facebook

Hadoop Architecture and its Usage at Facebook

Dhruba Borthakur
Project Lead, Apache Hadoop Distributed File System
dhruba@apache.org
Presented at Microsoft Research, Seattle
October 16, 2009

facebook

Outline

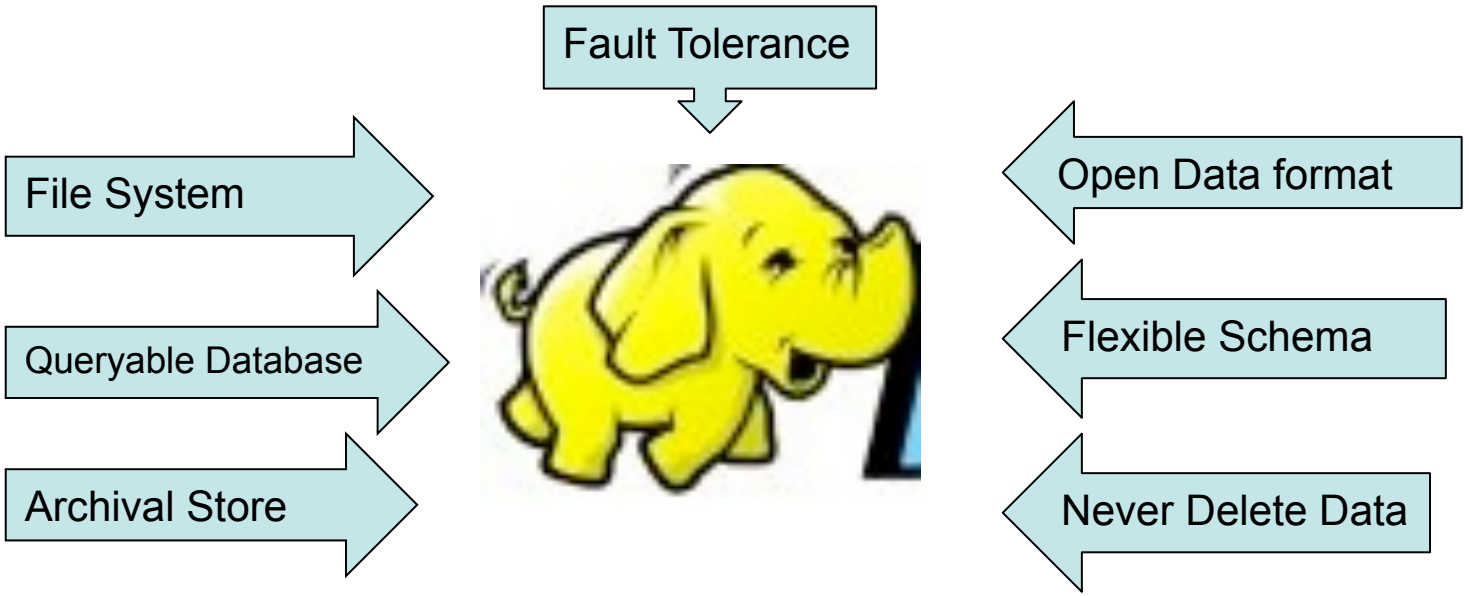
- Introduction
- Architecture of Hadoop Distributed File System
- Hadoop Usage at Facebook

Who Am I?

- **Hadoop FileSystem (HDFS) Project Lead**
  - Core contributor since Hadoop's infancy
- **Facebook** (Hadoop, Hive, Scribe)
- **Yahoo!** (Hadoop in Yahoo Search)
- **Veritas** (San Point Direct, Veritas File System)
- **IBM Transarc** (Andrew File System)
- **UW Computer Science Alumni (**Condor Project)

facebook

A Confluence of Trends



HADOOP: A Massively Scalable Queryable Store and Archive

Hadoop, Why?

- **Need to process Multi Petabyte Datasets**
- **Data may not have strict schema**
- **Expensive to build reliability in 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, Open Source Apache License

Is Hadoop a Database?

- Hadoop triggered upheaval in Database Research
  - "A giant step backward in the programming paradigm", Dewitt et el
  - "DBMS performance outshines Hadoop" – Stonebraker, Dewitt, SIGMOD 2009
- Parallel Databases
  - A few scales to low hundreds of nodes and about 5 PB
  - Primary design goal is "performance"
  - Requires homogeneous hardware
  - Anomalous behavior is not well tolerated:
    - A slow network can cause serious performance degradation
    - Most queries fail when one node fails
- Scalability and Fault Tolerance: Hadoop to the rescue!

Hadoop History

- **Dec 2004** - Google GFS paper published
- **July 2005** - Nutch uses MapReduce
- **Feb 2006** - Starts as a Lucene subproject
- **Apr 2007** - Yahoo! on 1000-node cluster
- **Jan 2008** - An Apache Top Level Project
- **Jul 2008** - A 4000 node test cluster
- **May 2009** - Hadoop sorts Petabyte in 17 hours

facebook

Who uses Hadoop?

