Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT
2017

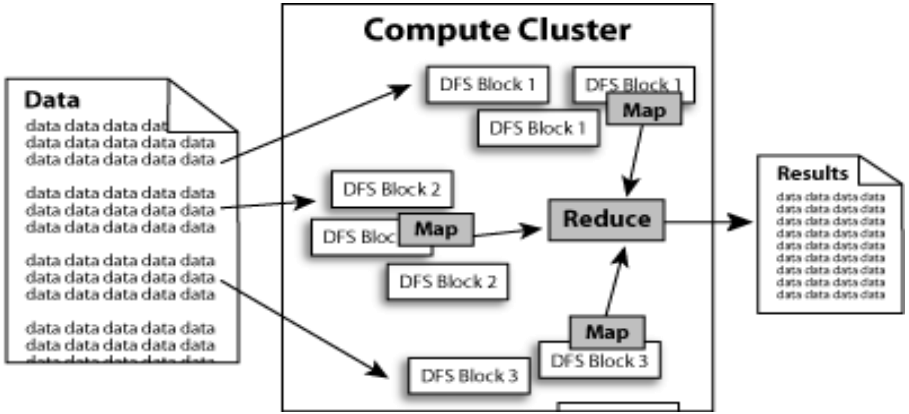
Hadoop: A Software Framework for Data Intensive Computing Applications

What is Hadoop?

- Software platform that lets one easily write and run applications that process vast amounts of data. It includes:
  - MapReduce – offline computing engine
  - HDFS – Hadoop distributed file system
  - HBase (pre-alpha) – online data access
- Yahoo! is the biggest contributor
- Here's what makes it especially useful:
  - **Scalable:** It can reliably store and process petabytes.
  - **Economical:** It distributes the data and processing across clusters of commonly available computers (in thousands).
  - **Efficient:** By distributing the data, it can process it in parallel **on the nodes where the data is located.**
  - **Reliable:** It automatically maintains multiple copies of data and automatically redeploys computing tasks based on failures.

What does it do?

- Hadoop implements Google's MapReduce, using HDFS
- MapReduce divides applications into many small blocks of work.
- HDFS creates multiple replicas of data blocks for reliability, placing them on compute nodes around the cluster.
- MapReduce can then process the data where it is located.
- Hadoop 's target is to run on clusters of the order of 10,000-nodes.



Hadoop: Assumptions

It is written with large clusters of computers in mind and is built around the following assumptions:

- Hardware *will* fail.
- Processing will be run in batches. Thus there is an emphasis on high throughput as opposed to low latency.
- Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size.
- It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance.
- Applications need a **write-once-read-many** access model.
- Moving Computation is Cheaper than Moving Data.
- Portability is important.

Apache Hadoop Wins Terabyte Sort Benchmark (July 2008)

- One of Yahoo's Hadoop clusters sorted 1 terabyte of data in **209 seconds**, which beat the previous record of 297 seconds in the annual general purpose (daytona) terabyte sort benchmark. The sort benchmark specifies the input data (10 billion 100 byte records), which must be completely sorted and written to disk.
- The sort used 1800 maps and 1800 reduces and allocated enough memory to buffers to hold the intermediate data in memory.
- The cluster had 910 nodes; 2 quad core Xeons @ 2.0ghz per node; 4 SATA disks per node; 8G RAM per a node; 1 gigabit ethernet on each node; 40 nodes per a rack; 8 gigabit ethernet uplinks from each rack to the core; Red Hat Enterprise Linux Server Release 5.1 (kernel 2.6.18); Sun Java JDK 1.6.0_05-b13

Example Applications and Organizations using Hadoop

- A9.com – Amazon: To build Amazon's product search indices; process millions of sessions daily for analytics, using both the Java and streaming APIs; clusters vary from 1 to 100 nodes.
- Yahoo! : More than 100,000 CPUs in ~20,000 computers running Hadoop; biggest cluster: 2000 nodes (2*4cpu boxes with 4TB disk each); used to support research for Ad Systems and Web Search
- AOL : Used for a variety of things ranging from statistics generation to running advanced algorithms for doing behavioral analysis and targeting; cluster size is 50 machines, Intel Xeon, dual processors, dual core, each with 16GB Ram and 800 GB hard-disk giving us a total of 37 TB HDFS capacity.
- Facebook: To store copies of internal log and dimension data sources and use it as a source for reporting/analytics and machine learning; 320 machine cluster with 2,560 cores and about 1.3 PB raw storage;
- FOX Interactive Media : 3 X 20 machine cluster (8 cores/machine, 2TB/machine storage) ; 10 machine cluster (8 cores/machine, 1TB/machine storage); Used for log analysis, data mining and machine learning
- University of Nebraska Lincoln: one medium-sized Hadoop cluster (200TB) to store and serve physics data;

More Hadoop Applications

- Adknowledge - to build the recommender system for behavioral targeting, plus other clickstream analytics; clusters vary from 50 to 200 nodes, mostly on EC2.
- Contextweb - to store ad serving log and use it as a source for Ad optimizations/ Analytics/reporting/machine learning; 23 machine cluster with 184 cores and about 35TB raw storage. Each (commodity) node has 8 cores, 8GB RAM and 1.7 TB of storage.
- Cornell University Web Lab: Generating web graphs on 100 nodes (dual 2.4GHz Xeon Processor, 2 GB RAM, 72GB Hard Drive)
- NetSeer - Up to 1000 instances on Amazon EC2 ; Data storage in Amazon S3; Used for crawling, processing, serving and log analysis
- The New York Times : Large scale image conversions ; EC2 to run hadoop on a large virtual cluster
- Powerset / Microsoft - Natural Language Search; up to 400 instances on Amazon EC2 ; data storage in Amazon S3

