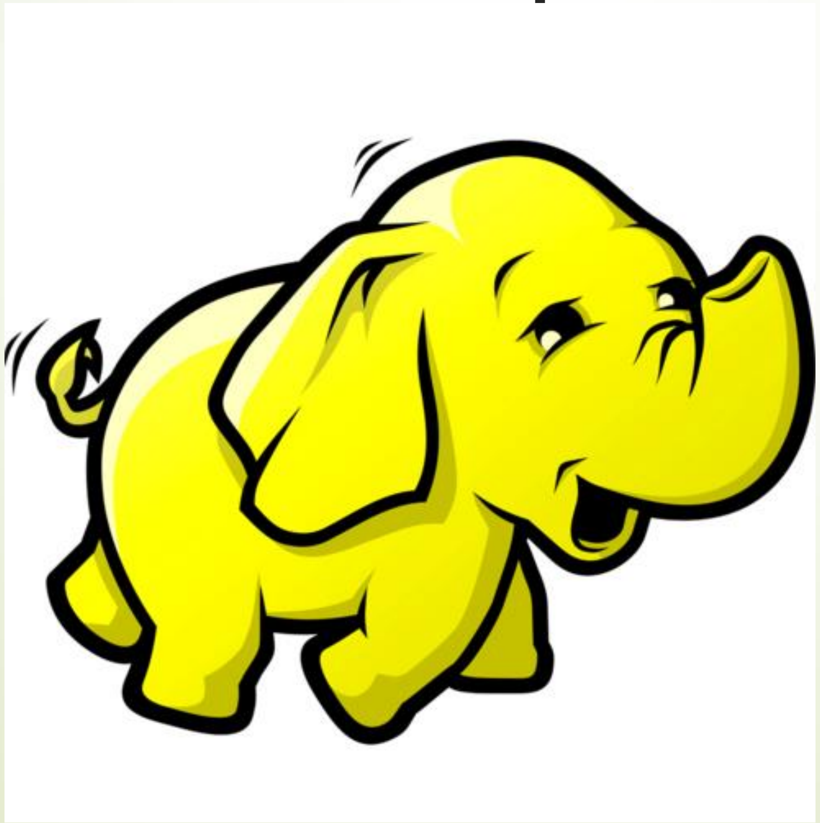
Introduction to Hadoop



New York Oracle User Group
Vikas Sawhney

GENERAL AGENDA

- Driving Factors behind BIG-DATA NOSQL Database
- 2014 Database Landscape
- Hadoop Architecture
- Map/Reduce
- Hadoop Eco-system
- Hadoop Releases
- Hadoop 2.0

NO-SQL BIG Data Revolution

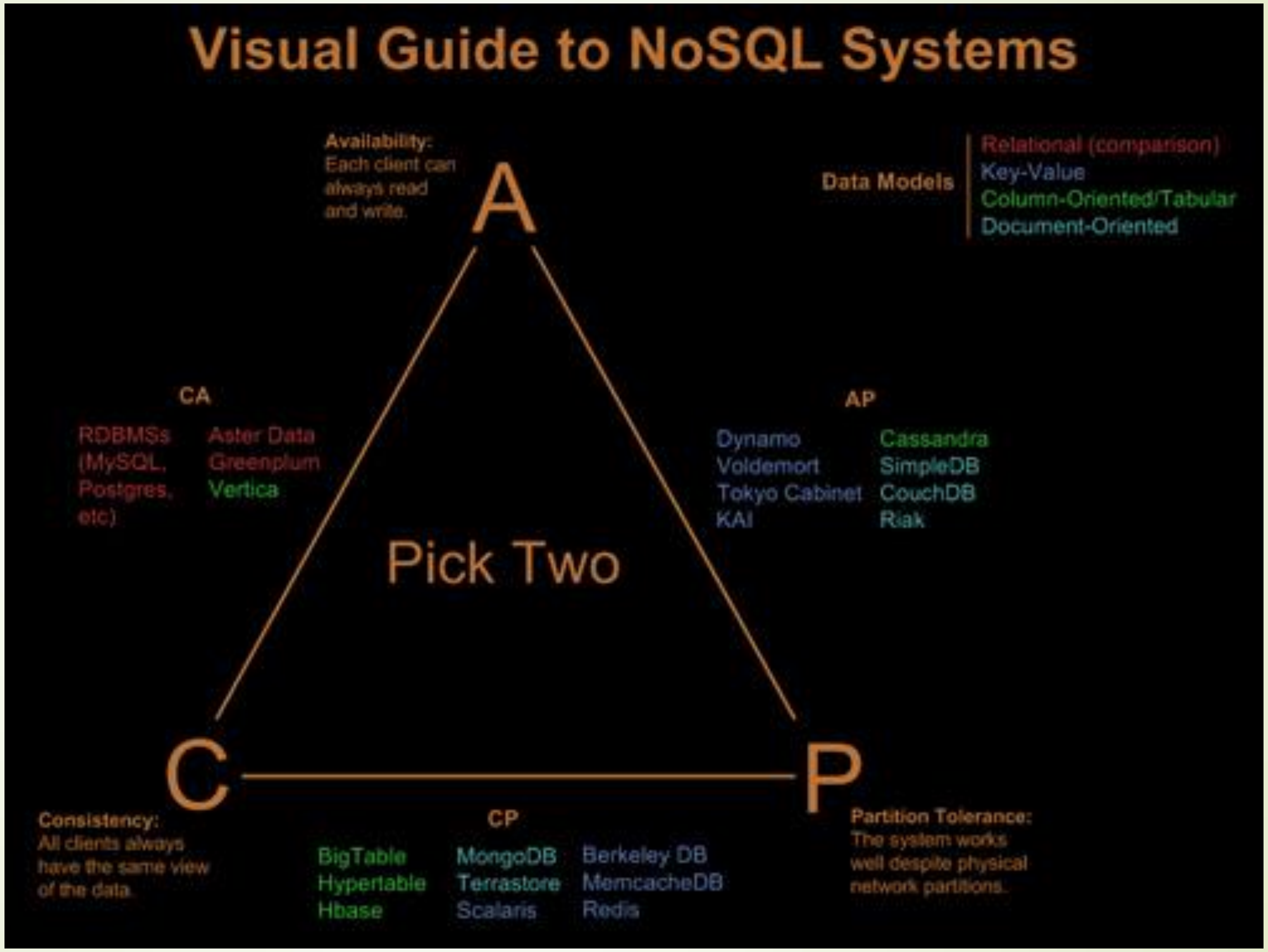
- Ease of elastic scalability horizontal scalability buy bigger hardware box with higher costs
- Costs associated with RDMBS to achieve horizontal (add hardware) scalability and vertically (larger boxes)
- Schema-less design for dynamic streams which does not find into a standard data model
- Ability to manage large volume, velocity, and variety of data.
- RDBMS databases are feature rich but lose on some of the scalability feature due various constraints referential integrity and data consistency

NO-SQL DATABASE DEFINED

- "A **NoSQL** or **Not Only SQL** database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. The data structure (e.g., key-value, graph, or document) differs from the RDBMS, and therefore some operations are faster in NoSQL and some in RDBMS. There are differences though and the particular suitability of a given NoSQL DB depends on the problem to be solved (e.g., does the solution use graph algorithms?).

