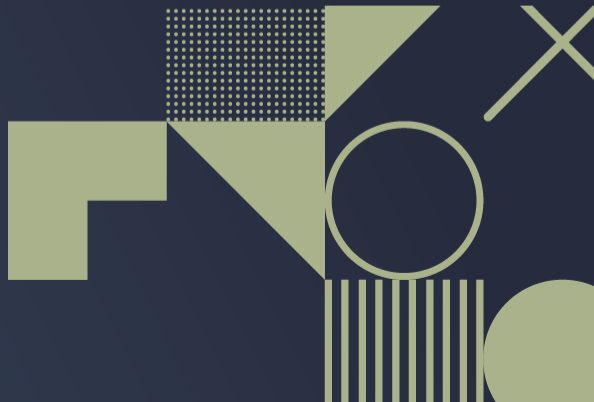
DeltaFS: A Scalable No-Ground-Truth Filesystem for Massively-Parallel Computing

Qing Zheng, Chuck Cranor, Greg Ganger, Garth Gibson, George Amvrosiadis
Brad Settlemyer, Gary Grider

Carnegie Mellon University, Los Alamos National Lab

LA-UR-21-30729



DeltaFS: A Scalable No-Ground-Truth Filesystem For Massively-Parallel Computing

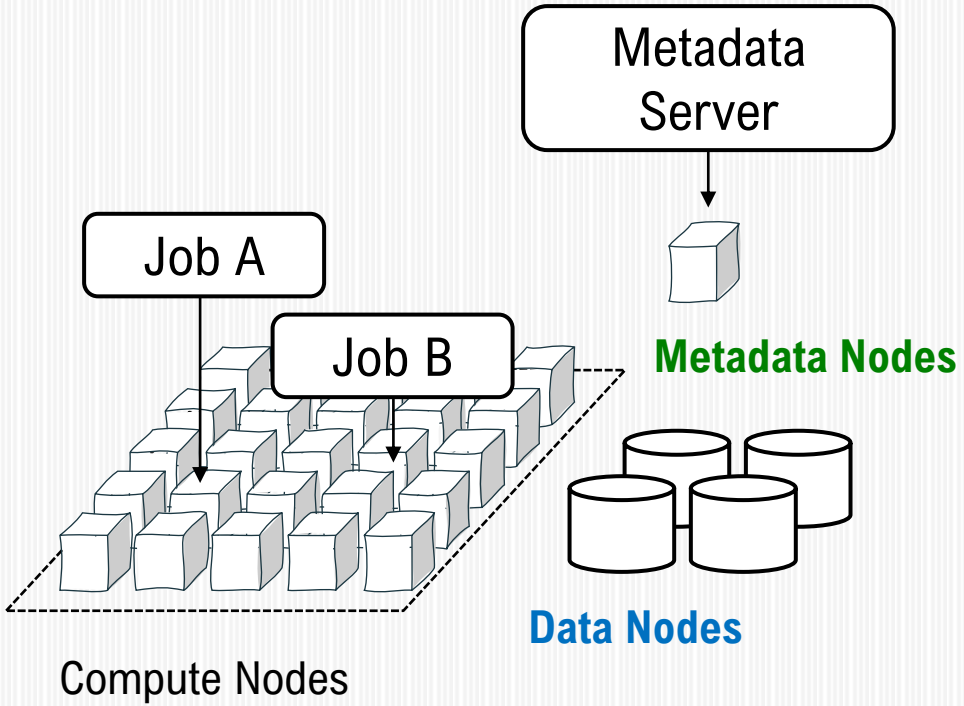
Qing Zheng

Chuck Cranor, Greg Ganger, Garth Gibson, George Amvrosiadis
Brad Settlemyer, Gary Grider

Carnegie Mellon University, Los Alamos National Lab

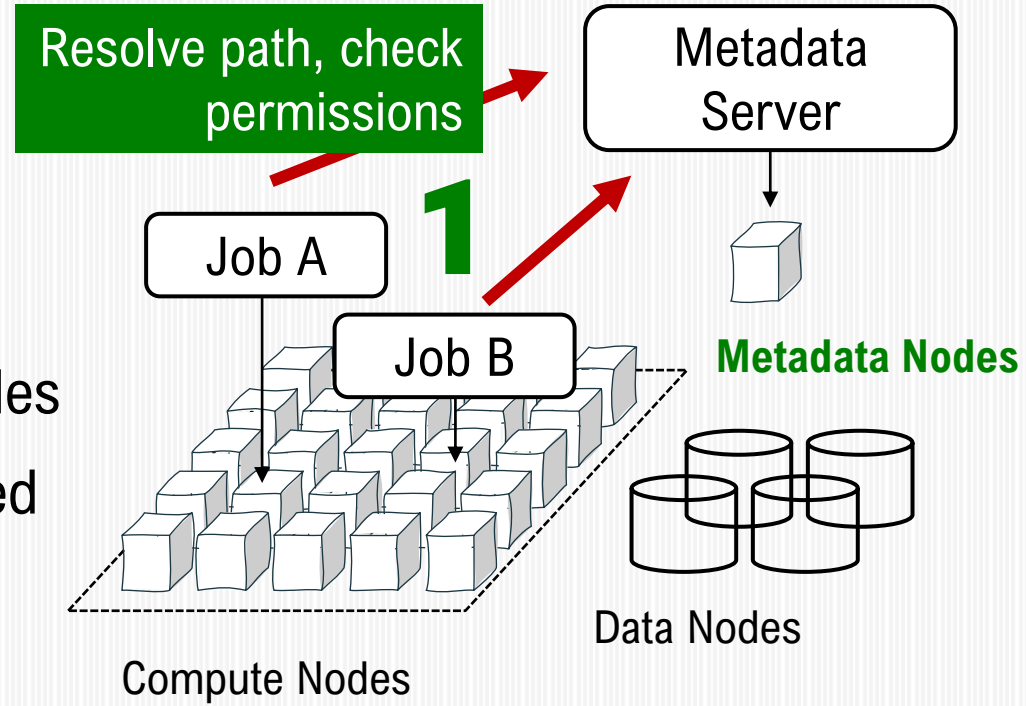
Background

- HPC stores data as **files**
- Files are managed by a **filesystem**
- Clients run on compute nodes
- Filesystems run on dedicated **data** and **metadata** server nodes



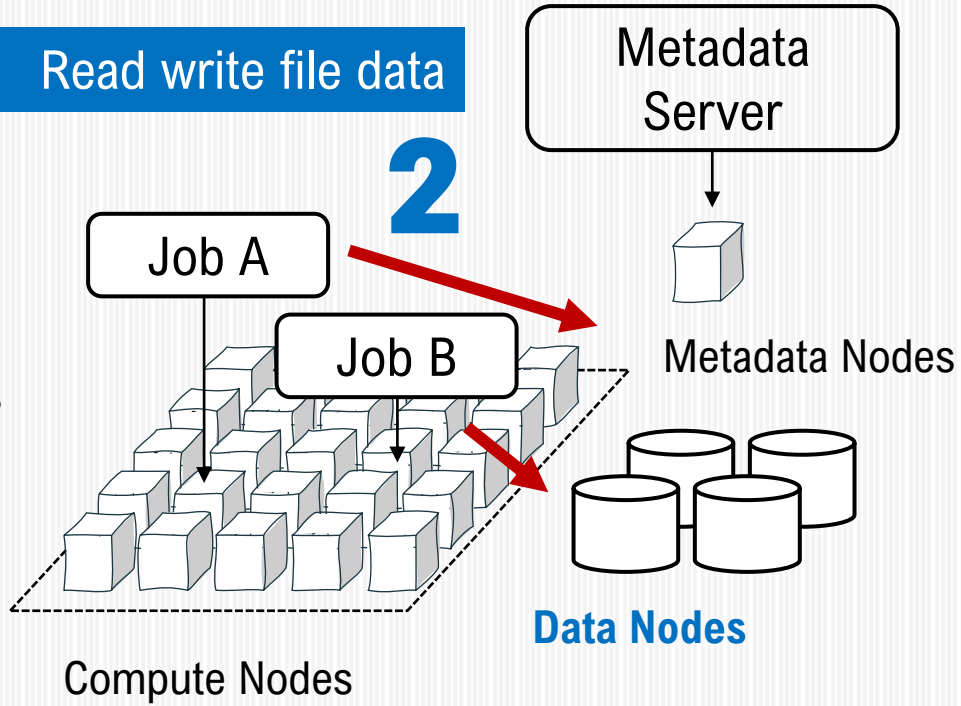
Background

- HPC stores data as **files**
- Files are managed by a **filesystem**
- Clients run on compute nodes
- Filesystems run on dedicated **data** and **metadata** server nodes



