Hadoop Distributed File System

Dhruba Borthakur
Apache Hadoop Project Management Committee
dhruba@apache.org
dhruba@facebook.com

Hadoop, Why?

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

Hadoop History

- **Dec 2004 – Google paper published**
- **July 2005 – Nutch uses new MapReduce implementation**
- **Jan 2006 – Doug Cutting joins Yahoo!**
- **Feb 2006 – Hadoop becomes a new Lucene subproject**
- **Apr 2007 – Yahoo! running Hadoop on 1000-node cluster**
- **Jan 2008 – An Apache Top Level Project**
- **Feb 2008 – Yahoo! production search index with Hadoop**
- **July 2008 – First reports of a 4000-node cluster**

Who uses Hadoop?

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

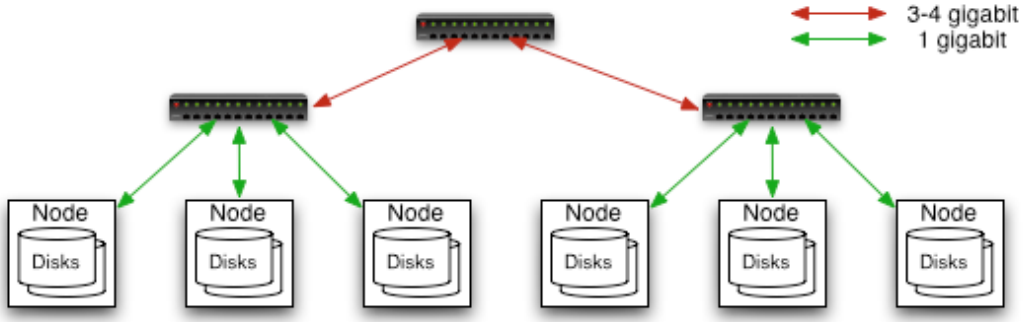
What is Hadoop used for?

- Search
  - Yahoo, Amazon, Zvents,
- Log processing
  - Facebook, Yahoo, ContextWeb. Joost, Last.fm
- Recommendation Systems
  - Facebook
- Data Warehouse
  - Facebook, AOL
- Video and Image Analysis
  - New York Times, Eyealike

Public Hadoop Clouds

- Hadoop Map-reduce on Amazon EC2 instances
  - http://wiki.apache.org/hadoop/AmazonEC2
- IBM Blue Cloud
  - Partnering with Google to offer web-scale infrastructure
- Global Cloud Computing Testbed
  - Joint effort by Yahoo, HP and Intel