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

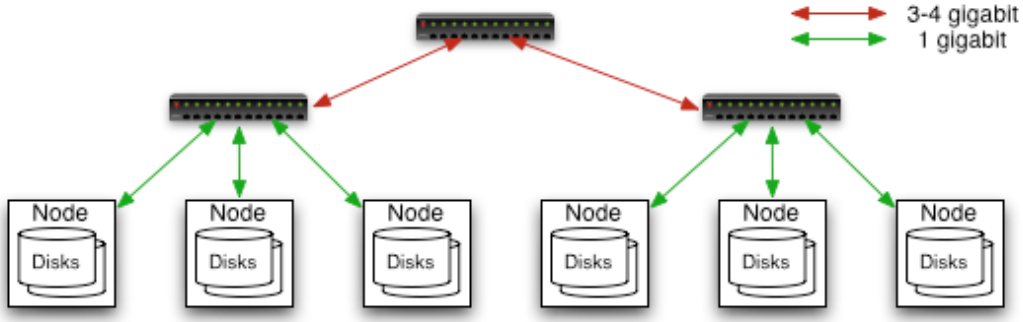
What is Hadoop used for?

- 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
  - 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
  - http://www.opencloudconsortium.org/testbed.html

Commodity Hardware



Typically in 2 level architecture

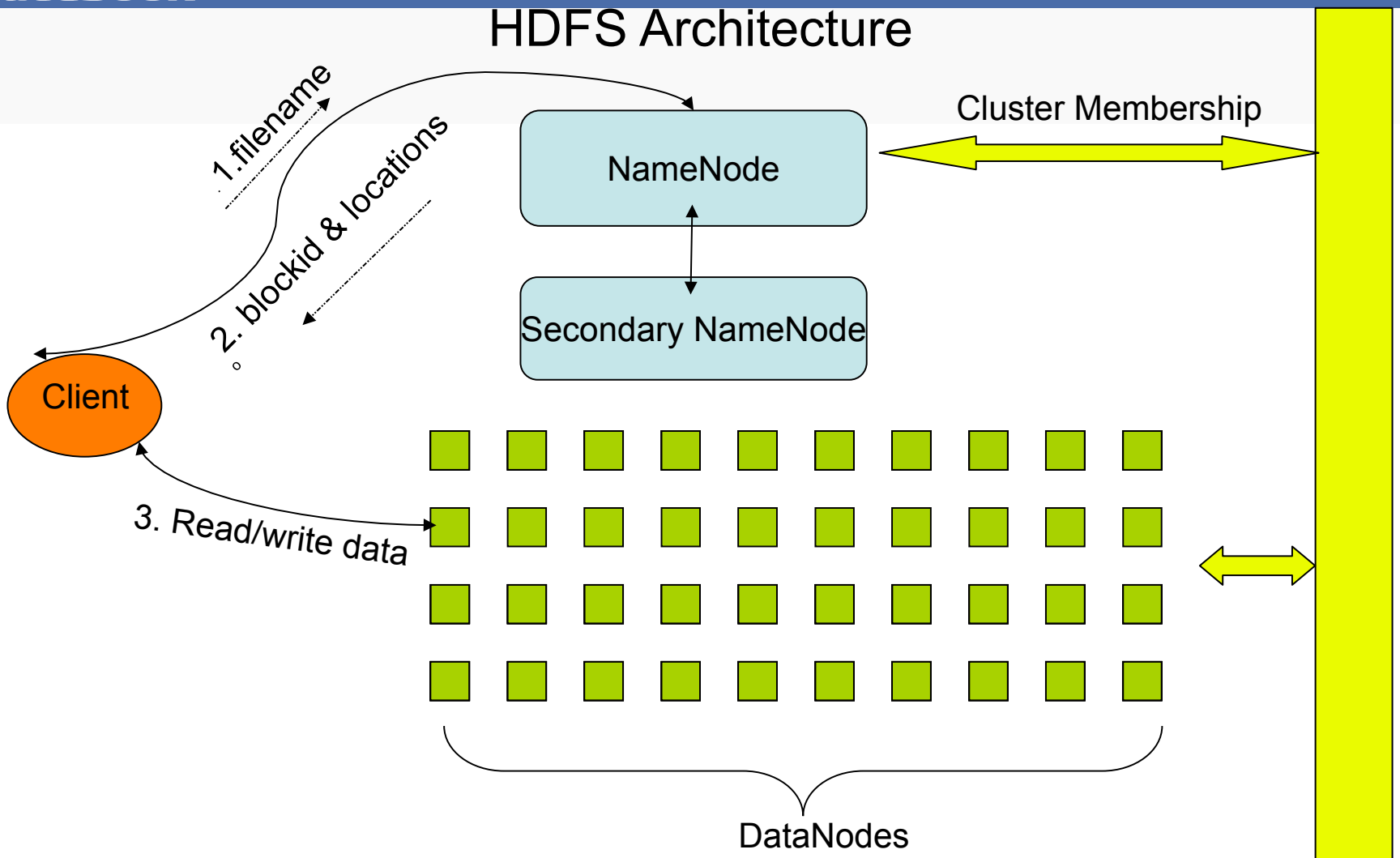
– Nodes are commodity PCs
– 30-40 nodes/rack
– Uplink from rack is 3-4 gigabit
– Rack-internal is 1 gigabit

