The Hadoop Distributed File System

Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler
Yahoo!
Sunnyvale, California USA
{Shv, Hairong, SRadia, Chansler}@Yahoo-Inc.com

***Abstract*—The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. We describe the architecture of HDFS and report on experience using HDFS to manage 25 petabytes of enterprise data at Yahoo!.**

Keywords: Hadoop, HDFS, distributed file system

developed at Facebook. Pig [4], ZooKeeper [6], and Chukwa were originated and developed at Yahoo! Avro was originated at Yahoo! and is being co-developed with Cloudera.

HDFS is the file system component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favor of improved performance for the applications at hand.

HDFS stores file system metadata and application data separately. As in other distributed file systems, like PVFS [2][14], Lustre [7] and GFS [5][8], HDFS stores metadata on a

HDFS: abstract

- Store large datasets reliably
- Transfer them to applications at a high bandwidth
- In a cluster, servers both store data and compute tasks
- Resources can grow with demand

1. Introduction

Hadoop is open source

Yahoo! has Hadoop clusters with 25,000 servers and 25PB of data

Yahoo! developed 80% of Hadoop

HDFS	Distributed file system Subject of this paper!
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

Table 1. Hadoop project components

Main architectural features

- Separates data (DataNodes) from metadata (NameNode)
- Data is replicated on multiple DataNodes
  - Increases reliability
  - Increases data transfer bandwidth

2. Architecture

- DataNodes
  - Store files, split in blocks of 128MB (selectable)
  - Blocks independently replicated to multiple DataNodes (default: 3)
- NameNode
  - Records permissions, access time, etc
  - Maintains the directory tree and mapping of blocks to DataNodes
  - Metadata is called an *image*

Reading data

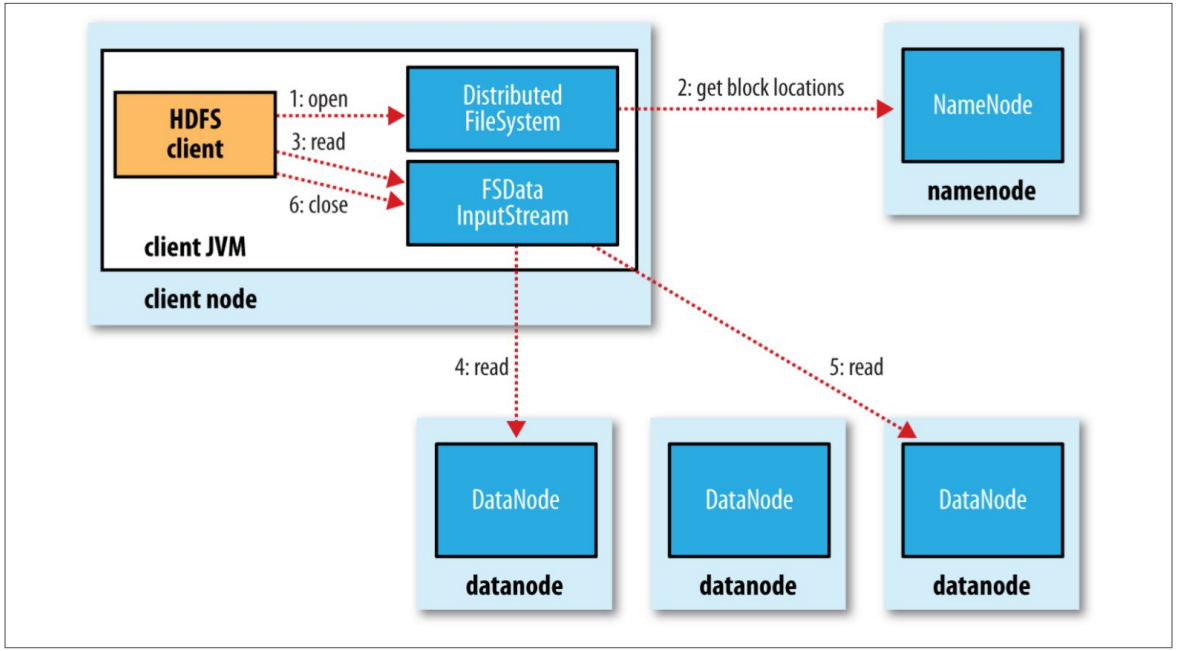

Figure 3-2. A client reading data from HDFS

Hadoop: The Definitive Guide, *Tom White*, 4th edition.

