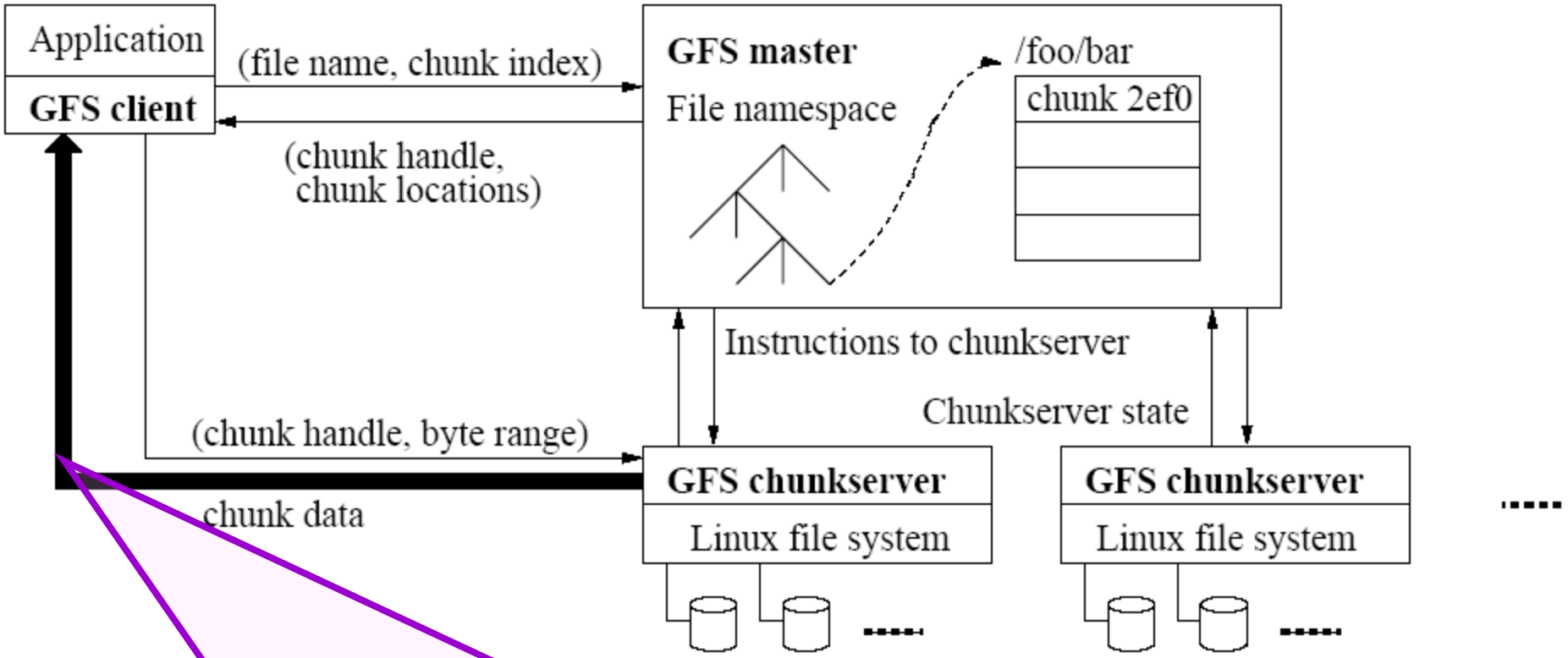


Replies with chunk handle & location of chunkserver replicas (including which is 'primary')



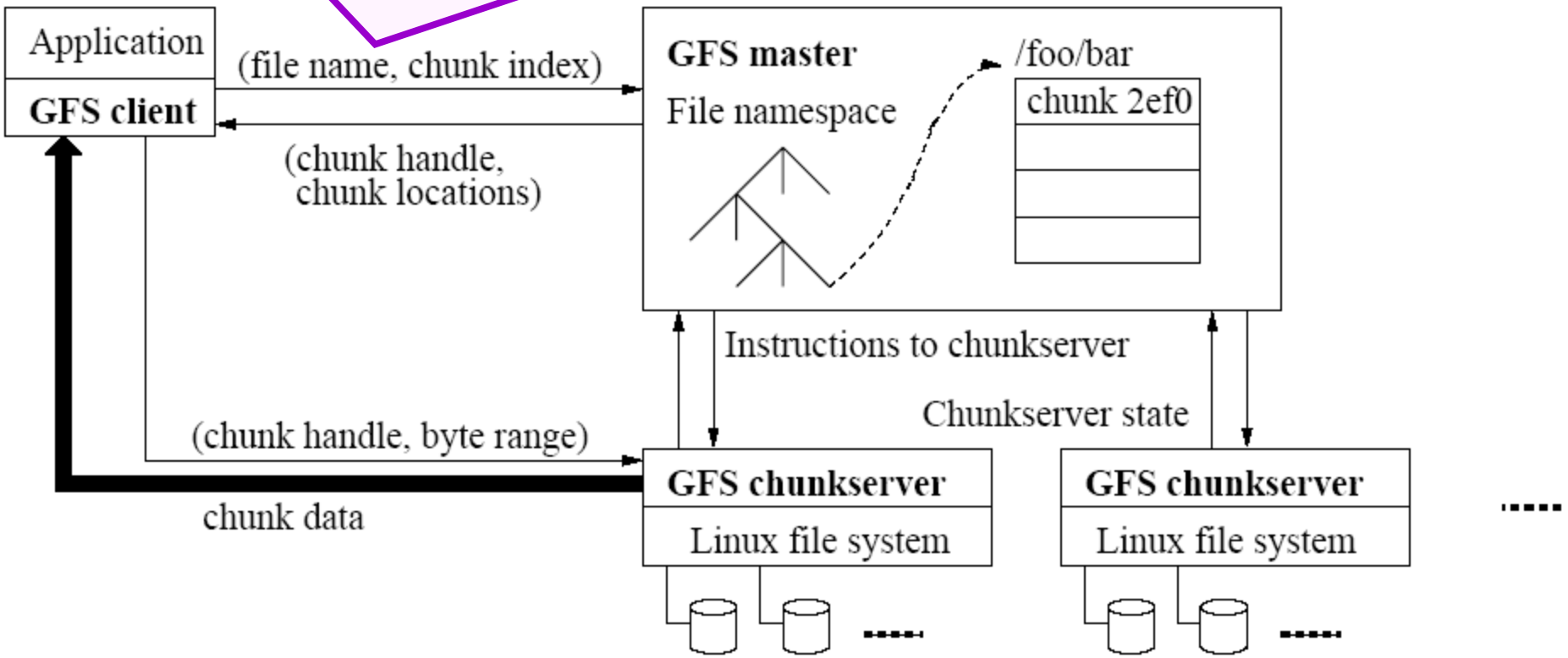
Cache info
using filename & chunk index as key



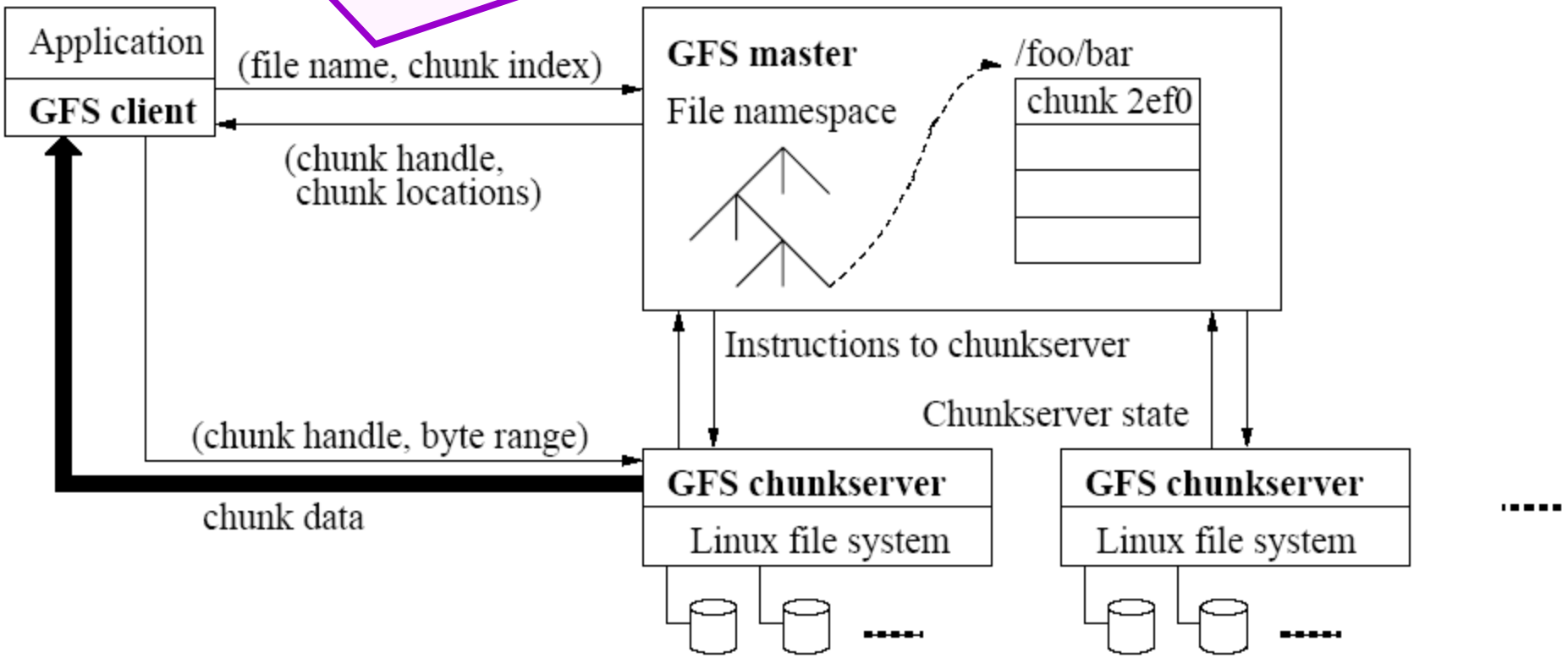


Request data from nearest chunkserver
“chunkhandle & index into chunk”

