

CS x55: DISTRIBUTED SYSTEMS [THE GOOGLE FILE SYSTEM]

So Long, and Thanks for All the Fish

Outside of CS, there are few who care
For how we conjure things in software
Of stripe sets and hash buckets
And how networks route packets
Or wonder why accesses to memory are blazingly quick
But those to disks, so eternally slow
Or ponder what makes MapReduce tick
Here's to the pursuit of being in the know!

Shrideep Pallickara
Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT



1

Frequently asked questions from the previous class survey

- What's the key advantage of virtual nodes over single identifiers?
- When using virtual nodes, the failure of a physical node may impact several key spaces simultaneously?
- Does the $O(N)$ routing table pose issues for Dynamo's scalability?
- What's the preferred file system for systems such as Dynamo or GFS?



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.2

2

Topics covered in this lecture

- The Google File System



COLORADO STATE UNIVERSITY

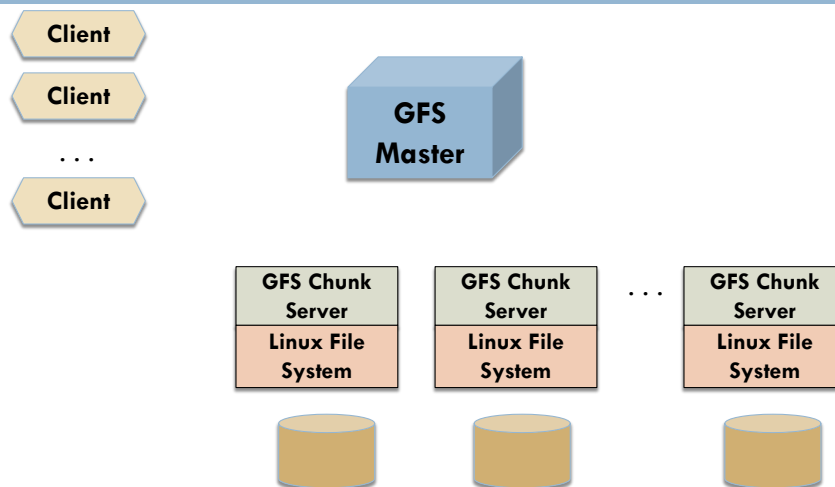
Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.3

3

Architecture of GFS



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.4

4

In GFS a file is broken up into fixed-size chunks

- Obvious reason
 - The file is too big
- **Set the stage** for computations that operate on this data
 - Parallel I/O
 - I/O seek times are 14×10^6 slower than CPU clock cycles

Map-Reduce



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.5

5

In GFS a file is broken up into fixed-size chunks

- Each chunk has a 64-bit globally unique ID
 - Assigned by the Master
- Chunks are stored by chunk servers
 - On local disks as LINUX files
- Each chunk is **replicated**
 - Default is 3



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.6

6

Master operations

- Manage system **metadata**
- **Leasing** of chunks
- **Garbage collection** of orphaned chunks
- Chunk **migrations**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.7

7

ALL system metadata is managed by the Master and stored in **main memory**

- ① File and chunk namespaces
- ② Mapping from files to chunks
- ③ Location of chunks

} *Logs mutations
into a permanent log*



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.8

8

Why have a single Master?

- Vastly **simplifies** design
- Easy to use global knowledge to **reason about**
 - Chunk placements
 - Replication decisions



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.9

9

Communications with the chunk servers

- Periodic communications using **heartbeats**
 - Instructions to the chunk server
 - Collect/retrieve state from the chunk server



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.10

10

Chunk size

- This is fixed at **64 MB**
 - ▣ Much larger than typical filesystem block sizes (512B up to 4KB)
- **Lazy space allocation**
 - ▣ Stored as plain Linux file
 - ▣ Extended only as needed



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.11

11

But why this big?

- **Reduces client interaction** with the master
 - ▣ Can cache info for a multi-TB working set
- Reduce network **overhead**
 - ▣ With a large chunk, client performs more operations
 - ▣ Persistent connections
- Reduce **size of metadata** stored in the master
 - ▣ 64 bytes of metadata per 64 MB chunk



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.12

12

Why keep the entire metadata in memory?