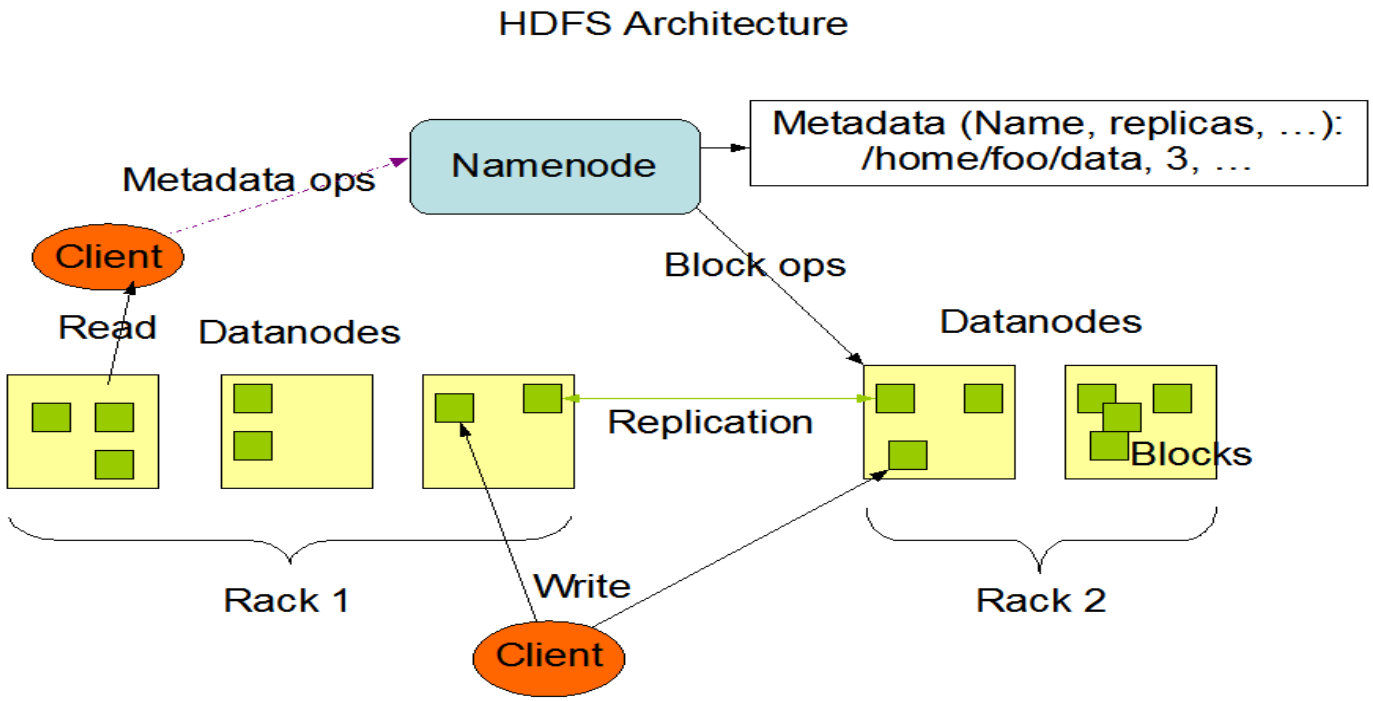
MapReduce Paradigm

- Programming model developed at Google
- Sort/merge based distributed computing
- Initially, it was intended for their internal search/indexing application, but now used extensively by more organizations (e.g., Yahoo, Amazon.com, IBM, etc.)
- It is functional style programming (e.g., LISP) that is naturally parallelizable across a large cluster of workstations or PCS.
- **The underlying system takes care of the partitioning of the input data, scheduling the program's execution across several machines, handling machine failures, and managing required inter-machine communication. (This is the key for Hadoop's success)**

HDFS

- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant.
  - highly fault-tolerant and is designed to be deployed on low-cost hardware.
  - provides high throughput access to application data and is suitable for applications that have large data sets.
  - relaxes a few POSIX requirements to enable streaming access to file system data.
  - part of the Apache Hadoop Core project. The project URL is http://hadoop.apache.org/core/.

HDFS Architecture



Example runs [1]

- Cluster configuration: ≈1800 machines; each with two 2GHz Intel Xeon processors with 4GB of memory (1-1.5 GB reserved for other tasks), two 160GB IDE disks, and a gigabit Ethernet link. All of the machines were in the same hosting facility and therefore the round-trip time between any pair of machines was less than a millisecond.
- Grep: *Scans through 10^{10} 100-byte records (distributed over 1000 input file by GFS)*, searching for a relatively rare three-character pattern (the pattern occurs in 92,337 records). The input is split into approximately 64MB pieces (M = 15000), and the entire output is placed in one file (R = 1). The entire computation took approximately 150 seconds from start to finish including 60 seconds to start the job.
- Sort: *Sorts 10^{10} 100-byte records (approximately* 1 terabyte of data). As before, the input data is split into 64MB pieces (M = 15000) and R = 4000. Including startup overhead, the entire computation took 891 seconds.

Execution overview

1. The MapReduce library in the user program first splits input files into M pieces of typically 16 MB to 64 MB/piece. It then starts up many copies of the program on a cluster of machines.
2. One of the copies of the program is the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
3. A worker who is assigned a map task reads the contents of the assigned input split. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.
4. The locations of these buffered pairs on the local disk are passed back to the master, who forwards these locations to the reduce workers.

Execution overview (cont.)