  "NoSQL databases are finding significant and growing industry use in big data and real-time web applications. NoSQL systems are also referred to as "Not only SQL" to emphasize that they may in fact allow SQL-like query languages to be used. Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, the lack of standardized interfaces, and the huge investments already made in SQL by enterprises " – Ref: Wikipedia

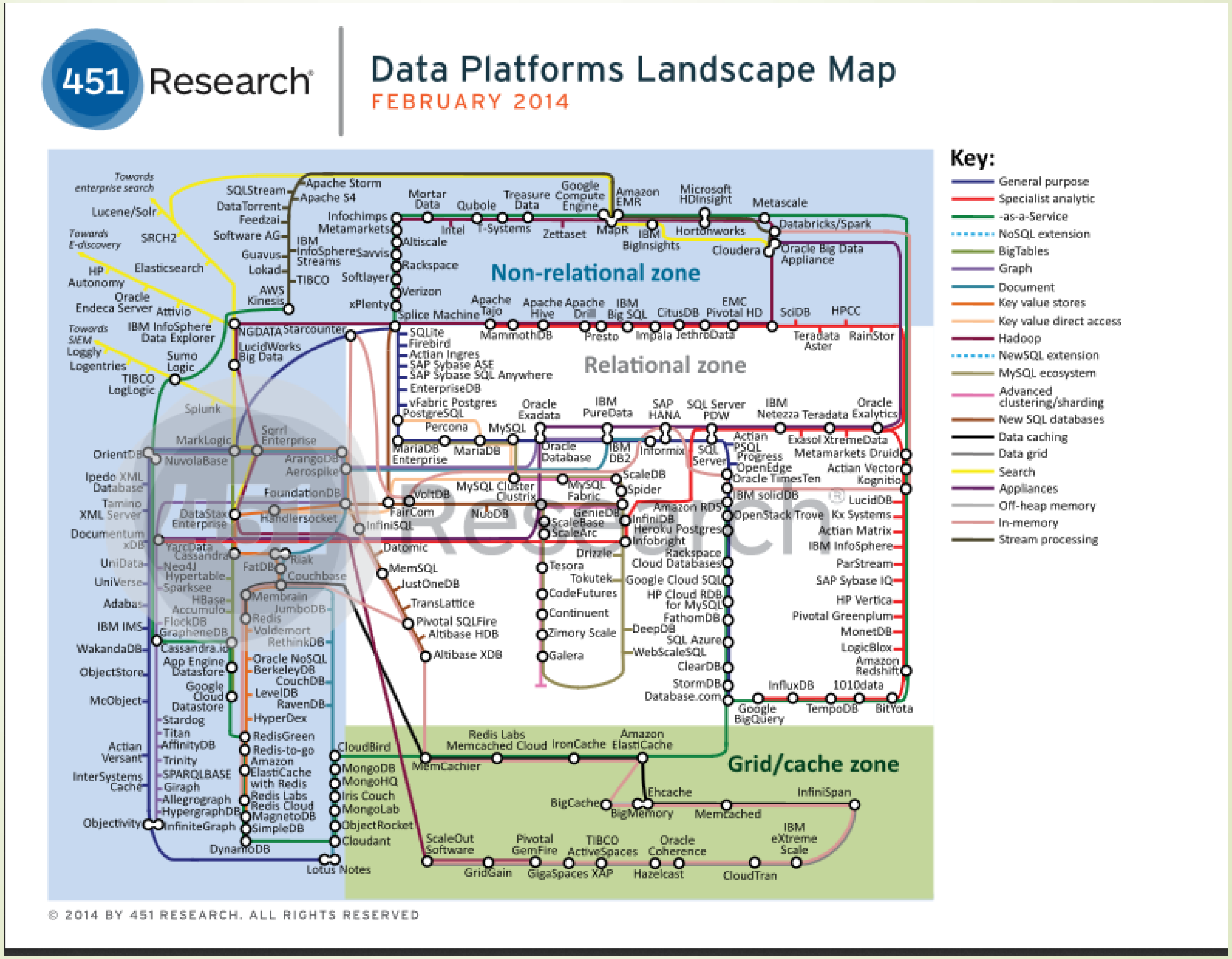
CAP Theorem


Visual Guide to NoSQL Systems
Availability: Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency: All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance: The system works well despite physical network partitions.

NO-SQL Database using the Hadoop framework

- Hadoop refers to an ecosystem of software packages, including MapReduce, HDFS, and a whole host of other software packages to support the import and export of data into and from HDFS (the Hadoop Distributed FileSystem). When someone says, "I have a Hadoop cluster," they generally mean a cluster of machines all running in this general ecosystem with a large distributed filesystem to support large scale computation.
- NoSQL is referring to non-relational or at least non-SQL database solutions such as HBase (also a part of the Hadoop ecosystem), Cassandra, MongoDB, Riak, CouchDB,

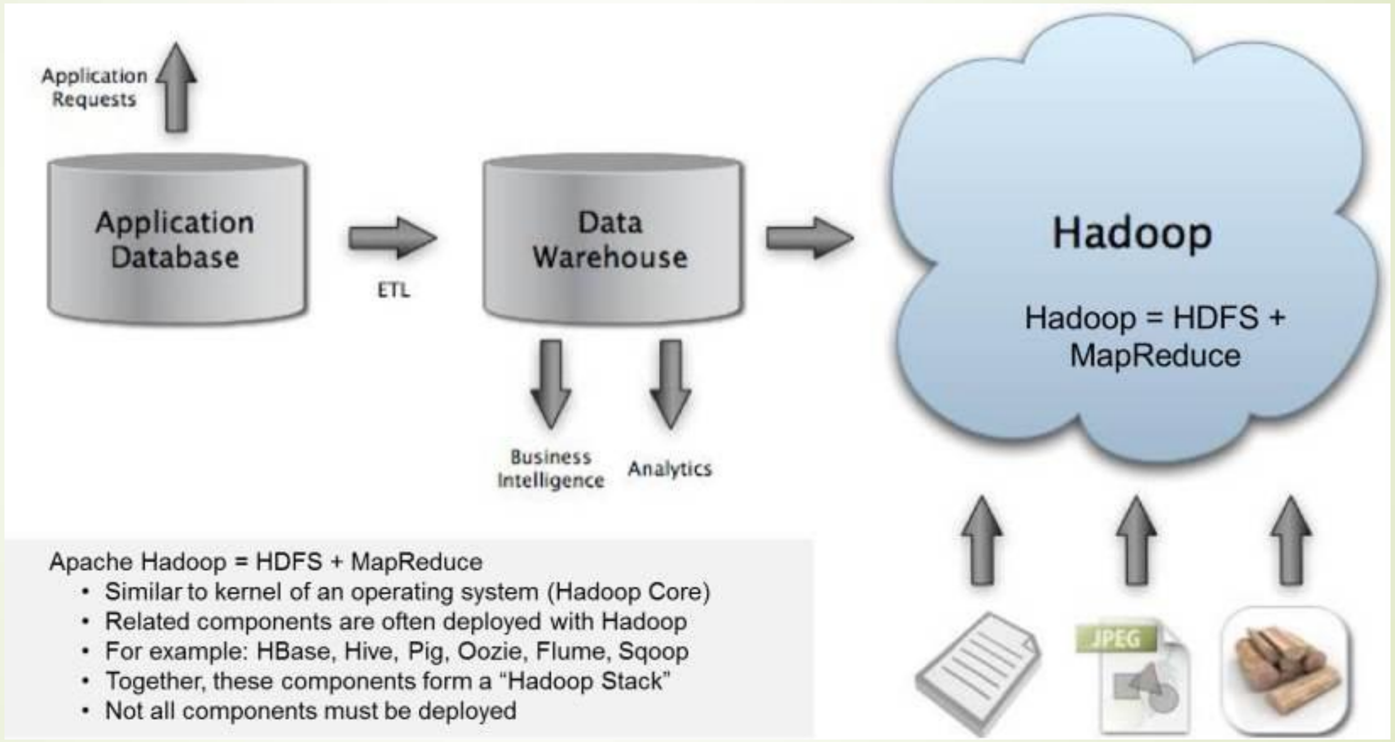
The NYC Subway MAP of Data Platforms



HADOOP

- Open source software platform for scalable, distributed computing framework for analysis of data
- Hadoop provides fast and reliable analysis of both structured data, unstructured data, polymorphic data
- Apache Hadoop is essentially a framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model.
- Hadoop can scale up from single servers to thousands of machines, each offering local computation and storage.

