Distributed OS

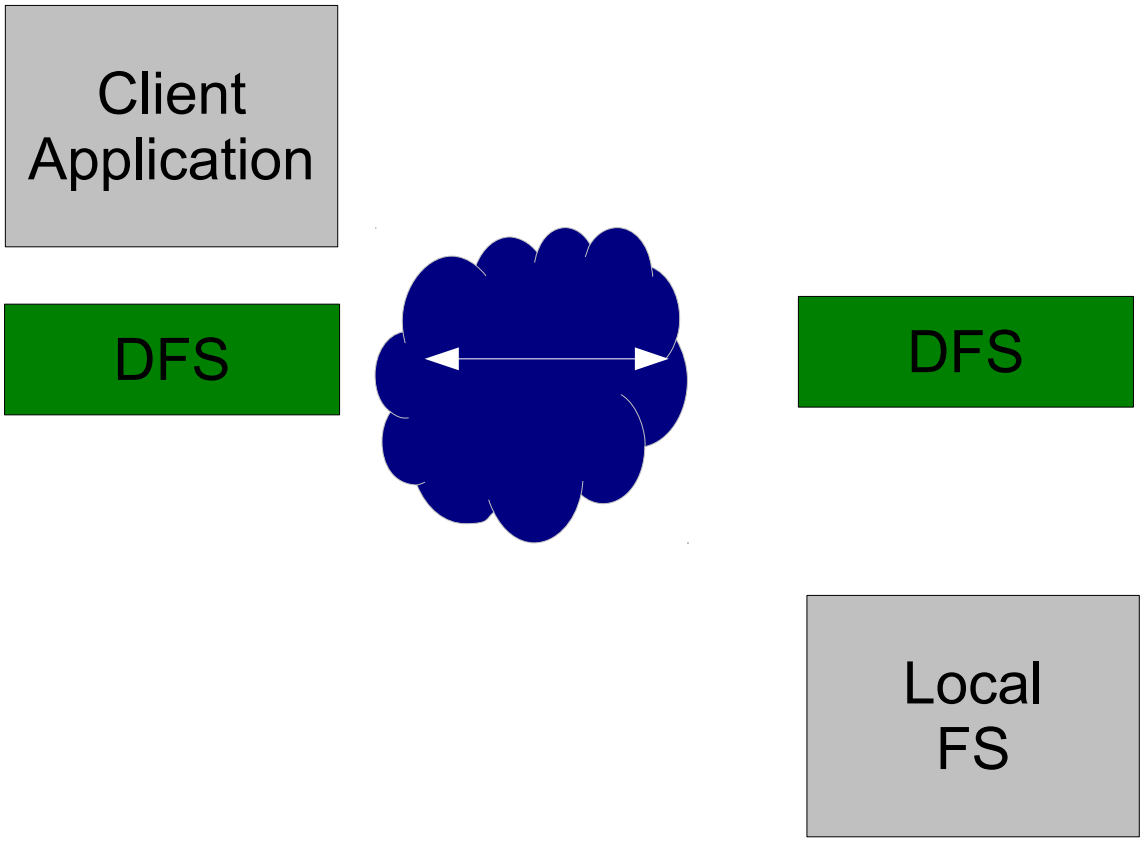
Hermann Härtig

File System Scalability

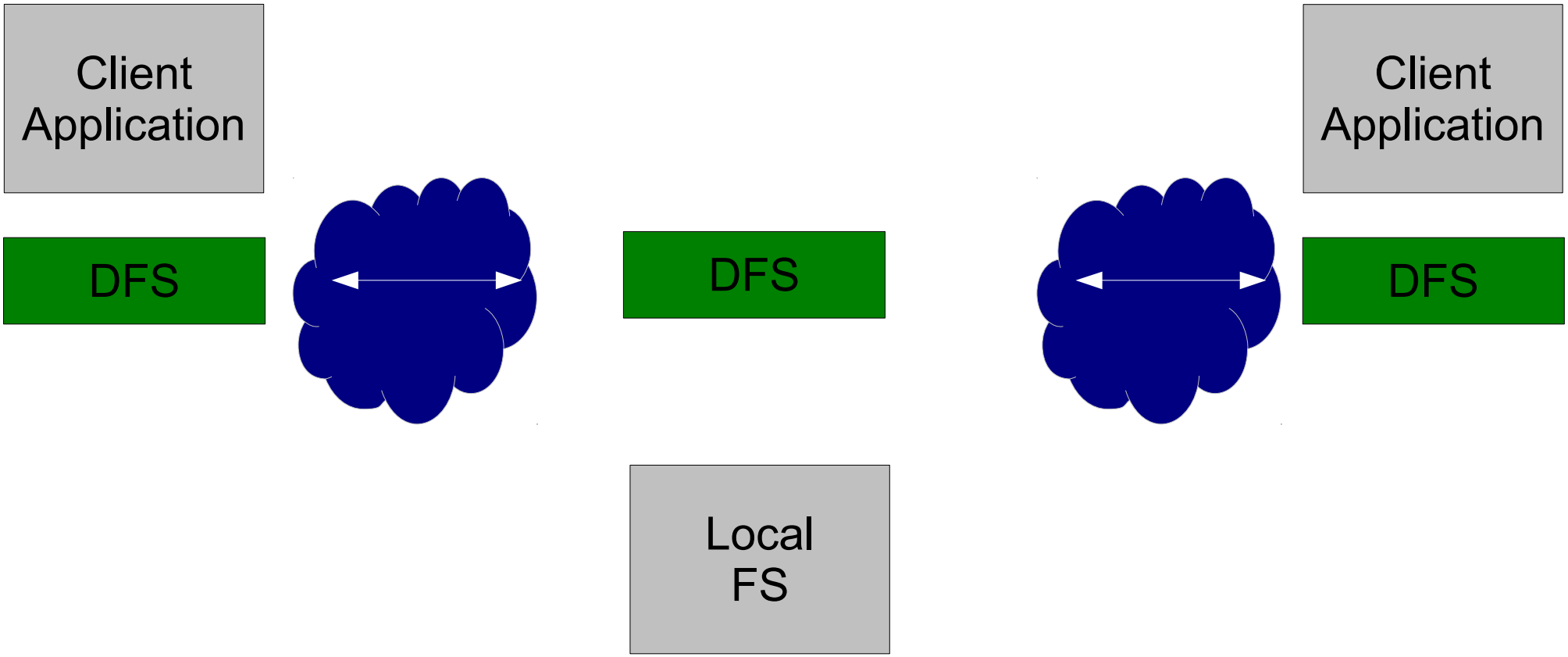
(NFS → AFS →) G(oogle-)FS

TECHNISCHE UNIVERSITÄT DRESDEN

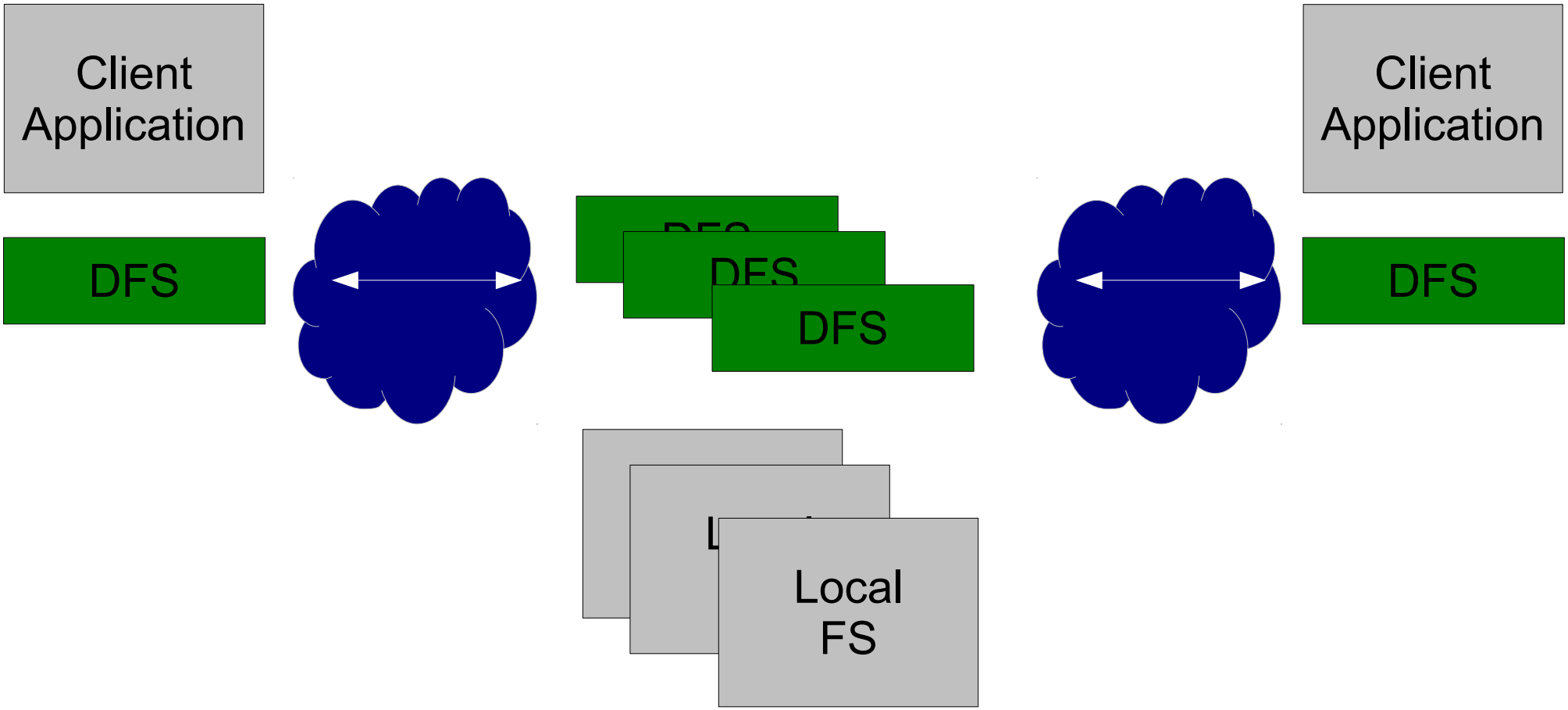
Distributed FS



Distributed FS



Distributed FS



Network File System (NFS, SUN v2)

Design Principles

- Interface: as close as possible to Unix
- names/"open": message to file server for each part of pathname
- Read/Write: use block cached in client
- Cache: blocks
- Replication: none (only very rudimentary)
- Consistency: exchange consistency messages every few s
- Faults: write through on server (v2),
  hold up client when server crashes

Andrew File System

Design Principles

- Interface: as close as possible to Unix
- names/”open”: name resolution on client using caches directories
- Read/Write: use file cached in client
- Cache: whole files, use server replicates to fetch files to cache
- Replication: some (performance)
- Consistency: session, explicit invalidation of cache through “call backs”
- Faults: ?

Scalability limitations

- Very general usage model
  - Posix consistency model
    - cache validation (distributed caches)
    - write semantics (need to load block or file before writing)
  - Too many small messages
- One-to-one mapping of files
- Distribution-related metadata and user-data on same server
- No inherent fault tolerance
- No or simple replication