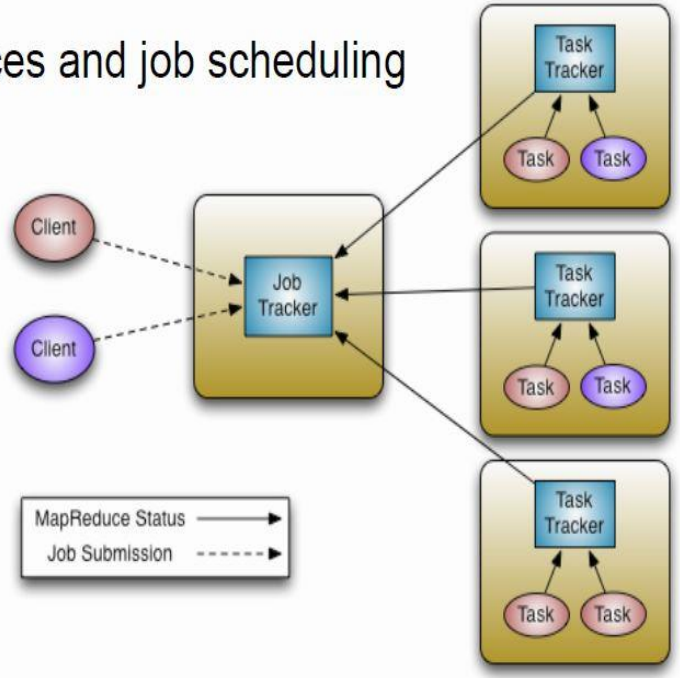
5. When a reduce worker is notified by the master about these locations, it uses RPC remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together.
6. The reduce worker iterates over the sorted intermediate data and for each **unique intermediate key** encountered, it passes the key and the corresponding set of intermediate values to the user's *Reduce function*. The output of the *Reduce function is appended* to a final output file for this reduce partition.
7. When all map tasks and reduce tasks have been completed, the master wakes up the user program---the MapReduce call in the user program returns back to the user code. The output of the mapreduce execution is available in the R output files (one per reduce task).

YARN

- Yet Another Resource Negotiator
- YARN Application Resource Negotiator(Recursive Acronym)
- Remedies the scalability shortcomings of “classic” MapReduce
- Is more of a general purpose framework of which classic mapreduce is one application.

Hadoop MapReduce Classic

- JobTracker
  - Manages cluster resources and job scheduling
- TaskTracker
  - Per-node agent
  - Manage tasks



Current MapReduce Limitations

- Scalability
  - Maximum Cluster Size – 4000 Nodes
  - Maximum Concurrent Tasks – 40000
  - Coarse synchronization in Job Tracker
- Single point of failure
  - Failure kills all queued and running jobs
  - Jobs need to be resubmitted by users
- Restart is very tricky due to complex state

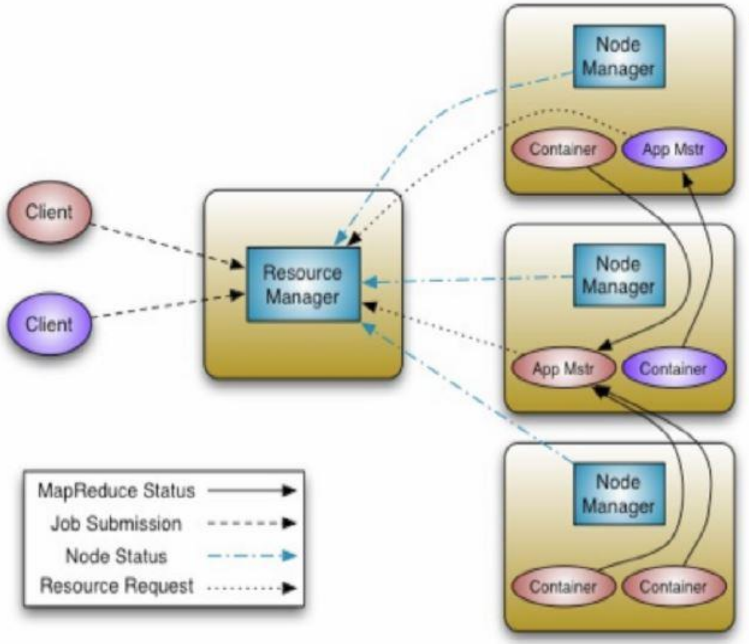
Yet Another Resource Manager

- Split up the two major responsibilities of the JobTracker/TaskTracker into separate entities
- - a global ResourceManager
- - a per-application ApplicationMaster
- - a per-node slave NodeManager
- - a per-application Container running on a NodeManager

YARN

- Splits up the two major functions of JobTracker
  - Global Resource Manager - Cluster resource management
  - Application Master - Job scheduling and monitoring (one per application). The Application Master negotiates resource containers from the Scheduler, tracking their status and monitoring for progress. Application Master itself runs as a normal *container*.
- Tasktracker
  - NodeManager (NM) - A new per-node slave is responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting to the Resource Manager.
- YARN maintains compatibility with existing MapReduce applications and users.

YARN – Architectural Overview



- Scalability - Clusters of 6,000-10,000 machines
  - Each machine with 16 cores, 48G/96G RAM, 24TB/36TB disks
  - 100,000+ concurrent tasks
  - 10,000 concurrent jobs

Classic MapReduce vs. YARN

- Fault Tolerance and Availability
  - Resource Manager
    - No single point of failure – state saved in ZooKeeper
    - Application Masters are restarted automatically on RM restart
  - Application Master
    - Optional failover via application-specific checkpoint
    - MapReduce applications pick up where they left off via state saved in HDFS
- Wire Compatibility
  - Protocols are wire-compatible
  - Old clients can talk to new servers
  - Rolling upgrades

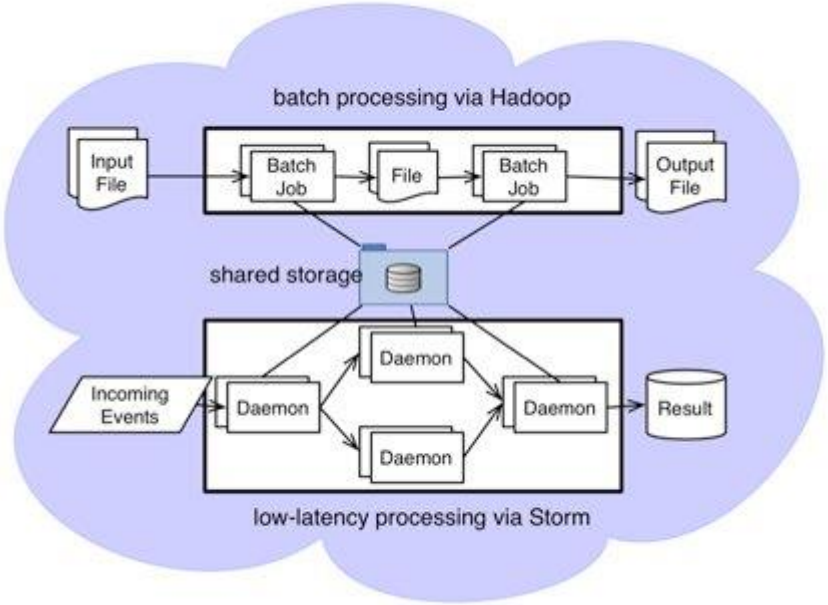
Classic MapReduce vs. YARN

- Support for programming paradigms other than MapReduce (Multi tenancy)
  - Tez – Generic framework to run a complex DAG
  - HBase on YARN(HOYA)
  - Machine Learning: Spark
  - Graph processing: Giraph
  - Real-time processing: Storm
  - Enabled by allowing the use of paradigm-specific application master
  - *Run all on the same Hadoop cluster!*

