

Google File System, Replication

Amin Vahdat

CSE 123b

May 23, 2006

Annoucements

- Third assignment available today
 - Due date June 9, 5 pm
- Final exam, June 14, 11:30-2:30

Google File System

(thanks to Mahesh Balakrishnan)

The Google File System

- Specifically designed for Google's backend needs
- Web Spiders append to huge files
- Application data patterns:
 - Multiple producer – multiple consumer
 - Many-way merging
- GFS $\leftrightarrow$ Traditional File Systems

Design Space Coordinates

- Commodity Components
- Very large files – Multi GB
- Large sequential accesses
- Co-design of Applications and File System
- Supports small files, random access writes and reads, but not efficiently

GFS Architecture

- Interface:
 - Usual: create, delete, open, close, etc
 - Special: snapshot, record append
- Files divided into fixed size chunks
- Each chunk replicated at chunkservers
- Single master maintains metadata
- Master, Chunkservers, Clients: Linux workstations, user-level process

Client File Request

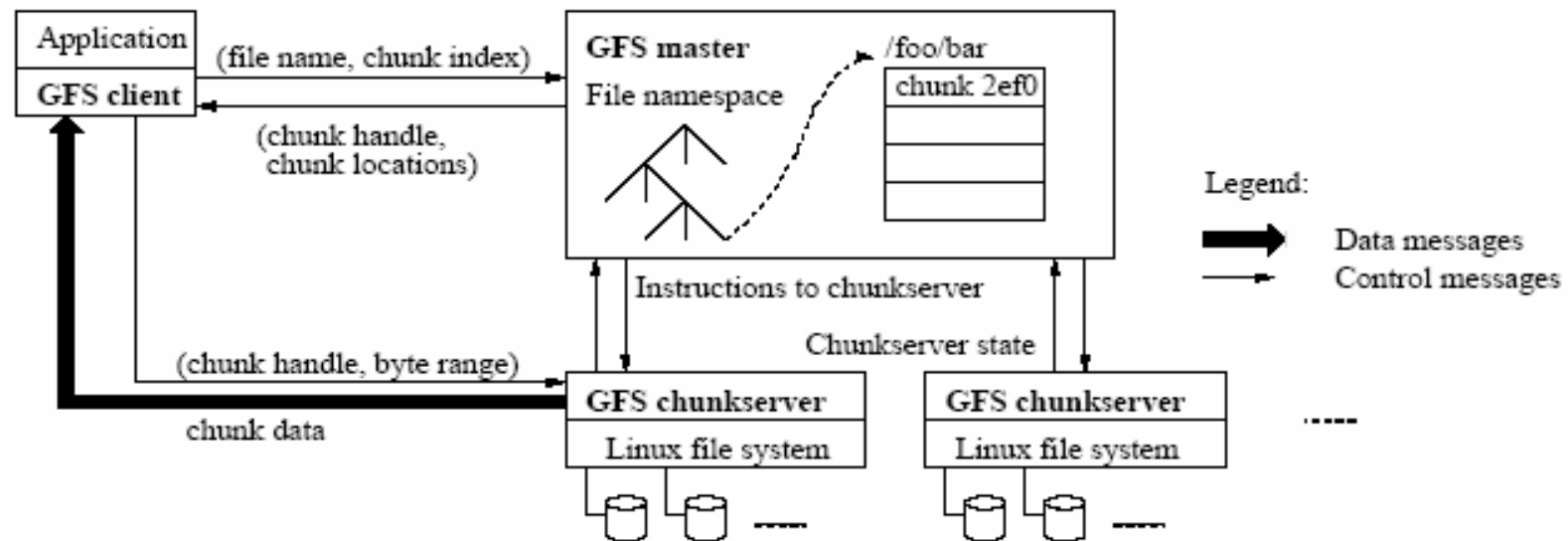


Figure 1: GFS Architecture

- Client finds chunkid for offset within file
- Client sends <filename, chunkid> to Master
- Master returns chunk handle and chunkserver locations

Design Choices: Master

- Single master maintains all metadata
 - Simple Design
 - Global decision making for chunk replication and placement
 - Bottleneck?
 - Single Point of Failure?