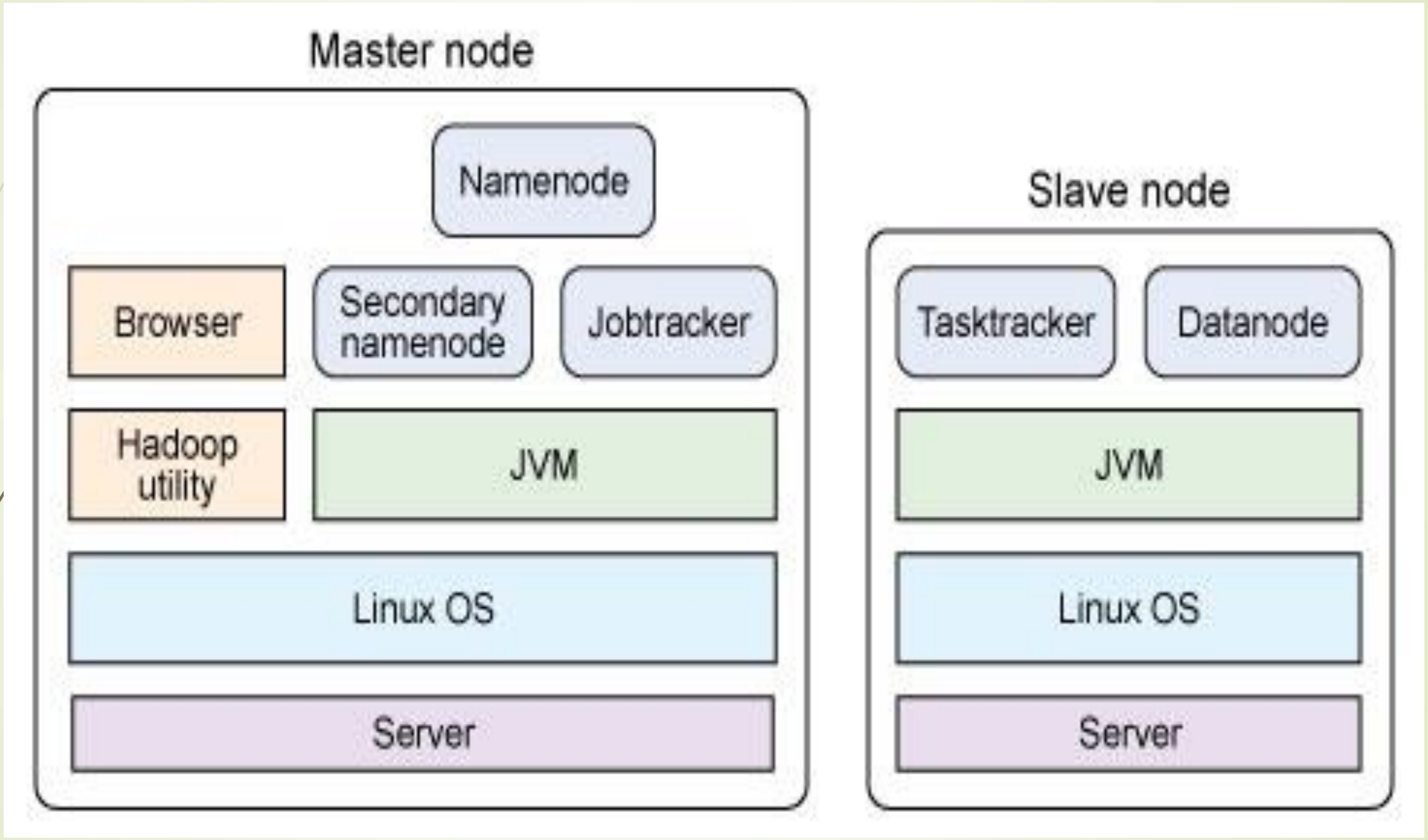
HADOOP ARCHITECTURE



Apache Hadoop = HDFS + MapReduce

- Similar to kernel of an operating system (Hadoop Core)
- Related components are often deployed with Hadoop
- For example: HBase, Hive, Pig, Oozie, Flume, Sqoop
- Together, these components form a “Hadoop Stack”
- Not all components must be deployed

Hadoop ARCHITECTURE



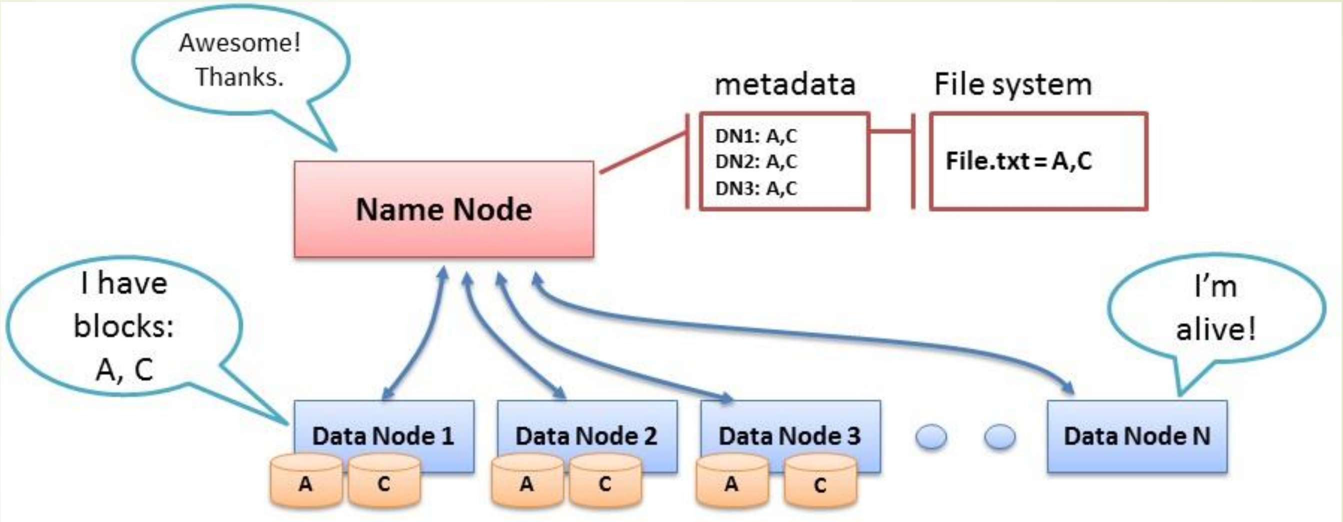
Hadoop cluster requirements

- **Prerequisites:**
  - System: Mac OS / Linux
  - Java 6 installed
  - Dedicated user for hadoop
  - SSH configured

