Google File System
+
BigTable

Database seminar, Spring 2012
School of Computing, University of Utah

Google File System
+
BigTable

Database seminar, Spring 2012
School of Computing, University of Utah

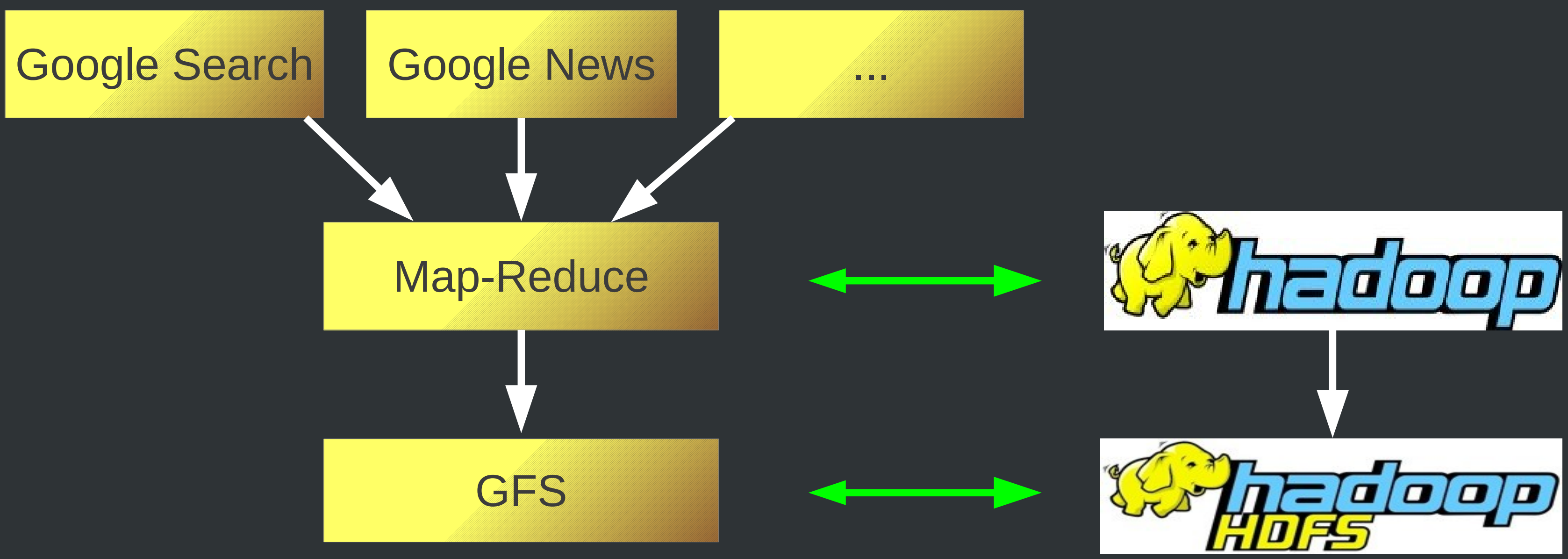
The Google File System(GFS)

- Introduction
- Motivations
- Design Overview
- Fault Tolerance and Replication Management
- Performance Evaluation

The Google File System: Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, Google, SOSP '03

GFS - Introduction

- A scalable distributed file system for large distributed data-intensive applications



Google's Implementation

Open-source Implementation

GFS - Motivations

- Component failures are the norm.
  - A storage cluster is built from hundreds or thousands of inexpensive commodity servers.
- Files are huge: multi-GB
- Most data is appended, rather than overwritten
- Co-designing applications with the file system API increases flexibility

GFS – Design Overview

- Features
  - Recover from component failures
  - Manage huge files efficiently
  - Support for large streaming reads
  - Support for concurrent large appends to the same file
  - High sustained bandwidth

GFS - Interface

- Hierarchical directories
- Operations:
  - Create, delete, open, close, read and write
  - Snapshot: creates a copy of a directory tree at low cost
  - Record append: efficient atomic appends