Design Choices: Master

- Single master maintains all metadata in memory
 - Fast master operations
 - Allows background scans of entire data
 - Memory Limit?
 - Fault Tolerance?

Relaxed Consistency Model

- File Regions are
 - Consistent: All clients see the same thing
 - Defined: After mutation, all clients see exactly what the mutation wrote
- Ordering of Concurrent Mutations –
 - For each chunk's replica set, Master gives one replica primary lease
 - Primary replica decides ordering of mutations and sends to other replicas

Anatomy of a Mutation

- 1 2 Client gets chunkserver locations from master
- 3 Client pushes data to replicas, in a chain
- 4 Client sends write request to primary; primary assigns sequence number to write and applies it
- 5 6 Primary tells other replicas to apply write
- 7 Primary replies to client

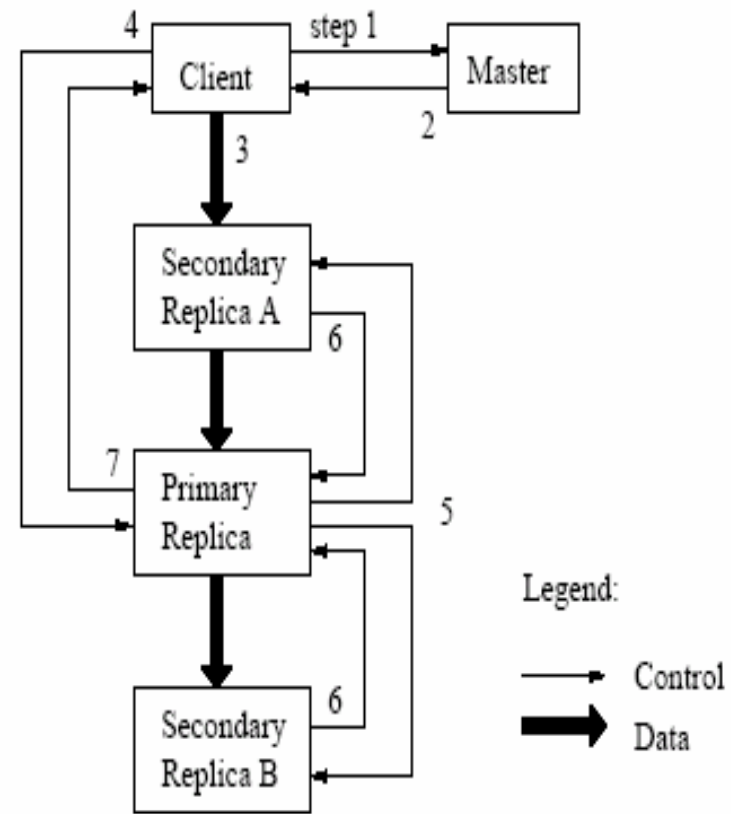


Figure 2: Write Control and Data Flow

Connection with Consistency Model

	Write	Record Append
Serial success	<i>defined</i>	<i>defined</i> interspersed with <i>inconsistent</i>
Concurrent successes	<i>consistent</i> but <i>undefined</i>	
Failure	<i>inconsistent</i>	

Table 1: File Region State After Mutation

- Secondary replica encounters error while applying write (step 5): region Inconsistent.
- Client code breaks up single large write into multiple small writes: region Consistent, but Undefined.

Special Functionality

- Atomic Record Append
 - Primary appends to itself, then tells other replicas to write at that offset
 - If secondary replica fails to write data (step 5),
duplicates in successful replicas, padding in failed ones
region defined where append successful, inconsistent where failed
- Snapshot
 - Copy-on-write: chunks copied lazily to same replica

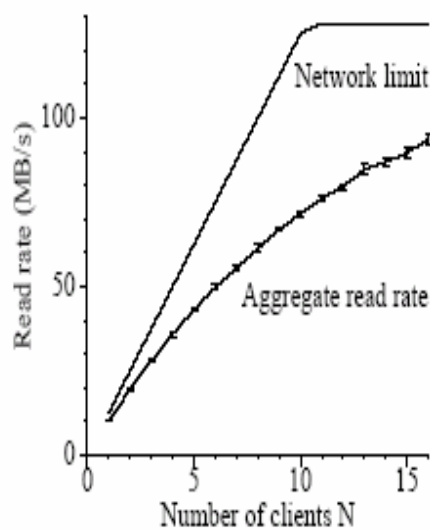
Master Internals

- Namespace management
- Replica Placement
- Chunk Creation, Re-replication, Rebalancing
- Garbage Collection
- Stale Replica Detection

