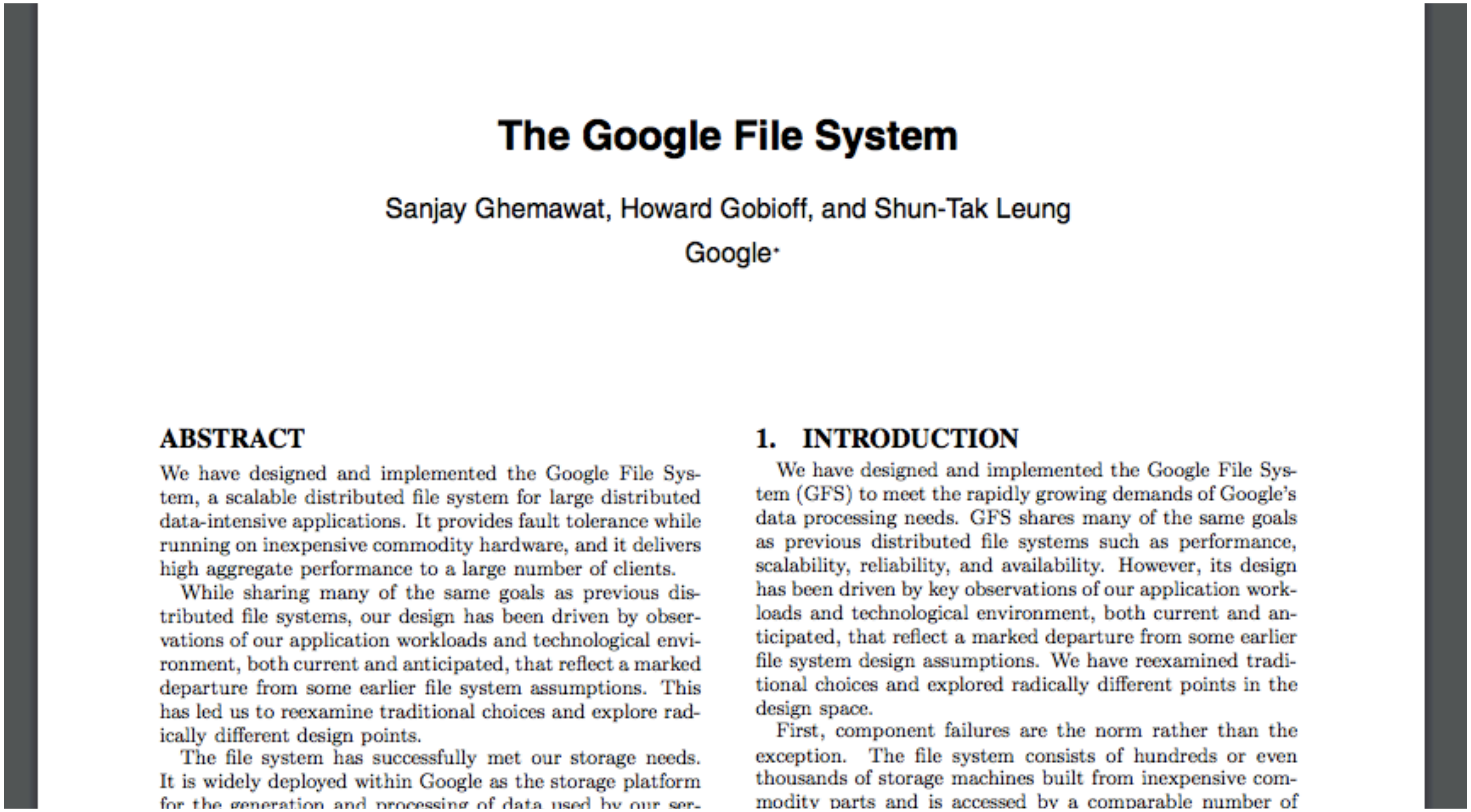
Case-study of primary-backup replication



