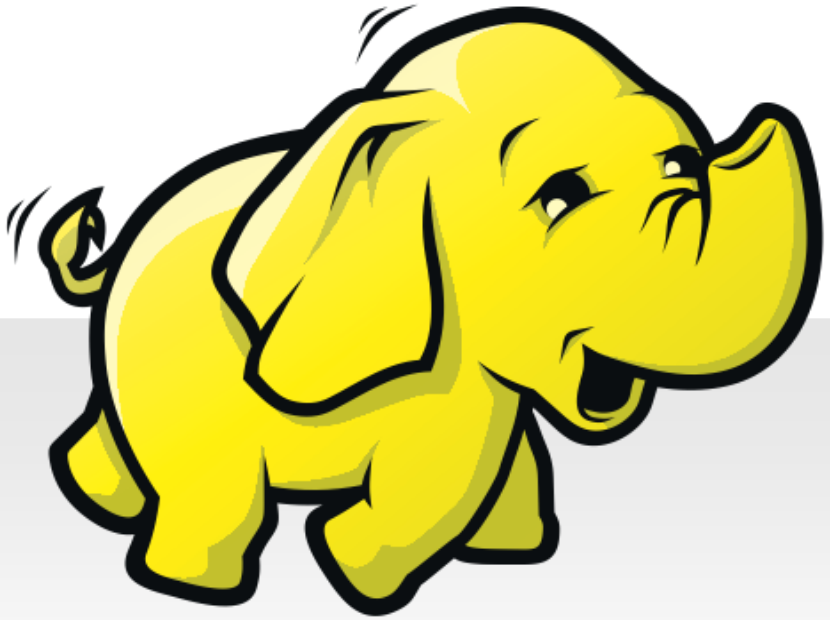
The Hadoop Distributed File System

Konstantin Shvachko
Hairong Kuang
Sanjay Radia
Robert Chansler



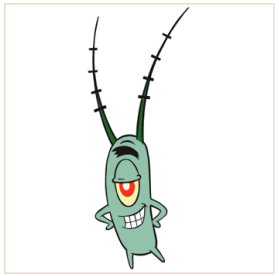
MSST 10
6 May 2010

Hadoop in Perspective

Hadoop scales computation capacity, storage capacity, and I/O bandwidth by adding commodity servers.

Hadoop is a member of the Apache Foundation community.

Hadoop processes data for over 100 organizations world-wide.



HDFS	**Distributed file system**
MapReduce	Distributed computation
HBase	Column store
Pig	Dataflow language
Hive	Data warehouse
Zookeeper	Distributed coordination
Chukwa	Data Collection
Avro	Data Serialization

The NameNode Application Manages File System Metadata

The name space is a hierarchy of files and directories.

Names map to *inodes* that record permissions, modification times, and directory quotas for names and space.

Files are a sequence of blocks (typically 128 MB).

Blocks are replicated at DataNodes (typically 3 times).

The NameNode keeps the entire name space in RAM.

The durability of the name space is maintained by a write-ahead *journal* and *checkpoints*.

A *Secondary NameNode* or a *BackupNode* creates periodic checkpoints.

The DataNode Application Serves Block Replicas

A block replica is two files, one for data bytes and one for metadata.

The metadata for a replica includes the length of the data, checksums, and generation stamp.

DataNodes register with the NameNode, and provide periodic *block reports* that list the block replicas on hand.

DataNodes send *heartbeats* to the NameNode.