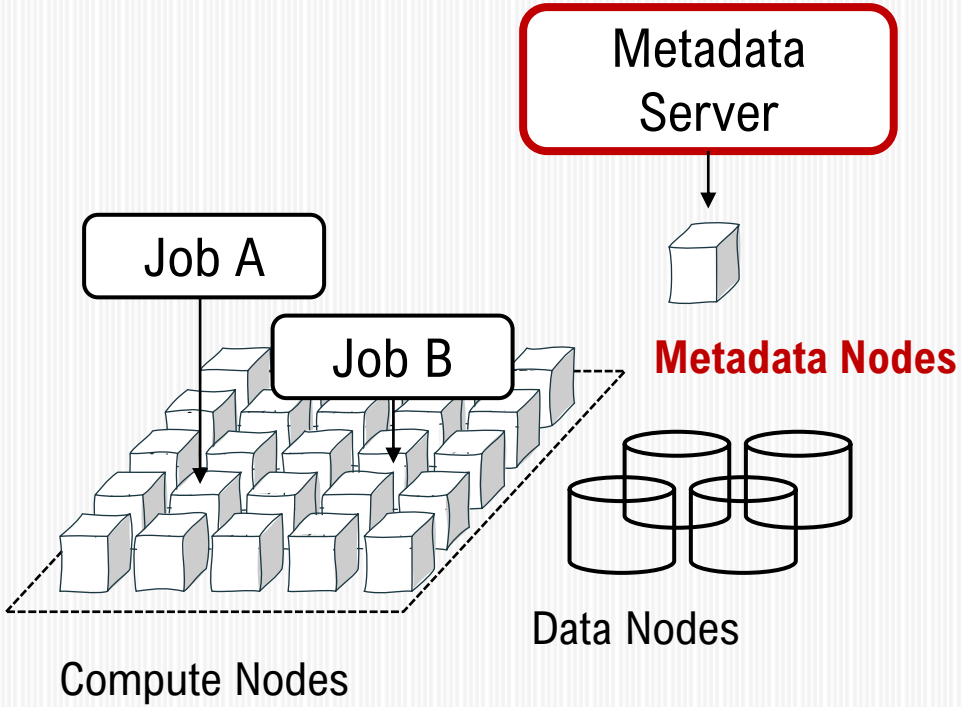
Background

- HPC stores data as **files**
- Files are managed by a **filesystem**
- Clients run on compute nodes
- Filesystems run on dedicated **data** and **metadata** server nodes



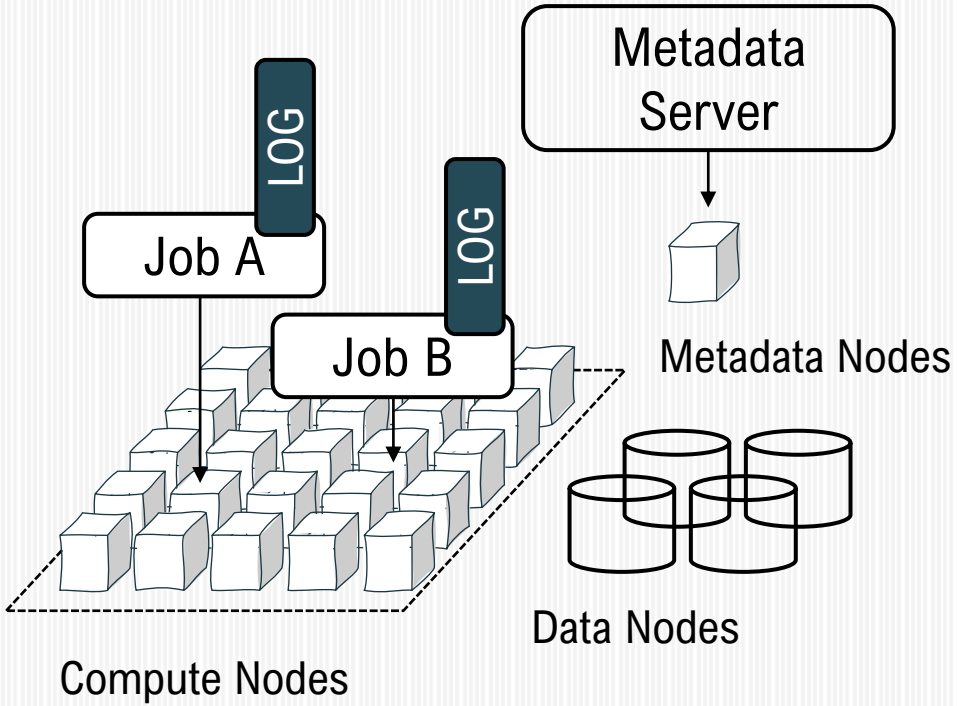
Metadata Bottlenecks

- HPC cluster increasingly parallel (millions of client CPU cores)
- Metadata server becomes a **bottleneck**



Existing Technique: Write Buffering

- Clients commit metadata writes in **local logs**
- Defer costly server communication until reads
- Reads still slow

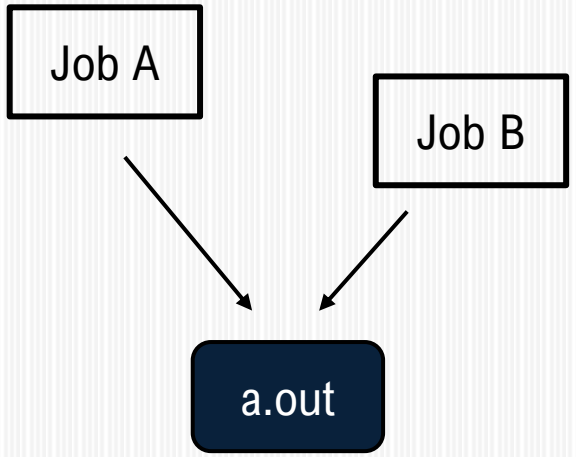


Can We Speed up Metadata In All Cases?

- Impossible without challenging **fundamental FS design principles**
- DeltaFS allows for fast metadata **reads and writes**

Today's FS: One View for All Clients

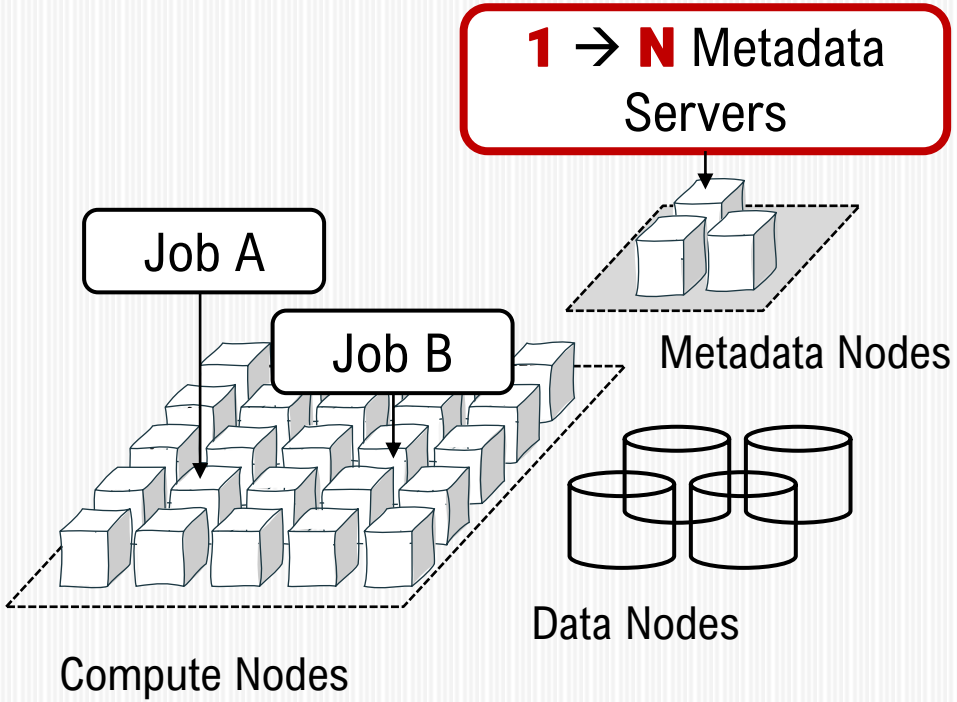
- FS maintains a single FS view (**state of all files**) for all clients
- We call this view the **Ground Truth** of a FS
- **Problem**: Limits processing to servers