- **Speed**
- Master can **scan** its **state** in the background
 - Implement chunk garbage collection
 - Re-replicate if there are failures
 - Chunk migration to balance load and space
- **Add** extra memory to increase file system size



13

Size of the file system with 1 TB of RAM: Assume file sizes are exact multiples of chunk sizes

- Number of entries = $2^{40}/2^6$
- MAXIMUM SIZE of the file system
 - = Number of entries x Chunk size
 - = $\frac{2^{40}}{2^6} \times 2^6 \times 2^{20}$
 - = $2^{60} = 1 \text{ EB}$



14

Tracking the chunk servers

- Master **does not** keep a persistent copy of the location of chunk servers
- List maintained via **heart-beats**
 - Allows list to be in *sync* with reality despite failures
 - Chunk server has final word on chunks it holds



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.15

15

Caching at the client/chunk servers

- Clients **do not cache** file data
 - At client, the working set may be *too large*
 - Simplify client; eliminate *cache-coherence* problems
- Chunk servers **do not cache** file data either
 - Chunks are stored as local files
 - Linux's buffer cache *already keeps* frequently accessed data in memory




COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS


L38.16

16



Mutation: It Is The Key To Our Evolution.
Charles Xavier, X-Men.


MANAGING MUTATIONS

COMPUTER SCIENCE DEPARTMENT  COLORADO STATE UNIVERSITY

17

Mutations

- **Mutation** changes the content or metadata of a chunk
 - Write
 - Append
- Each mutation is performed at **all** chunk replicas

 COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT GFS L38.18

18

GFS uses leases to maintain consistent mutation order across replicas

- Master grants **lease** to one of the replicas
 - **PRIMARY**
- Primary picks **serial-order**
 - For all mutations to the chunk
 - Other replicas *follow* this order
 - When applying mutations



Lease mechanism designed to minimize communications with the master

- Lease has initial *timeout* of 60 seconds
- As long as chunk is being mutated
 - Primary can request and receive **extensions**
- Extension requests/grants piggybacked over heart-beat messages



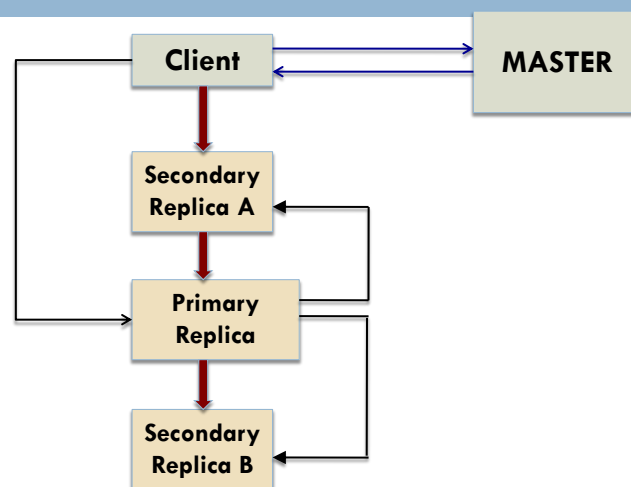
Revocation and transfer of leases

- Master may **revoke** a lease before it expires
- If communications lost with primary
 - Master can safely give lease to another replica
 - **ONLY AFTER** the lease period for old primary *elapses*



21

How a write is actually performed



22

Client orchestrates writing data to the replicas [1/2]

- Each chunk server stores data in an **LRU buffer** until
 - ▣ Data is used
 - ▣ Aged out



Client orchestrates writing data to the replicas [2/2]

- When chunk servers acknowledge receipt of data
 - ▣ Client sends a *write request to primary*
- Primary assigns *consecutive serial numbers* to mutations
 - ▣ Forwards to replicas



Data flow is **decoupled** from the control flow to utilize network efficiently

- Utilize each machine's network bandwidth
- Avoid network bottlenecks
- Avoid high-latency links
- **Leverage** network topology
 - Estimate distances from IP addresses



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.25

25

What if the secondary replicas could not finish the write operation?

- Client request is considered **failed**
- Modified region is **inconsistent**
 - *No attempt* to delete this from the chunk
 - Client must handle this inconsistency
- Client **retries** the failed mutation



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.26

26

GFS client code implements the file system API

- Communications with master and chunk servers done *transparently*
 - On behalf of apps that read or write data
- Interact with master for **metadata**
- **Data-bearing** communications directly to chunk servers



Traditional writes

- Client specifies **offset** at which data needs to be written
- Concurrent writes to same region
 - Not serializable
 - Region ends up containing **data fragments from multiple clients**



Atomic record appends

- Client specifies **only** the data **not** the offset
- GFS **appends** it to the file
 - **At least once atomically**
 - At an offset of GFS' choosing
- No need for a distributed lock manger