HDFS Daemons

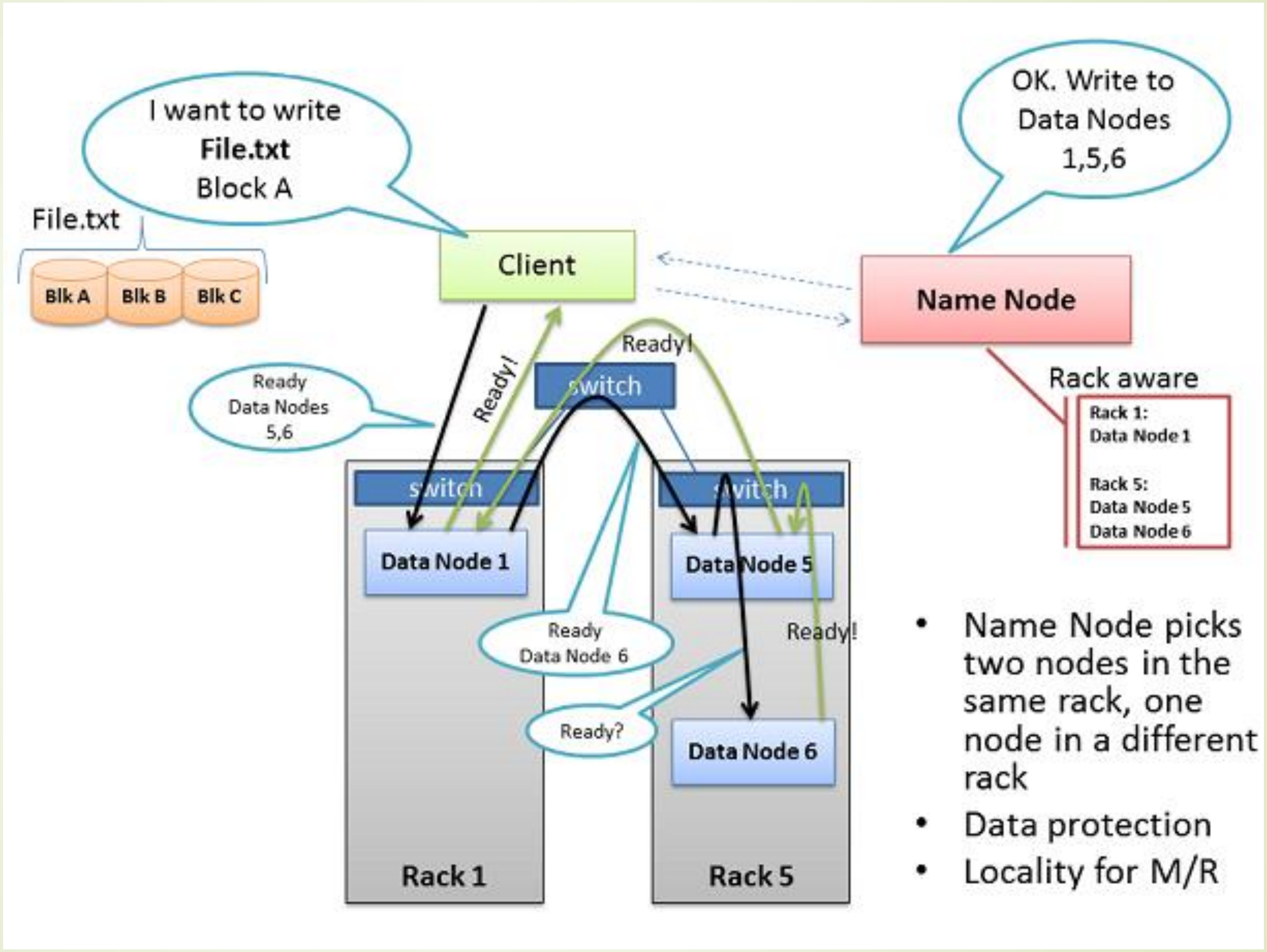
- **Name Node** : Keeps the metadata of all files/blocks in the file system, and tracks where across the cluster the file data is kept
- **Data Node :** DataNode actually stores data in the Hadoop Filesystem
- **Secondary Name Node :** perform periodic checkpoints to Name Node Metadata

Name Node



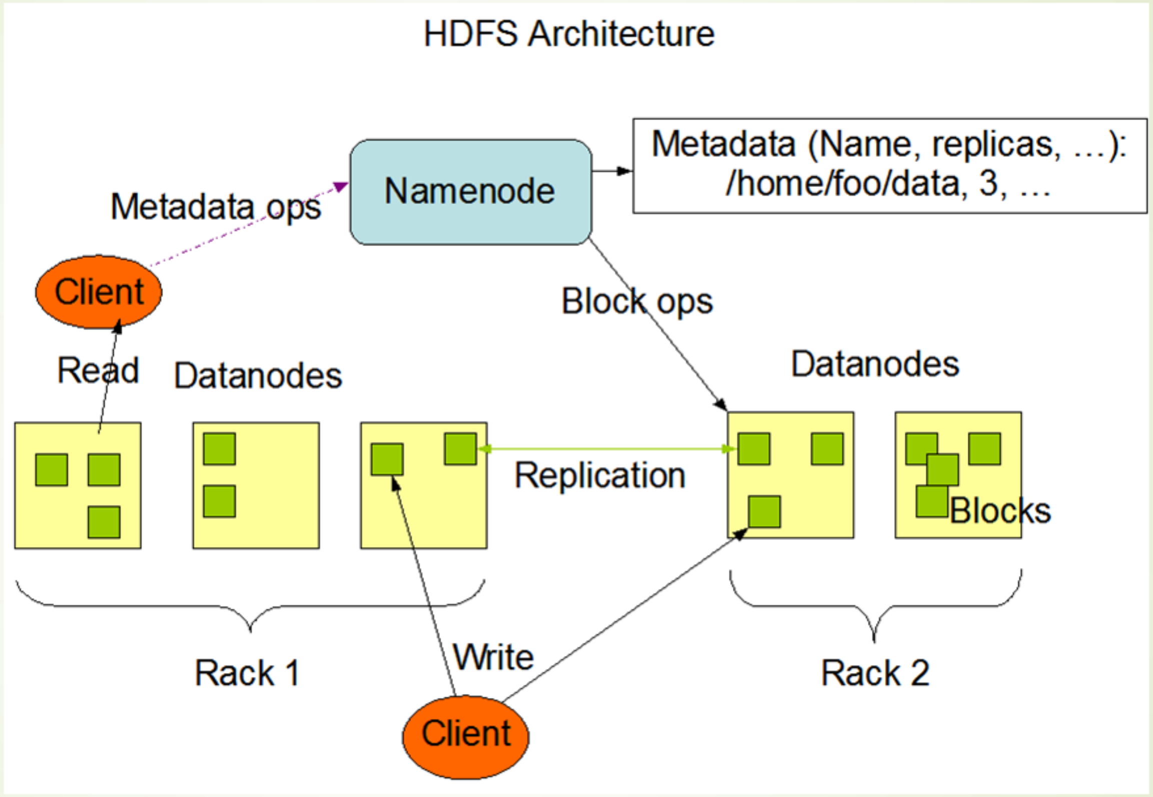
- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds
- If Name Node is down, HDFS is down

DATA Replication



- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

HDFS ARCHITECTURE



HDFS

- ✓ A distributed file system that provides high-throughput access to application data
- ✓ HDFS uses a master/slave architecture in which one device (master) termed as NameNode controls one or more other devices (slaves) termed as DataNode.
- ✓ It breaks Data/Files into small blocks (128 MB each block) and stores on DataNode and each block replicates on other nodes to accomplish fault tolerance.
- ✓ NameNode keeps the track of blocks written to the DataNode.