Writing data

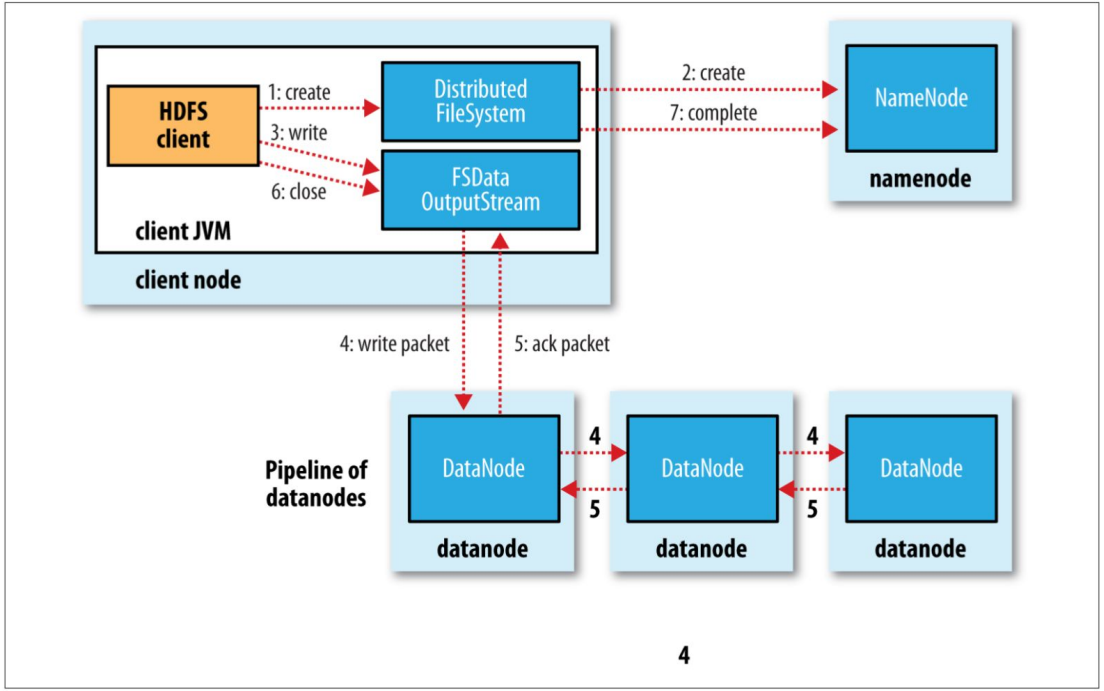

Figure 3-4. A client writing data to HDFS

Hadoop: The Definitive Guide, *Tom White*, 4th edition.

3. File I/Os and replica management

- Single-writer, multiple-reader model
  - Once written to a file, bytes cannot be changed (why?)
  - New data can be appended
  - A writer is granted a *lease* during which no other client can write to the file
- When a client reads a file, it fetches a list of blocks
  - Blocks are ordered by their distance to the reader
  - Client attempts to read the closest one, and try the next replicas if it fails