The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

ABSTRACT
We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our ser-

1. INTRODUCTION
We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of

Symposium of Operating Systems Principles (SOSP) 2003

Goal of GFS

- Many computation stores huge amounts of data and demands high throughput
  - many concurrent readers / writers
- Example: parallel web crawler

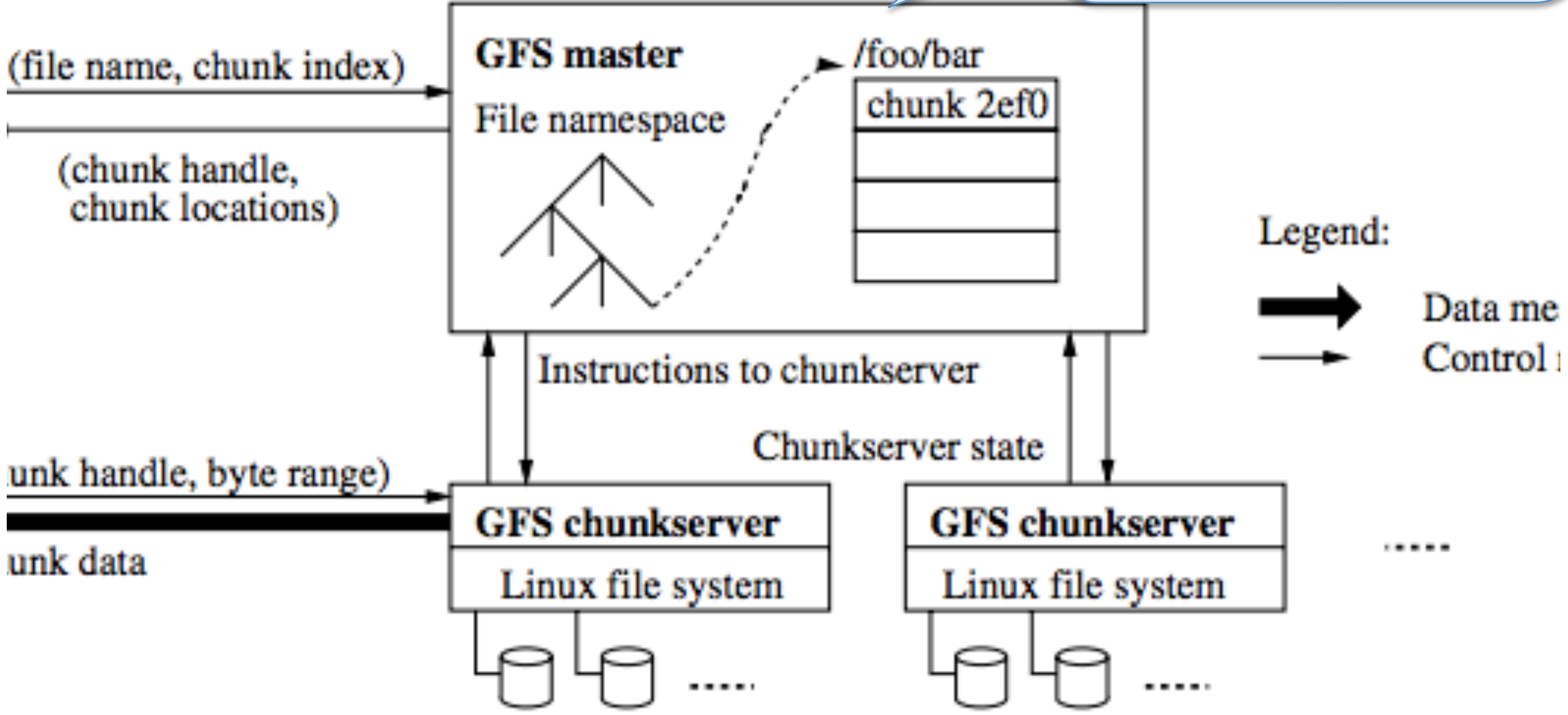
GFS high level design

- The API
  - File system API: directories, files, open/read/write/append
  - Not POSIX compatible

GFS architectu

A single point of control: master is responsible for:

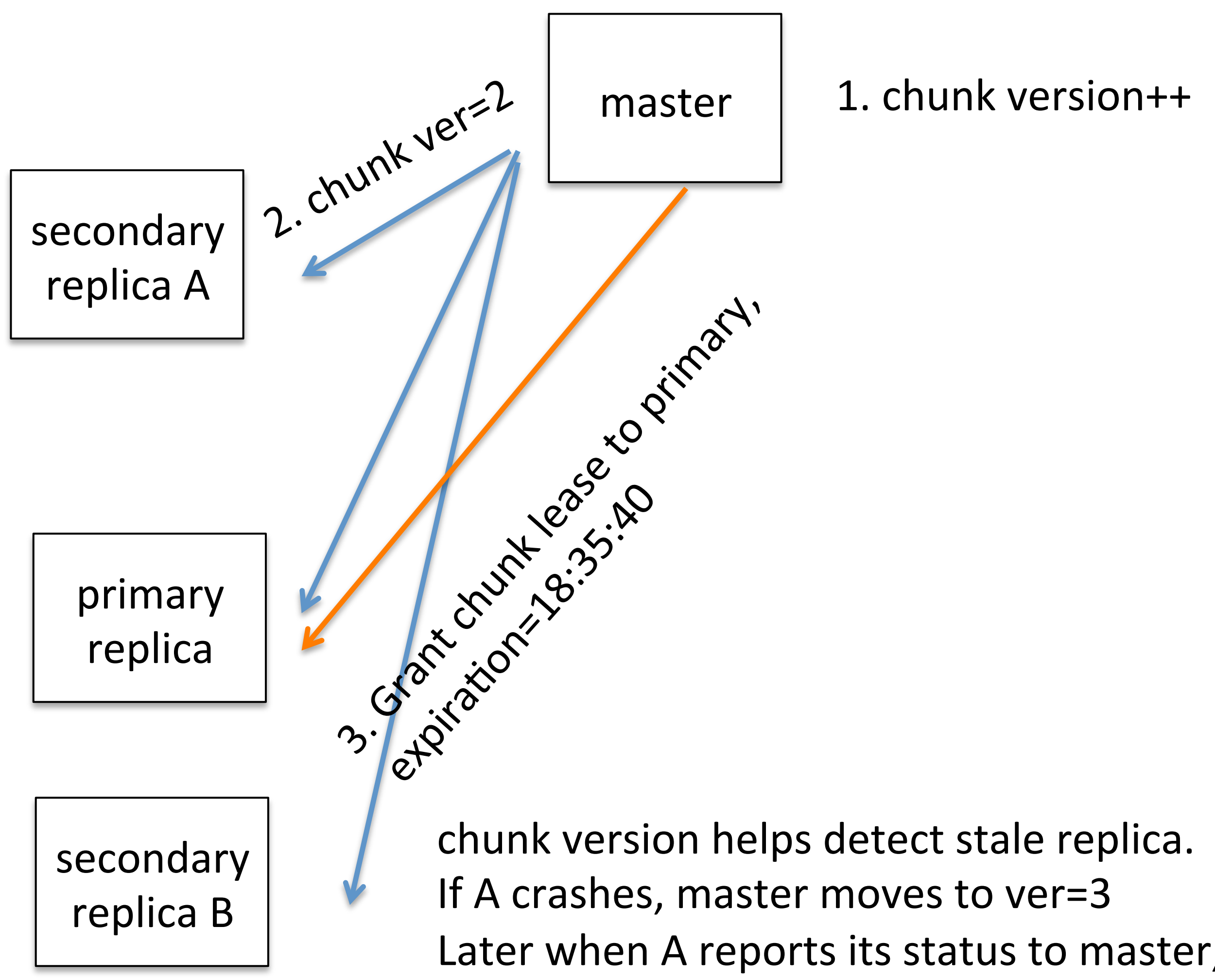
- file and chunk namespace
- file→chunks
- chunks→replica servers