Goals of HDFS

- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 - 100 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
- **User Space, runs on heterogeneous OS**

facebook

HDFS Architecture



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

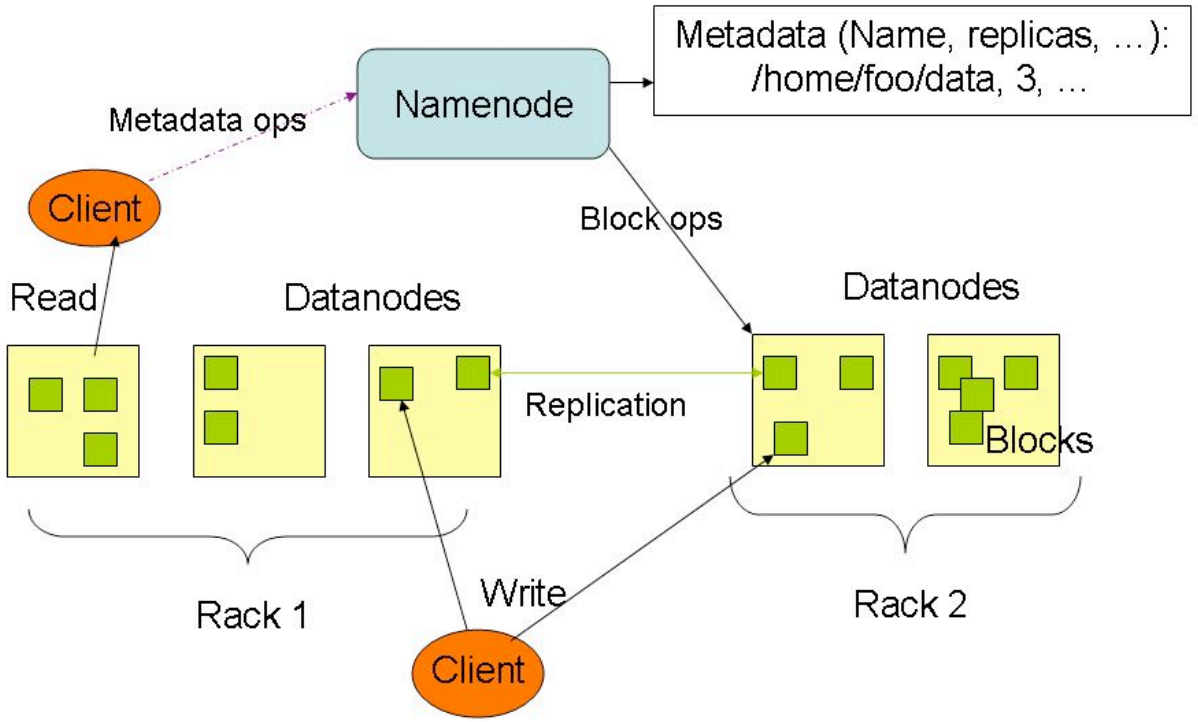
HIVE

hadoop

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



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. CRC32)
  - Serves data and meta-data to Clients
  - Periodic validation of checksums
- **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**
- **Pluggable policy for placing block replicas**
  - Co-locate datasets that are often used together
  - http://hadoopblog.blogspot.com/2009/09/hdfs-block-replica-placement-in-your.html

Data Pipelining

- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline, and so on
- When all replicas are written, the Client moves on to write the next block in file

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**
  - work in progress: BackupNode

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
- **Disadvantages**
  - Does not rebalance based on access patterns or load
  - No support for automatic handling of hotspots of data

Hadoop Map/Reduce

- **The Map-Reduce programming model**
  - 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

