Hadoop Distributed File System

Dhruba Borthakur
Apache Hadoop Project Management Committee
dhruba@apache.org
June 3rd, 2008

Who Am I?

- **Hadoop Developer**
  - Core contributor since Hadoop's infancy
  - Focussed on Hadoop Distributed File System
- **Facebook** (Hadoop)
- **Yahoo!** (Hadoop)
- **Veritas** (San Point Direct, VxFS)
- **IBM Transarc** (Andrew File System)

Hadoop, Why?

- **Need to process huge datasets on large clusters of computers**
- **Very expensive to build reliability into each application.**
- **Nodes fail every day**
  - Failure is expected, rather than exceptional.
  - The number of nodes in a cluster is not constant.
- **Need common infrastructure**
  - Efficient, reliable, easy to use
  - Open Source, Apache License

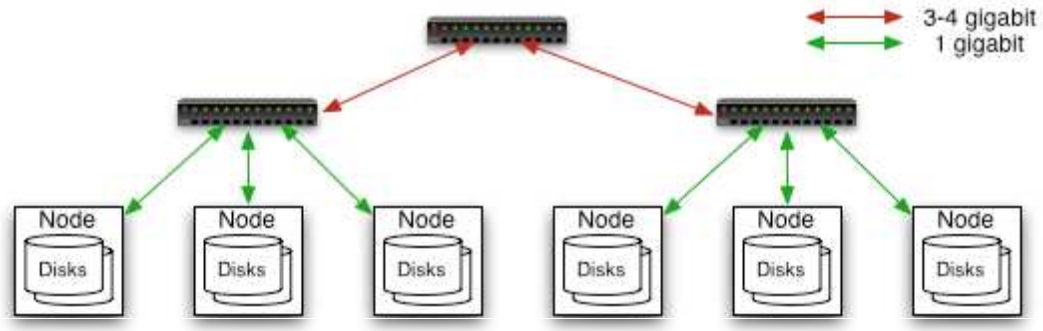
Hadoop History

- **Dec 2004 —** Google GFS paper published
- **July 2005 —** Nutch uses MapReduce
- **Jan 2006 —** Doug Cutting joins Yahoo!
- **Feb 2006 —** Becomes Lucene subproject
- **Apr 2007 —** Yahoo! on 1000-node cluster
- **Jan 2008 —** An Apache Top Level Project
- **Feb 2008 —** Yahoo! production search index

Who uses Hadoop?

- Amazon/A9
- Facebook
- Google
- IBM : Blue Cloud?
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