GFS master

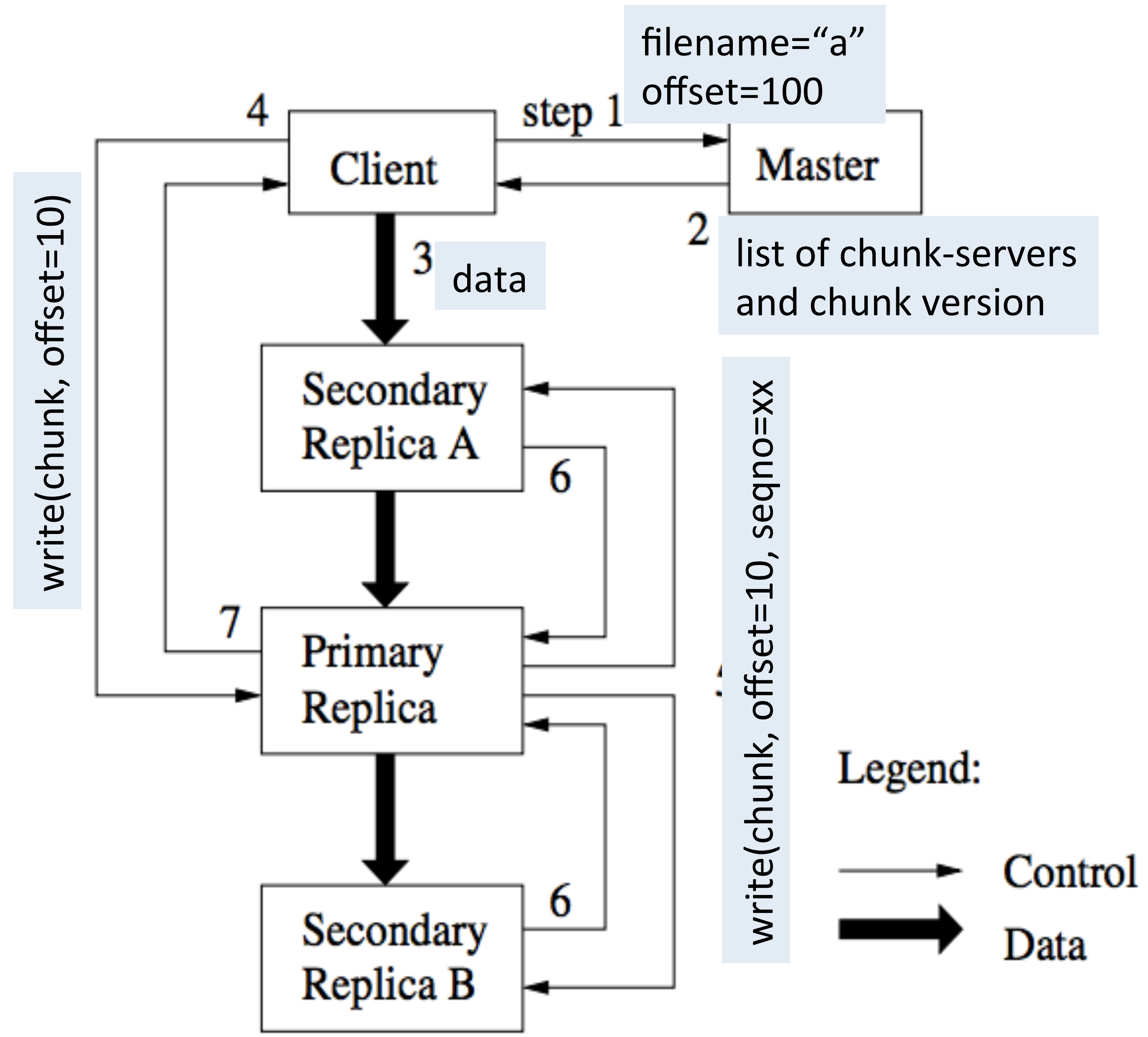
- GFS master server stores meta-data:
  - For a directory, what files are in it
  - For a file, the set of chunk servers for each 64 MB chunk
  - For a chunk, the set of replica servers storing it
- master keeps state in memory
  - 64 bytes of metadata per each chunk
  - master replicates operation log to master replicas
- External monitor performs master switching