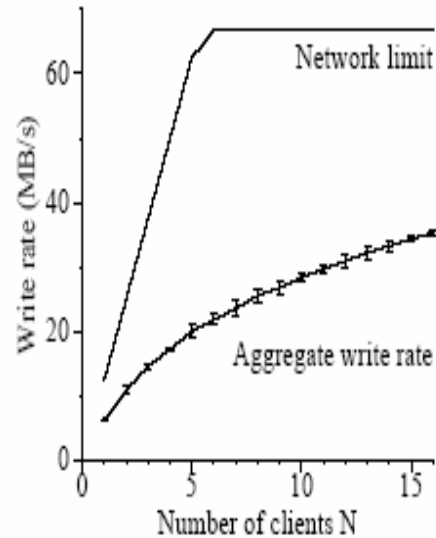
Dealing with Faults

- High availability
 - Fast master and chunkserver recovery
 - Chunk replication
 - Master state replication: read-only shadow replicas
- Data Integrity
 - Chunk broken into 64KB blocks, with 32 bit checksum
 - Checksums stored in memory, logged to disk
 - Optimized for appends, since no verifying required

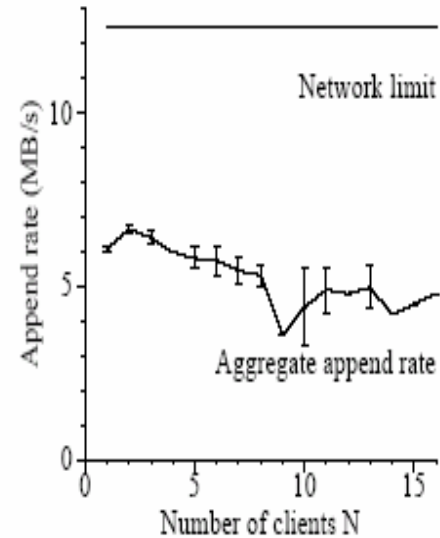
Micro-benchmarks



(a) Reads



(b) Writes



(c) Record appends

Storage Data for ‘real’ clusters

Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

Table 2: Characteristics of two GFS clusters

Performance

Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

Table 3: Performance Metrics for Two GFS Clusters

Workload Breakdown

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
0K	0.4	2.6	0	0	0	0
1B..1K	0.1	4.1	6.6	4.9	0.2	9.2
1K..8K	65.2	38.5	0.4	1.0	18.9	15.2
8K..64K	29.9	45.1	17.8	43.0	78.0	2.8
64K..128K	0.1	0.7	2.3	1.9	< .1	4.3
128K..256K	0.2	0.3	31.6	0.4	< .1	10.6
256K..512K	0.1	0.1	4.2	7.7	< .1	31.2
512K..1M	3.9	6.9	35.5	28.7	2.2	25.5
1M..inf	0.1	1.8	1.5	12.3	0.7	2.2