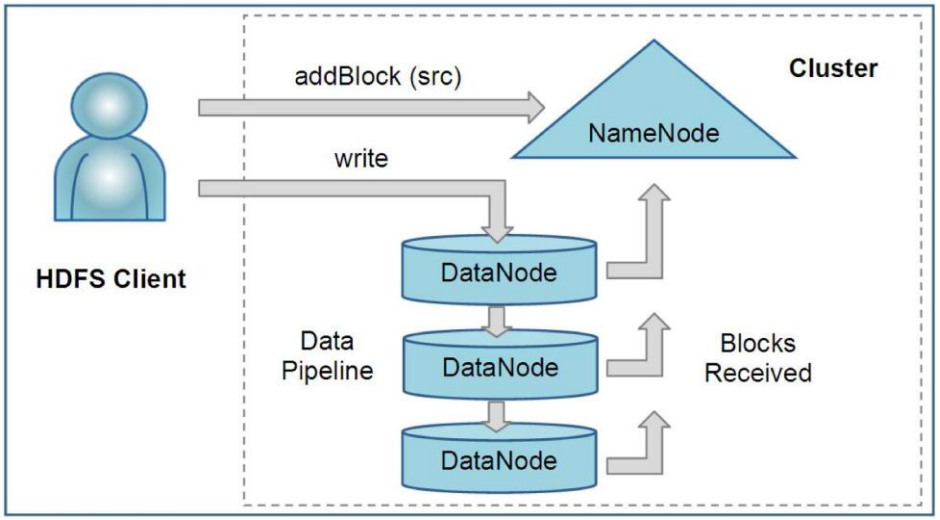
Heartbeat responses give instructions for managing replicas.

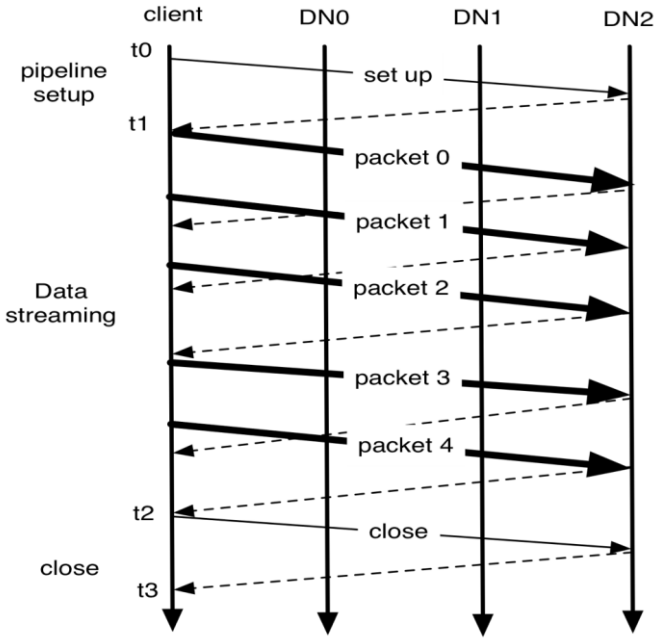
If no heartbeat is received during a 10-minute interval, the node is presumed to be lost, and the replicas hosted by that node to be unavailable.

The Client Library Connects User to NameNode and DataNodes

To write a block of a file, the client requests a list of candidate DataNodes from the NameNode, and organizes a write pipeline.

To read a block, the client requests the list of replica locations from the NameNode.





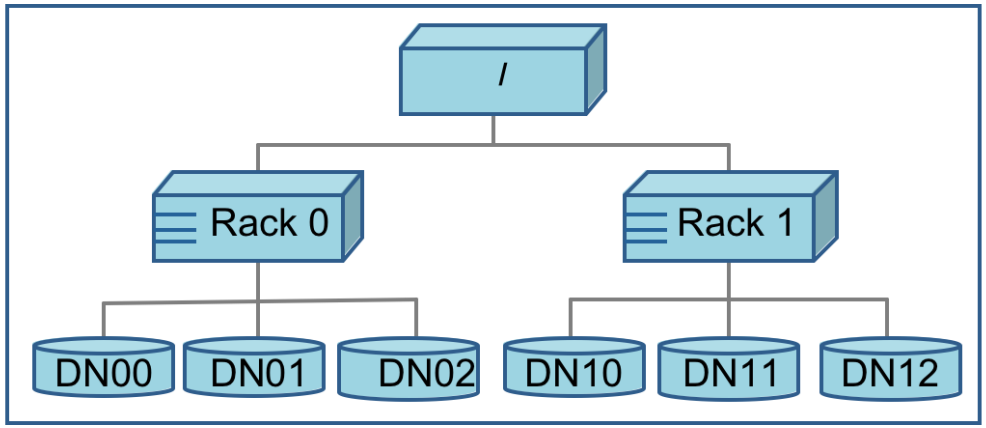
Replica Management

Replica locations are topology-aware.