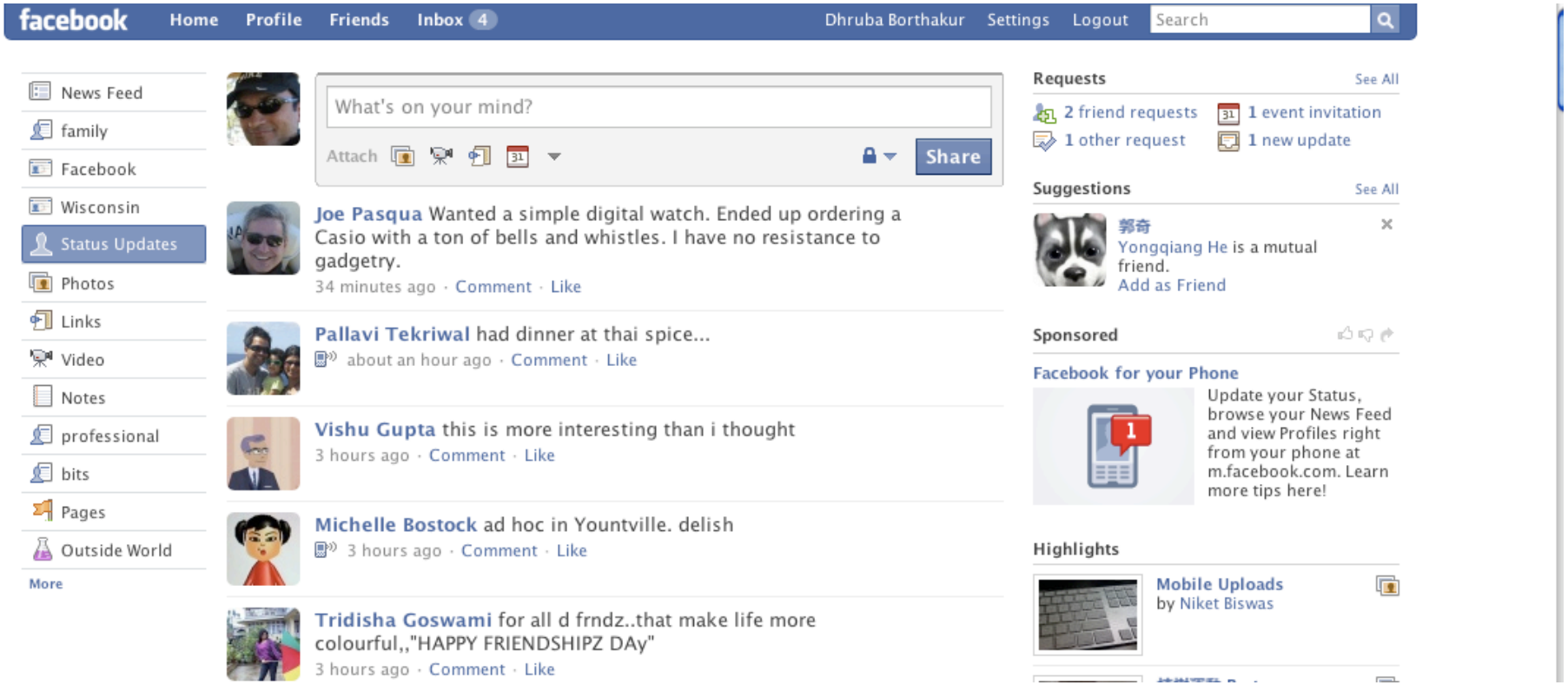
Map/Reduce and Storage

- **Clean API between Map/Reduce and HDFS**
- **Hadoop Map/Reduce and Storage Stacks**
  - Typical installations store data in HDFS
  - Hadoop Map/Reduce can run on data in MySQL
  - Demonstrated to run on IBM GPFS
- **External Schedulers and HDFS Storage**
  - Condor Job Scheduler on HDFS
  - Dryad-style DAG Scheduler on HDFS

Job Scheduling

- **Current state of affairs with Hadoop Scheduler**
  - Places computation close to data
  - FIFO and Fair Share scheduler
- **Work in progress**
  - Resource aware (cpu, memory, network)
  - Support for MPI workloads
  - Isolation of one job from another

Hadoop @ Facebook



facebook

Who generates this data?

- **Lots of data is generated on Facebook**
  - 300+ million active users
  - 30 million users update their statuses at least once each day
  - More than 1 billion photos uploaded each month
  - More than 10 million videos uploaded each month
  - More than 1 billion pieces of content (web links, news stories, blog posts, notes, photos, etc.) shared each week