Master coordinates replica-group for each chunk



chunk version helps detect stale replica.
If A crashes, master moves to ver=3
Later when A reports its status to master,
master knows A is out of date

Primary-backup replication in chunk servers



GFS chunk replication

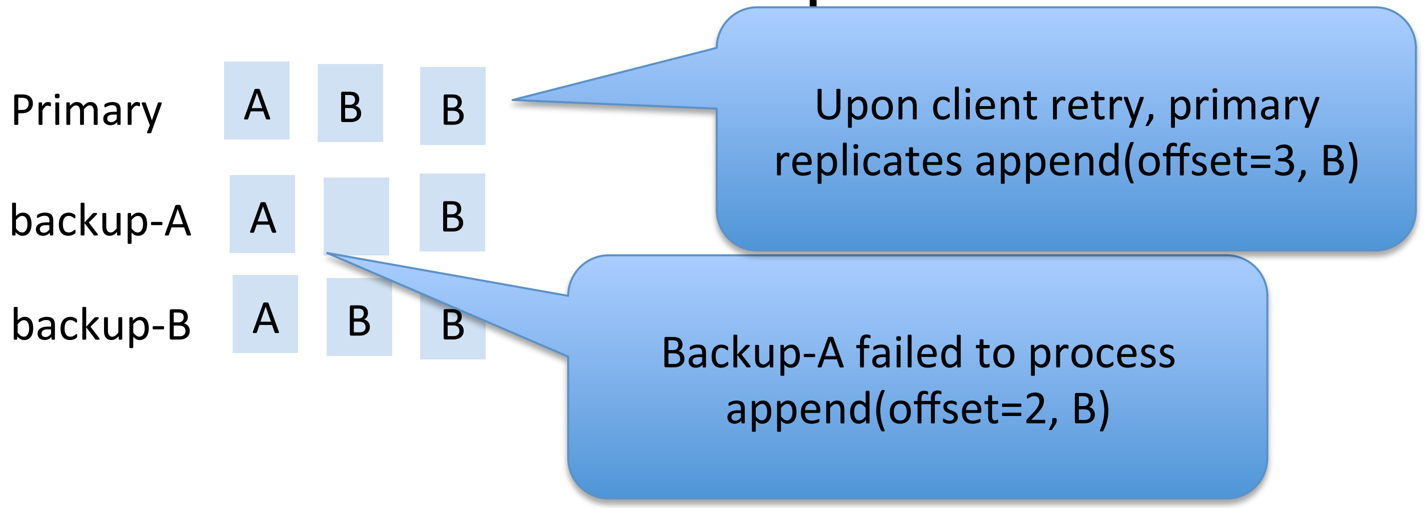
- No logging, chunk servers directly apply writes in-place
  - write(chunk-id, offset=100, “abcdefg”)
- What happens w/ concurrent writes?
  - data of different clients may be mingled
  - Client-1: write(0, 100-byte-of-data)
  - Client-2: write(50, 200-byte-of-data)

GFS atomic appends

- Clients issue append(chunk-id, “bbb”)
- Primary picks offset for append and replicates data at chosen offset.
- What happens during a chunk server failure?
  - client retry

Does GFS achieve "ideal" consistency for atomic append?

- No. A file can have duplicate or holes


Primary
A
B
B
backup-A
A
B
backup-B
A
B
B
Upon client retry, primary replicates append(offset=3, B)
Backup-A failed to process append(offset=2, B)

- No. A "unlucky" client can read stale data
  – primary succeeded in appending
  – client read from stale backup

Impact of GFS

- Google's first distributed system infrastructure
- Simplified design → fast development time
  - single master
  - allow inconsistency during failure
- Worked well for a decade 2003-2012
  - succeeded by Colossus
    - No more single master holding all meta-data

GFS vs. Viewstamp replication

- VR-replication for chunk server?
- VR-replication for master meta-data replication?