Commodity Hardware



Typically in 2 level architecture

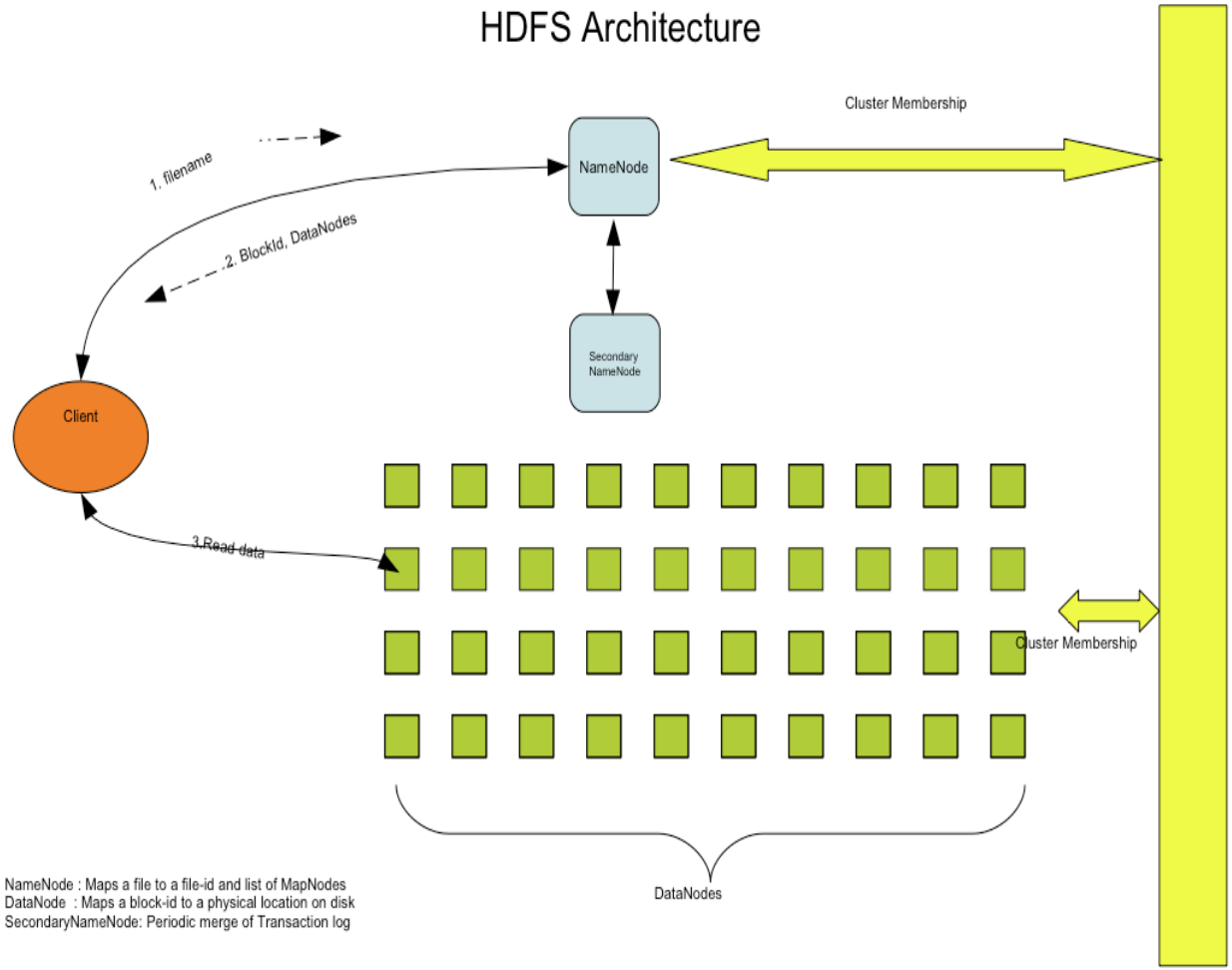
– Nodes are commodity PCs
– 30-40 nodes/rack
– Uplink from rack is 3-4 gigabit
– Rack-internal is 1 gigabit

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

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
Cluster Membership
1. filename
2. BlockId, DataNodes
NameNode
Secondary NameNode
Client
3.Read data
Cluster Membership
DataNodes
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 Metadata

- 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 access 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 random node on local rack
  -- Second replica on a random 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

