Hadoop Distributed File System

Dhruba Borthakur
June, 2007

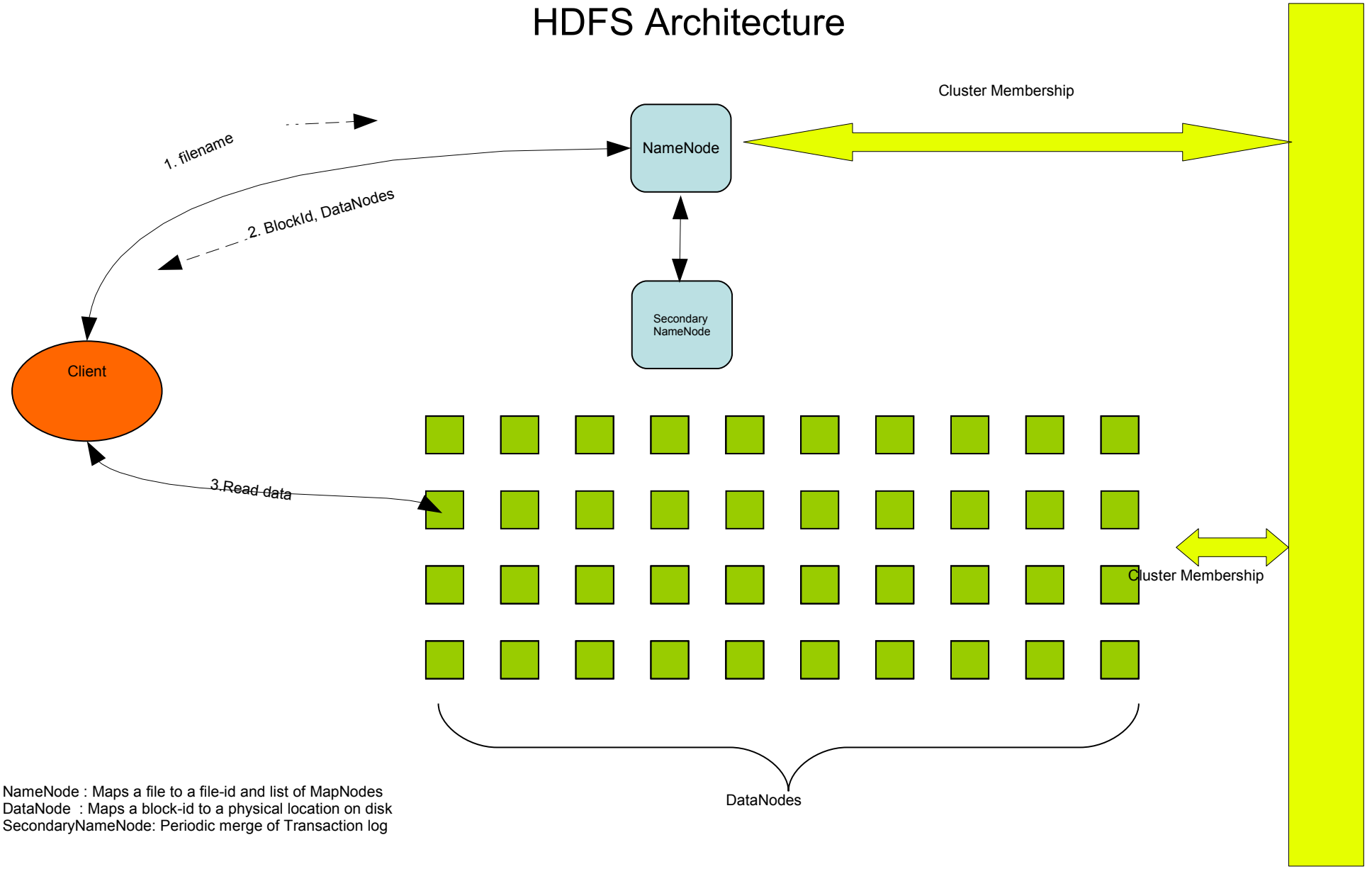
Goals of HDFS

- 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

Distributed File System

- Single Namespace for entire cluster
- Data Coherency
  - Write-once-read-many access model
  - Client does not see a file until the creator has closed it
- 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



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

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 Meta-data

- Meta-data in Memory
  - The entire metadata is in main memory
  - No demand paging of FS 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 fac-tor
- 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

- One replica on local node
- Another replica on a remote rack
- Third replica on local rack
- Additional replicas are randomly placed

HeartBeats

- DataNodes send heartbeat to the NameNode
- 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)

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 the next block in file

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

Useful Links

- HDFS Design:
  - http://lucene.apache.org/hadoop/hdfs_design.html
- HDFS API:
  - http://lucene.apache.org/hadoop/api/