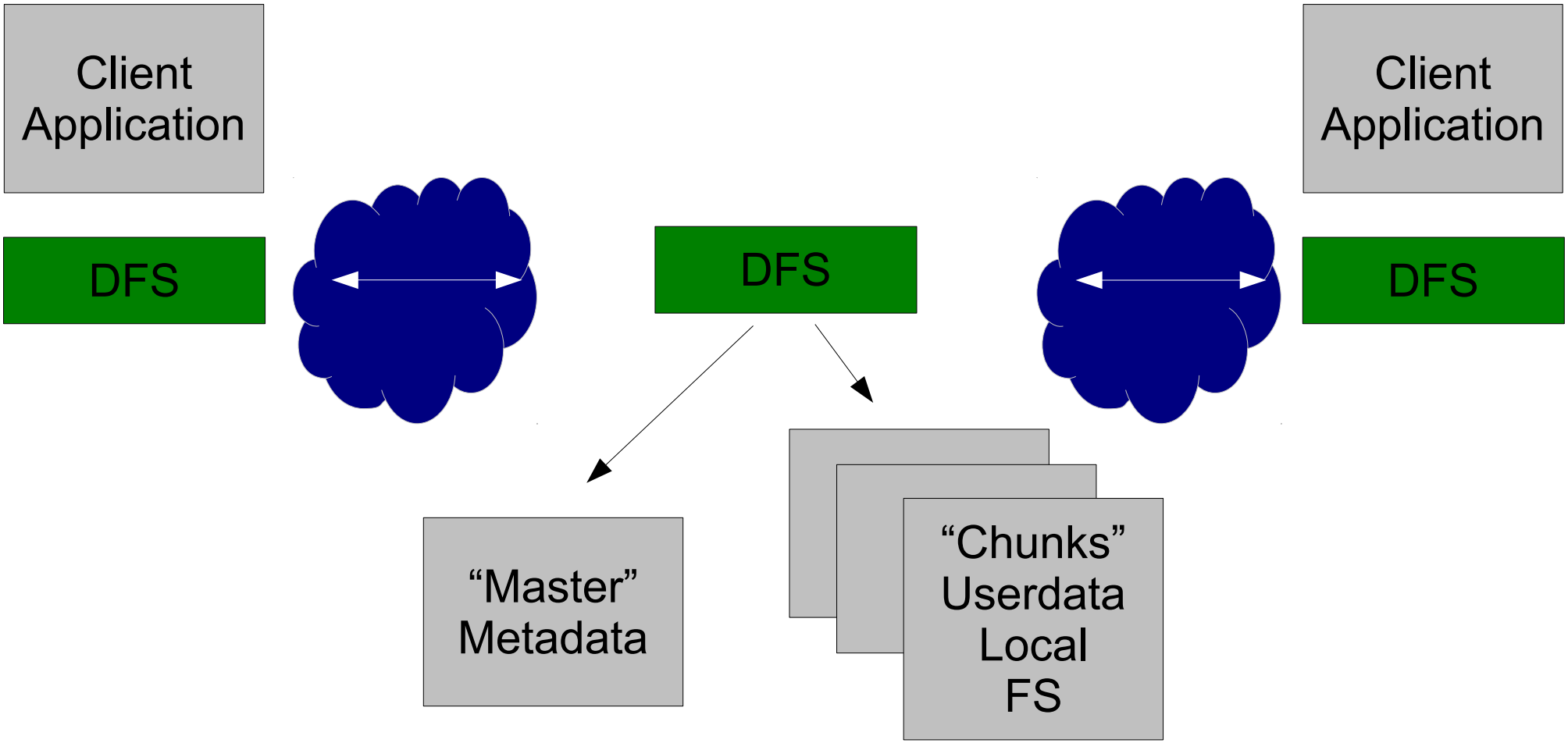
Google File System

Design Principles

- Interface: specialized for google applications
- names/"open": name resolution on "Master server"
- Read/Write: direct communication with "chunk servers"
- Cache: no caching of user data
- Replication: "chunks" are replicated
- Consistency: relaxed consistency (replicas need not be identical)
- Faults: inherent in design

Details: see slides and paper (presentation at SOSP 2003)