Storm on YARN

- Motivations
  - Collocating real-time processing with batch processing
  - Provides a huge potential for elasticity.
  - Reduces network transfer rates by moving storm closer to Mapreduce.

Storm on YARN @Yahoo



Storm on YARN @Yahoo

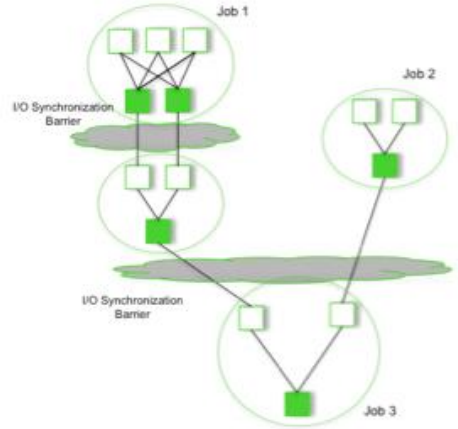
- Yahoo enhanced Storm to support Hadoop style security mechanisms
- Storm is being integrated into Hadoop YARN for resource management.
- Storm-on-YARN enables Storm applications to utilize the computational resources in our tens of thousands of Hadoop computation nodes.
- YARN is used to launch the Storm application master (Nimbus) on demand, and enables Nimbus to request resources for Storm application slaves (Supervisors).

Tez on YARN

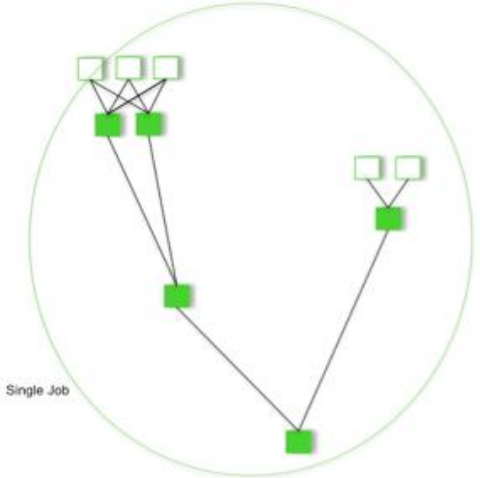
- Hindi for speed
- Provides a general-purpose, highly customizable framework that creates simplifies data-processing tasks across both small scale (low-latency) and large-scale (high throughput) workloads in Hadoop.
- Generalizes the MapReduce paradigm to a more powerful framework by providing the ability to execute a complex DAG
- Enables Apache Hive, Apache Pig and Cascading can meet requirements for human-interactive response times and extreme throughput at petabyte scale

Tez on YARN

- Original MapReduce requires disk I/O after each stage
- A series of MapReduce jobs following each other would result in lots of I/O
- Tez eliminates these intermediate steps, increasing the speed and lowering the resource usage



Pig/Hive - MR



Pig/Hive - Tez

Tez on YARN

- Performance gains over Mapreduce
  - Eliminates replicated write barrier between successive computations
  - Eliminates job launch overhead of workflow jobs
  - Eliminates extra stage of map reads in every workflow job
  - Eliminates queue and resource contention suffered by workflow jobs that are started after a predecessor job completes

HBase on YARN(HOYA)

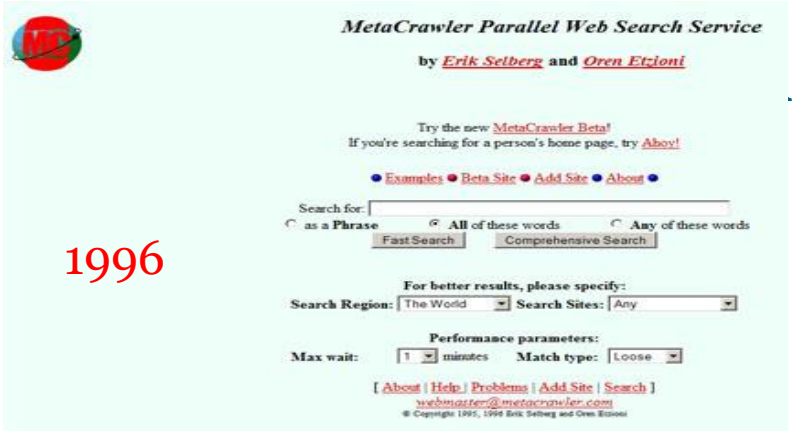
- Currently in prototype
- Be able to create on-demand HBase clusters easily -by and or in apps
  - With different versions of HBase potentially (for testing etc.)
- Be able to configure different HBase instances differently
  - For example, different configs for read/write workload instances
- Better isolation
  - Run arbitrary co-processors in user's private cluster
  - User will own the data that the hbase daemons create

HBase on YARN(HOYA)

- MR jobs should find it simple to create (transient) HBase clusters
  - For Map-side joins where table data is all in HBase, for example
- Elasticity of clusters for analytic / batch workload processing
  - Stop / Suspend / Resume clusters as needed
  - Expand / shrink clusters as needed
- Be able to utilize cluster resources better
  - Run MR jobs while maintaining HBase's low latency SLAs

What is Hadoop?

- Apache top level project, open-source implementation of frameworks for reliable, scalable, distributed computing and data storage.
- It is a flexible and highly-available architecture for large scale computation and data processing on a network of commodity hardware.