Example: Job A creates a file; Job B will see the file

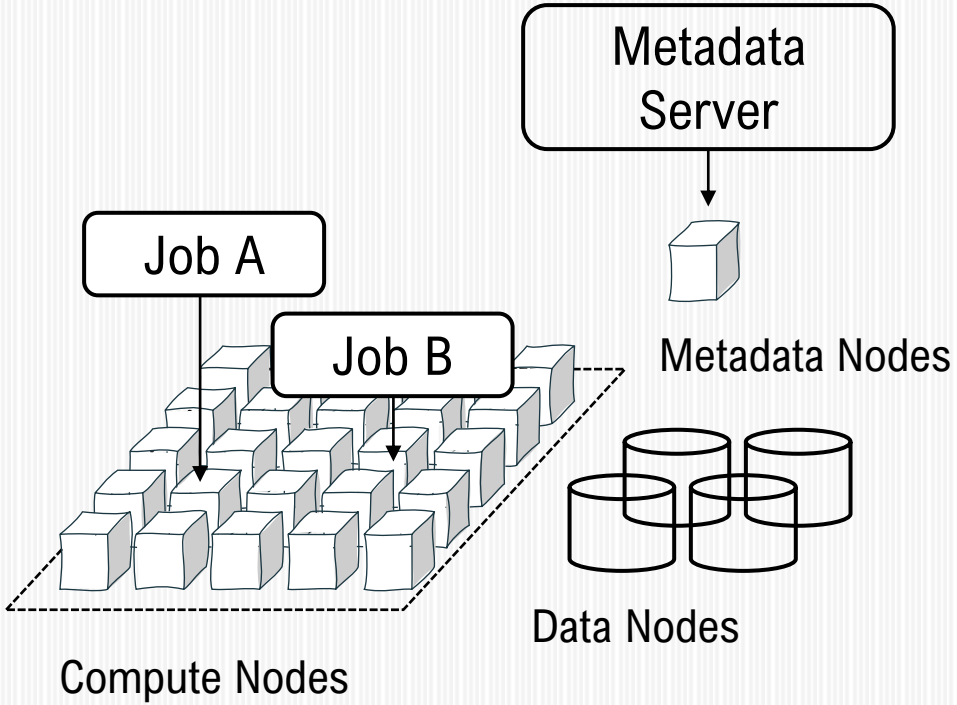
We Tried Multi Metadata Servers

- Run **many** (instead of 1) metadata server
- Each server manages a partition of the FS
- **Problem**: Requires potentially **lots of servers** for peak performance



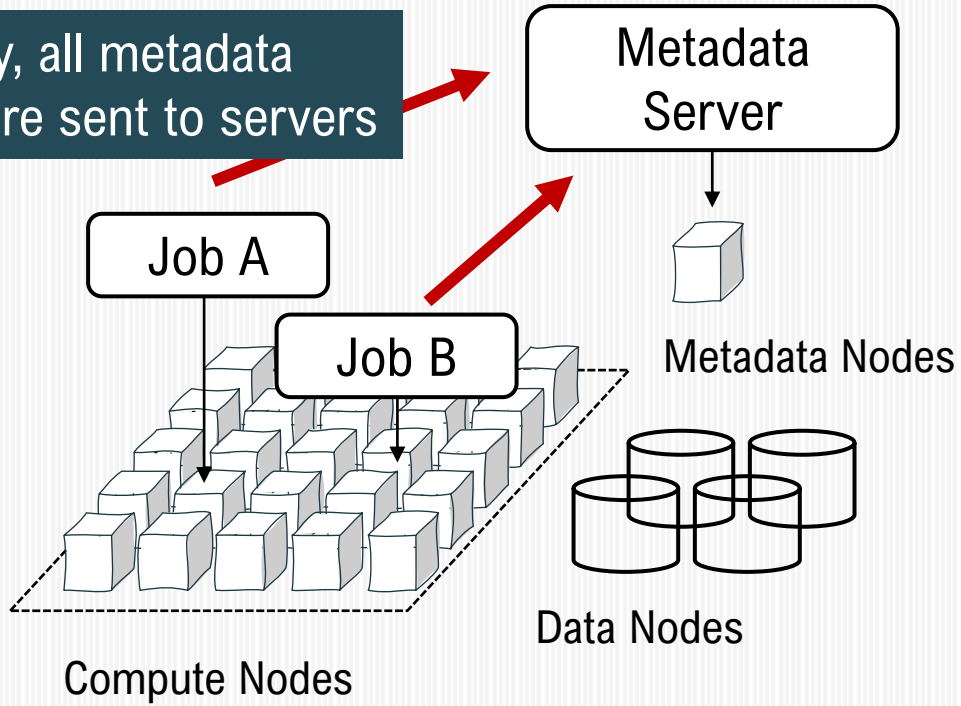
Write Buffering Partially Worked

- **Example:** IndexFS [SC14 Best Paper] allows clients to log metadata writes for **bulk insertion**



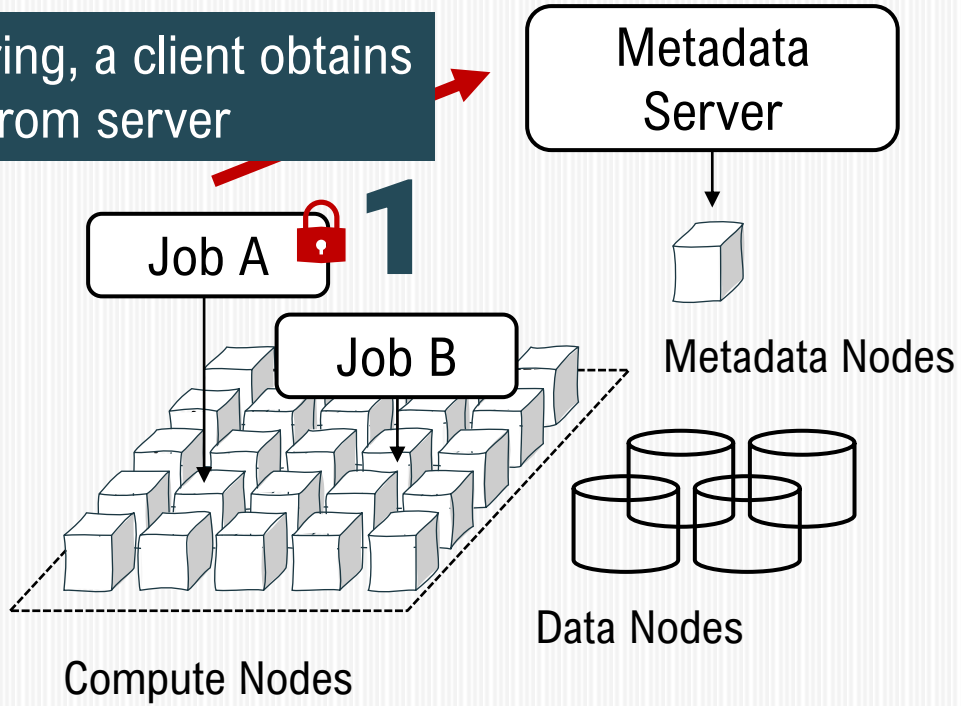
Write Buffering Partially Worked

- **Example:** IndexFS [`Best Paper`] allows clients to log metadata writes for **bulk insertion**



Write Buffering Partially Worked

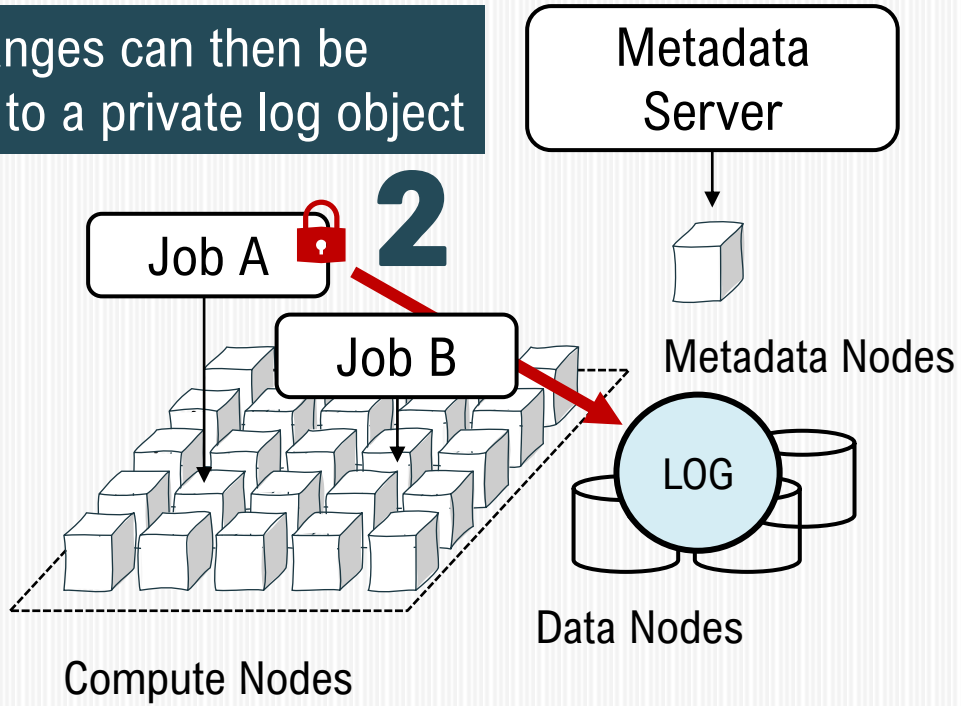
- **Example:** [Best Paper] allows clients to log metadata writes for **bulk insertion**