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 the next block in file

Secondary NameNode

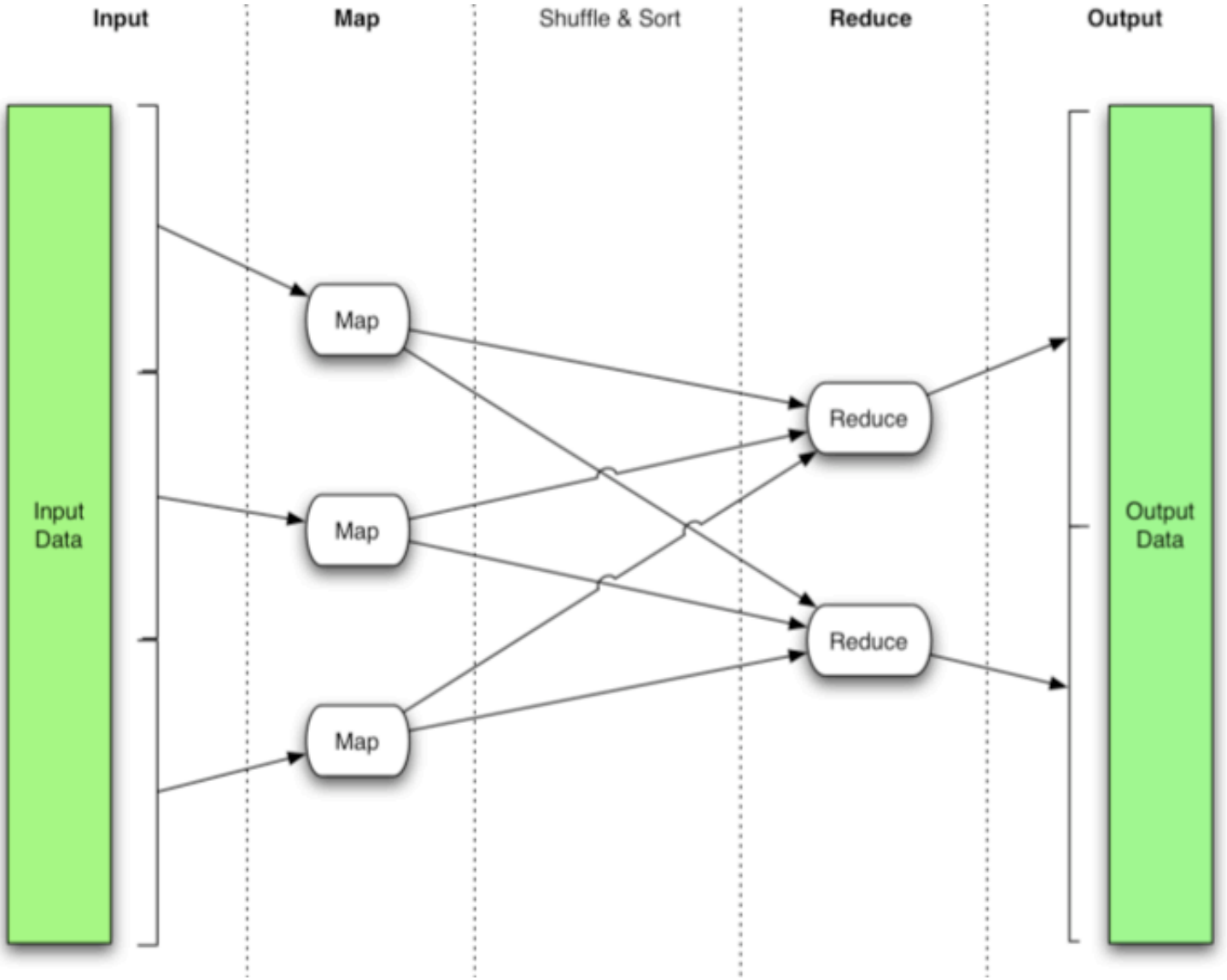
- Copies FsImage and Transaction Log from NameNode to a temporary directory
- Merges FSImage and Transaction Log into a new FSImage in temporary directory
- Uploads new FSImage to the NameNode
  - Transaction Log on NameNode is purged

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

Hadoop Map/Reduce

- **Implementation of the Map-Reduce programming model**
  – Framework for distributed processing of large data sets
  – Data handled as collections of key-value pairs
  – Pluggable user code runs in generic framework
- **Very common design pattern in data processing**
  – Demonstrated by a unix pipeline example:
  cat * | grep | sort | unique -c | cat > file
  input | **map** | shuffle | **reduce** | output
- Natural for:
  – Log processing
  – Web search indexing
  – Ad-hoc queries


Input
Map
Shuffle & Sort
Reduce
Output
Input Data
Map
Map
Map
Reduce
Reduce
Output Data

hadoop

Hadoop Subprojects

- Hive
  - A Data Warehouse with SQL support
- HBase
  - table storage for semi-structured data
- Zookeeper
  - coordinating distributed applications
- Mahout
  - Machine learning