No need to talk more
About this 64MB chunk
Until cached info expires or file reopened



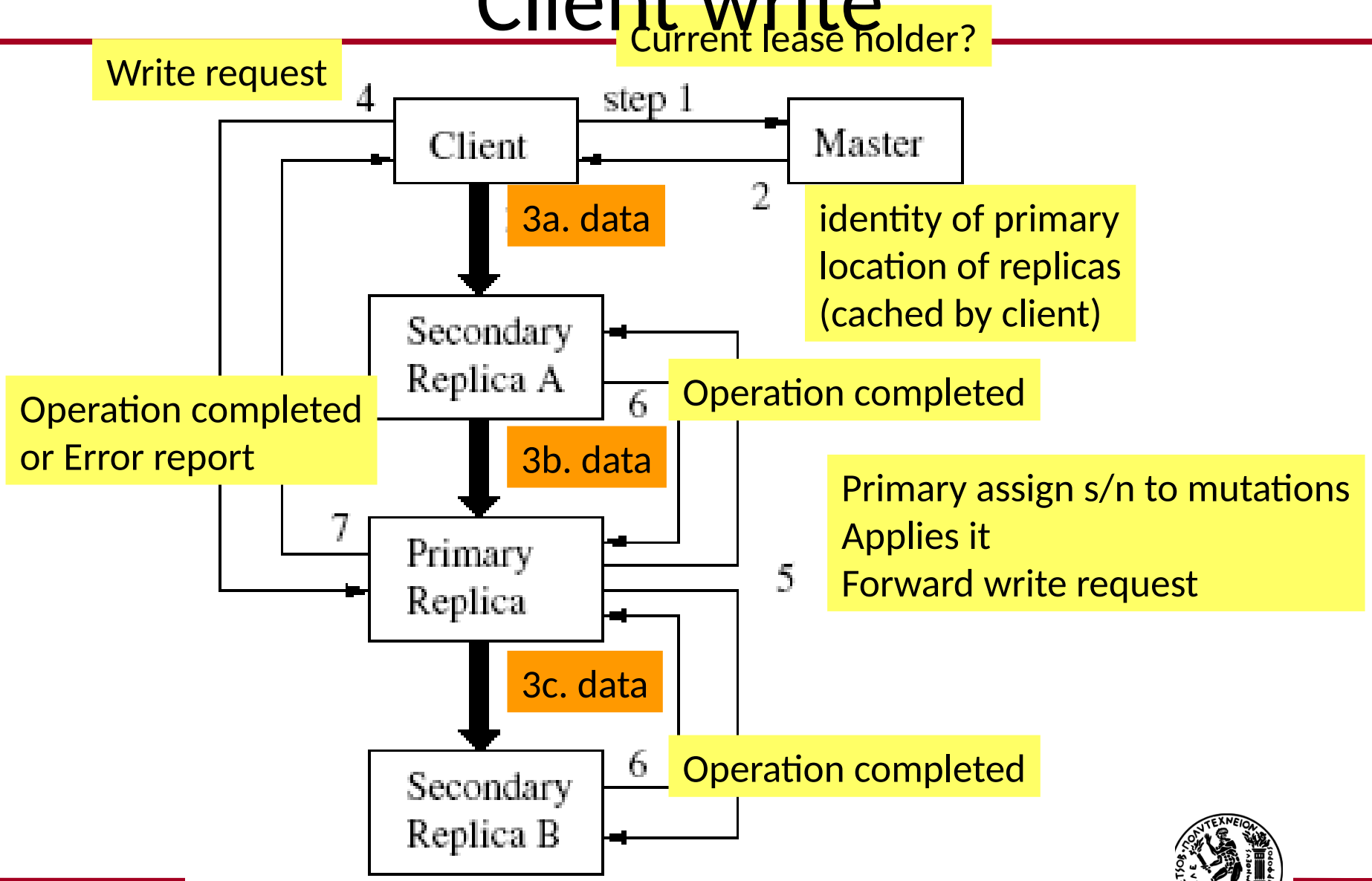
Often initial request asks about
Sequence of chunks



Lease, mutation και ροή δεδομένων

- Mutation είναι κάθε αλλαγή στα δεδομένα ενός αρχείου και εφαρμόζεται σε όλα τα αντίγραφα του.
- Objective
  - Ensure data consistency
  - Minimize load on master
- Ο master δίνει lease σε ένα replica -> primary και διαχειρίζεται τα mutation στα υπόλοιπα αντίγραφα.
- Η ροή δεδομένων διαχωρίζεται από τη ροή ελέγχου.
- Τα mutations διαδίδονται σειριακά από ένα chunkserver στον κοντινότερο με pipelining.

Client write



Atomic Appends

- Google uses large files as queues between multiple producers and consumers
- Same control flow as for writes, except…
- Client pushes data to replicas of last chunk of

file

- Client sends request to primary
- Common case: request fits in current last chunk:
  - Primary appends data to own replica
  - Primary tells secondaries to do same at same byte offset in theirs
  - Primary replies with success to client

Atomic Appends

- When data won't fit in last chunk:
  - Primary fills current chunk with padding
  - Primary instructs other replicas to do same
  - Primary replies to client, "retry on next chunk"
- If record append fails at any replica, client retries operation
  - So replicas of same chunk may contain different data - even duplicates of all or part of record data
- What guarantee does GFS provide on success?
  - Data written at least once in atomic unit

Consistency (1)

- Changes to namespace (i.e., metadata) are

Atomic

  - Done by single master server!
  - Master uses log to define global total order of namespace-changing operations

Consistency (2)

- Changes to data are ordered as chosen by a Primary
  - All replicas will be consistent
  - But multiple writes from the same client may be interleaved or overwritten by concurrent operations from other clients
- Record append completes at least once, at offset of GFS's choosing
  - Applications must cope with possible duplicates

Consistency Model

	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

Consistent = all clients see same data

Consistency Model

	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

Defined = consistent + clients see full effect of mutation
Key: all replicas must process chunk-mutation requests in ***same order***

Consistency Model

	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

Different clients may see different data

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.
- ΄Ισος 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

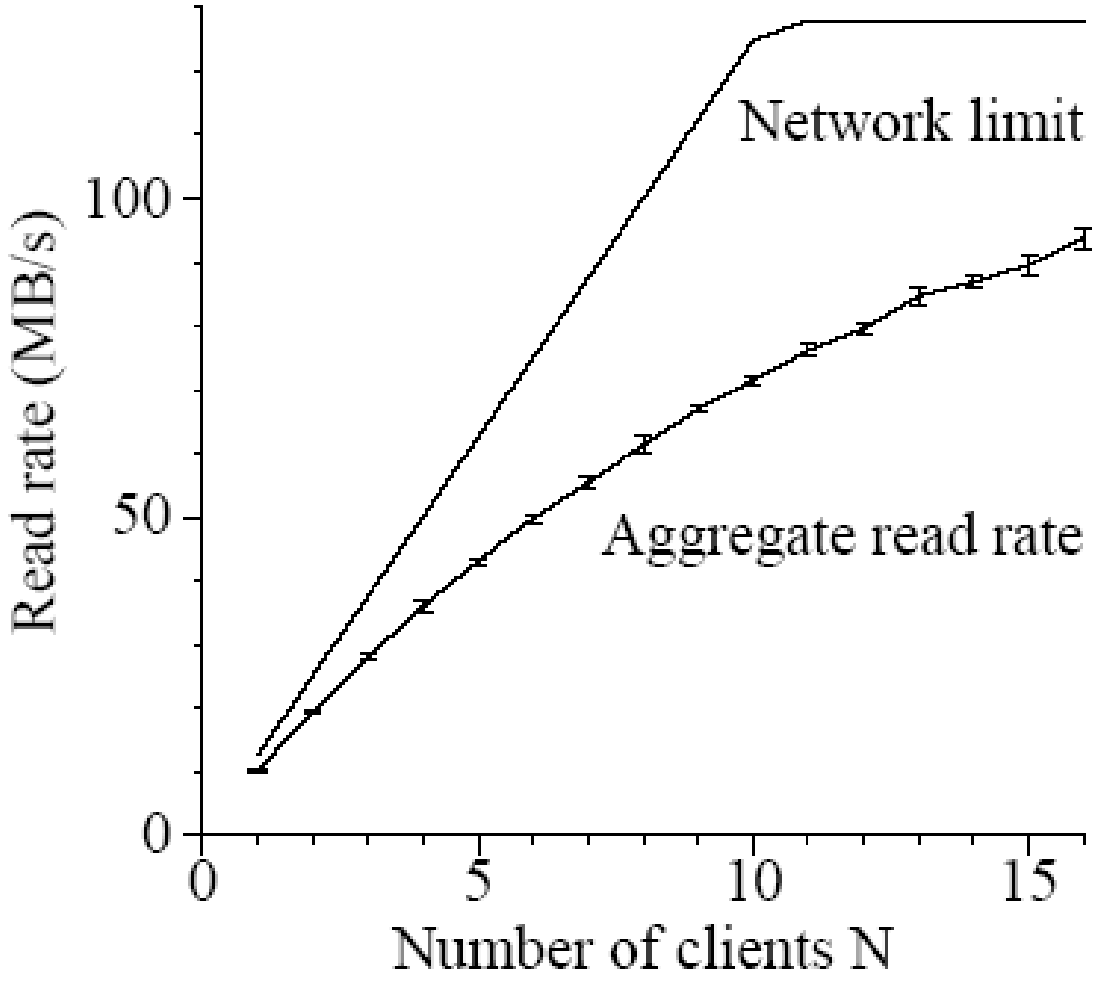
What If the Master Reboots?