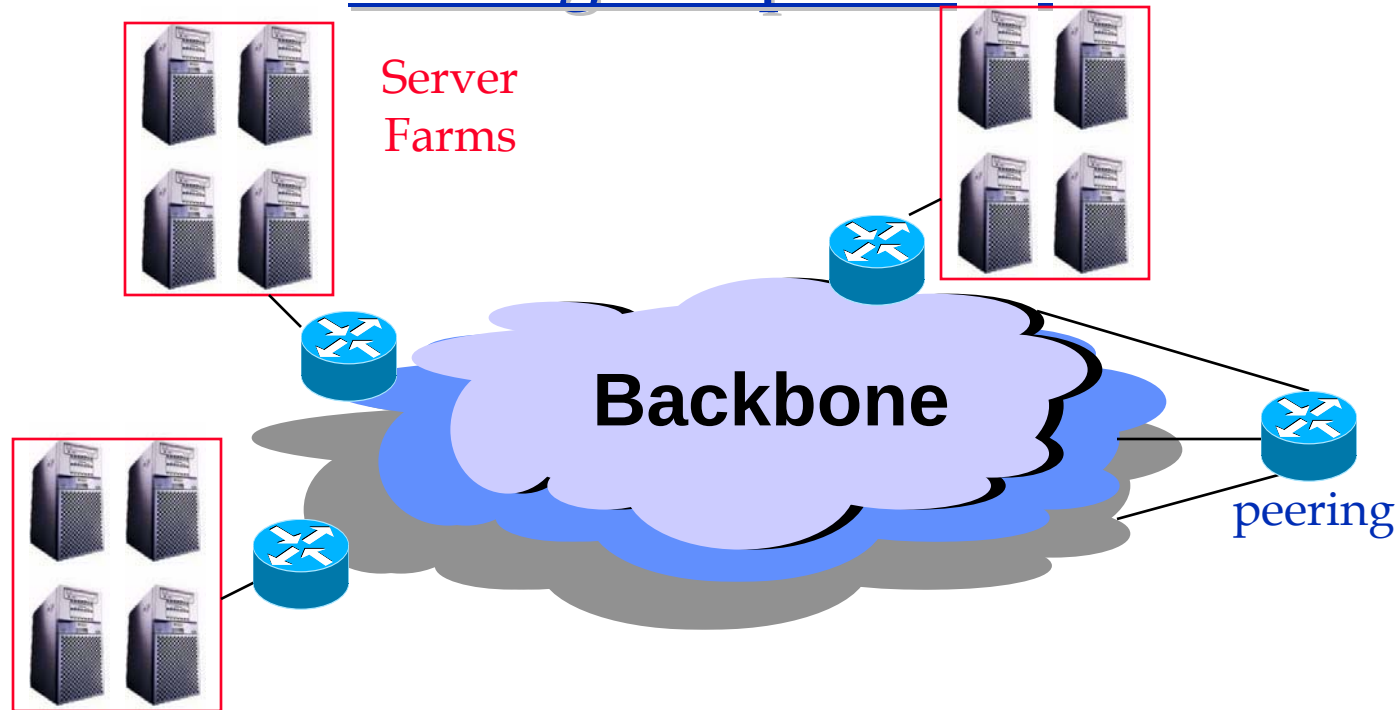
% of operations
for given size

% of bytes
transferred for
given operation
size

Operation	Read		Write		Record Append	
Cluster	X	Y	X	Y	X	Y
1B..1K	< .1	< .1	< .1	< .1	< .1	< .1
1K..8K	13.8	3.9	< .1	< .1	< .1	0.1
8K..64K	11.4	9.3	2.4	5.9	2.3	0.3
64K..128K	0.3	0.7	0.3	0.3	22.7	1.2
128K..256K	0.8	0.6	16.5	0.2	< .1	5.8
256K..512K	1.4	0.3	3.4	7.7	< .1	38.4
512K..1M	65.9	55.1	74.1	58.0	.1	46.8
1M..inf	6.4	30.1	3.3	28.0	53.9	7.4

Replication

High Performance and Availability Through Replication?



- Improve probability that nearby replica can handle request
- Increase system complexity

The Need for Replication

- Certain mission critical Internet services must provide 100% availability and predictable (high) performance to clients located all over the world
 - With scale of the Internet, high probability that some replica/some network link unavailable at *all* times
- Replication is the *only* way to provide such guarantees
 - Despite any increased complexities, must investigate techniques for addressing replication challenges