The control flow for record appends is similar to that of writes

- Client pushes data to replicas of the **last** chunk of file
- Primary replica checks if the record **fits** in this chunk



Primary replica checks if the record append will breach the size (64MB) threshold

- If chunk size would be breached
 - ▣ **Pad** the chunk to maximum size
 - ▣ Tell client, that operation should be retried on *next chunk*
- If the record fits, the primary
 - ▣ Appends data to its replica
 - ▣ Notifies secondaries to **write at the exact offset**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.31

31

Record sizes and fragmentation

- Size is restricted to $\frac{1}{4}$ the chunk size
- Minimizes worst-case fragmentation
 - ▣ Internal fragmentation in each chunk ...



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.32

32

What if record append fails at one of the replicas

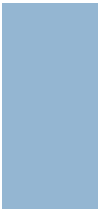
- Client **must retry** the operation
- Replicas of same chunk may contain
 - Different data
 - **Duplicates** of the same record
 - In whole or in part
- Replicas of chunks are **not bit-wise identical!**
 - In most systems, replicas are identical



GFS only guarantees that the data will be written at least once as an atomic unit


- For an operation to return *success*
 - Data must be **written at the same offset** on **all** the replicas
- After the write, all replicas are *as long as* the end of the record
 - Any future record will be assigned a higher offset or a different chunk





REPLICATION

COMPUTER SCIENCE DEPARTMENT




COLORADO STATE UNIVERSITY

35

Reasons why chunk replicas are created

- Chunk creation
- Re-replication
- Rebalancing



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.36

36

Chunk replica creation

- Place replicas on chunk servers with below average **disk space** utilization
- **Limit** the number of **recent creations** on a chunk server
 - Predictor of imminent heavy traffic
- Spread replicas **across racks**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.37

37

Re-replicate chunks when replication level drops

- How **far** is it from replication goal?
- Preference for chunks of **live** files
- Boost priority of chunks **blocking client progress**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.38

38

Rebalancing replicas

- **Examine** current replica distribution and **move** replicas
 - Better disk space
 - Load balancing
- **Removal** of existing replicas
 - Chunk servers with below-average disk space



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.39

39

Incorporating a new chunk server

- **Do not swamp** new server with lots of chunks
 - Concomitant traffic will bog down the machine
- **Gradually** fill up new server with chunks



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.40

40

So, so you think you can tell
Heaven from hell?
Blue skies from pain?
Can you tell a green field
From a cold steel rail?
A smile from a veil?
Do you think you can tell?
Wish You Were Here; Gilmour/Waters; Pink Floyd

CREATING SNAPSHOTS

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

41

Snapshots allow you to make a copy of a file very fast

- Master **revokes** outstanding leases for any chunks of the file (source) to be snapshot
- **Log** the operation to disk
- Update in-memory state
 - Duplicate metadata of the source file
- Newly created file points to the same chunks as the source



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.42

42

When a client wants to write to a chunk C after the snapshot operation

- Master sees the *reference count* to $C > 1$
- Pick **new chunk-handle C'**
- Ask chunk-server with current replica of C
 - ▣ Create new chunk C'
 - ▣ Data is *copied locally, not over the network*
- From this point chunk handling of C' is no different



GFS does not have a per-directory structure that lists files in the directory

- Name spaces represented as a **lookup** table
 - ▣ Maps **full pathnames** to metadata
- File creation does not require a lock on the directory structure
 - ▣ No inode needs to be protected from modification



Each master operation acquires a set of locks before it runs

- If operation involves `/d1/d2/.../dn/leaf`
 - Acquire read locks on directory names
 - `/d1`, `/d1/d2`, ..., `/d1/d2/.../dn`
 - Read or write lock on full pathname
 - `/d1/d2/.../dn/leaf`
- Used to **prevent** operations during snapshots
 - For e.g. cannot create `/home/user/foo`
 - While `/home/user` is being snapshotted to `/save/user`



Locks are used to prevent operations during snapshots

- For e.g. cannot create `/home/user/foo`
 - While `/home/user` is being snapshotted to `/save/user`
- Read locks on `/home` and `/save`
 - **Read lock prevents a directory from being deleted**
- Write lock on **`/home/user`** and `/save/user`
- File creation does not require write lock on parent directory ... there is no “directory”
 - Read locks on `/home` and **`/home/user`**
 - Write lock on `/home/user/foo`



CONSISTENCY IN GFS

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

47

In GFS the state of file region after mutation depends on ...

