ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

HY590.45
Modern Topics in Scalable Storage Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy590-45

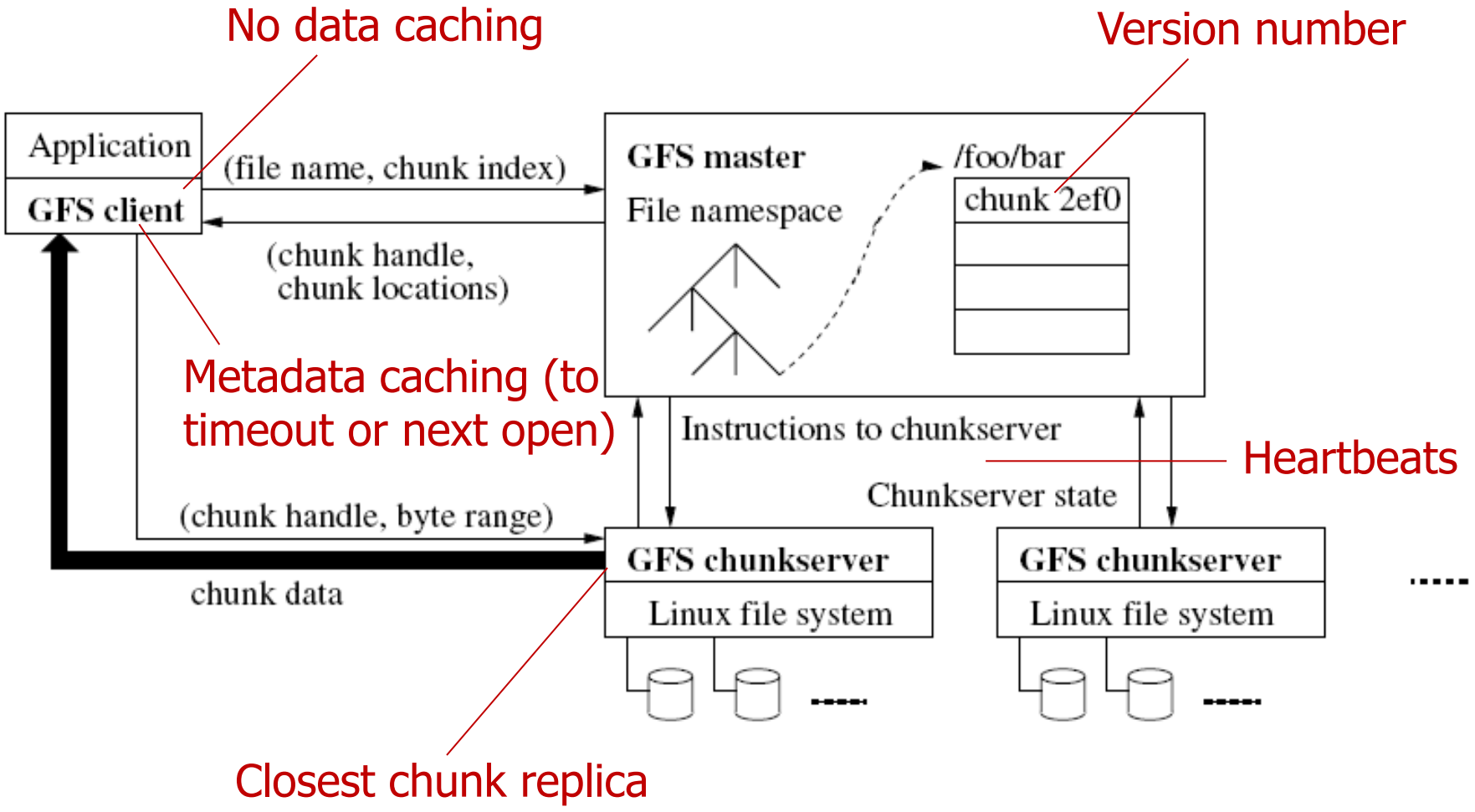
Assumptions

- Inexpensive commodity components
- Modest number of large files (multi-GB)
- Large streaming reads, small random reads
- Large sequential writes (appends)
- Need well-defined semantics for concurrent appends
- Bandwidth more important than latency

Interface

- Familiar but non-standard API
- Hierarchical, directory-based file namespace
- Create, delete, open, close, read, write
- Snapshot, record append