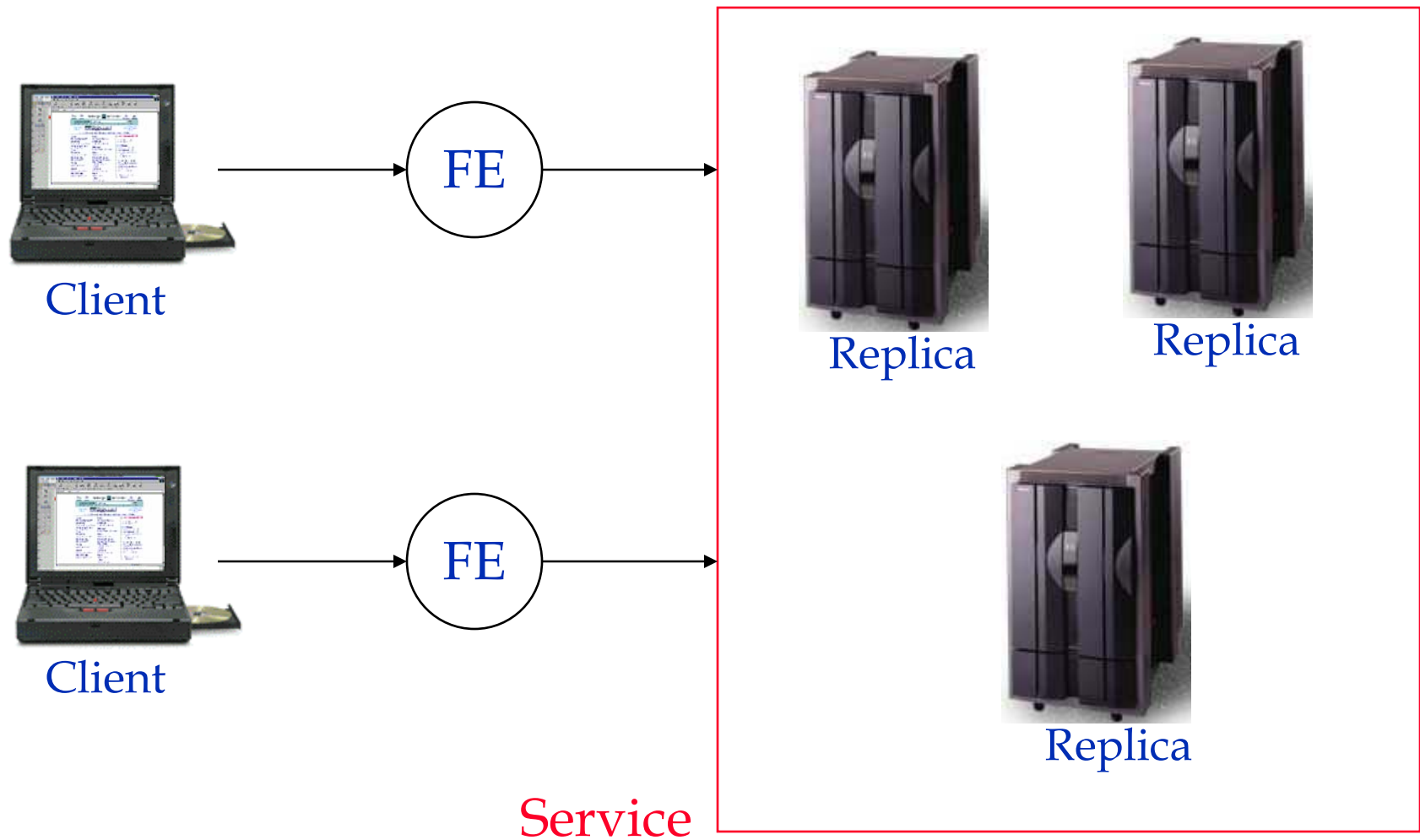
Replication Goals

- Replicate network service for:
 - Better performance
 - Enhanced availability
 - Fault tolerance
- How could replication lower performance, availability, and fault tolerance?

Replication Challenges

- Transparency
 - Mask from client the fact that there are multiple *physical* copies of a *logical* service or object
 - Expanded role of naming in networks/dist systems
- Consistency
 - Data updates must eventually be propagated to multiple replicas
 - Guarantees about latest version of data?
 - Guarantees about ordering of updates among replicas?
- Increased complexity...

Replication Model



How to Handle Updates?

- Problem: all updates must be distributed to all replicas
 - Different consistency guarantees for different services
 - Synchronous vs. asynchronous update distribution
 - Read/write ratio of workload
- Primary copy
 - All updates go to a single server (master)
 - Master distributes updates to all other replicas (slaves)
- Gossip architecture
 - Updates can go to any replica
 - Each replica responsible for eventually delivering local updates to all other replicas