- Replays log from disk
  - Recovers namespace (directory) information
  - Recovers file-to-chunk-ID mapping
- Asks chunkservers which chunks they hold
  - Recovers chunk-ID-to-chunkserver mapping
- If chunk server has older chunk, it's stale
  - Chunk server down at lease renewal
- If chunk server has newer chunk, adopt its version number
  - Master may have failed while granting lease

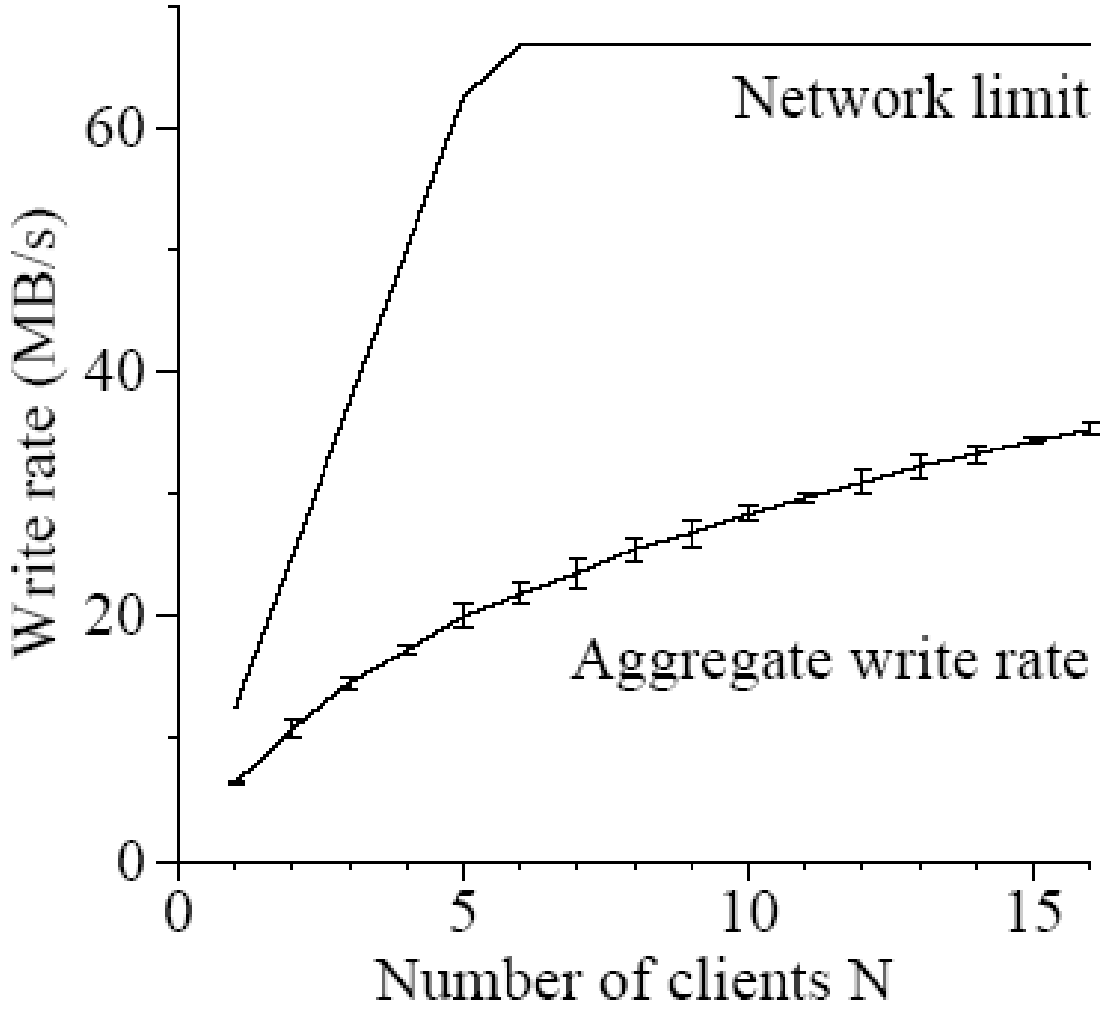
What if chunk server fails?

- Master notices missing heartbeats
  - Master decrements count of replicas for all chunks on dead chunkserver
- Master re-replicates chunks missing replicas in background
  - Highest priority for chunks missing greatest number of replicas