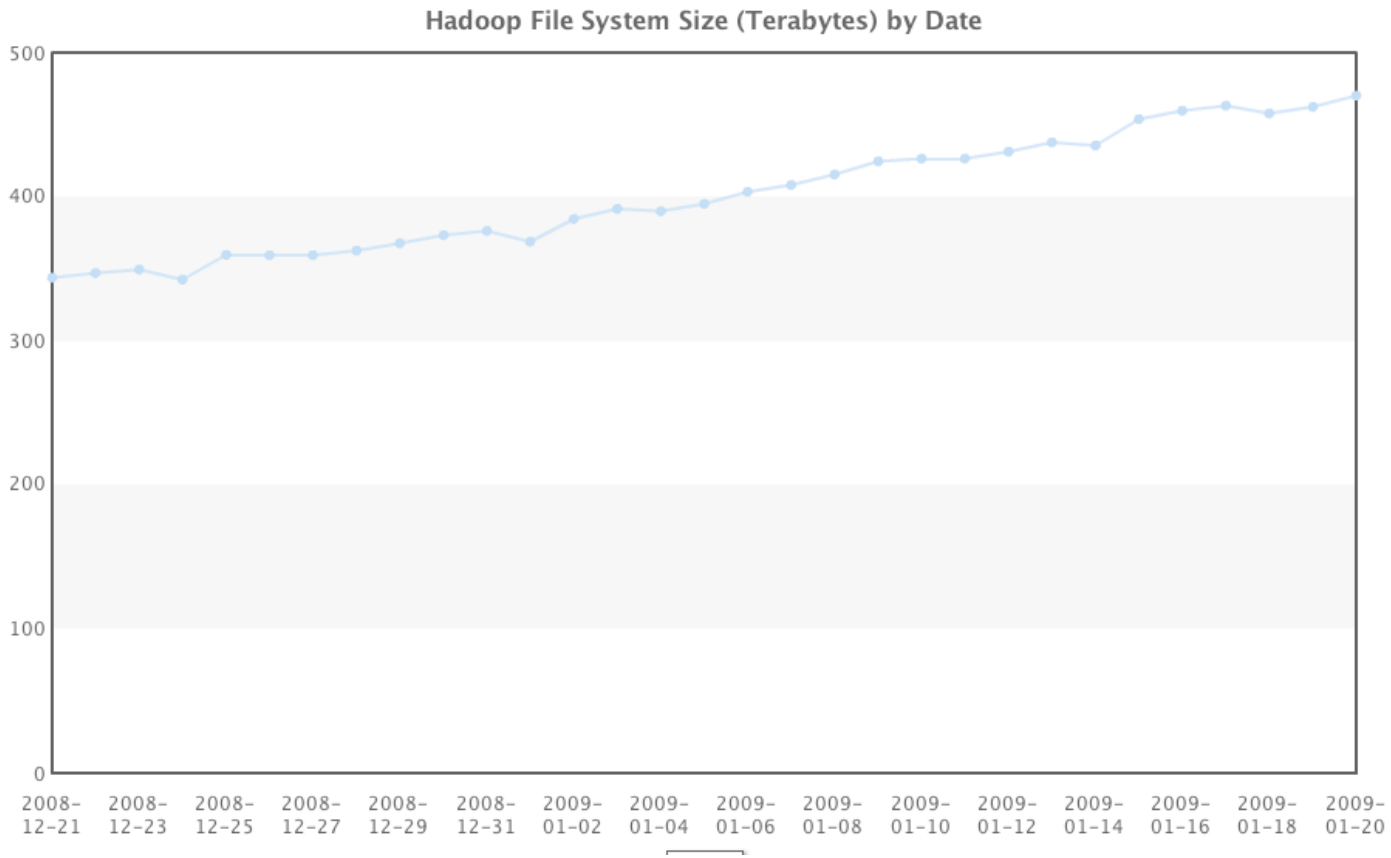
Hadoop at Facebook

- Hardware
  - 4800 cores, 600 machines, 16GB/8GB per machine – Nov 2008
  - 4 SATA disks of 1 TB each per machine
  - 2 level network hierarchy, 40 machines per rack