HDFS DAEMONS

- NAME NODE
- DATA NODE
- SECONDRY NAME NODE

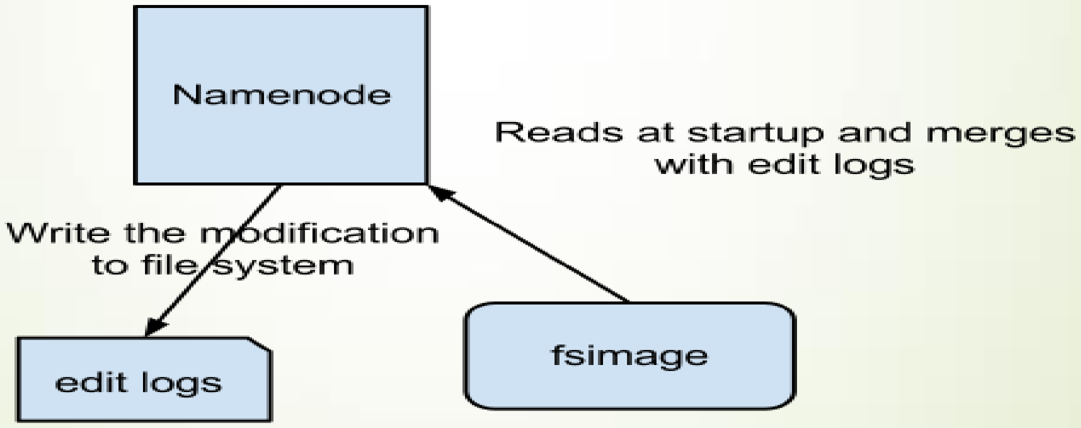
NAME NODE

- Keeps the metadata of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself.
- Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives
- NameNode Stores metadata in two files fsimage, Edit log

NAME NODE (Contd.)

fsimage - Its the snapshot of the filesystem when NameNode started

Edit logs - Its the sequence of changes made to the filesystem after NameNode started


Namenode
Reads at startup and merges with edit logs
Write the modification to file system
edit logs
fsimage

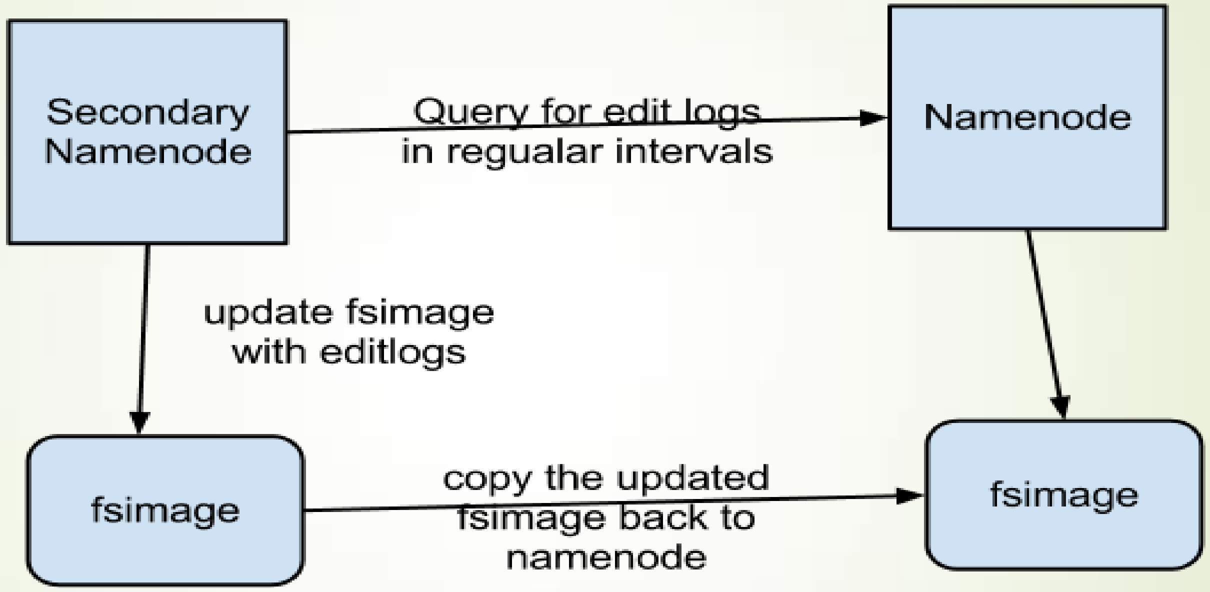
DATA NODE

- DataNode stores data in the Hadoop Filesystem
- A functional filesystem has more than one DataNode, with data replicated across them
- On startup, a DataNode connects to the NameNode; spinning until that service comes up. It then responds to requests from the NameNode for filesystem operations.
- Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data

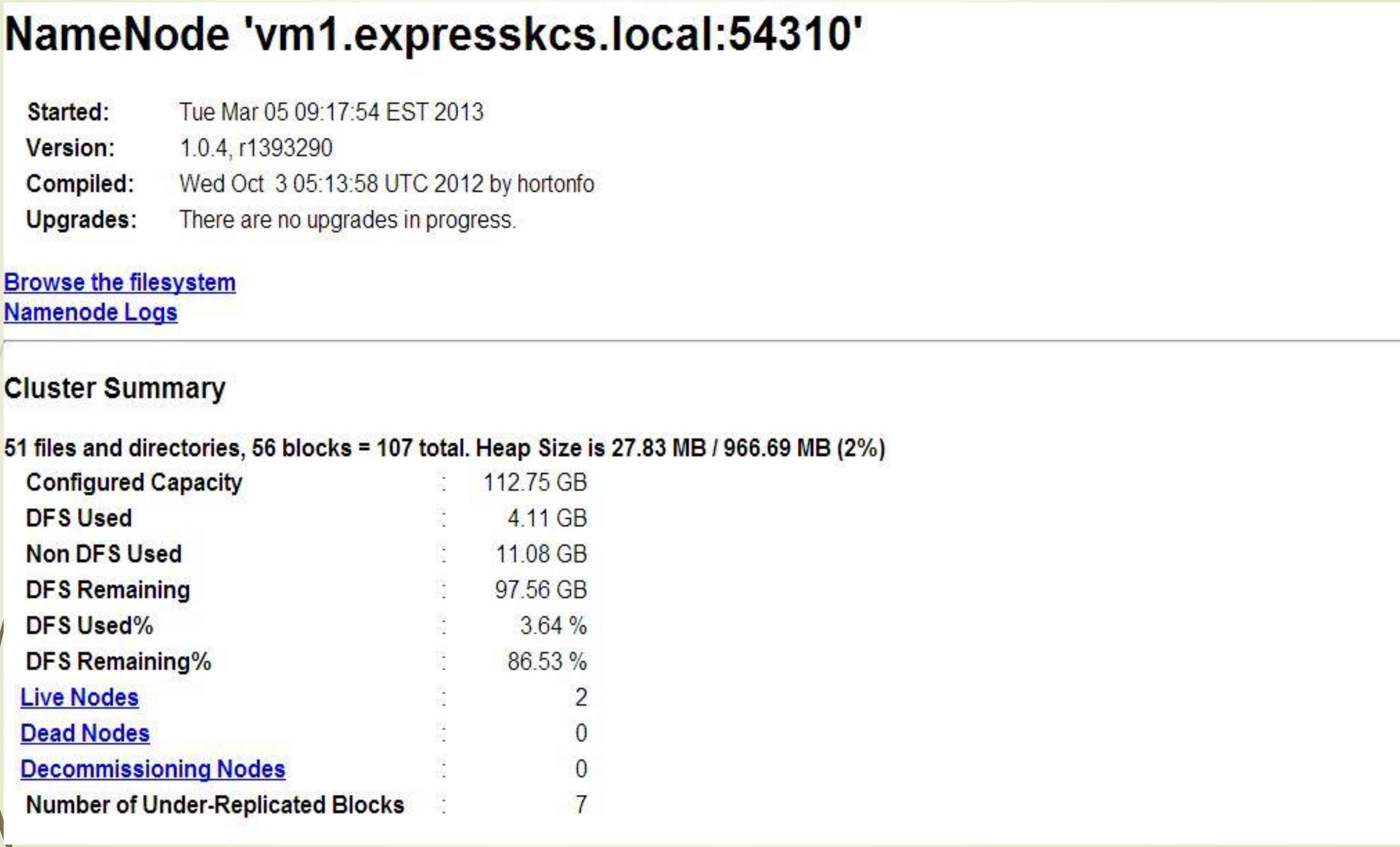
SECONDRY NAME NODE

- The term "secondary name-node" is somewhat misleading. It is not a name-node in the sense that data-nodes cannot connect to the secondary name-node, and in no event it can replace the primary name-node in case of its failure.
- The only purpose of the secondary name-node is to perform periodic checkpoints. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the (primary and the only) name-node

SECONDRY NAME NODE (Contd.)