Distributed FS



Architecture

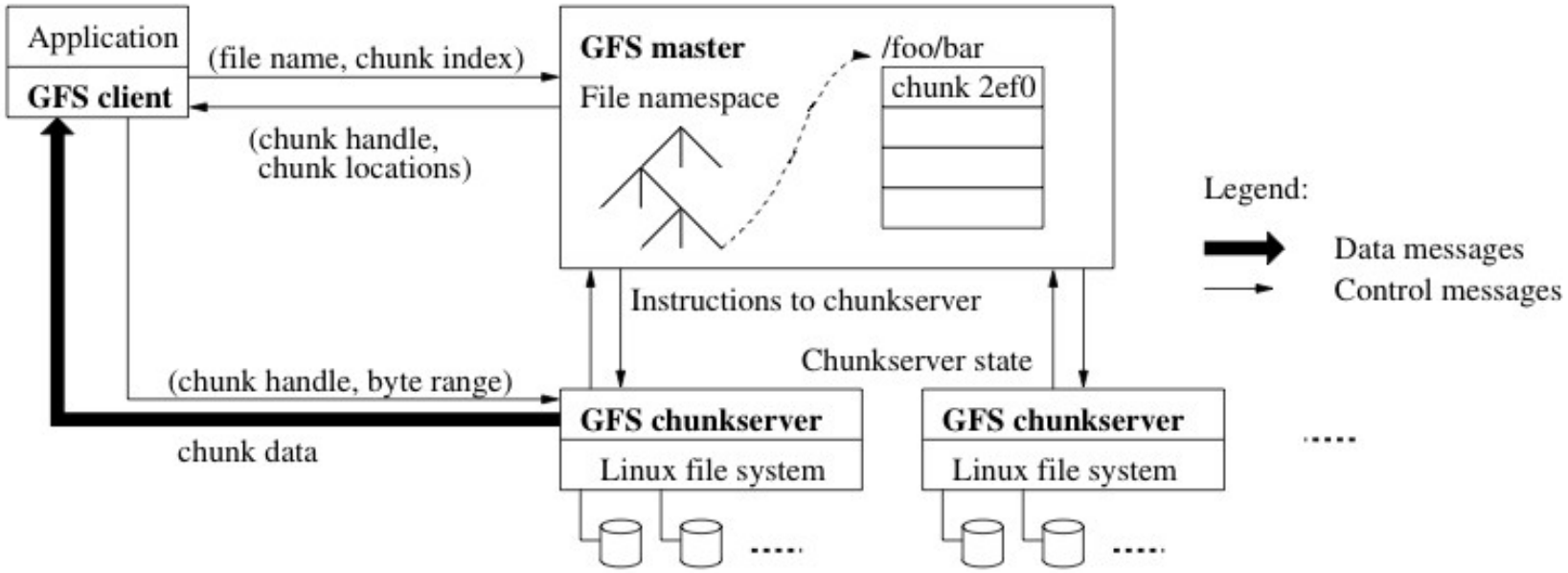


Figure 1: GFS Architecture

Figure from Ref 2(see last slide)

Write

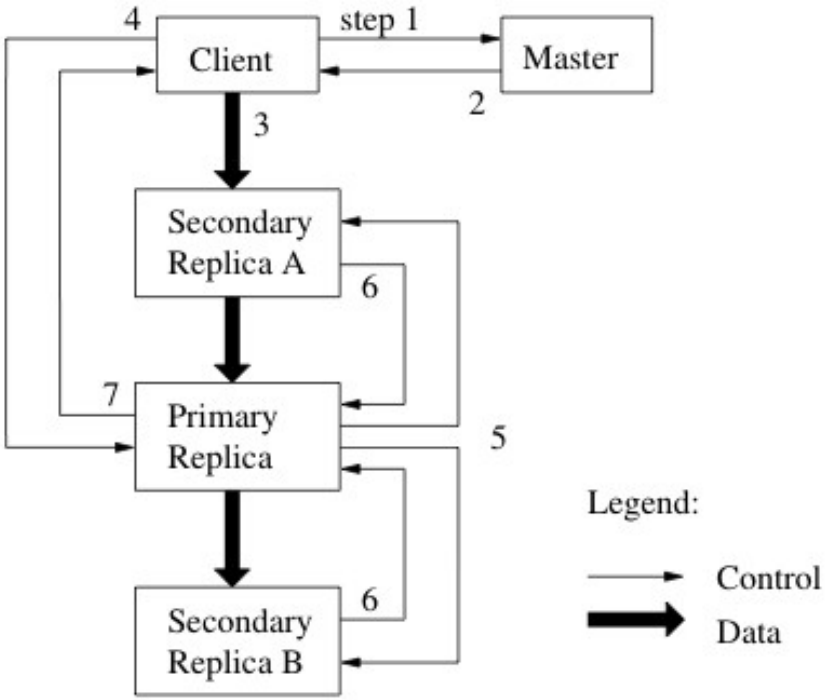


Figure 2: Write Control and Data Flow

Figure from Ref 2(see last slide)

Write 2

- No concurrency control → inconsistencies
- But Atomic “Append” with at least once semantics
- Replicas not identical (padding, multiple/aborted writes, …)

  → applications must recognize “records”

  → checksum per chunk-replicate
- Applications' data structures must be **designed** to tolerate this

Replicas

- Replicas are not always identical
- Replicas may become stale → version numbers
- Master stores current version in operation log
- Chunk servers register with master on start up

Fault Tolerance

- Little on fault handling in SOSP slides

  → read the SOSP paper and prepare to work on it in the exercises!!!

More Cluster File Systems

- Lustre for HPC
- G(lobal-)FS
- PVFS
- Tidy FS (to appear June 2011, Usenix ATC)
- Hadoop FS
-

References

1) NFS/AFS: many text books

2) Google FS:

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Symposium on OS Principles 2003