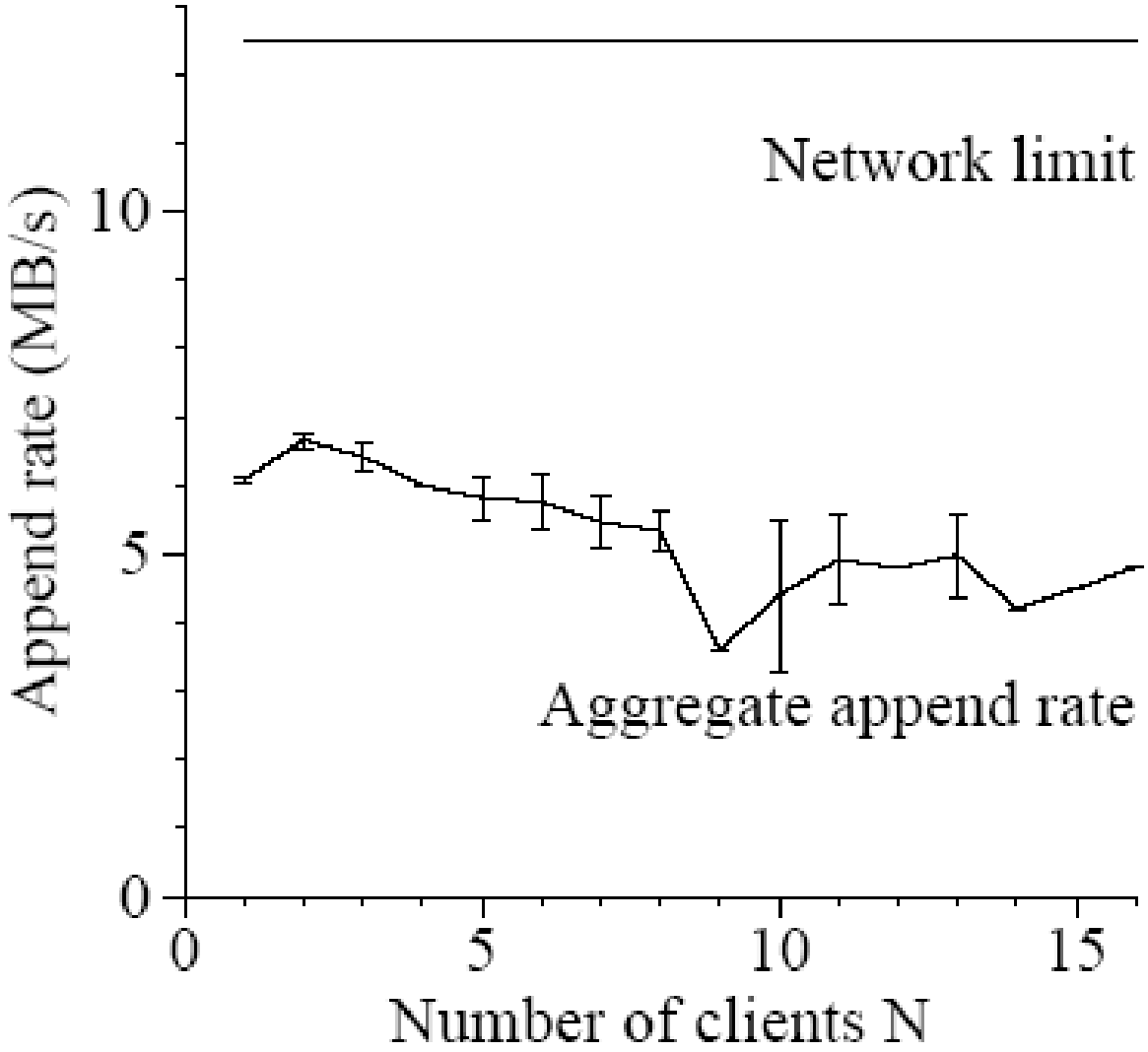
Read Performance



Write Performance



Record-Append Performance



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

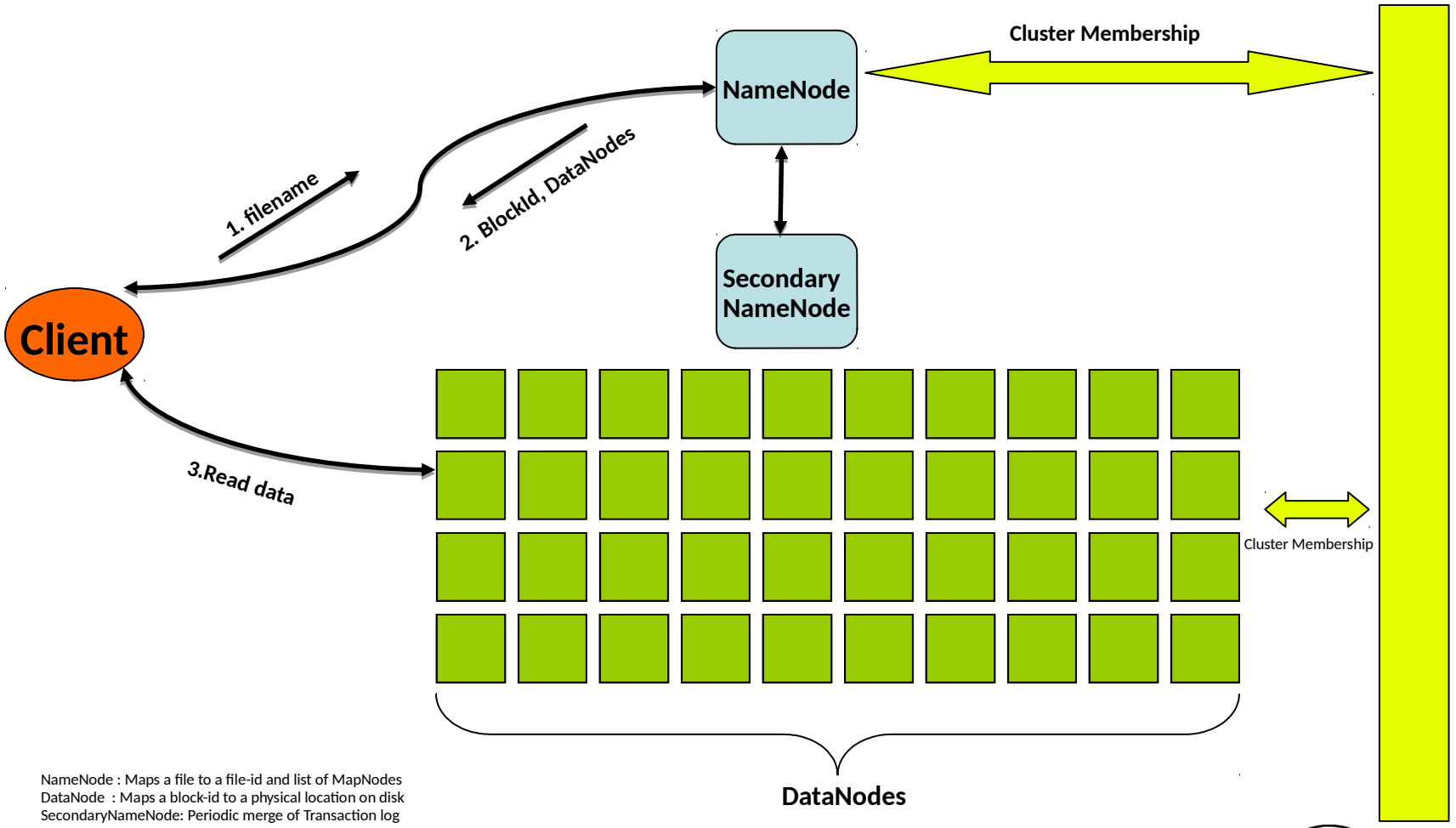
Πλεονεκτήματα του Hadoop HDFS

- 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 προσπελαύνουν 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

National Technical University of Athens
CSLab

NameNode DataNode

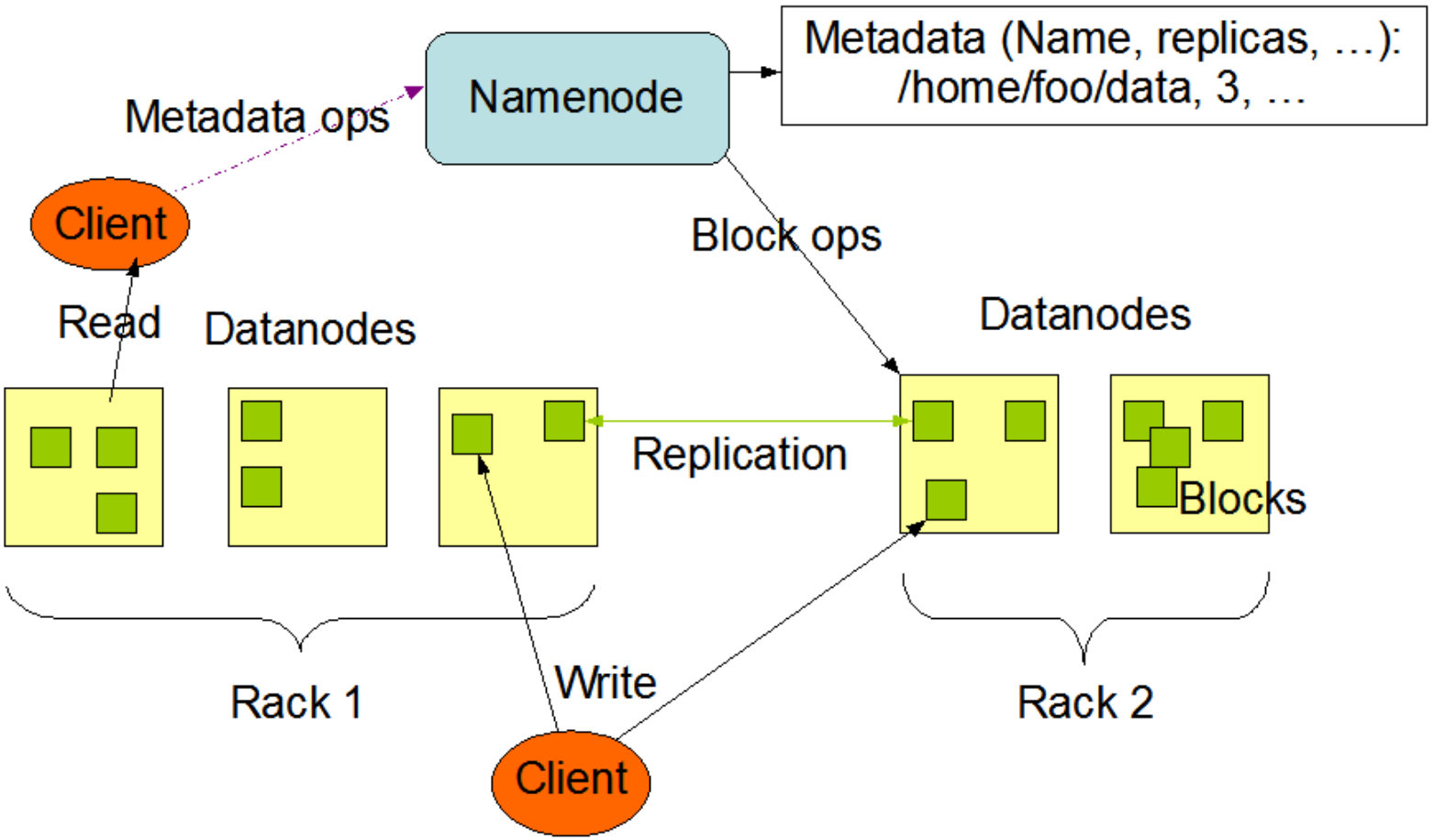
NameNode

- **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 μεταδεδομενα 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