Data Usage

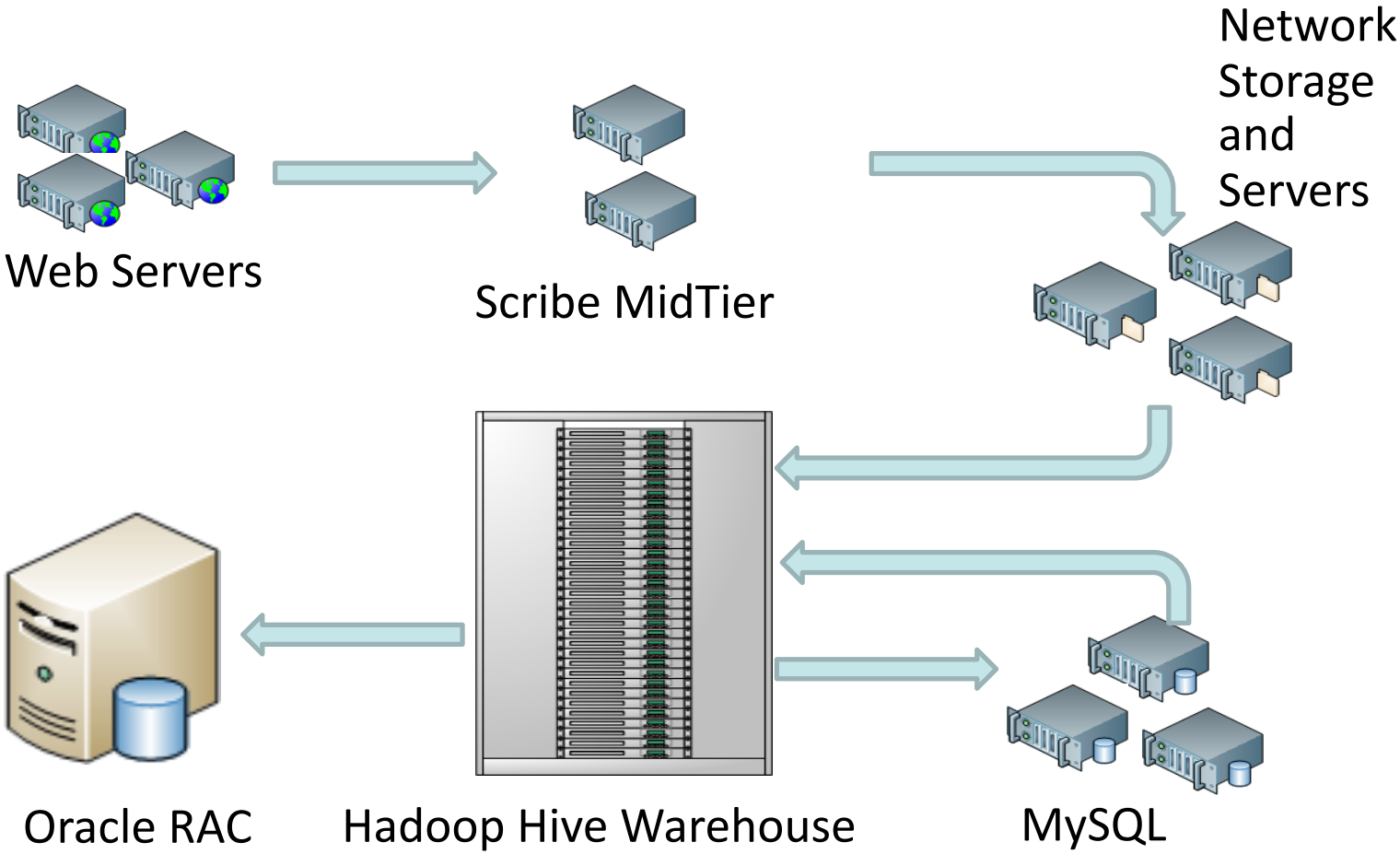
- **Statistics per day:**
  - 4 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs on production cluster per day
  - 80K compute hours per day
- **Barrier to entry is significantly reduced:**
  - New engineers go though a Hive training session
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive

Where is this data stored?

- **Hadoop/Hive Warehouse**
  - 4800 cores, 5.5 PetaBytes
  - 12 TB per node
  - Two level network topology
    - 1 Gbit/sec from node to rack switch
    - 4 Gbit/sec to top level rack switch