Network distance

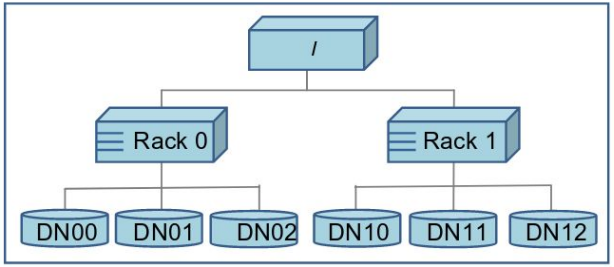

Figure 3. Cluster topology example

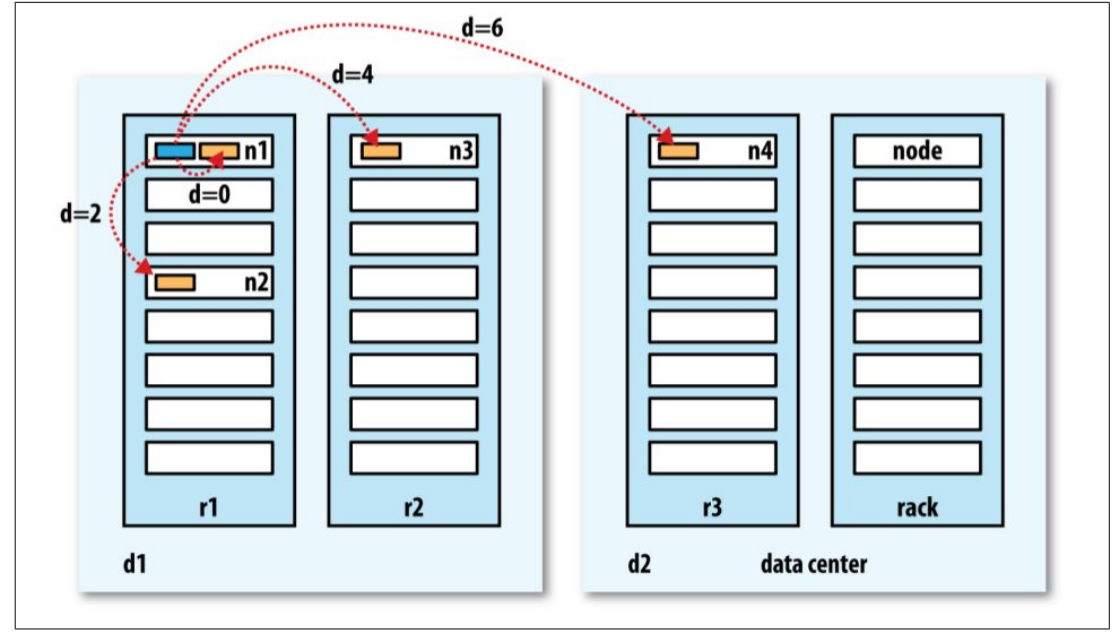

Figure 3-3. Network distance in Hadoop

Hadoop: The Definitive Guide, *Tom White*, 4th edition.

Block placement

- New blocks need to be placed carefully (why?)
- Default policy (customizable)
  - 1st replica: node where writer is located
  - 2nd and 3rd replica: on two different nodes in a different rack
  - Other replicas (if any): randomly
    - No more than 1 replica on a node
    - No more than 2 replicas in the same rack, when possible

What happens if a NameNode crashes?

All the files are lost!

- Fault tolerance mechanisms
  - Checkpointing and journaling
  - Secondary NameNode

Checkpointing and journaling

- A checkpoint is a persistent record of the image to disk
- The journal records the transactions done since the last checkpoint
- During startup, the NameNode:
  - Initializes the NameSpace from the last Checkpoint
  - Replays the transactions in the Journal
  - Writes a new Checkpoint
  - Empties the Journal

Main other mechanisms

- NameNode handles under- and over-replication
- Balancer (user program) balances disk space usage
- Block scanner (in DataNode) verifies block checksums

Conclusion: main properties of HDFS

- Adding new nodes increases the total storage capacity
- Files that are larger than any disk in the system can be stored
- Compute tasks can be located where the data is stored
- Fault tolerance through replication