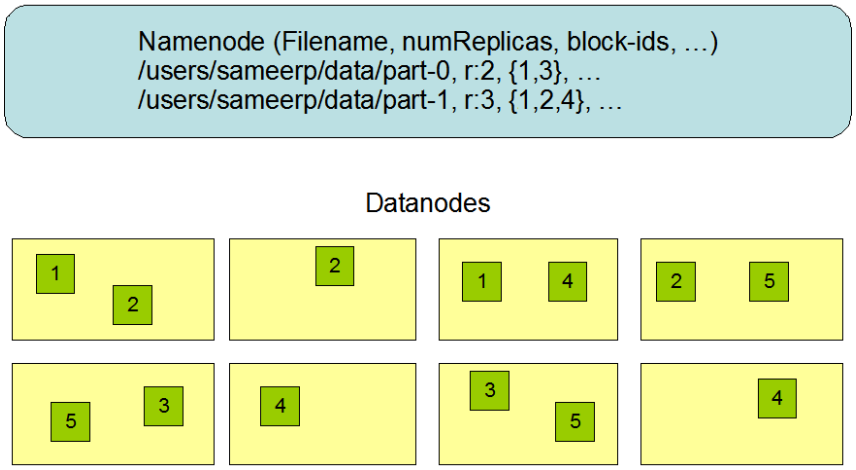
Εγγραφή – Ανάγνωση στο HDFS



Write Data Pipelining

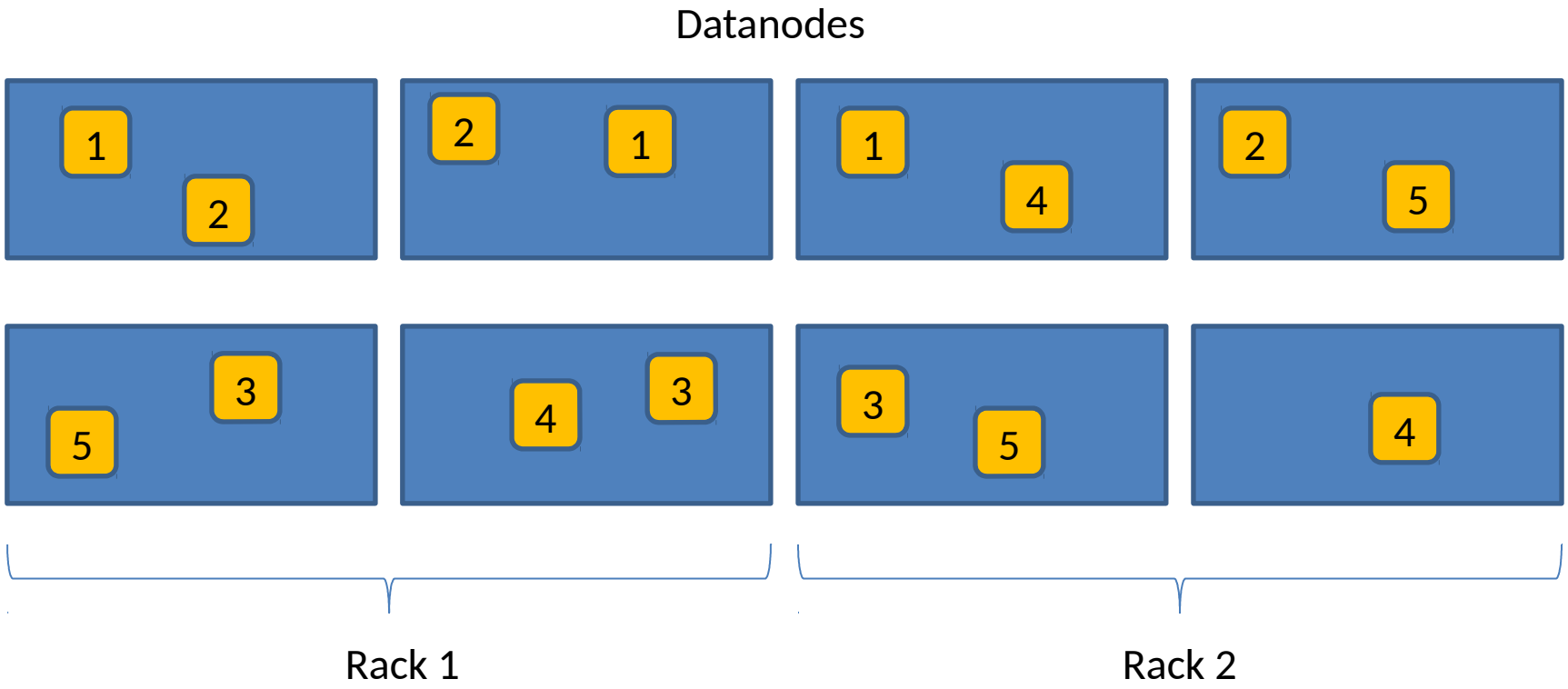
- Ο Client λαμβάνει μια λίστα από DataNodes στους οποίους θα δημιουργηθούν τα αντίγραφα του block
- Ο Client γράφει το block στον πρώτο DataNode
- Ο Πρώτος DataNode προωθεί τα δεδομένα στον επόμενο DataNode του με Pipeline
- Όταν όλα τα δεδομένα έχουν γραφτεί ο Client συνεχίζει την εγγραφή του επόμενου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

NameNode αντίγραφα Blocks



- **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 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
- 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

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

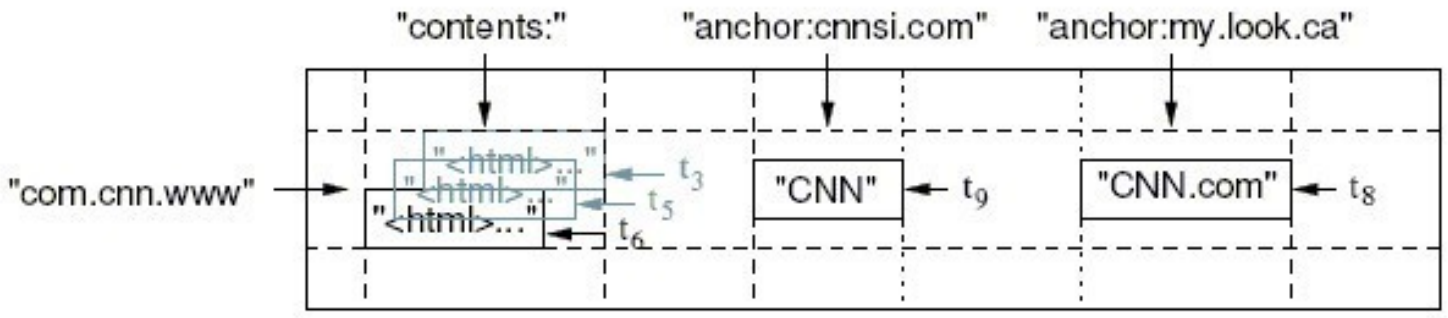
Στήλες (columns)

- 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?