facebook

Data Flow into Hadoop Cloud

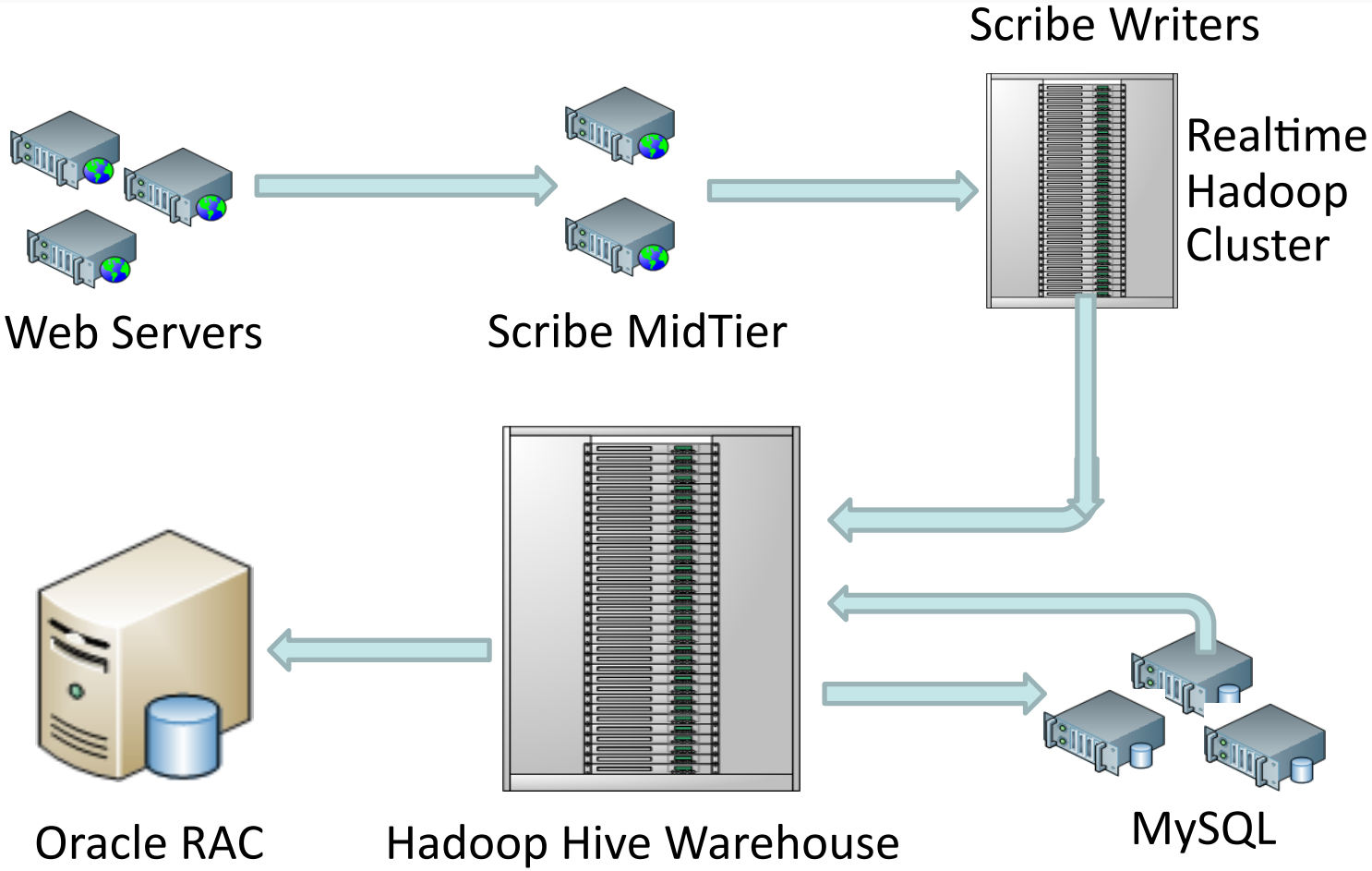


HIVE

hadoop

facebook

Hadoop Scribe: Avoid Costly Filers

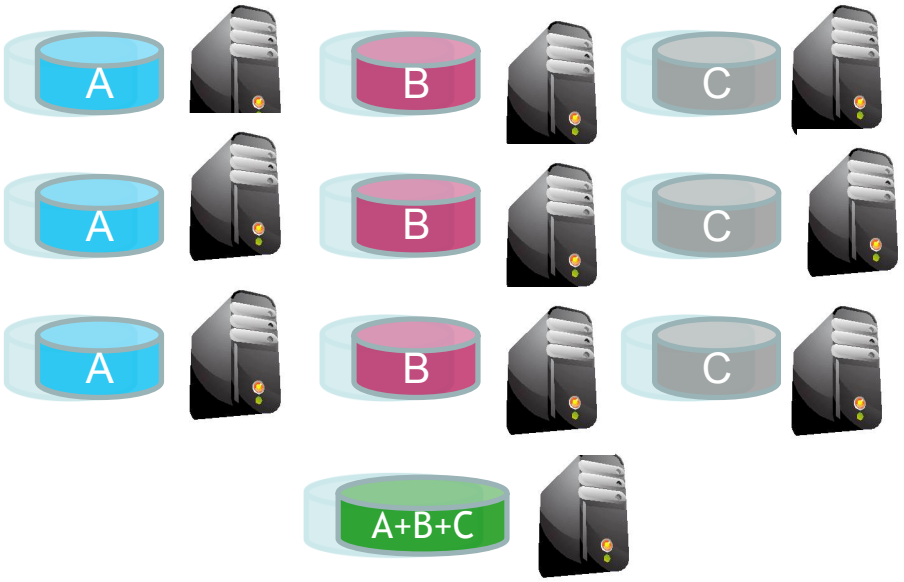


http://hadoopblog.blogspot.com/2009/06/hdfs-scribe-integration.html

facebook

HDFS Raid

- Start the same: triplicate every data block
- Background encoding
  - Combine third replica of blocks from a single file to create parity block
  - Remove third replica
  - Apache Hadoop 0.22 release
- DiskReduce from CMU
  - Garth Gibson research