Write Buffering Partially Worked

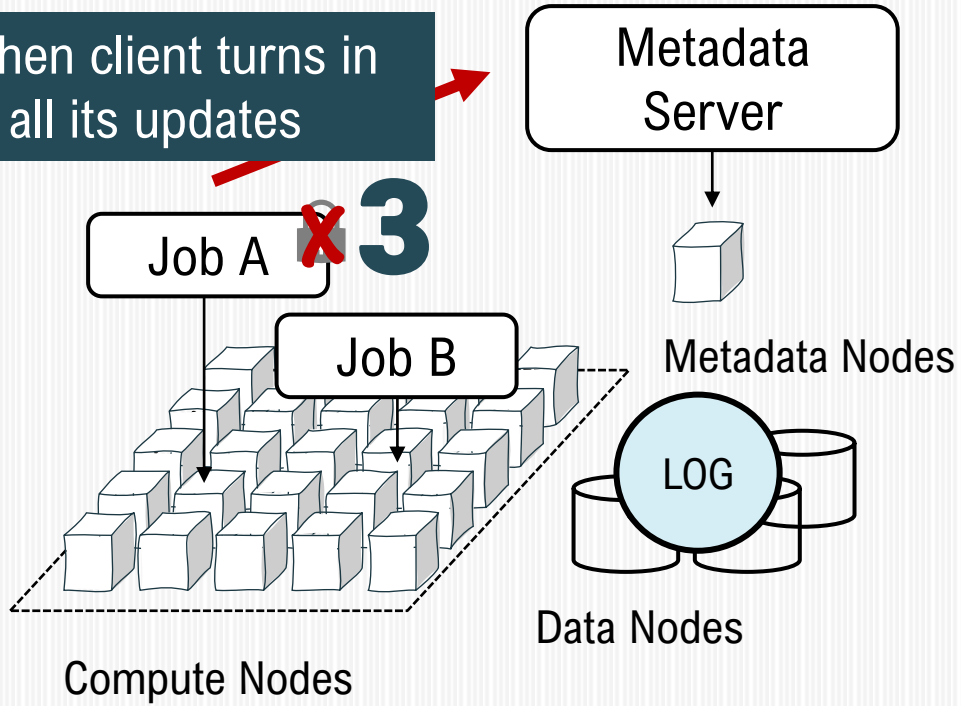
- **Example:** IndexFS [`Best Paper`] allows clients to log metadata writes for **bulk insertion**

All changes can then be committed to a private log object



Write Buffering Partially Worked

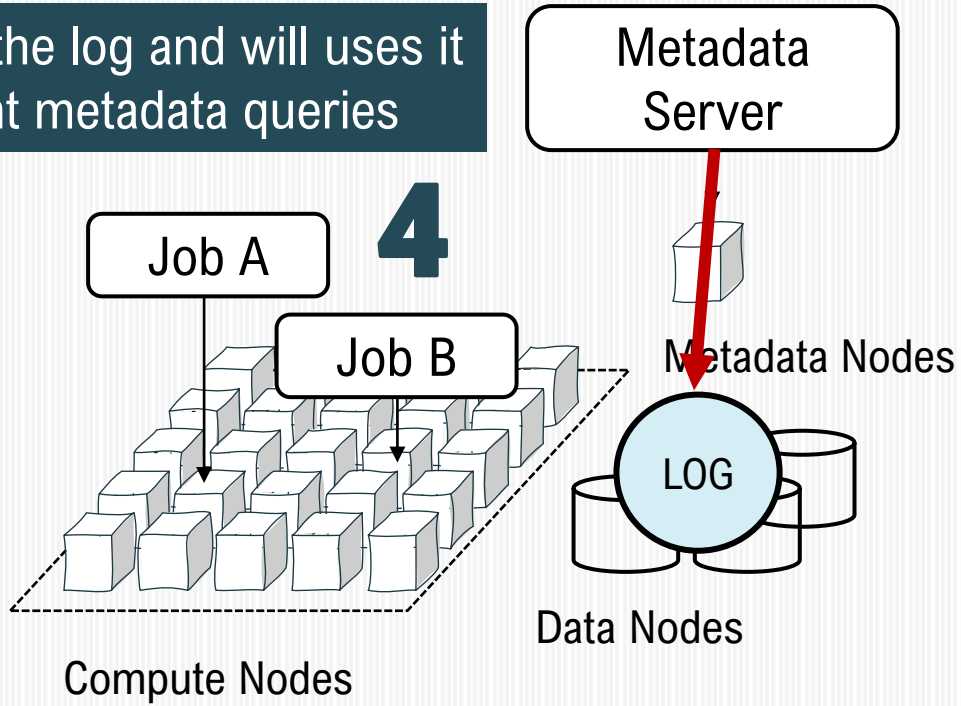
- **Example:** [Best Paper] allows clients to log metadata writes for **bulk insertion**



Write Buffering Partially Worked

- **Example:** Inde
  `Best Paper`] allows clients to log metadata writes for **bulk insertion**

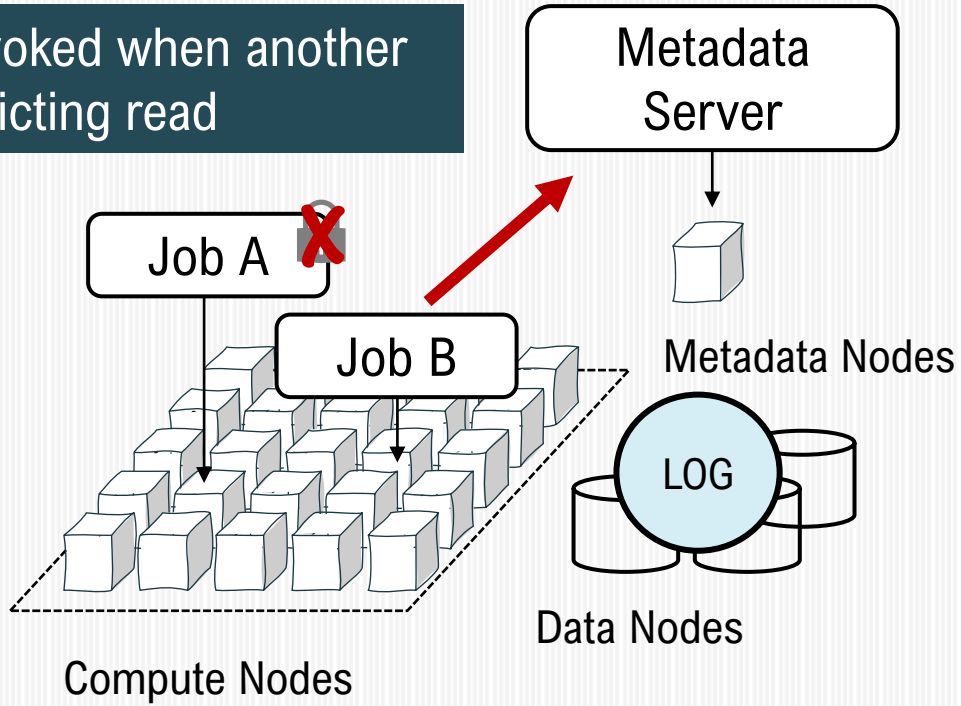
Server registers the log and will uses it for subsequent metadata queries