Secondary Namenode
Query for edit logs in regualar intervals
Namenode
update fsimage with editlogs
fsimage
copy the updated fsimage back to namenode
fsimage

NAME NODE INTERFACE



NameNode 'vm1.expresskcs.local:54310'

Started:	Tue Mar 05 09:17:54 EST 2013
Version:	1.0.4, r1393290
Compiled:	Wed Oct 3 05:13:58 UTC 2012 by hortonfo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

51 files and directories, 56 blocks = 107 total. Heap Size is 27.83 MB / 966.69 MB (2%)

Configured Capacity	:	112.75 GB
DFS Used	:	4.11 GB
Non DFS Used	:	11.08 GB
DFS Remaining	:	97.56 GB
DFS Used%	:	3.64 %
DFS Remaining%	:	86.53 %
Live Nodes	:	2
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	7

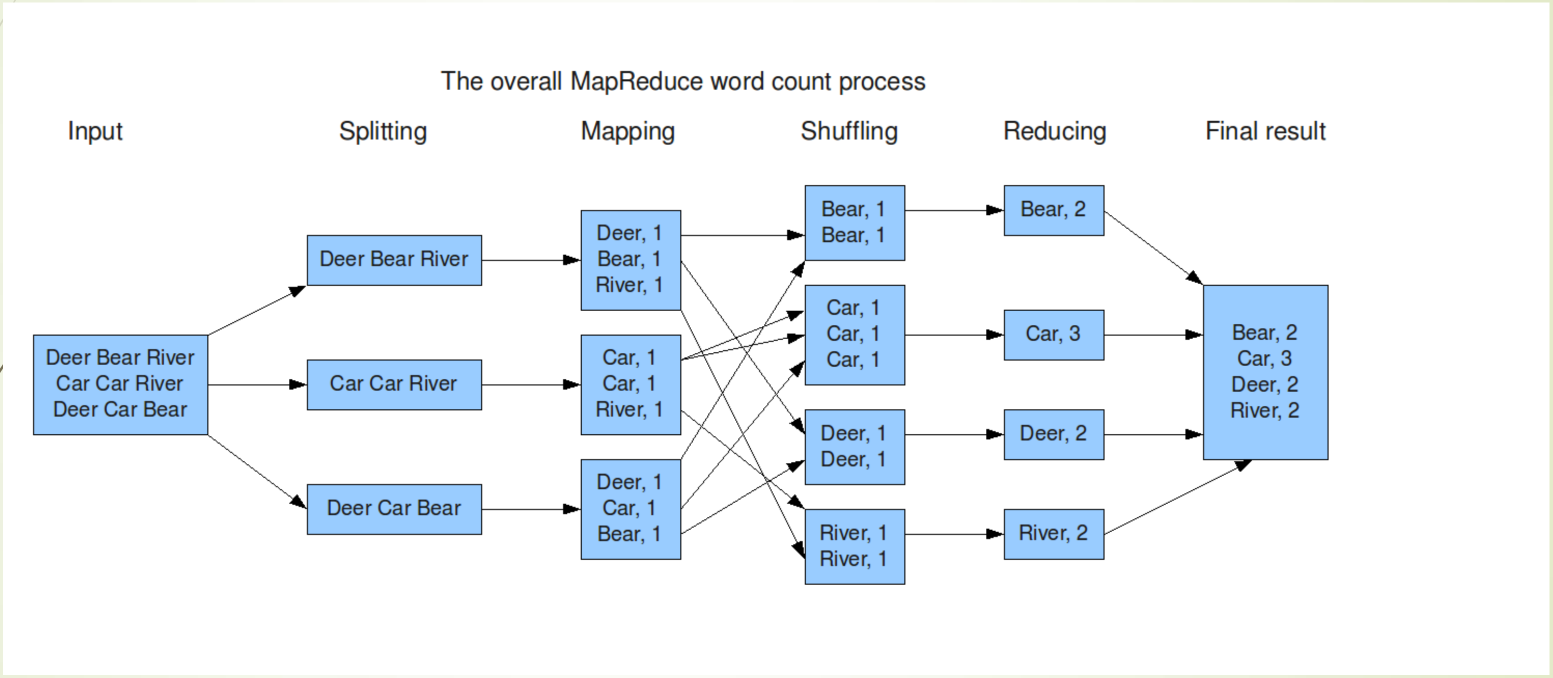
APACHE MAP/REDUCE

- A software framework for distributed processing of large data sets
- The framework takes care of scheduling tasks, monitoring them and re-executing any failed tasks.
- It splits the input data set into independent chunks that are processed in a completely parallel manner.
- MapReduce framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file system.

MAP REDUCE DATAFLOW

- An input reader
- A Map function
- A partition function
- A compare function
- A Reduce function
- An output writer

MAP/REDUCE ARCHITECTURE



MAPREDUCE DAEMONS

- JOB TRACKER
- TASK TRACKER

JOBTRACKER – Hadoop v1

- JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop

- JobTracker performs following actions in Hadoop :
  - It accepts the MapReduce Jobs from client applications
  - Talks to NameNode to determine data location
  - Locates available TaskTracker Node
  - Submits the work to the chosen TaskTracker Node

TASKTRACKER – Hadoop v1

- A TaskTracker node accepts map, reduce or shuffle operations from a JobTracker
- Its configured with a set of *slots*, these indicate the number of tasks that it can accept
- JobTracker seeks for the free slot to assign a job
- TaskTracker notifies the JobTracker about job success status.
- TaskTracker also sends the heartbeat signals to the job tracker to ensure its availability, it also reports the no. of available free slots with it.