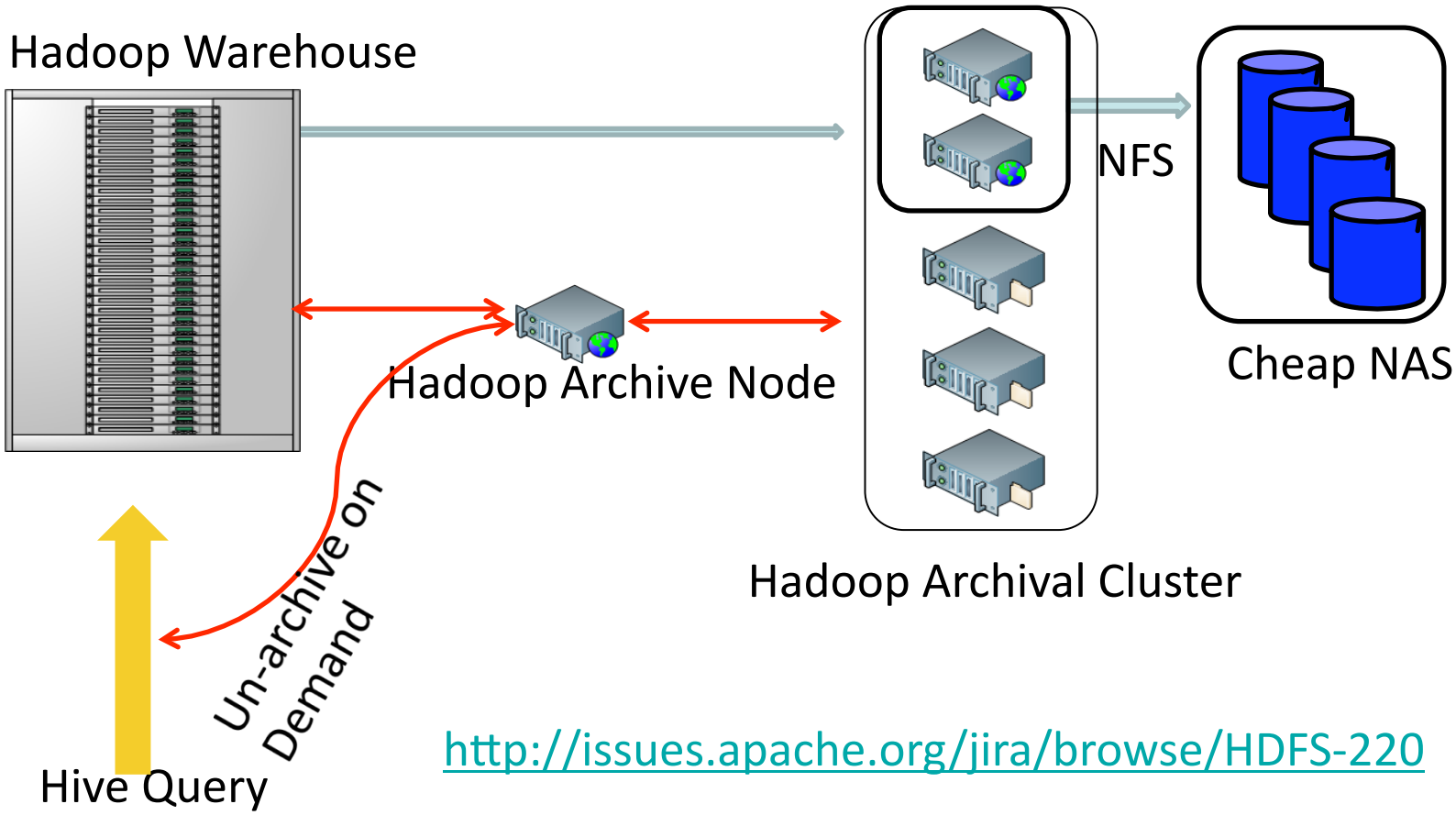


A file with three blocks A, B and C

http://hadoopblog.blogspot.com/2009/08/hdfs-and-erasure-codes-hdfs-raid.html

facebook

Archival: Move old data to cheap storage



http://issues.apache.org/jira/browse/HDFS-220

Dynamic-size MapReduce Clusters

- **Why multiple compute clouds in Facebook?**
  - Users unaware of resources needed by job
  - Absence of flexible Job Isolation techniques
  - Provide adequate SLAs for jobs
- **Dynamically move nodes between clusters**
  - Based on load and configured policies
  - Apache Jira MAPREDUCE-1044

facebook

Resource Aware Scheduling (Fair Share Scheduler)

- **We use the Hadoop Fair Share Scheduler**
  - Scheduler unaware of memory needed by job
- **Memory and CPU aware scheduling**
  - RealTime gathering of CPU and memory usage
  - Scheduler analyzes memory consumption in realtime
  - Scheduler fair-shares memory usage among jobs
  - Slot-less scheduling of tasks (in future)
  - Apache Jira MAPREDUCE-961