Write Buffering Partially Worked

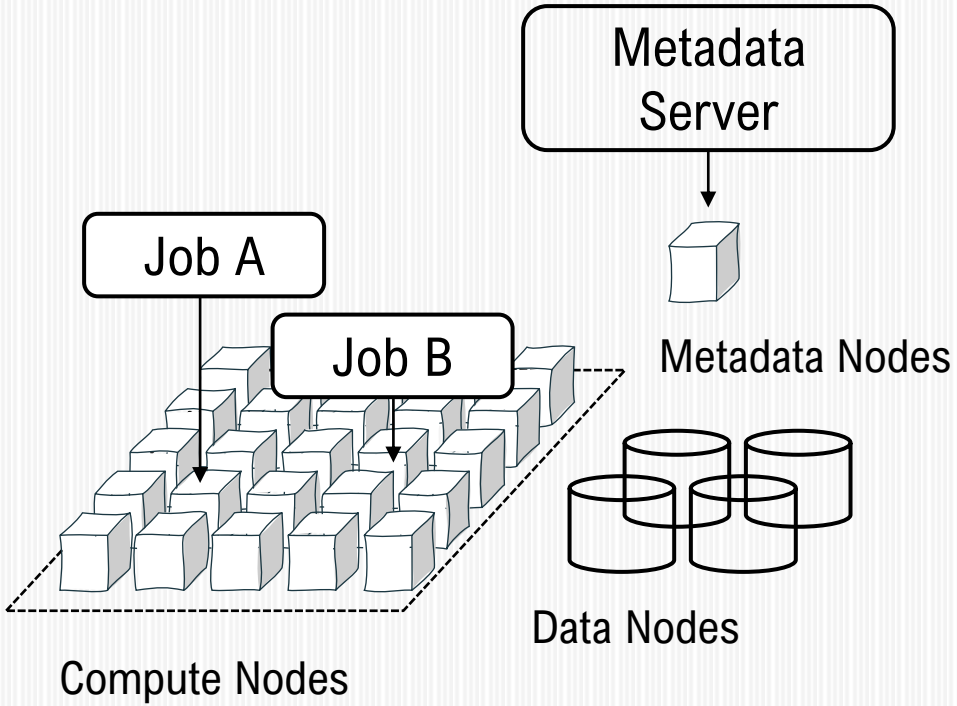
- **Exa**
  `Best Paper`] allows clients to log metadata writes for **bulk insertion**

Note that locks are also revoked when another client does a conflicting read



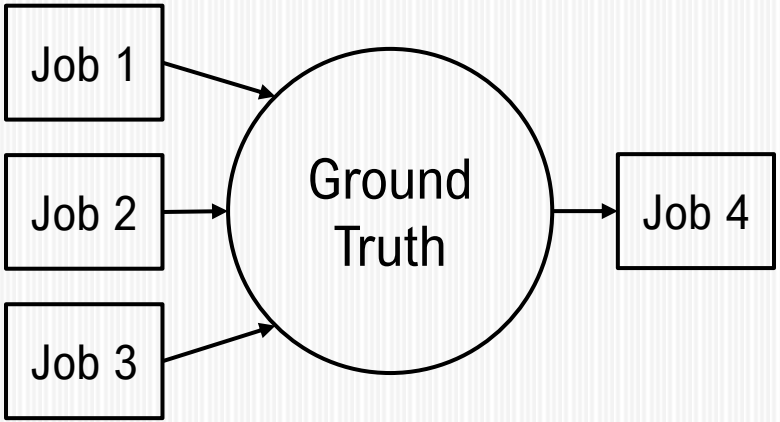
A Job Cannot Hold Its Lock Forever

- **Bulk insertion is inevitable** because we need it for ground truth
- **All jobs** must eventually turn in all logs
- **Back to slow mode** after bulk insertion



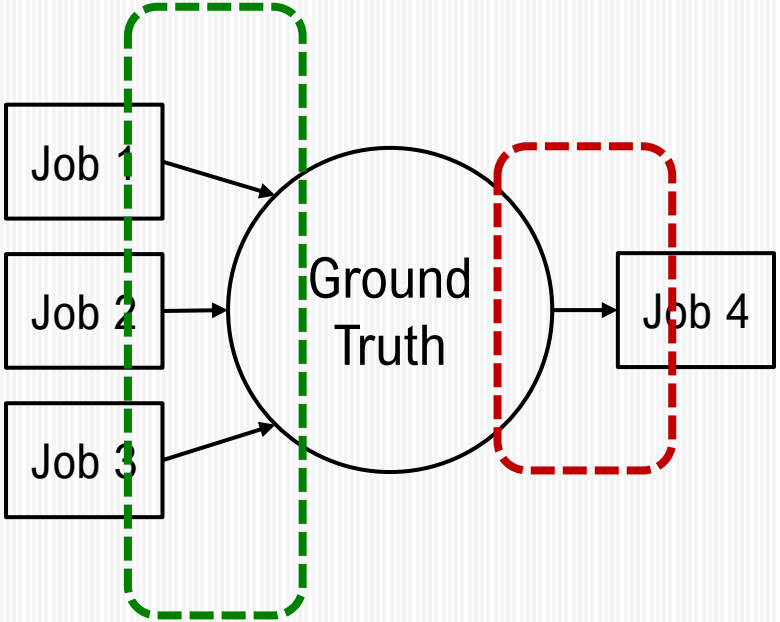
Why Do We Maintain Ground Truth?

- Because **cross-job data sharing**
- **Example**: A followup job needs to see the output of a previous job



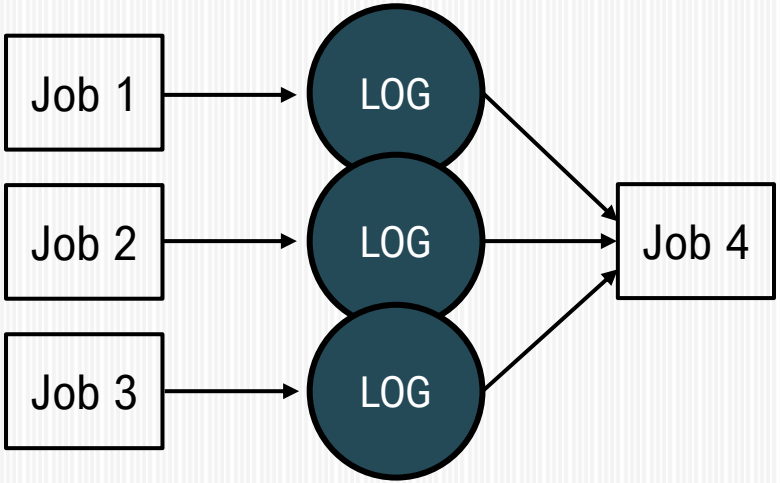
Maintaining A Single View is Costly

- **Performance limited by servers**
- **Writes** may be fast
- **Reads** continue to be slow



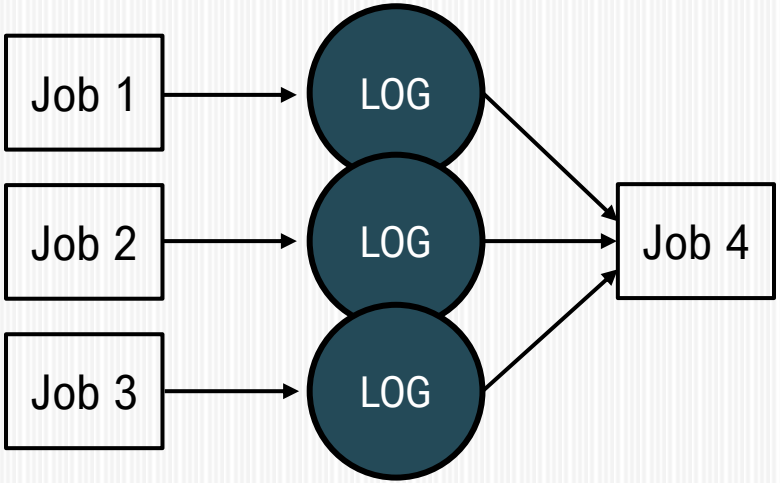
What If We Go No Ground Truth

- **Jobs** publish logs to servers
- **Servers** simply register logs but do nothing else
- **Followup jobs** themselves merge logs for metadata queries



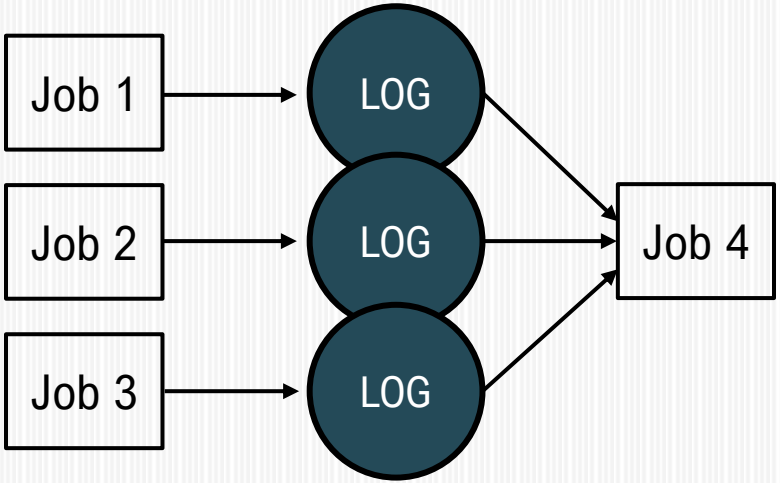
Performance Not Limited by Servers

- All metadata operations are **parallelized** across client CPU cores
- Metadata performance scales as cluster size increases