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

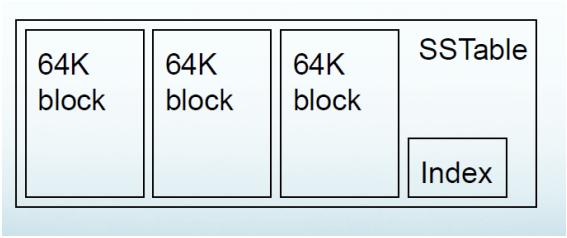
Αρχιτεκτονική

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

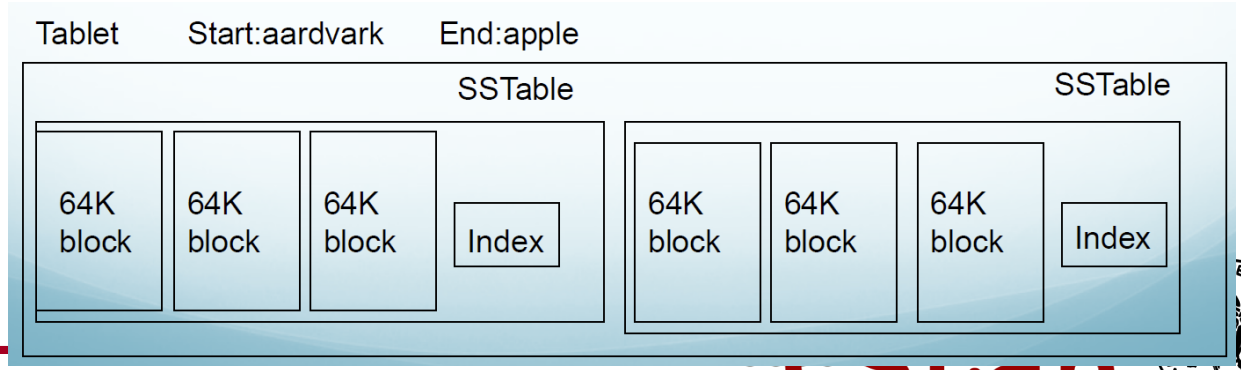
Tablets

- 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

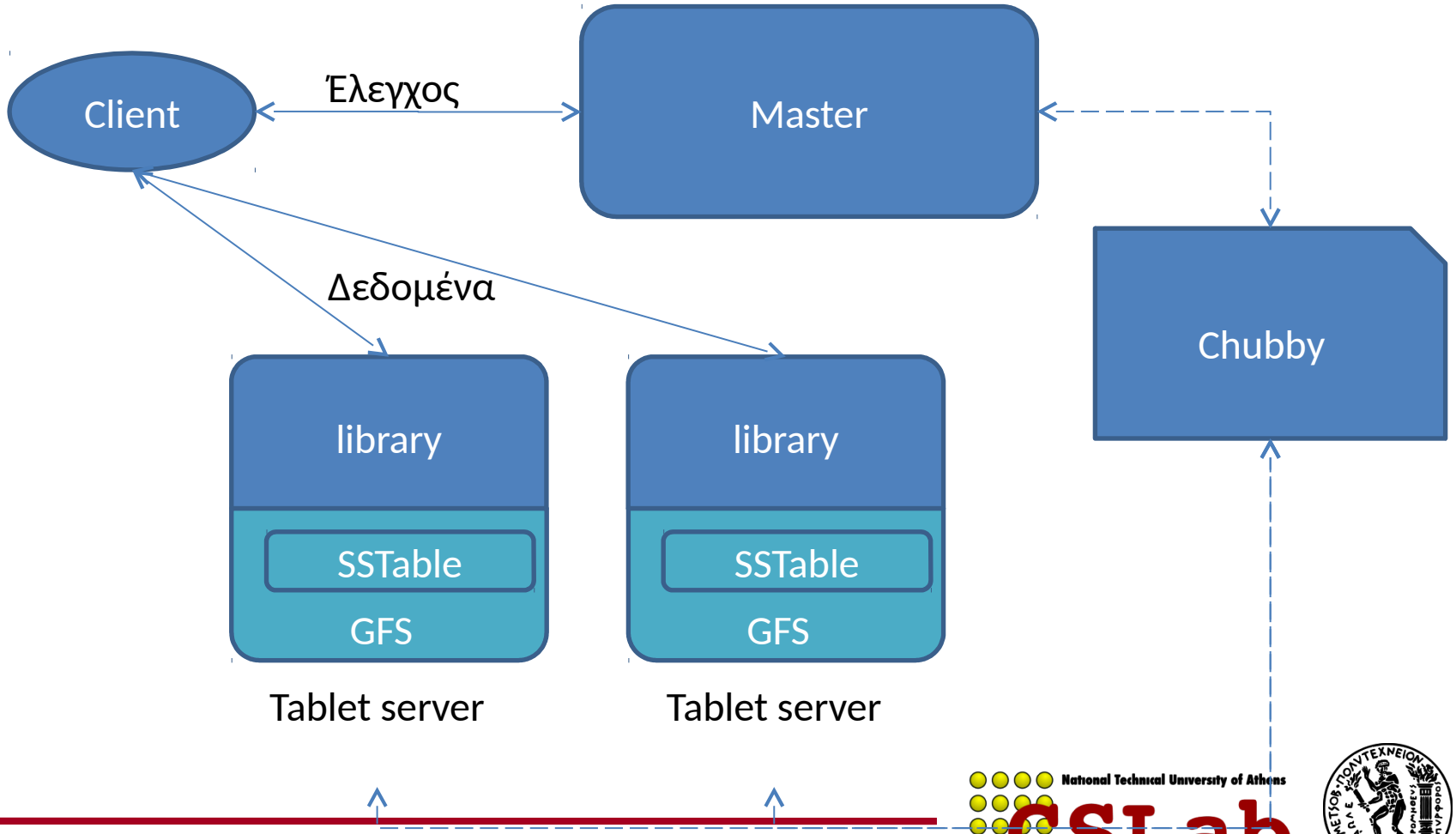
SSTable:



Tablet:



Αρχιτεκτονική



Master

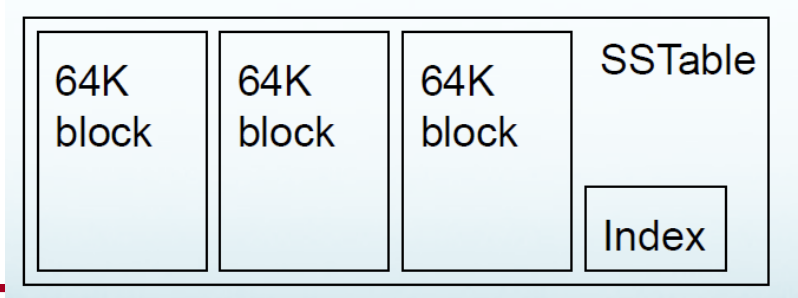
- Ανάθεση tablet σε κάποιον tablet server
- Εντοπισμός νέων tablet servers
- Εξισσόροπηση φόρτου
- Συλλογή σκουπιδιών από το GFS
- Διαχείριση schema
- Δεν περνάνε δεδομένα από αυτόν
- Τρέχει μαζί με τον Master του GFS.

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.



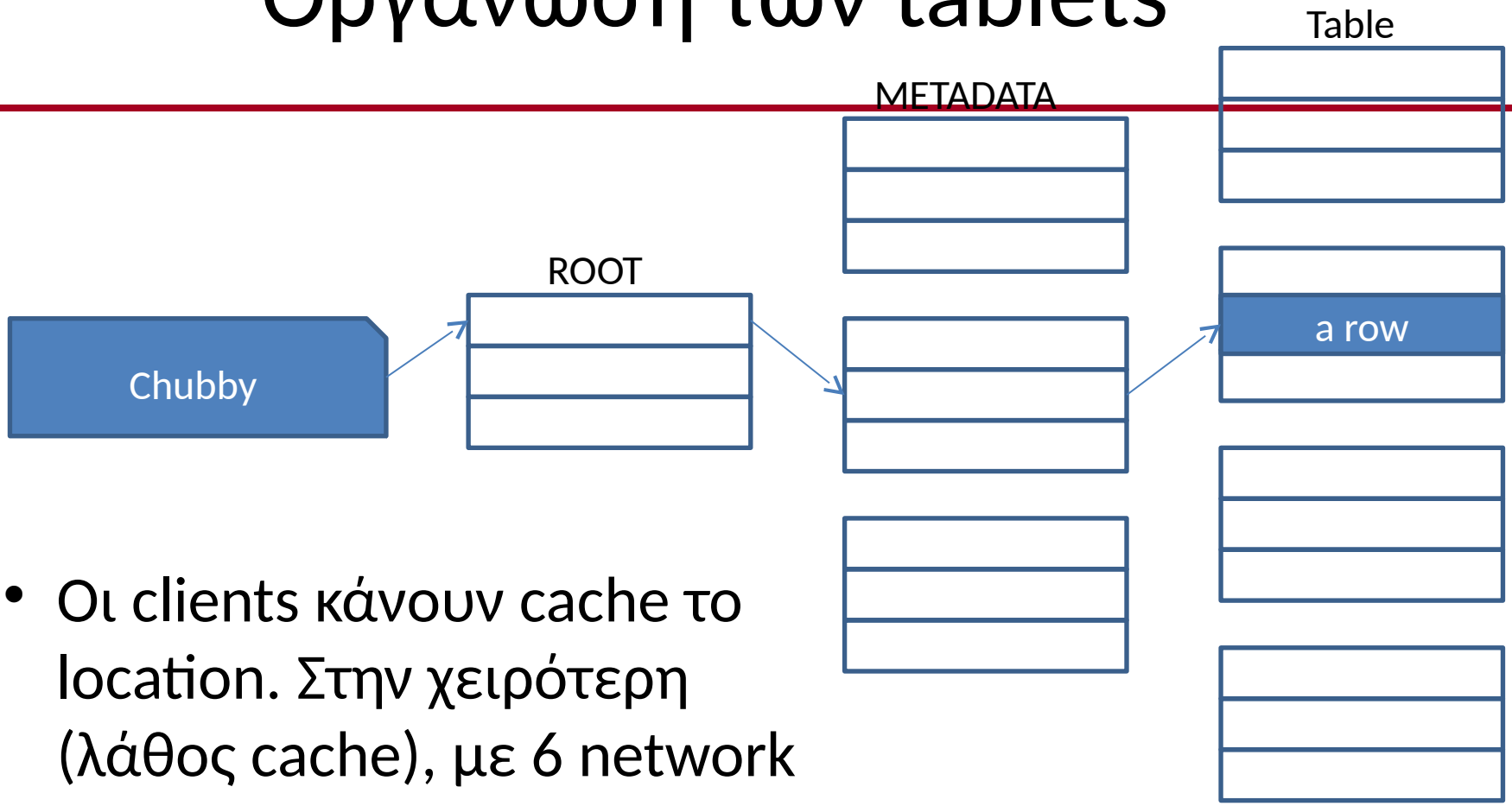
Chubby

- 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.

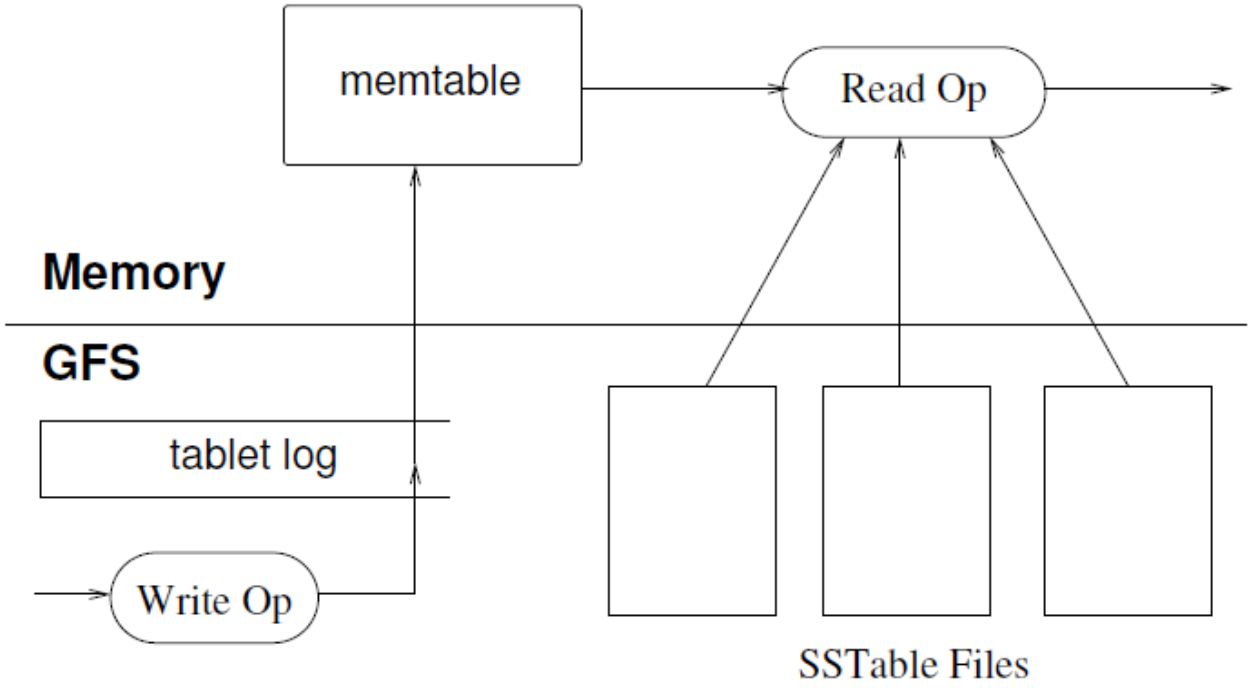
Ανάθεση των tablets

- 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

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)