facebook

Hive – Data Warehouse

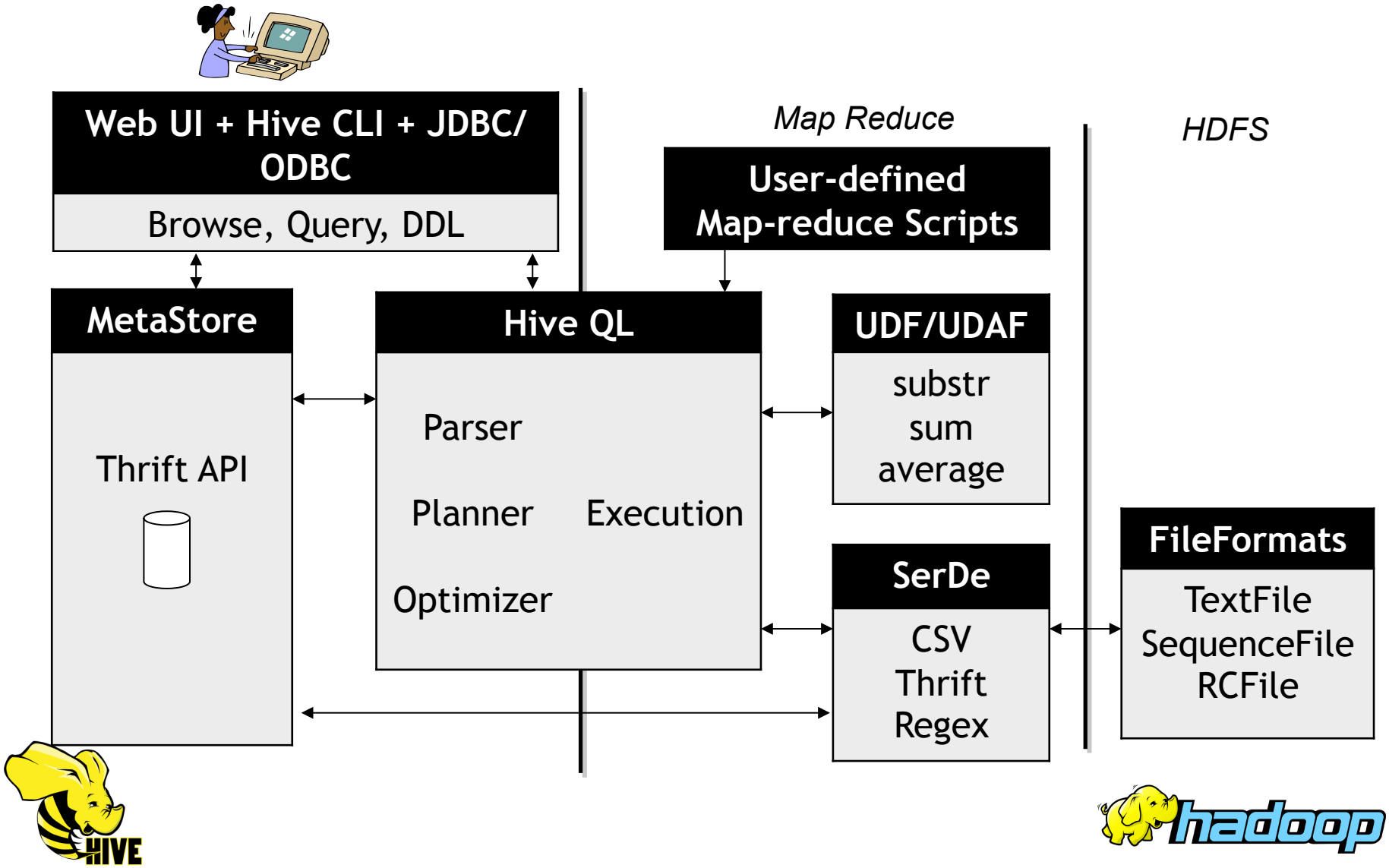
- Efficient SQL to Map-Reduce Compiler

- Mar 2008: Started at Facebook
- May 2009: Release 0.3.0 available
- Now: Preparing for release 0.4.0

- Countable for 95%+ of Hadoop jobs @ Facebook
- Used by ~200 engineers and business analysts at Facebook every month

facebook

Hive Architecture



File Formats

- TextFile:
  - Easy for other applications to write/read
  - Gzip text files are not splittable
- SequenceFile:
  - Only hadoop can read it
  - Support splittable compression
- RCFile: Block-based columnar storage
  - Use SequenceFile block format
  - Columnar storage inside a block
  - 25% smaller compressed size
  - On-par or better query performance depending on the query

SerDe

- Serialization/Deserialization
- Row Format
  - CSV (LazySimpleSerDe)
  - Thrift (ThriftSerDe)
  - Regex (RegexSerDe)
  - Hive Binary Format (LazyBinarySerDe)
- LazySimpleSerDe and LazyBinarySerDe
  - Deserialize the field when needed
  - Reuse objects across different rows
  - Text and Binary format

Useful Links

- **HDFS Design:**
  - http://hadoop.apache.org/core/docs/current/hdfs_design.html
- **Hadoop API:**
  - http://hadoop.apache.org/core/docs/current/api/
- **My Hadoop Blog:**
  - http://hadoopblog.blogspot.com/