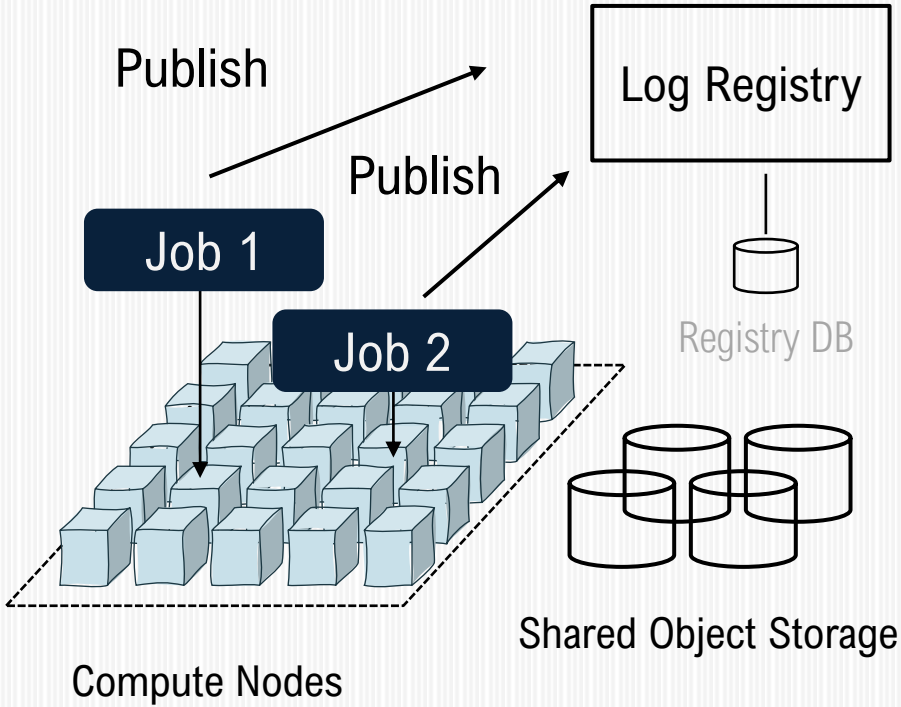
Not All Logs Need Merging

- **Small metadata footprint** per job --- jobs only pay for what they need to see



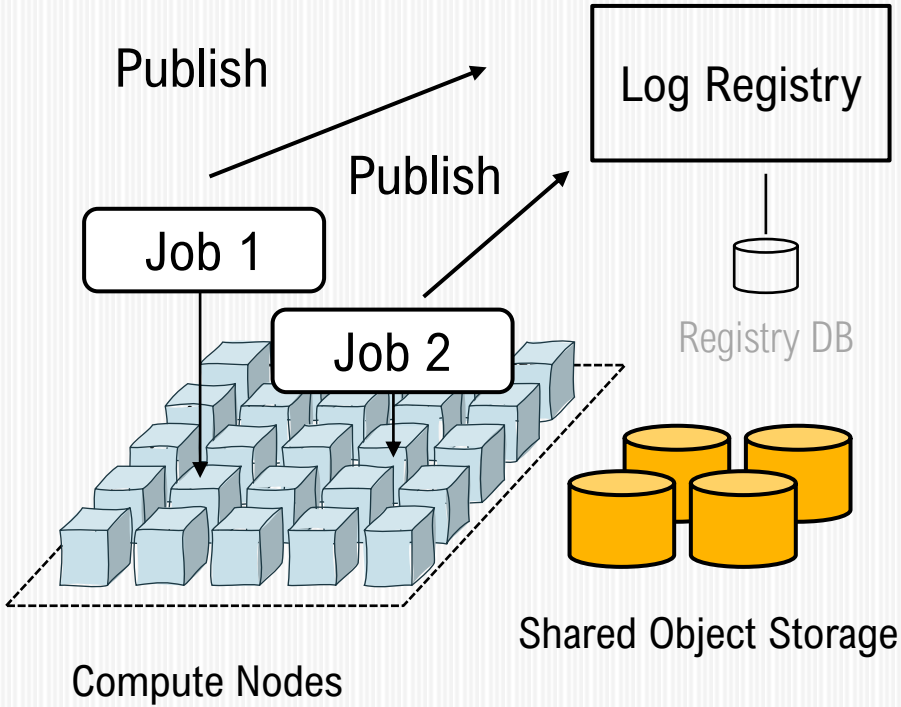
DeltaFS Overview

- Each job takes previous jobs' logs as **input**, and publishes **output** as a new log



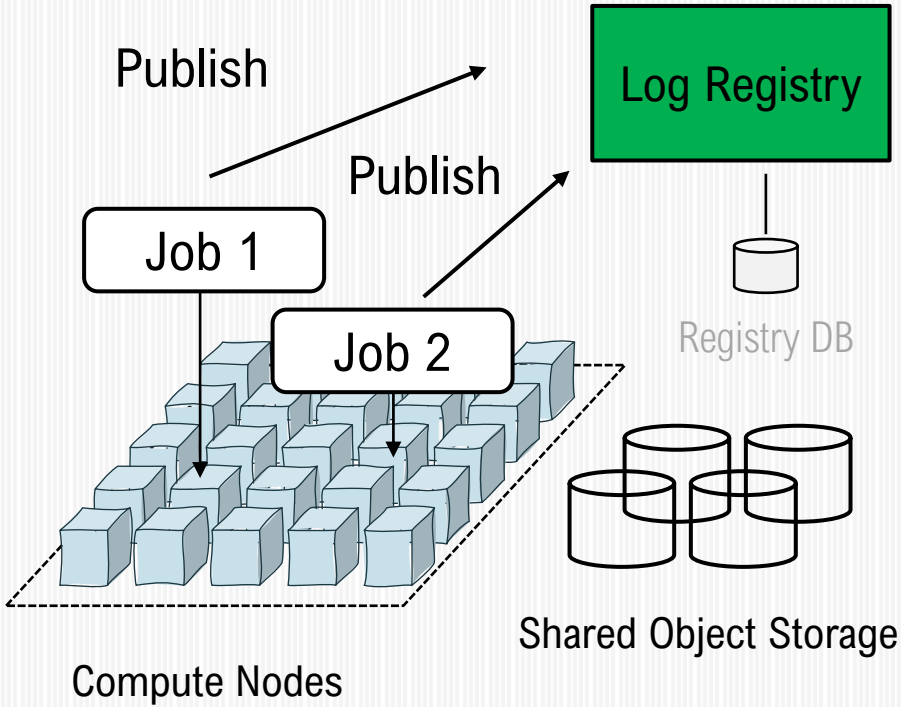
DeltaFS Overview

- Each job takes previous jobs' logs as **input**, and publishes **output** as a new log
- A shared **underlying object store** stores all logs

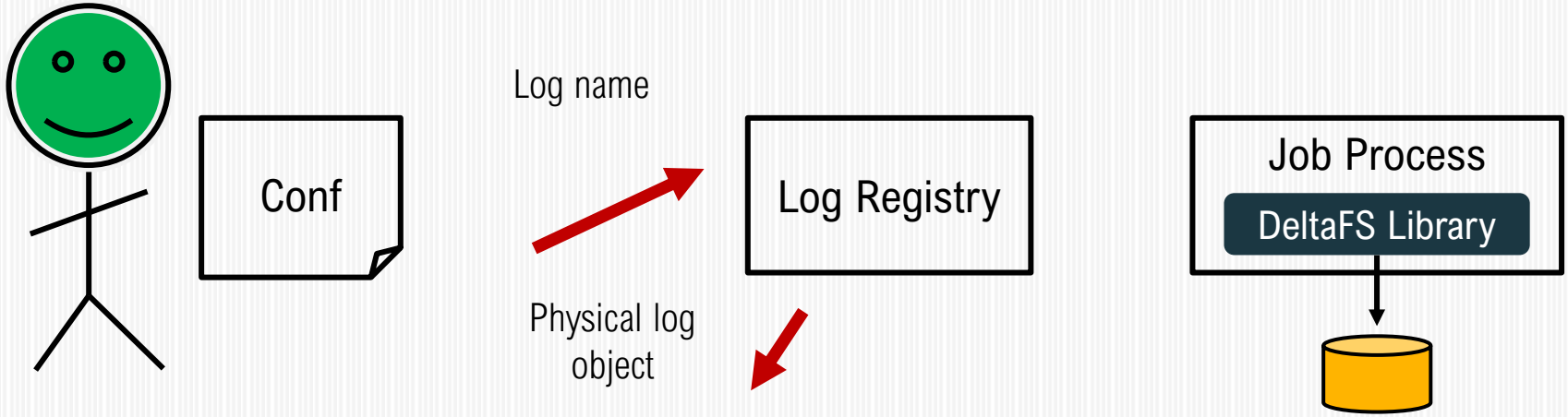


DeltaFS Overview

- Each job takes previous jobs' logs as **input**, and publishes **output** as a new log
- A shared **underlying object store** stores all logs
- A **log registry** catalogs all logs