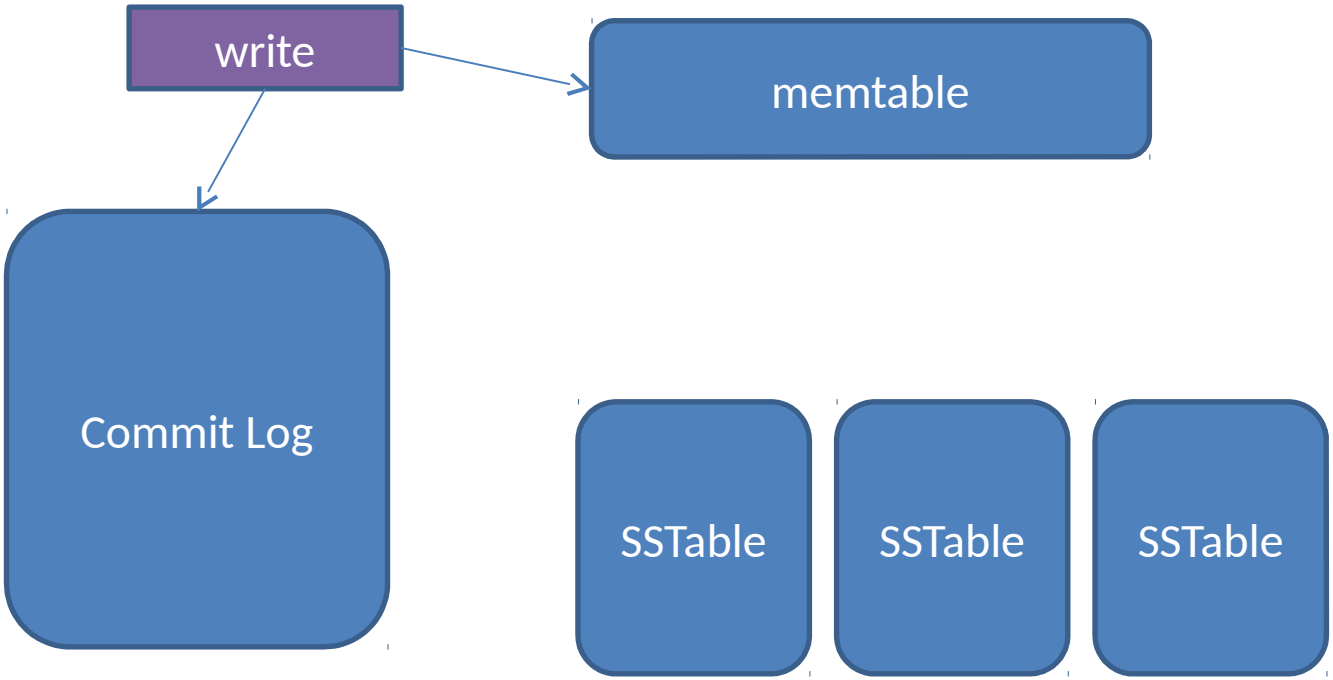
Εξυπηρέτηση αιτήσεων



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

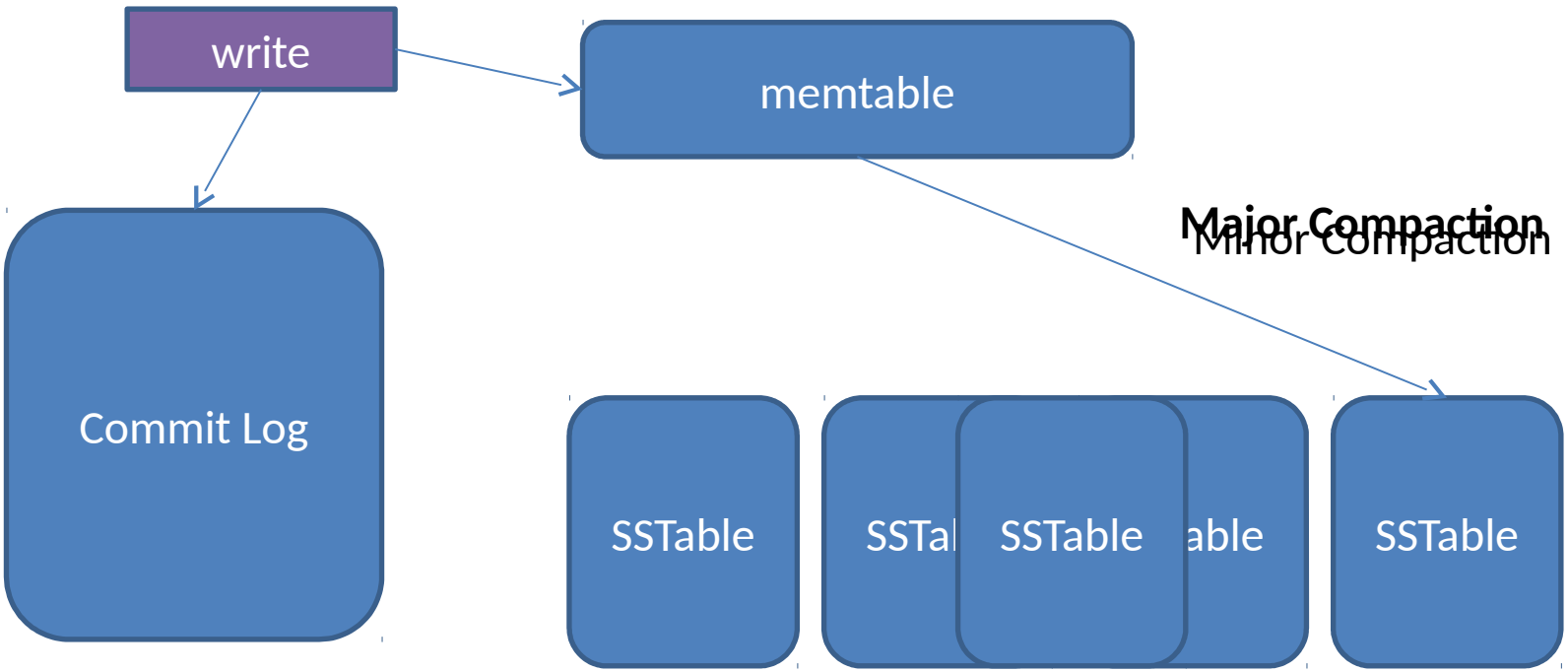
Εγγραφή δεδομένων



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 συγχωνευονται
  - Αποφεύγεται η δημιουργία πολλών αρχείων από πολλά minor compactions