Commodity Hardware



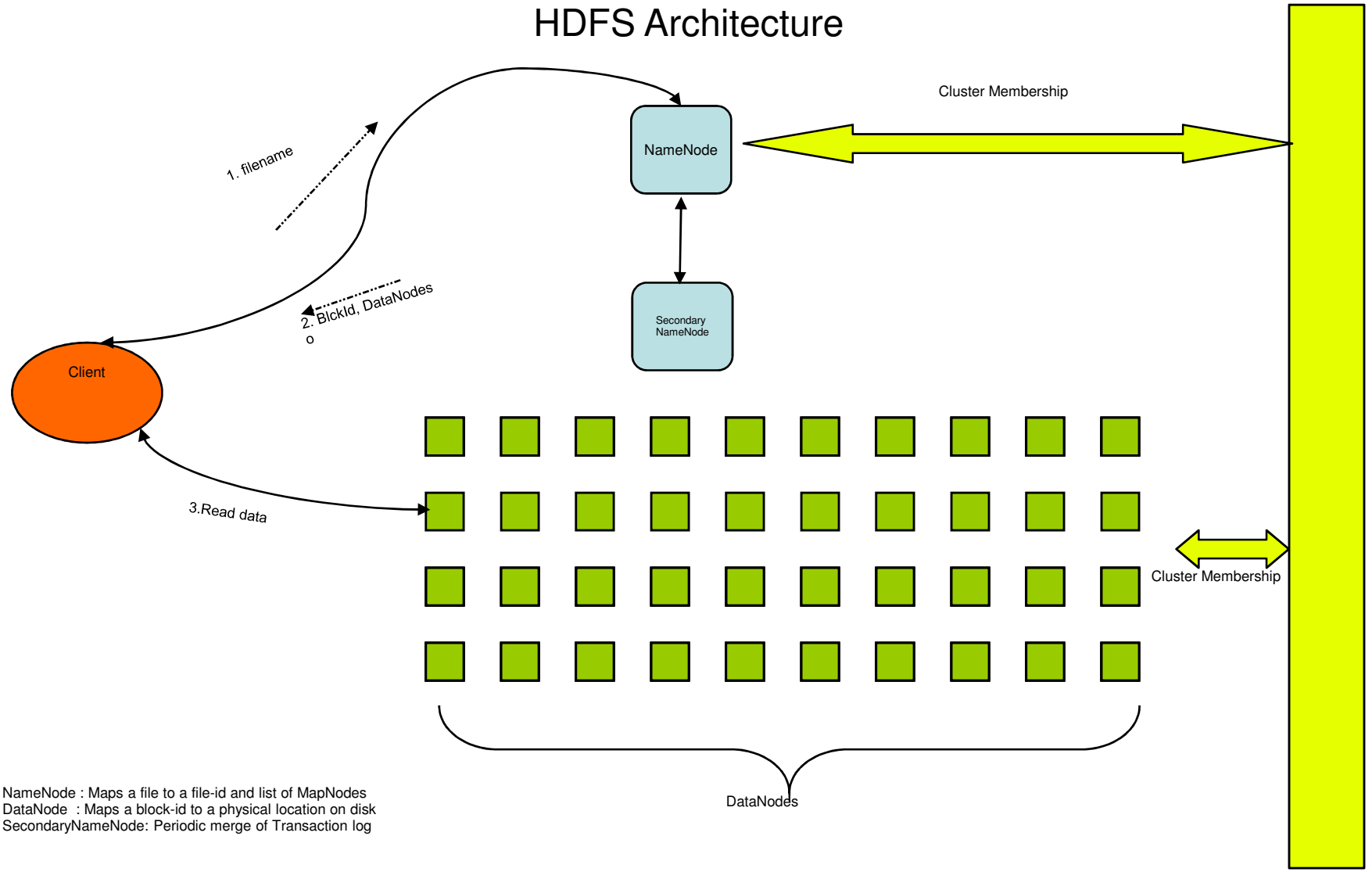
Typically in 2 level architecture

– Nodes are commodity PCs
– 30-40 nodes/rack
– Uplink from rack is 3-4 gigabit
– Rack-internal is 1 gigabit

Goals of HDFS

- **Very Large Distributed File System**
  – 10K nodes, 100 million files, 10 PB
- **Assumes Commodity Hardware**
  – Files are replicated to handle hardware failure
  – Detect failures and recovers from them
- **Optimized for Batch Processing**
  – Data locations exposed so that computations can move to where data resides
  – Provides very high aggregate bandwidth