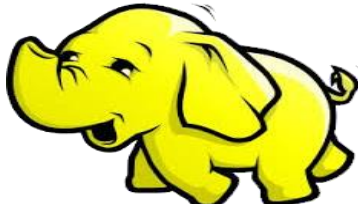


1996

1996

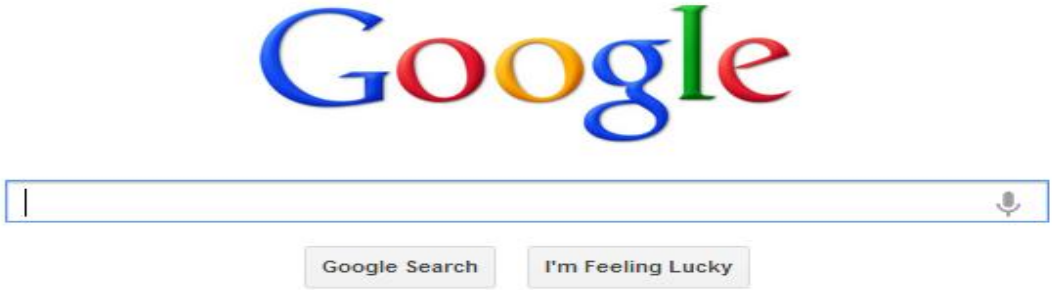
1996

1997

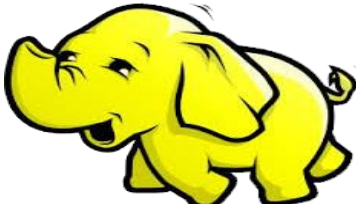


Google search engines

1998



2013



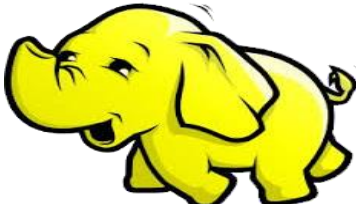
Hadoop's Developers

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.



Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

→ hadoop HDFS

2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat
jeff@google.com, sanjay@google.com
Google, Inc.

→ hadoop MapReduce

2006

Bigtable: A Distributed Storage System for Structured Data

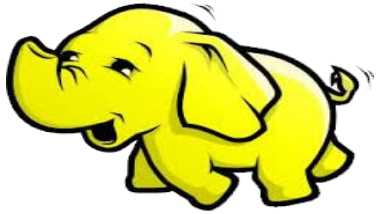
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

→ APACHE HBASE

Abstract

igtable is a distributed storage system for managing
tured data that is designed to scale to a very large
petabytes of data across thousands of commodity
rs. Many projects at Google store data in Bigtable,
uding web indexing, Google Earth, and Google Fi-
e. These applications place very different demands
igtable, both in terms of data size (from URLs to

achieved scalability and high performance, but Big
provides a different interface than such systems. Big
does not support a full relational data model; inste
provides clients with a simple data model that sup
dynamic control over data layout and format, an
lows clients to reason about the locality properties o
data represented in the underlying storage. Data i
dexed using row and column names that can be arbi
strings. Bigtable also treats data as uninterpreted str



Some Hadoop Milestones

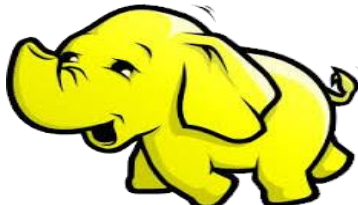
- **2008 - Hadoop Wins Terabyte Sort Benchmark (**sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)

- 2009 - Avro and Chukwa became new members of Hadoop Framework family

- 2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework

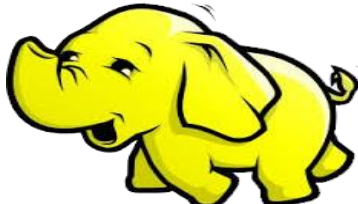
- **2011 - ZooKeeper Completed**

- **2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.**
  - Ambari, Cassandra, Mahout have been added