Job Bootstrapping

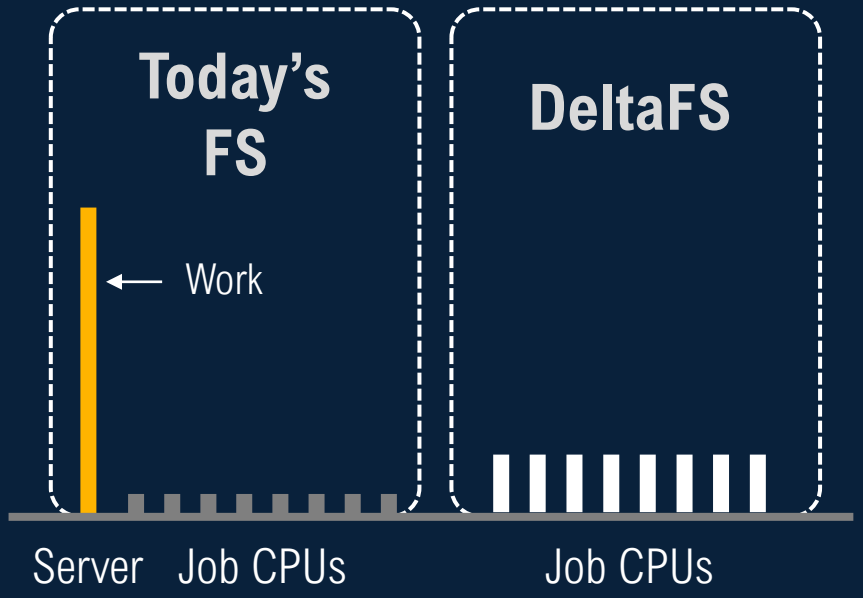


1. User specifies input and output logs

2. Registries map log names to physical log objects

3. DeltaFS reads logs for metadata information

Key Idea



- **Today's FS** limits metadata work to servers
- **DeltaFS** allows parallelizing work among client compute nodes

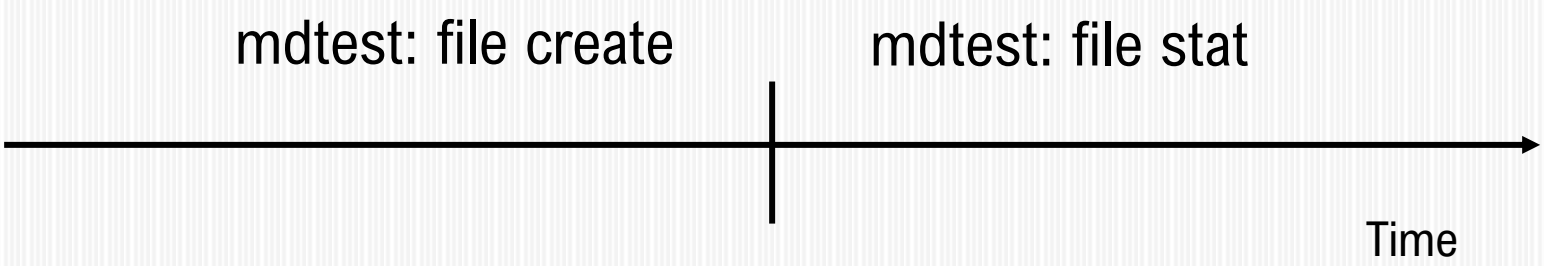
Micro Results

Baseline: IndexFS [SC14 Best Paper]

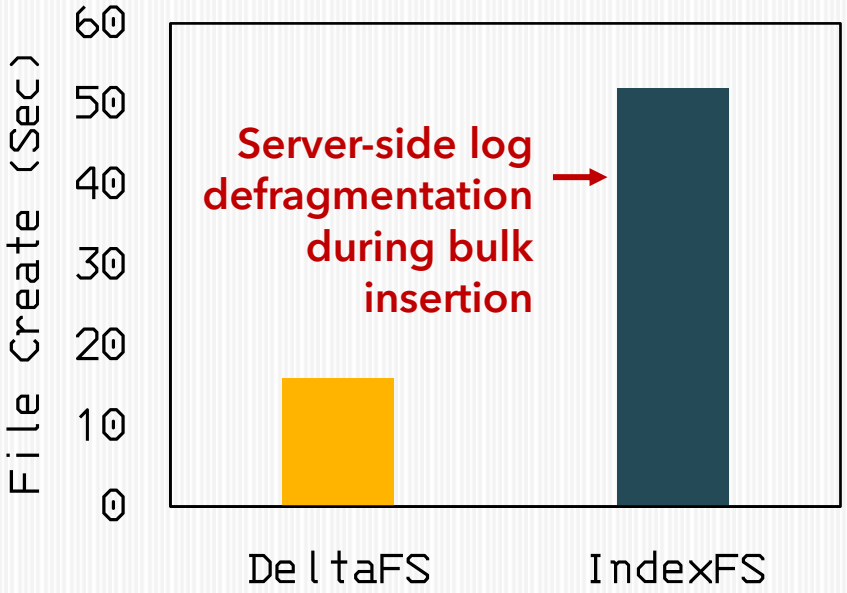
- **Scalable** parallel filesystem **metadata plane**
- Up to **351x** faster than Lustre, PanFS, and HDFS
- Current state-of-the-art

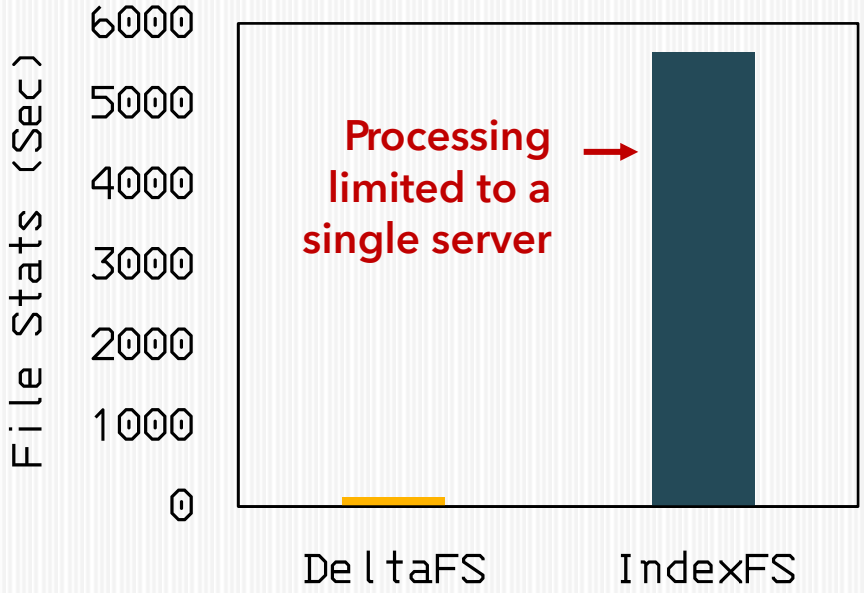
https://doi.org/10.1109/SC.2014.25

Workload



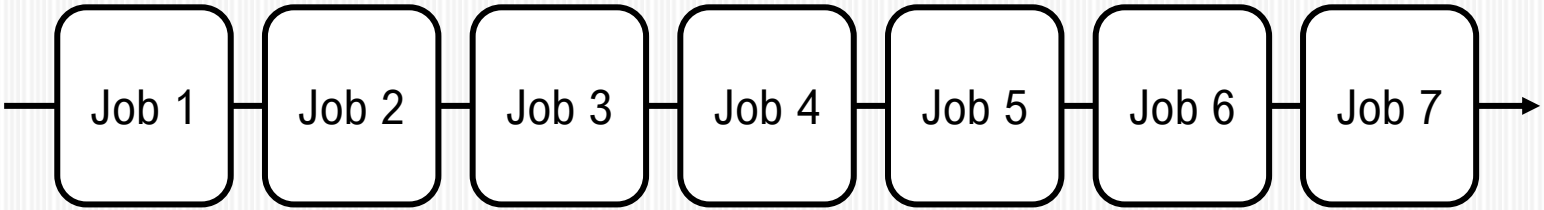
Results


File Create (Sec)
60
50
40
30
20
10
0
Server-side log defragmentation during bulk insertion
DeltaFS
IndexFS