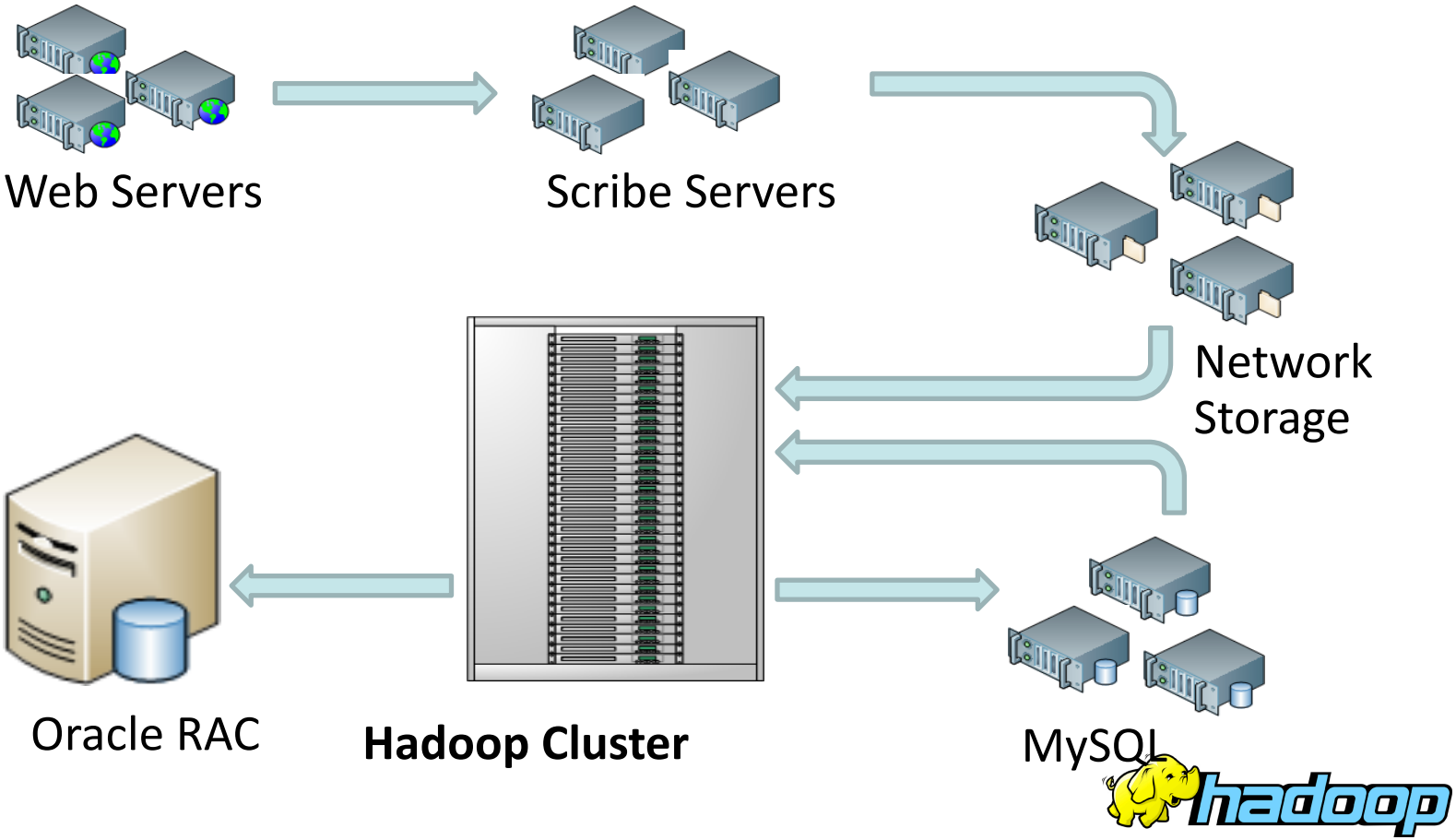
Hadoop at Facebook

- Single HDFS cluster across all cores
  - 2 PB raw capacity
  - Ingest rate is 2 TB compressed per day
    - 10 TB uncompressed
  - 12 Million files

Hadoop Growth at Facebook



Data Flow at Facebook



Hadoop Usage at Facebook

- Statistics per day:
  - 55TB of compressed data scanned per day
  - 3200+ jobs on production cluster per day
  - 80M compute minutes per day
- Barrier to entry is significantly reduced:
  - SQL like language called Hive
  - http://hadoop.apache.org/hive/

Useful Links

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