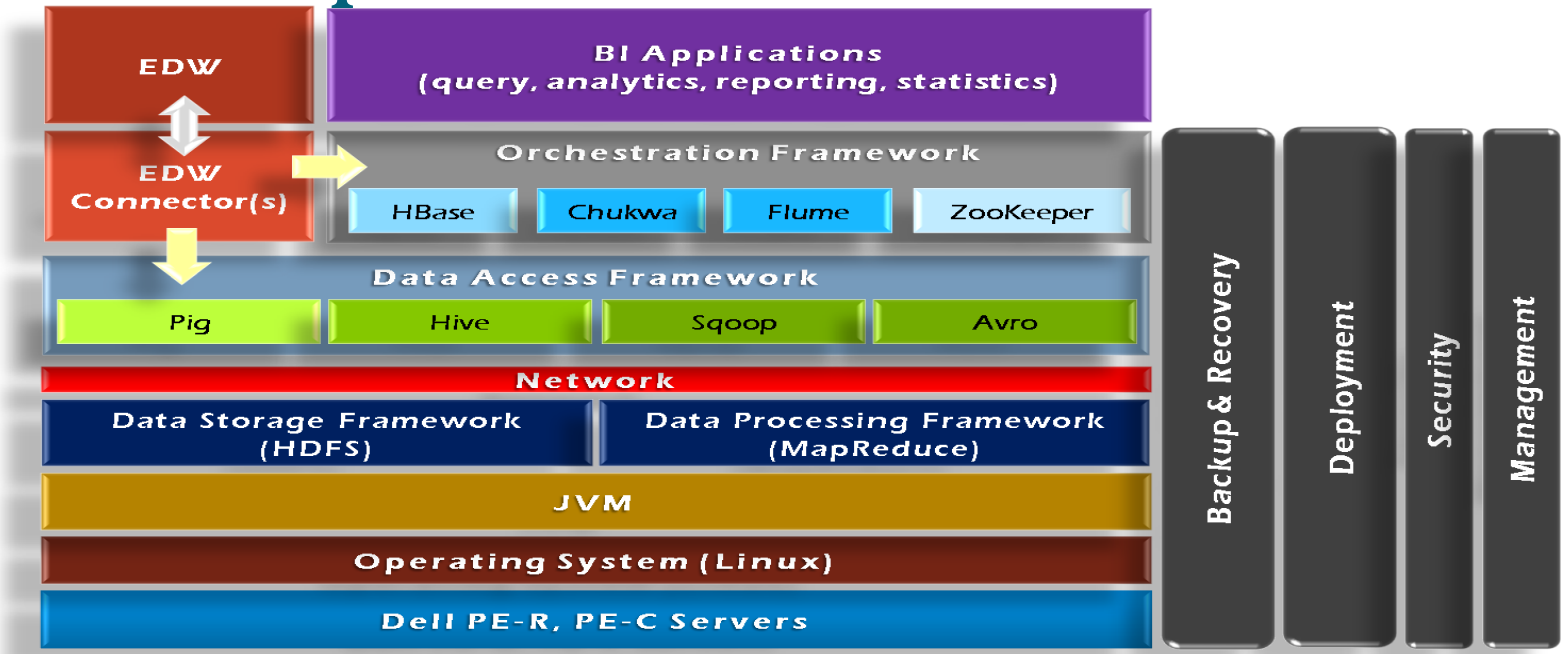


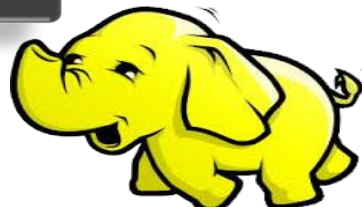
What is Hadoop?

- **Hadoop:**
  - an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license.
- **Goals / Requirements:**
  - Abstract and facilitate the storage and processing of large and/or rapidly growing data sets
    - Structured and non-structured data
    - Simple programming models
  - High scalability and availability
  - Use commodity (cheap!) hardware with little redundancy
  - Fault-tolerance
  - Move computation rather than data



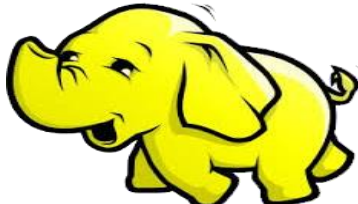
Hadoop Framework Tools



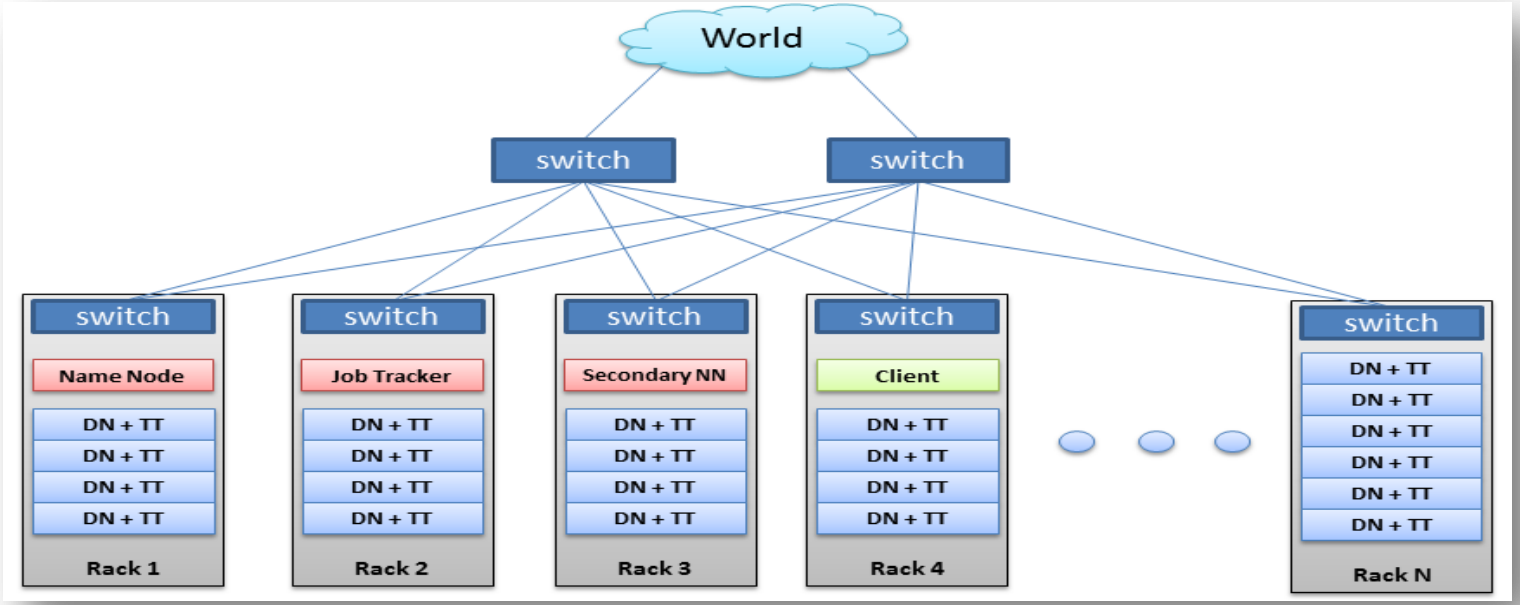


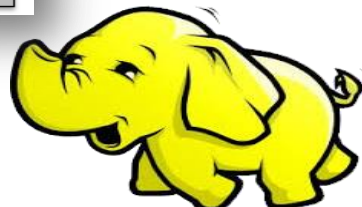
Hadoop's Architecture

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker
- Written in Java, also supports Python and Ruby



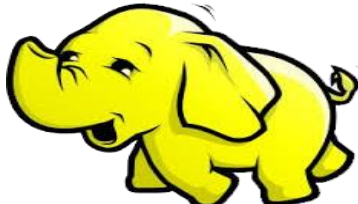
Hadoop's Architecture





Hadoop's Architecture

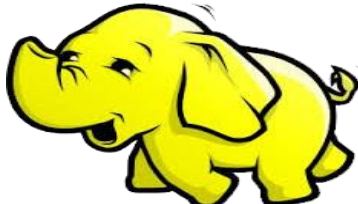
- Hadoop Distributed Filesystem
- Tailored to needs of MapReduce
- Targeted towards many reads of filestreams
- Writes are more costly
- High degree of data replication (3x by default)
- No need for RAID on normal nodes
- Large blocksize (64MB)
- Location awareness of DataNodes in network