HDFS Architecture

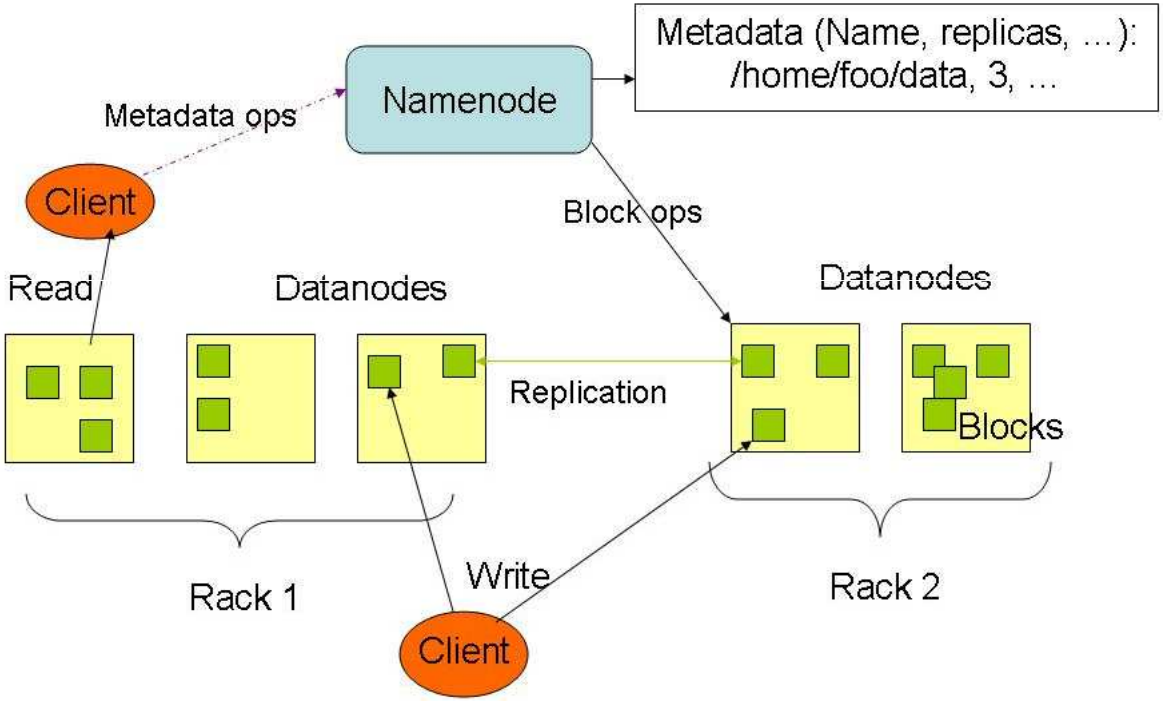


NameNode : Maps a file to a file-id and list of MapNodes
DataNode : Maps a block-id to a physical location on disk
SecondaryNameNode: Periodic merge of Transaction log

Distributed File System

- **Single Namespace for entire cluster**
- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- **Files are broken up into blocks**
  - Typically 128 MB block size
  - Each block replicated on multiple DataNodes
- **Intelligent Client**
  - Client can find location of blocks
  - Client accesses data directly from DataNode

HDFS Architecture



Functions of a NameNode

- **Manages File System Namespace**
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides
- **Cluster Configuration Management**
- **Replication Engine for Blocks**

NameNode Metadata

- **Meta-data in Memory**
  - The entire metadata is in main memory
  - No demand paging of meta-data
- **Types of Metadata**
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g creation time, replication factor
- **A Transaction Log**
  - Records file creations, file deletions. etc

DataNode

- **A Block Server**
  – Stores data in the local file system (e.g. ext3)
  – Stores meta-data of a block (e.g. CRC)
  – Serves data and meta-data to Clients
- **Block Report**
  – Periodically sends a report of all existing blocks to the NameNode
- **Facilitates Pipelining of Data**
  – Forwards data to other specified DataNodes

Block Placement

- **Current Strategy**

  -- One replica on local node

  -- Second replica on a remote rack

  -- Third replica on same remote rack

  -- Additional replicas are randomly placed
- **Clients read from nearest replica**
- **Would like to make this policy pluggable**

Heartbeats

- **DataNodes send heartbeat to the NameNode**
  - Once every 3 seconds
- **NameNode used heartbeats to detect DataNode failure**

Replication Engine

- **NameNode detects DataNode failures**
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

Data Correctness

- **Use Checksums to validate data**
  - Use CRC32
- **File Creation**
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- **File access**
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas

NameNode Failure

- **A single point of failure**
- **Transaction Log stored in multiple directories**
  - A directory on the local file system
  - A directory on a remote file system (NFS/CIFS)
- **Need to develop a real HA solution**

Data Pipelining

- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the Client moves on to write the next block in file

Rebalancer

- **Goal: % disk full on DataNodes should be similar**
  - Usually run when new DataNodes are added
  - Cluster is online when Rebalancer is active
  - Rebalancer is throttled to avoid network congestion
  - Command line tool

Secondary NameNode

- Copies FsImage and Transaction Log from NameNode to a temporary directory
- Merges FSImage and Transaction Log into a new FSImage in temporary directory
- Uploads new FSImage to the NameNode
  - Transaction Log on NameNode is purged

User Interface

- Command for HDFS User:
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
- Command for HDFS Administrator
  - hadoop dfsadmin -report
  - hadoop dfsadmin -decommission datanodename
- Web Interface
  - http://host:port/dfshealth.jsp

Hadoop Map/Reduce

- **The Map-Reduce programming model**
  – Framework for distributed processing of large data sets
  – Pluggable user code runs in generic framework
- **Common design pattern in data processing**
  cat * | grep | sort | unique -c | cat > file
  input | **map** | shuffle | **reduce** | output
- **Natural for:**
  – Log processing
  – Web search indexing
  – Ad-hoc queries

Hadoop Subprojects

- **Pig** (Initiated by Yahoo!)
  - High-level language for data analysis
- **HBase** (initiated by Powerset)
  - Table storage for semi-structured data
- **Zookeeper** (Initiated by Yahoo!)
  - Coordinating distributed applications
- **Hive** (initiated by Facebook, coming soon)
  - SQL-like Query language and Metastore
- **Mahout**
  - Machine learning

Useful Links

- **HDFS Design:**
  - http://hadoop.apache.org/core/docs/current/hdfs_design.html
- **Hadoop API:**
  - http://hadoop.apache.org/core/docs/current/api/