Google FS : Architecture



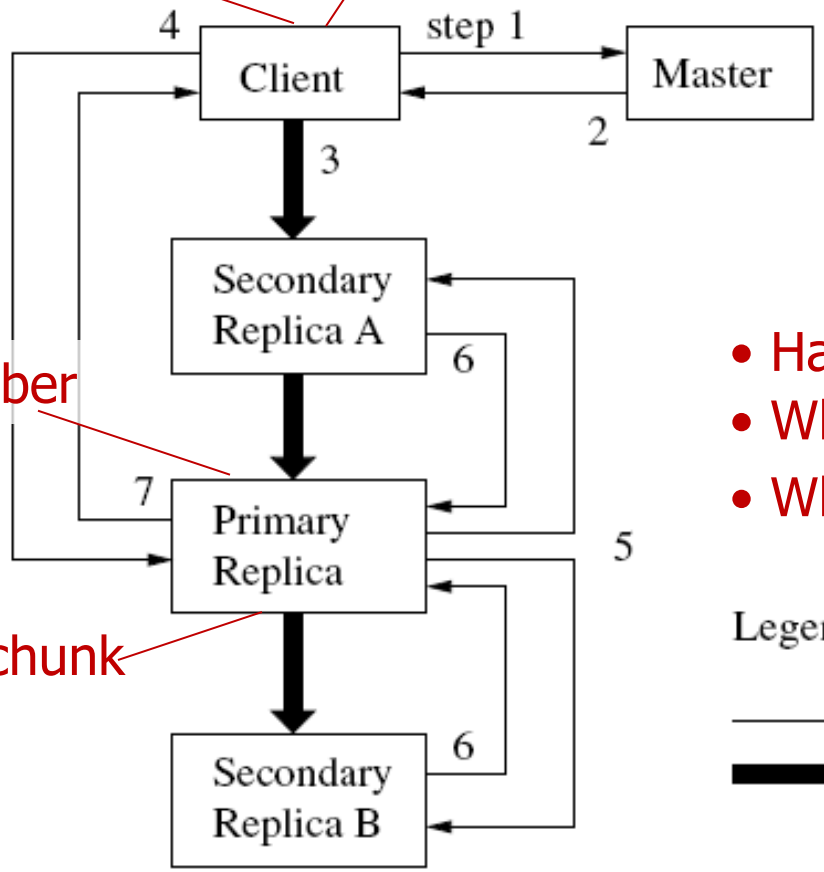
Metadata

- Types of metadata
  - File and chunk namespaces
  - Mapping from files to chunks
  - Location of chunk replicas
- Operation log
  - Persistent record of metadata; defines order
- Management activities
  - Chunk garbage collection
  - Chunk re-replication in case of chunkserver failures
  - Chunk migration to balance load

Write control and data flow

Caches metadata

Break write request into chunks



Assign serial number

Holds lease on chunk

- Handling of appends
- What happens on failure
- What if client fails

Legend:

Control

Data

Consistency model (1)

- File region is defined as
  - "Consistent": Same on all replicas
  - "Defined": Consistent and clients will see what the mutation wrote in its entirety (implies serializability)
- POSIX semantics are identical to "defined"
- Namespace mutations are atomic
  - Handled exclusively by the master

Consistency model (2)

- Data mutations depend on
  - Type of mutation: Write | Record append
  - Concurrency
  - Failures

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

Implication for applications

- Rely on appends rather than overwrites
  - More efficient and resilient to failures than regular writes
- Write self-validating, self-identifying records
- Checkpointing
  - Allows writers to restart incrementally
- Readers
  - Read only up to last checkpoint
  - Identify and discard padding using checksums
  - Tolerate duplicates through record version numbers

Snapshot

- On snapshot request, revoke leases on all chunks
- Log the operation to disk
- Duplicate metadata for source file or directory
- On write request to chunk C, clone to C’

Locking

- Acquire read locks on path names
- Acquire write lock on file or directory to operate on
- Example
  - /home/user is snapshotted to /save/user
  - Application tries to create /home/user/foo
- Allows concurrent mutations on same directory
- Avoids deadlock

Chunk creation

- Factors to consider in placing new replicas
  - Choose chunk servers with below-average disk utilization
  - Limit the number of new creations on each server
  - Spread replicas of a chunk across racks

- Re-replicate if # replicas < threshold
  - Prioritize chunks based on how many replicas lost