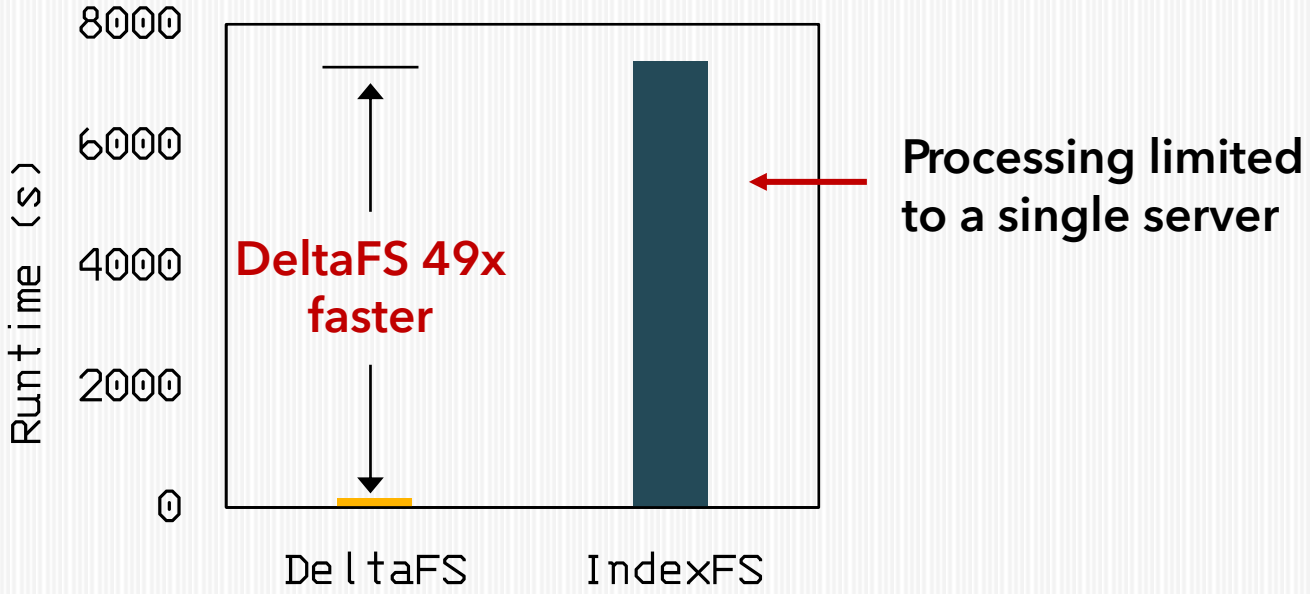

File Stats (Sec)
6000
5000
4000
3000
2000
1000
0
Processing limited to a single server
DeltaFS
IndexFS

Macro Results

Workload: A 7-Stage Workflow

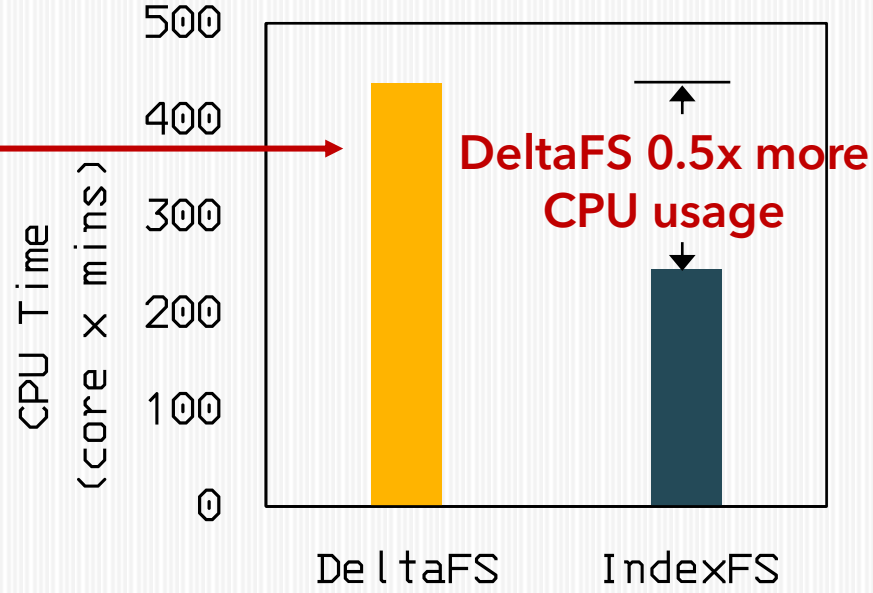


High Cross-Job Performance



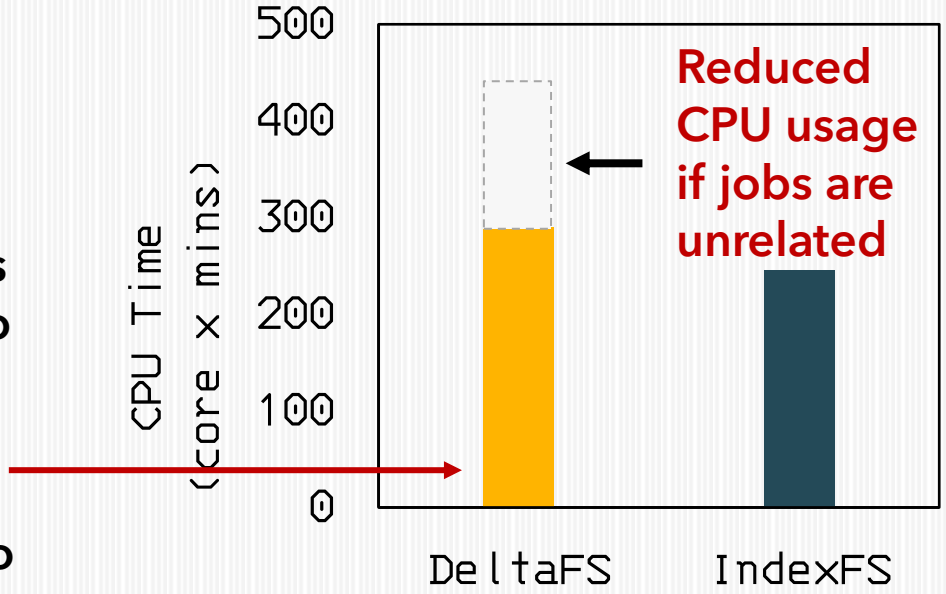
Cost of No Ground Truth

DeltaFS causes repeated log defragmentation work at each workflow stage



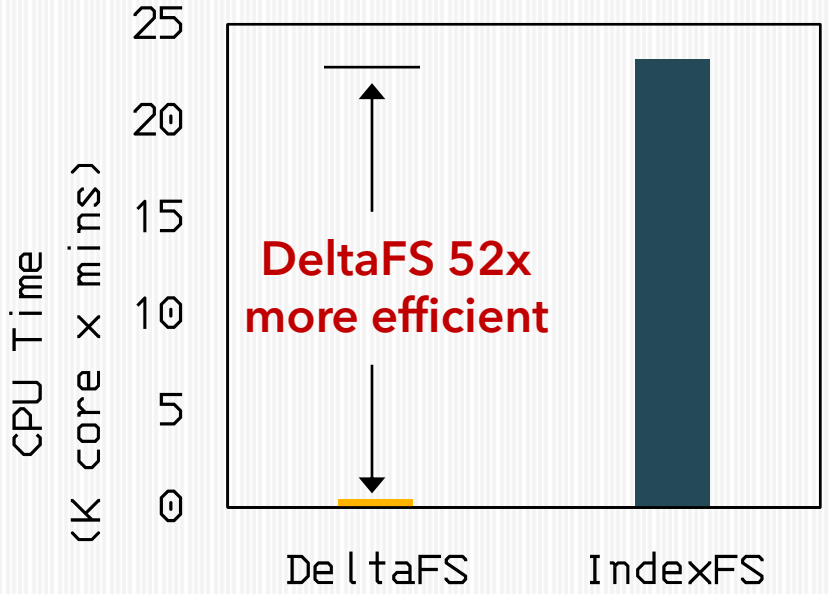
Jobs Only Pay for What They Need

Unrelated jobs do not need to read or defragment each other's logs leading to reduced work



Better Overall Resource Usage

- **IndexFS** has clients wait for server forfeiting their compute power
- **DeltaFS** leverages client resources for scalable metadata improving overall system (client + server) resource usage



Conclusion

- Ground truth (a single FS view) limits processing to servers
- DeltaFS transforms **one ground truth** to a collection of **facts** (logs) that jobs can use to compose their own FS views
- **Parallelizing** metadata processing on compute nodes enables higher, scalable performance

DeltaFS Also Has a Data Plane

- This paper presents DeltaFS **metadata plane**
- Please see our SC18 and ACM Trans. Storage papers for DeltaFS data plane

Thanks!

Please see our paper for more DeltaFS information and results

Backup Slides

Anonymous Synchronization

- **User** declares subtrees as atomic
- DeltaFS **forwards** all atomic requests to an external service (e.g., ZooKeeper) for synchronization
- **Key point**: No need to treat *every* metadata operation as atomic