The first replica is at the client's node; the next two replicas are at random nodes of another rack.

The list of replica locations returned to the client when reading a block is sorted by proximity to the client.

Replacement replicas obey the topology rules: no node has two replicas of a block, no rack has more than two replicas of a block.



Quiz: What Is the Common Attribute?

Active Management for Data Durability

End-to-end checksums

Continuous replica maintenance

Periodic checksum verification

Balancing storage utilization

Decommissioning nodes for service

Surveillance and reporting

Practice at Yahoo!

25 PB of application data across 25,000 servers

Largest cluster is 4,000 servers

- 2 quad core Xeon processors @ 2.5ghz
- Red Hat Enterprise Linux Server Release 5.1
- Sun Java JDK 1.6.0_13-b03
- 4 directly attached SATA drives (two TB each)
- 16G RAM
- 1-gigabit Ethernet
- 40 servers in a rack connected by an IP switch
- Rack switches connected by 8 core switches

Benchmarks

DFSIO

- Read: 66 MB/s
- Write: 40 MB/s

Observed on busy cluster

- Read: 1.02 MB/s
- Write: 1.09 MB/s

Sort ("Very carefully tuned user application")

Bytes (TB)	Nodes	Maps	Reduces	Time	HDFS I/O Bytes/s	
					Aggregate (GB/s)	Per Node (MB/s)
1	1460	8000	2700	62 s	32	22.1
1000	3558	80,000	20,000	58,500 s	34.2	9.35

Thoughts from Tuesday/Wednesday

You can apologize for being slow, but there is no forgiveness for losing data.

Know who your neighbors are!

HDFS systems grew by two order of magnitude in 4 years.
Exa-scale in 2014?

HDFS Team at Yahoo!

- Konstantin Shvachko
- Mahadev Konar
- Hairong Kuang
- Sanjay Radia
- Nicholas Sze
- Suresh Srinivas
- Boris Shkolnik
- Jakob Homan
- Kan Zhang
- Erik Steffl
- Rob Chansler

Gary Murry

Jagane Sundar

Ravi Phulari

Raj Merchia

And many friends in

Grid Development

Grid Solutions

Grid Operations

Y! Research

HDFS and GPFS

- HDFS
  - Data-store specially designed for Hadoop (MapReduce workloads)
    - Optimized for sequential workloads with large files
  - Highly scalable
  - Reliability features focused on commodity data centers
    - Supports rack-aware replication (arbitrary # replicas, default 3)
  - 3 yr old project (1st release April 2006)
  - Cannot be used as a traditional filesystem
  - Does not provide normal filesystem interface (POSIX interface, multiple writers, write to existing file)

- GPFS
  - Mature filesystem - many production installations
  - High performance, Highly scalable
  - Reliability features focused on SAN environments
    - Supports rack-aware 2-way replication
  - POSIX interface
  - Supports shared disk (SAN) and shared-nothing setups
  - Not optimized for MapReduce workloads
  - Does not expose data location information
  - largest block size = 16 MB

  - IBM, presented at HUG Sunnyvale 2009

HDFS and PVFS

- HDFS
  - Compute and storage on one node (beneficial to Hadoop/MapReduce model where computation is moved closer to the data)
  - Only one writer
  - Not optimized for small files; but client- side buffering will only send a write to a server once
  - Open-for-append, but not re-write
  - Client buffers one *chunk* (64KB)
  - Exposes block locations
  - Three replicas
  - Client API designed for the requirements of data-intensive applications
- PVFS
  - Separate compute and storage nodes (easy manageability and incremental growth); also used by Amazon S3
  - Guarantees POSIX sequential consistency for non-conflicting writes
  - Lack of client-side caching incurs high I/O overhead for small file ops
  - Append not supported
  - No client-side buffering
  - PVFS maintains stripe layout information as extended attributes (pNFS has support for layout delegations)
  - Redundancy through RAID on/across storage devices
  - Supports UNIX I/O interface and (most) POSIX semantics

From Tantisiriroj, et al.