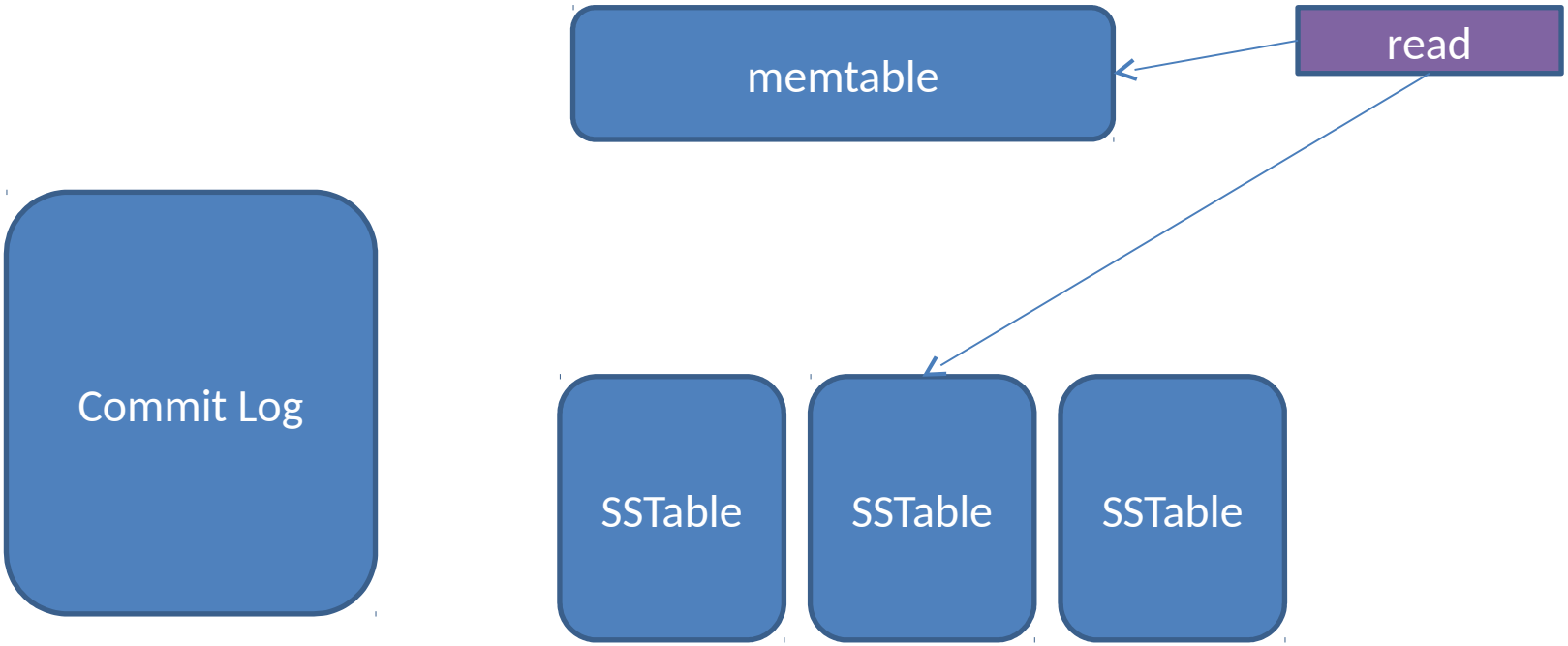
Συμπύκνωση


write
memtable
Commit Log
Major Compaction
Minor Compaction
SSTable
SSTable
SSTable

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 δημιουργειται για καθένα
  - Διευκολύνουν την διαχείριση των πινάκων

Βελτιώσεις

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

Βελτιώσεις

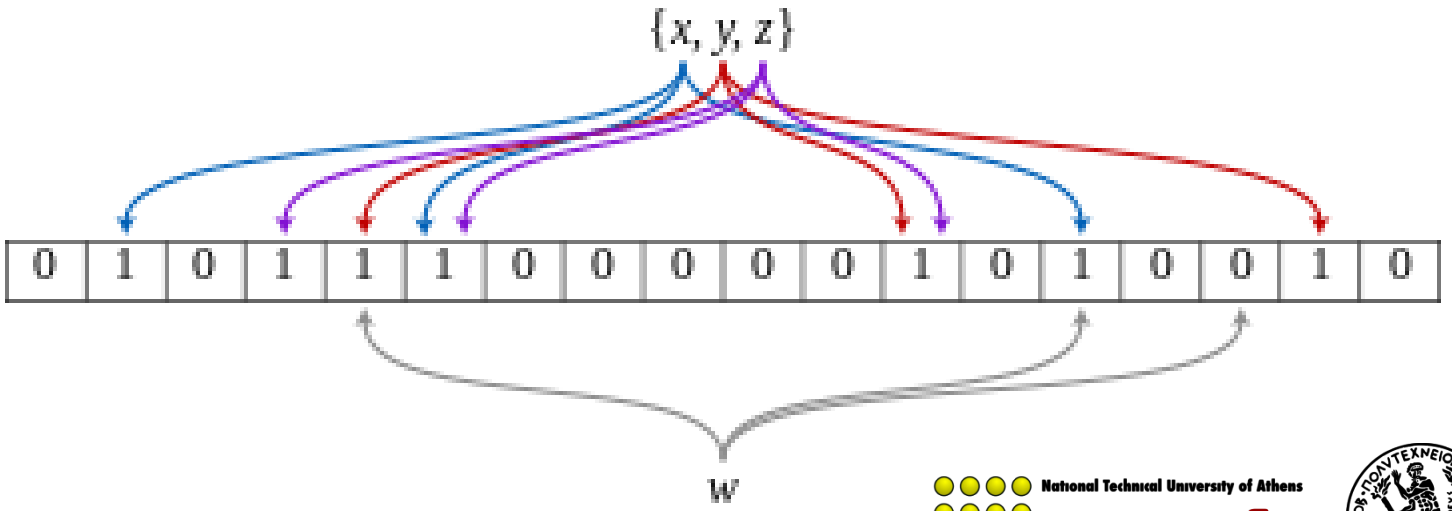
- Caching
  - Πραγματοποιείται σε δύο επίπεδα
  - Η Scan Cache διατηρεί τα ζευγη key-value(υψηλό επίπεδο)
  - Η Block Cache διατηρεί ολόκληρα blocks από το SSTable(χαμηλό επίπεδο)

Βελτιώσεις

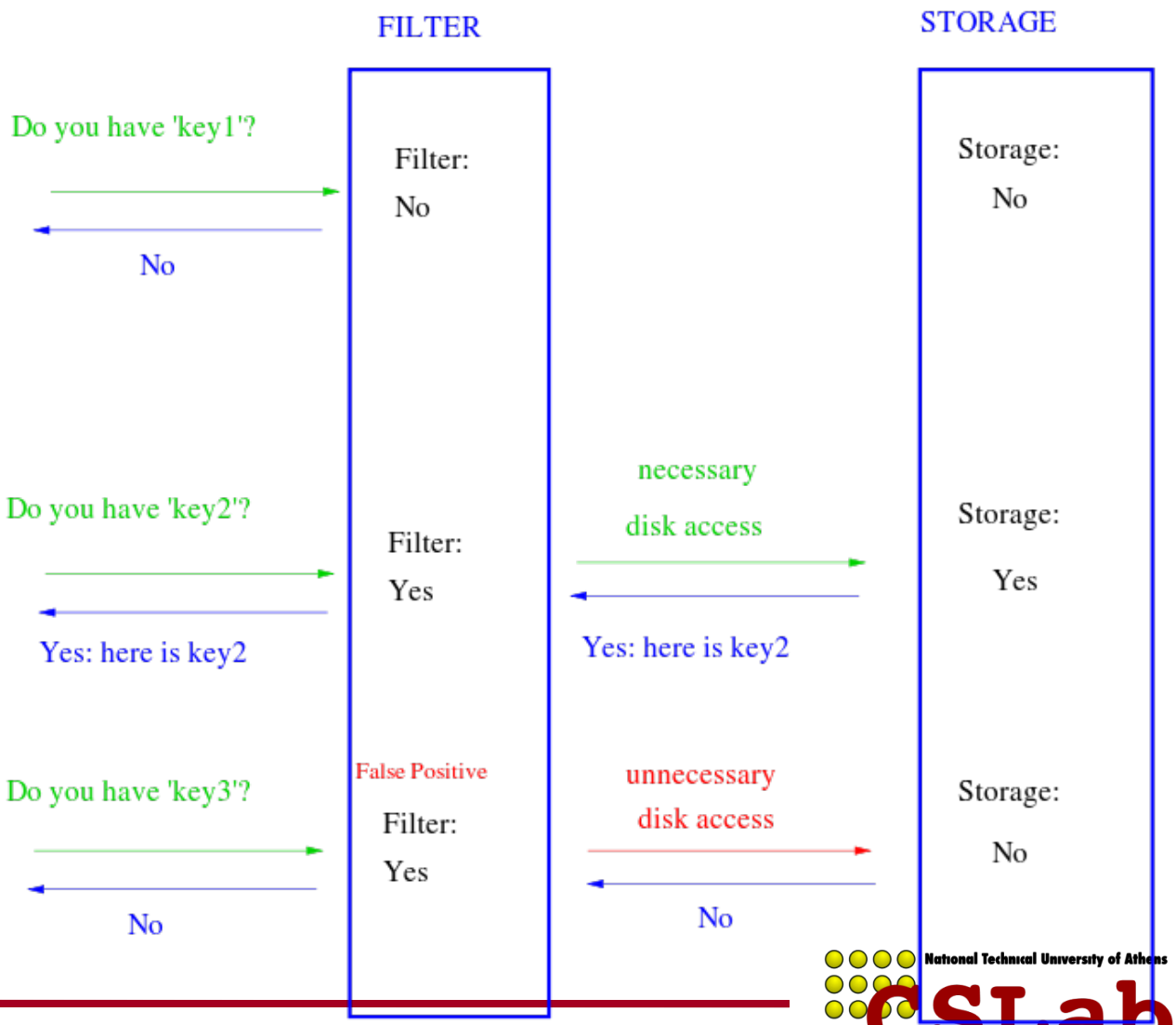
- Bloom filters
  - Ο χρήστης καθορίζει αν θα χρησιμοποιηθούν
  - Χρησιμοποιείται για να καθοριστεί αν κάποιο SSTable περιέχει δεδομένα από συγκεκριμένο row χωρίς να το ανακτήσουμε
  - Με λίγο αποθηκευτικό χώρο παραπάνω μπορούμε να αποκλείουμε αμέσως μεγάλο αριθμό sstables.
  - Πιθανά false positives: 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 -> all 0s
- Add element: k hash functions -> k positions set to 1



Παράδειγμα Bloom Filter



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?