Hadoop's Architecture

NameNode:

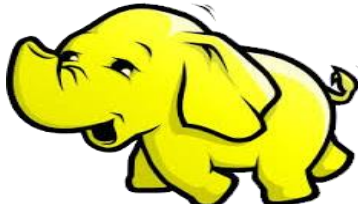
- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure



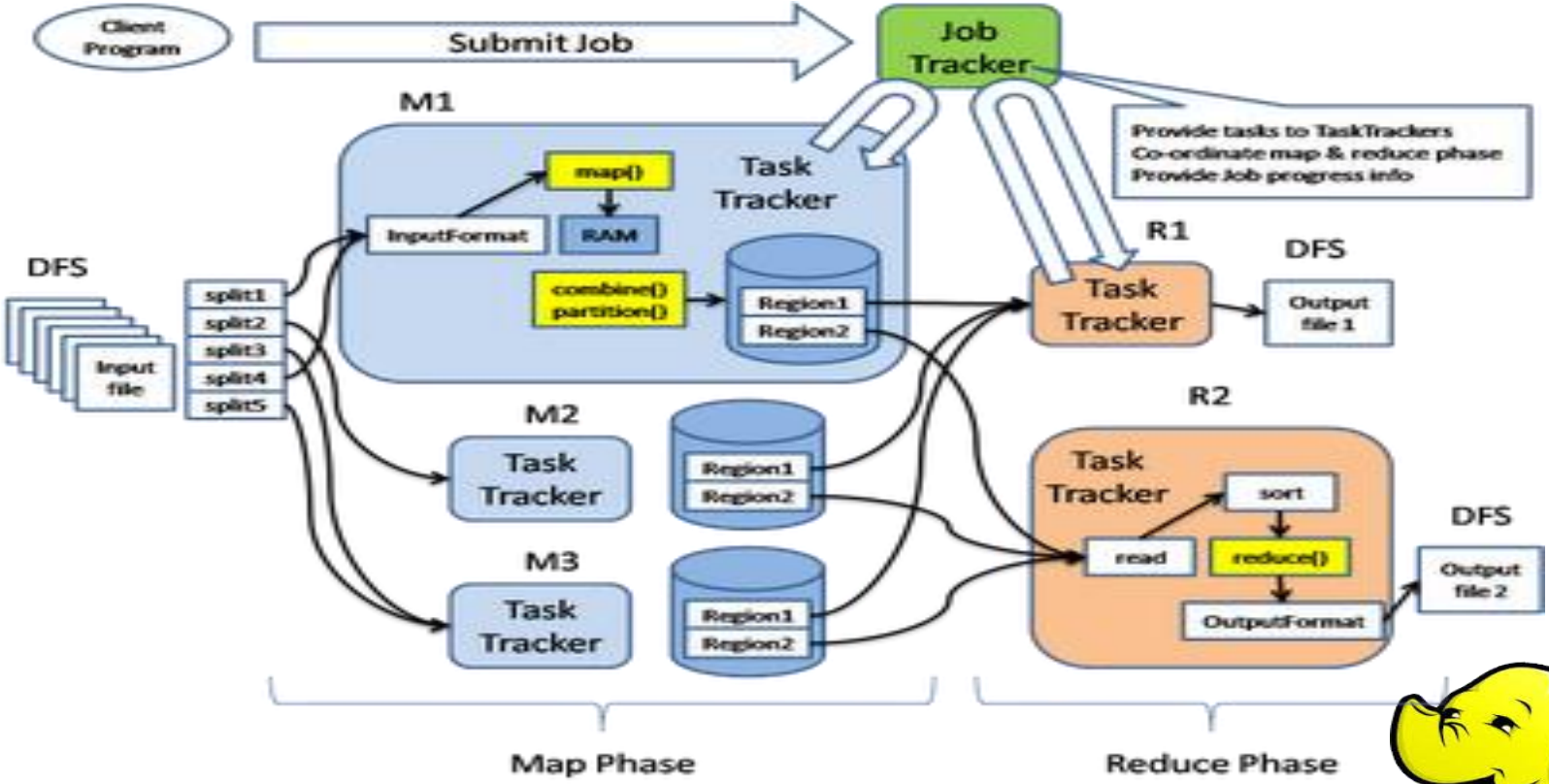
Hadoop's Architecture

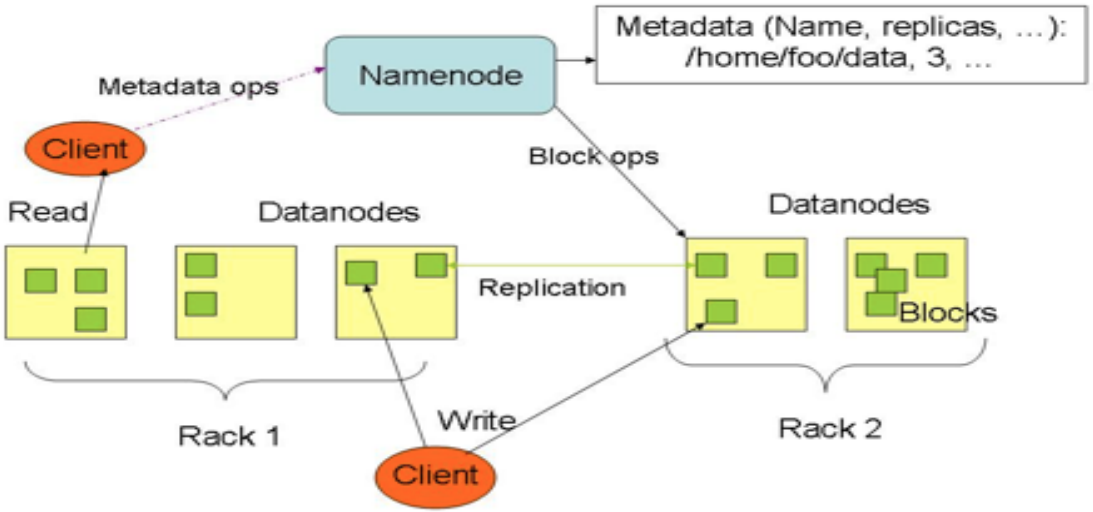
DataNode:

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere



Hadoop's Architecture: MapReduce Engine

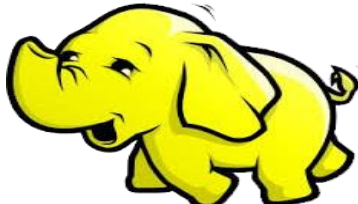



Metadata (Name, replicas, …):
/home/foo/data, 3, …
Namenode
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Client
Rack 2

Hadoop's Architecture

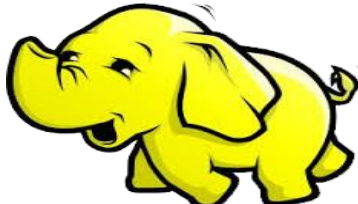
MapReduce Engine:

- JobTracker & TaskTracker
- JobTracker splits up data into smaller tasks("Map") and sends it to the TaskTracker process in each node
- TaskTracker reports back to the JobTracker node and reports on job progress, sends data ("Reduce") or requests new jobs



Hadoop's Architecture

- None of these components are necessarily limited to using HDFS
- Many other distributed file-systems with quite different architectures work
- Many other software packages besides Hadoop's MapReduce platform make use of HDFS

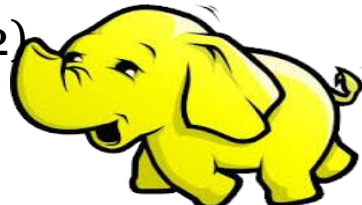


Hadoop in the Wild

- Hadoop is in use at most organizations that handle big data:
  - Yahoo!
  - Facebook
  - Amazon
  - Netflix
  - Etc…

- Some examples of scale:
  - Yahoo!'s Search Webmap runs on 10,000 core Linux cluster and powers Yahoo! Web search

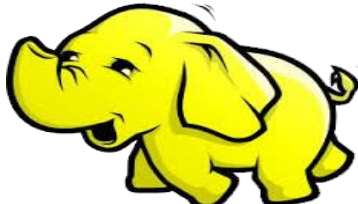
  - FB's Hadoop cluster hosts 100+ PB of data (July, 2012) & growing at ½ PB/day (Nov, 2012)



Hadoop in the Wild

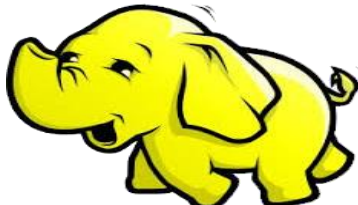
Three typical applications of Hadoop:

- Advertisement (Mining user behavior to generate recommendations)
- Searches (group related documents)
- Security (search for uncommon patterns)



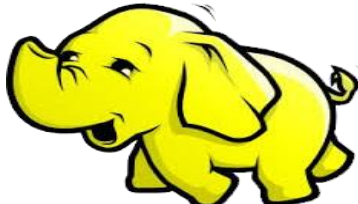
Hadoop in the Wild

- Non-realtime large dataset computing:
  - NY Times was dynamically generating PDFs of articles from 1851-1922
  - Wanted to pre-generate & statically serve articles to improve performance
  - Using Hadoop + MapReduce running on EC2 / S3, converted 4TB of TIFFs into 11 million PDF articles in 24 hrs



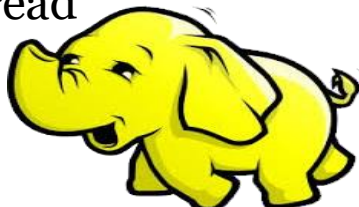
Hadoop in the Wild: Facebook Messages

- Design requirements:
  - Integrate display of email, SMS and chat messages between pairs and groups of users
  - Strong control over who users receive messages from
  - Suited for production use between 500 million people immediately after launch
  - Stringent latency & uptime requirements



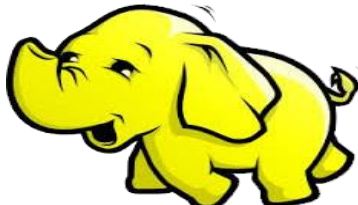
Hadoop in the Wild

- System requirements
  - High write throughput
  - Cheap, elastic storage
  - Low latency
  - High consistency (within a single data center good enough)
  - Disk-efficient sequential and random read performance



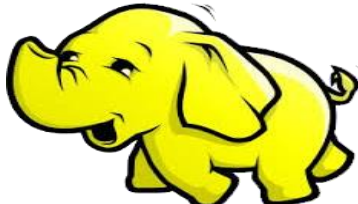
Hadoop in the Wild

- Classic alternatives
  - These requirements typically met using large MySQL cluster & caching tiers using Memcached
  - Content on HDFS could be loaded into MySQL or Memcached if needed by web tier
- Problems with previous solutions
  - MySQL has low random write throughput… BIG problem for messaging!
  - Difficult to scale MySQL clusters rapidly while maintaining performance



Hadoop in the Wild

- Facebook's solution
  - Hadoop + HBase as foundations
  - Improve & adapt HDFS and HBase to scale to FB's workload and operational considerations
    - Major concern was availability: NameNode is SPOF & failover times are at least 20 minutes
    - Proprietary "AvatarNode": eliminates SPOF, makes HDFS safe to deploy even with 24/7 uptime requirement
    - Performance improvements for realtime workload: RPC timeout. Rather fail fast and try a different DataNode



Hadoop Subprojects - Summary

- Pig
  - High-level language for data analysis
- HBase
  - Table storage for semi-structured data
- Zookeeper
  - Coordinating distributed applications
- Hive
  - SQL-like Query language and Metastore
- Mahout
  - Machine learning