HADOOP WEB INTERFACES

- MapReduce Job Tracker Web Interface

  http://localhost:50030/

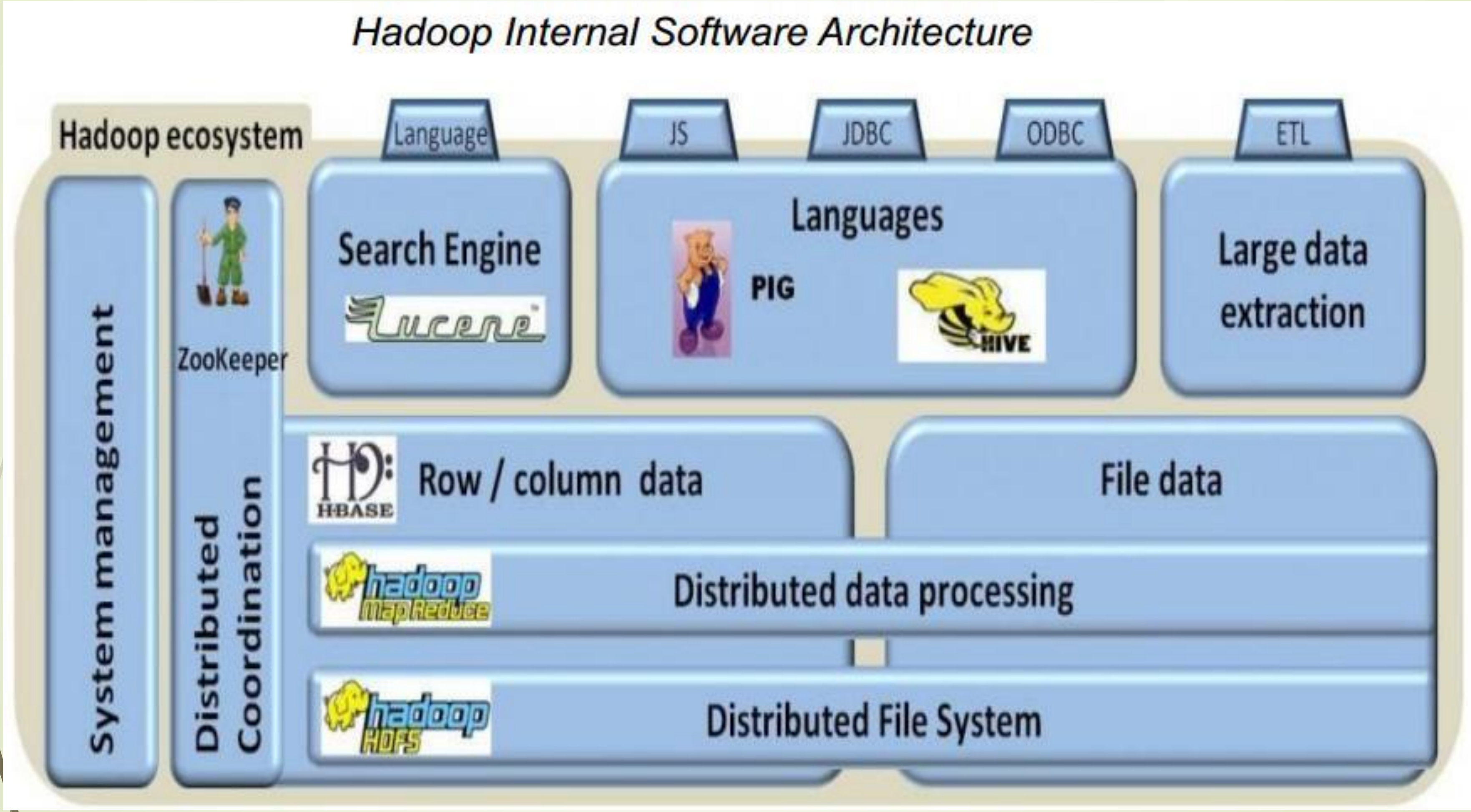
- Task Tracker Web Interface

  http://localhost:50060/

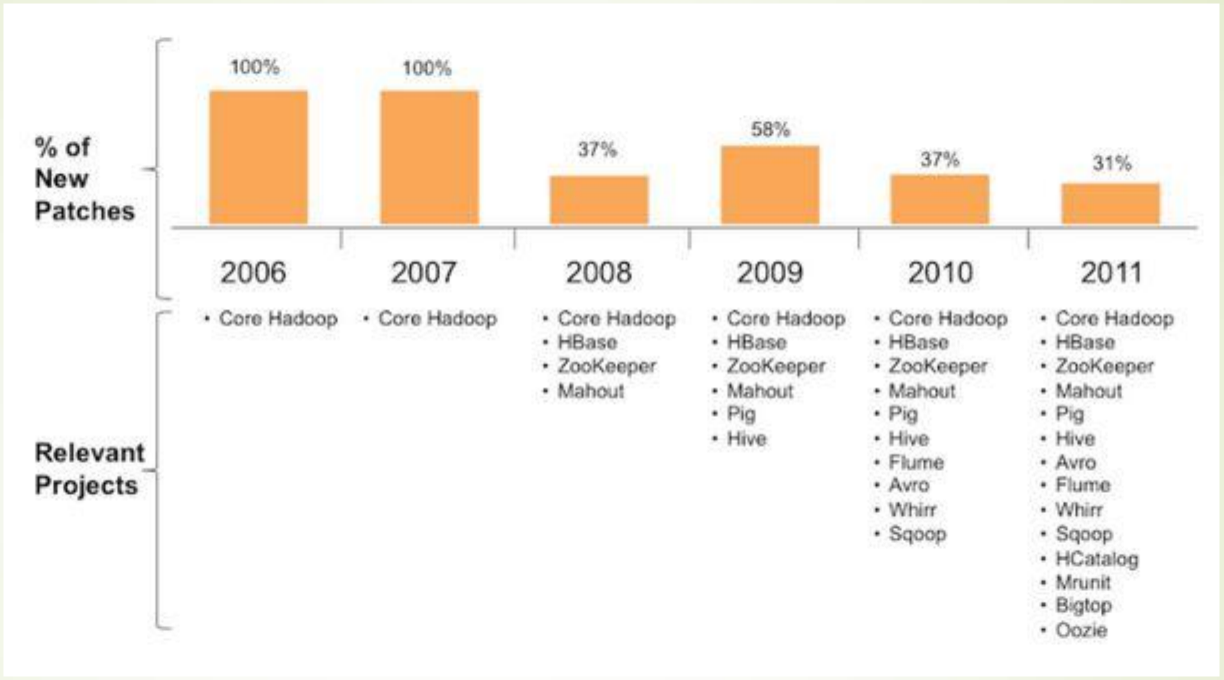
- HDFS Name Node Web Interface

  http://localhost:50070/

HADOOP STACK



Hadoop Ecosystems Project



	2006	2007	2008	2009	2010	2011
% of New Patches	100%	100%	37%	58%	37%	31%
Relevant Projects	• Core Hadoop	• Core Hadoop	• Core Hadoop • HBase • ZooKeeper • Mahout	• Core Hadoop • HBase • ZooKeeper • Mahout • Pig • Hive	• Core Hadoop • HBase • ZooKeeper • Mahout • Pig • Hive • Flume • Avro • Whirr • Sqoop	• Core Hadoop • HBase • ZooKeeper • Mahout • Pig • Hive • Avro • Flume • Whirr • Sqoop • HCatalog • Mrunit • Bigtop • Oozie

HADOOP ECOSYSTEMS PROJECTS

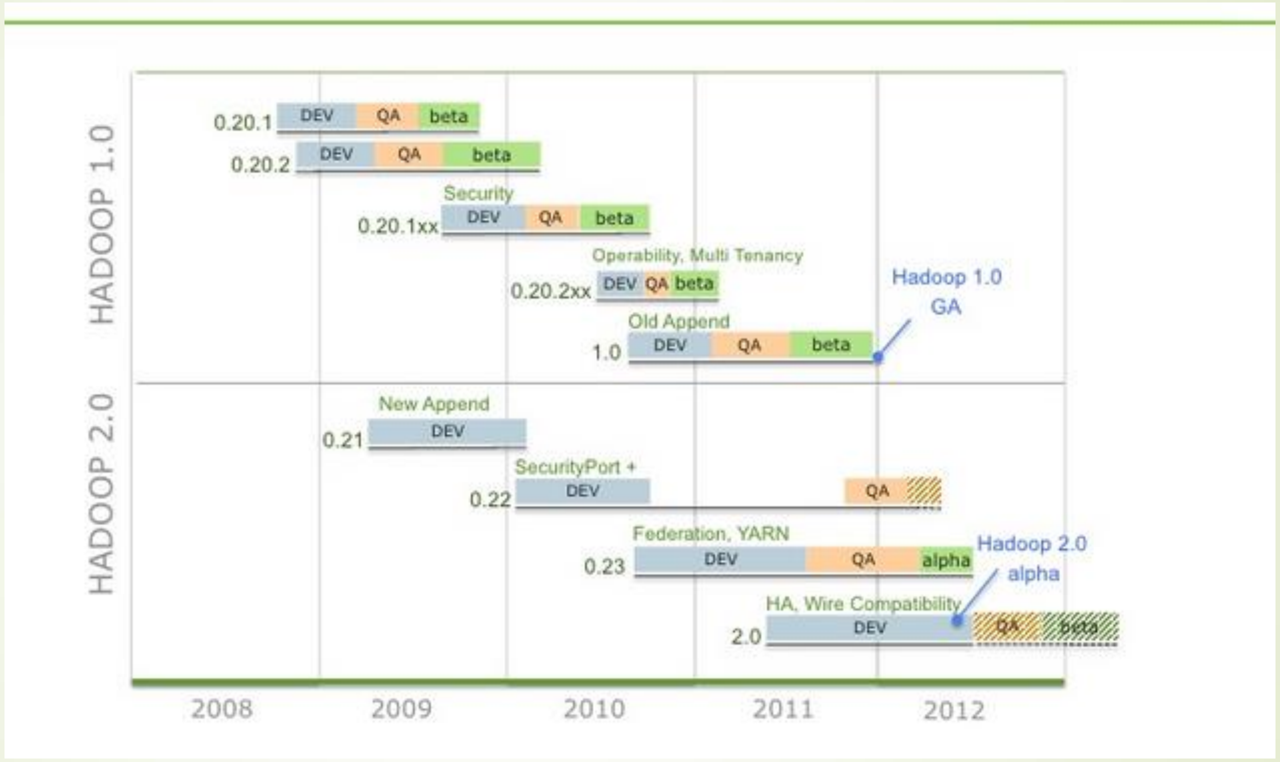
- Hadoop Ecosystem Projects includes:
  - Hadoop Common utilities
  - Avro: A data serialization system with scripting languages.
  - Chukwa: managing large distributed systems.
  - HBase: A scalable, distributed database for large tables.
  - HDFS: A distributed file system.
  - Hive: data summarization and ad hoc querying.
  - MapReduce: distributed processing on compute clusters.
  - Pig: A high-level data-flow language for parallel computation.
  - ZooKeeper: coordination service for distributed applications.