GFS - Architecture

- Minimize the master's involvement

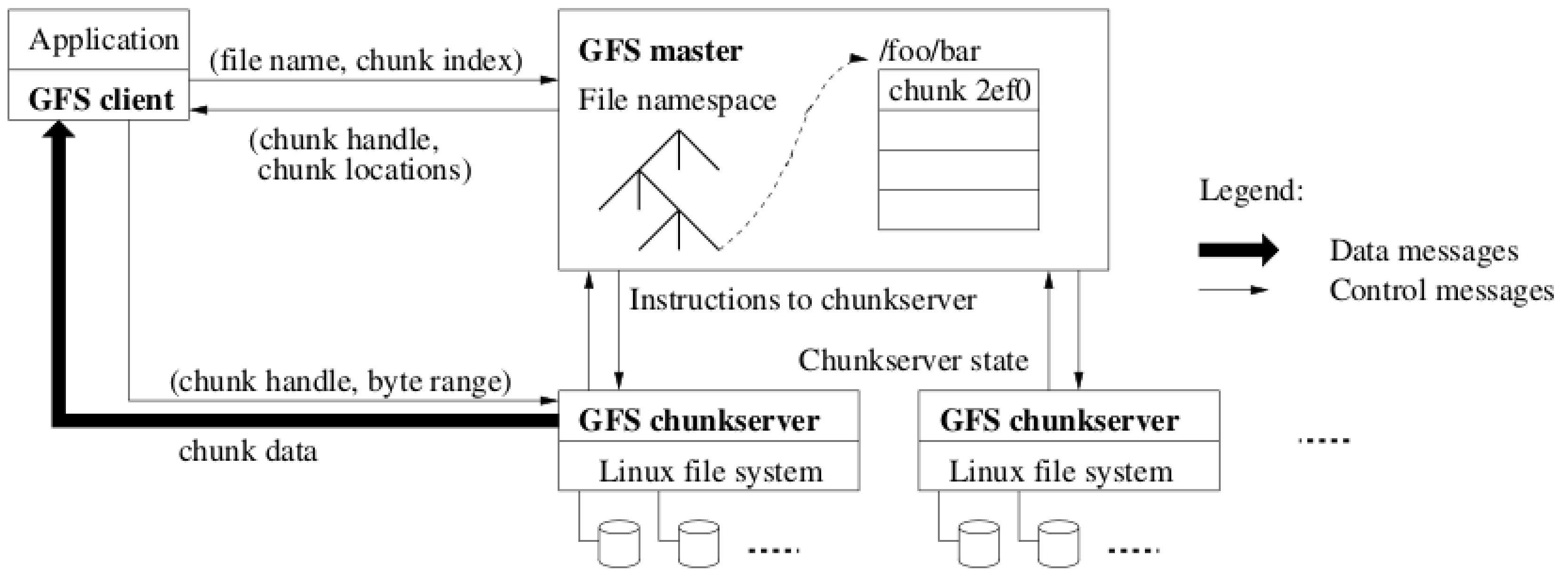


Figure 1: GFS Architecture

GFS – Architecture Cont.

- Master
  - Maintian all metadata in memory
  - Makes chunk placement and replication decision, using global knowledge
  - Operation log for persistence
    - Replicated on remote machines
    - Do checkpoints for quick recovery
  - Chunk Locations: polls chunkservers
    - Chunkservers join and leave frequently
    - A chunkserver knows what chunks it has

GFS – Architecture Cont.

- Chunkserver
  - Stores each chunk as a Linux file
  - Check data integrity
- Client:
  - Linked to apps using the file system API
  - Communicates with master for metadata
  - Communicates with chunkservers for data
  - Only caches metadata information

GFS – Architecture Cont.

- Chunksize: a key design parameter(64 MB)

  Larger chunksize => fewer chunks

  - Reduce client-master interaction
  - Reduce network connections
  - Reduce metadata size