- **TYPE** of the mutation
- **SUCCESS/FAILURE** of the mutation
- Whether there were **CONCURRENT** mutations



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.48

48

GFS has a relaxed consistency model

- **Consistent:** See the same data
 - On all replicas

- **Defined**
 - Clients see mutation writes in its **entirety**



File state region after a mutation

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



Implications for applications

- Rely on **appends** instead of overwrites
- Checkpoint
- Write **records** that are
 - ▣ Self-validating
 - ▣ Self-identifying



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.51

51

DELETION OF FILES & GARBAGE COLLECTION

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

52

Garbage collection in GFS

- After a file is deleted, GFS does not reclaim space immediately
- Done **lazily** during garbage collection at
 - File and chunk levels



Master logs a file's deletion immediately

- File is **renamed to a hidden name**
 - Includes deletion timestamp
- Master scans the file system namespace
 - Delete if hidden file existed for more than 3 days
- When file removed from namespace
 - *In memory metadata* is also removed
 - **Severs links** to all its chunks!



Garbage collection: When Master scans its chunk namespace

- Identifies **orphaned chunks**
 - ▣ Not reachable from any file
- Erase metadata for these chunks



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.55

55

The role of heart-beats in garbage collection

- Chunk server reports subset of chunks it currently has
- Master replies with identity of chunks no longer present
 - ▣ Chunk server free to delete its replica of such chunks



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.56

56

Stale chunks and issues

- If a chunk server fails
 - ▣ AND misses mutations to the chunk
 - ▣ The chunk replica becomes **stale**
- Working with a **stale replica** causes problems with:
 - ▣ Correctness
 - ▣ Consistency



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.57

57

Aiding the detection of stale chunks

- Master maintains a **chunk version number** for each chunk
 - ▣ Distinguish between stale and up-to-date chunks
 - When master grants a new lease on chunk
 - ▣ Increase version number
 - ▣ Inform replicas
 - ▣ Record new version
- } Occurs BEFORE any client can write to chunk



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.58

58

If a replica is unavailable its **version number** will not be advanced

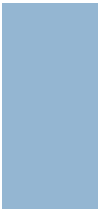
- When a chunk server restarts, it reports to the Master with the following:
 - ▣ Set of Chunks
 - ▣ Corresponding version numbers
- Used to **detect** stale replicas
- **Remove stale replicas** in regular garbage collection



Additional safeguards against stale replicas


- Include chunk version number
 - ▣ When **client requests** chunk information
 - Client/Chunk server verify version to make sure things are up-to-date
 - ▣ During cloning operations
 - Clone the most up-to-date chunk
- Clients and chunk servers expected to **verify** versioning information





DATA INTEGRITY

COMPUTER SCIENCE DEPARTMENT




COLORADO STATE UNIVERSITY

61

Data Integrity

- *Impractical* to detect chunk corruptions *across replicas*
 - ▣ Not bitwise identical in any case!
- Detection of corruption should be **self-contained**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.62

62

Data Integrity

- Break chunks into **64 KB** data blocks
- Compute 32-bit checksum for block
 - Keep in chunk server memory
 - Store persistently, **separate** from the data
- Verify checksums of data blocks that **overlap** read range



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.63

63

INEFFICIENCIES

COMPUTER SCIENCE DEPARTMENT



COLORADO STATE UNIVERSITY

64

The master server is a single point of failure

- Master server **restart** takes several seconds
- **Shadow servers** exist
 - Can handle reads of files
 - In place of the master
 - But not writes
- Requires a **massive main memory**



The system is optimized for large files

- But **not for** a very large number of very **small files**
- Primary operation on files
 - Long, sequential reads/writes
 - Large number of **random overwrites** will **clog** things up quite a bit



Consistency Issues: GFS expects clients to resolve inconsistencies

- File chunks may have **gaps or duplicates** of some records
 - ▣ The client has to be able to deal with this
- Imagine doing this for a scientific application
 - ▣ Portions of a massive array are corrupted
 - Clients would have to detect this
 - Detection is possible of course, but **onerous!**



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.67

67

Security model

- **None**
 - ▣ Operation is expected to be in a trusted environment



COLORADO STATE UNIVERSITY

Professor: SHRIDEEP PALICKARA
COMPUTER SCIENCE DEPARTMENT

GFS

L38.68

68

The contents of this slide-set are based on the following references

- Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: The Google file system. Proceedings of SOSP 2003: 29-43.