GFS – Chunk Replication

- Replication Protocal
- Data Flow: closest machine and pipelining

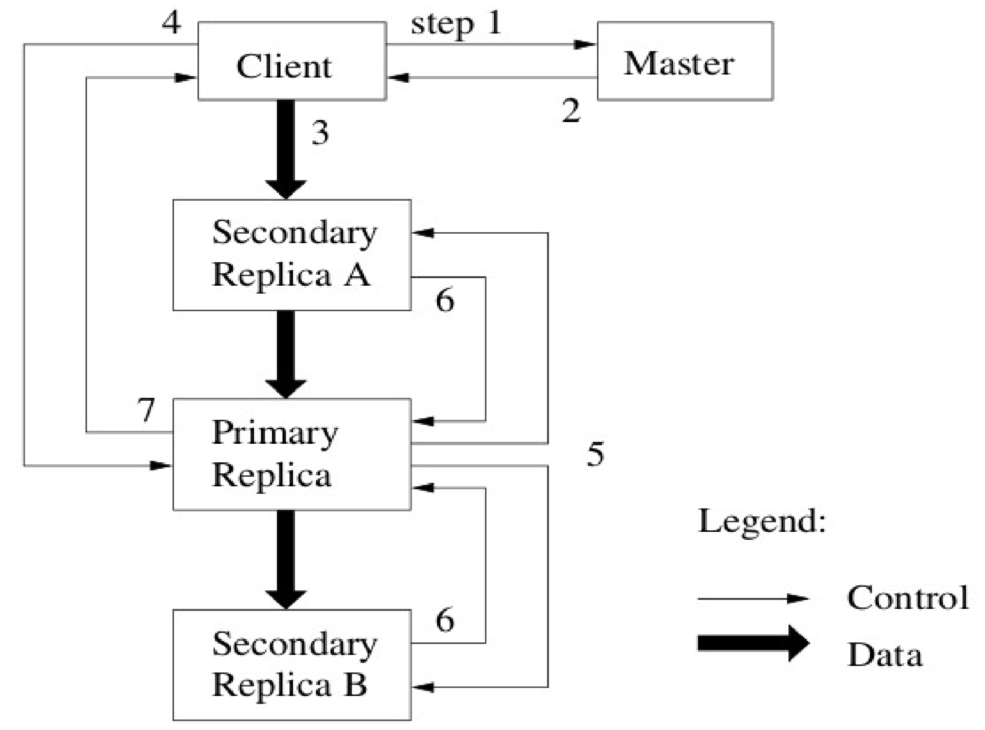
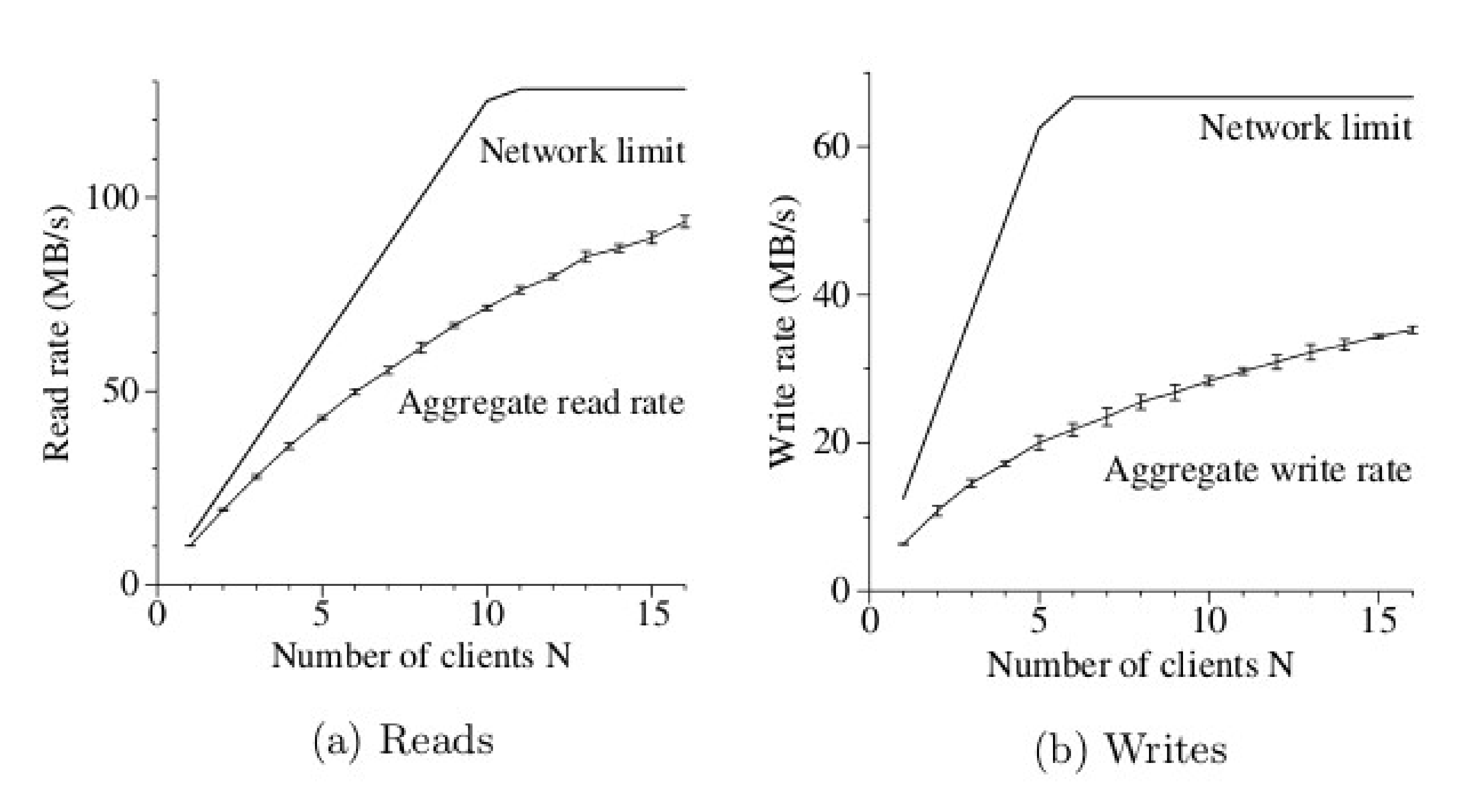


Figure 2: Write Control and Data Flow

GFS – Other Cool Designs

- Snapshot: new chunks are created on the same chunkservers as the original chunks
- Prefix compression for compressing full pathnames
- Replica placement:
  - Chunkservers with below-average disk utilization
  - Limit “recent” creations numbers
  - Spread across racks

GFS – Evaluations



GFS – Evaluations Cont.

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

GFS – Evaluation Cont.

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