Hadoop releases

- Available Versions

- **1.2.X -** current stable version, 1.2 release
- **2.2.X -** current stable version, 2.x release
- **2.4.x** – current beta release
- **0.2.3.x** similar to 2.x.x but missing NN HA.

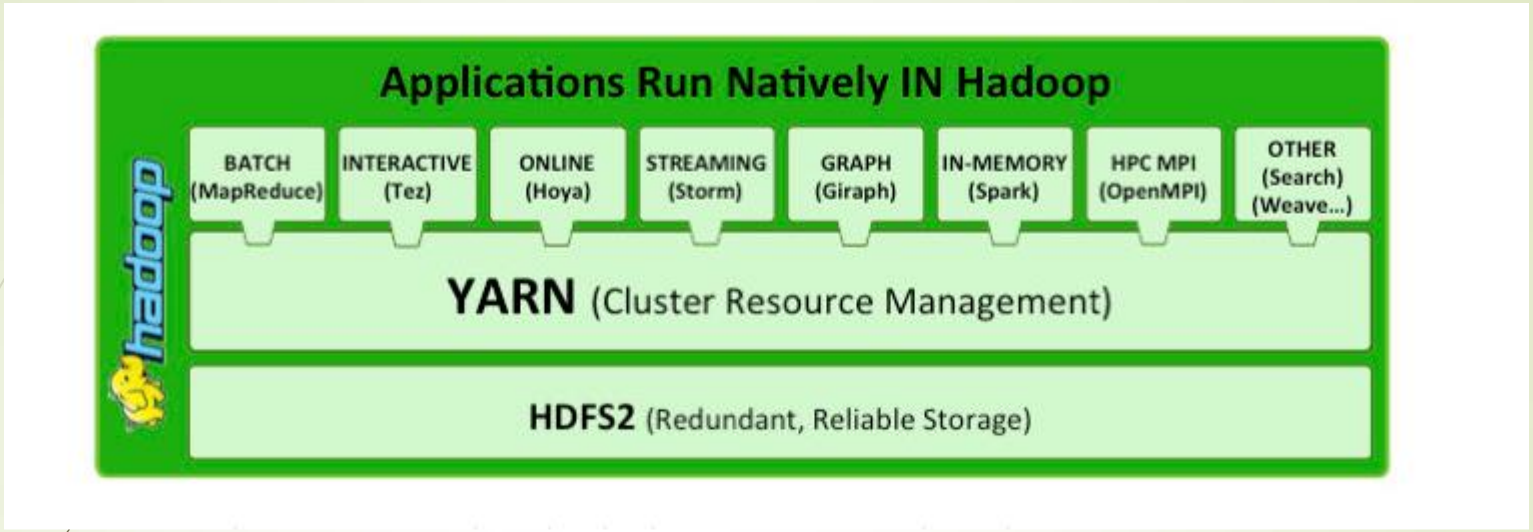
Release Lifecycle



Notable editions Hadoop 2.0

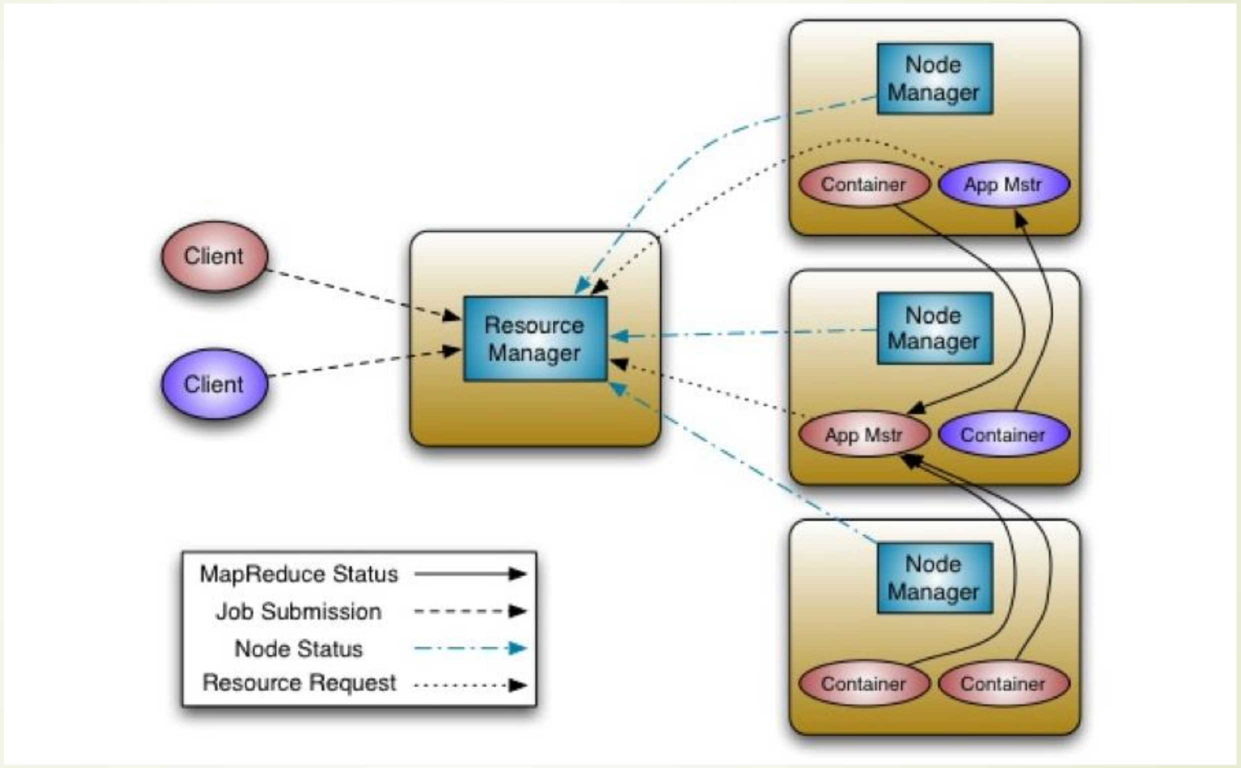
- **Map Reduce process restructured with YARN**
- **NameNode HA:** automated failover with a hot standby and resiliency for the NameNode master service.
- **Snapshots:** point-in-time recovery for backup, disaster recovery and protection against use errors.
- **Federation:** allow for multiple namenode to manage namespace for a Hadoop cluster. With HDFS federation, multiple Namenode servers manage namespaces and this allows for horizontal scaling, performance improvements, and multiple namespaces.
- Speed improvement in SQL query language HIVE

NextGen MapReduce (YARN)

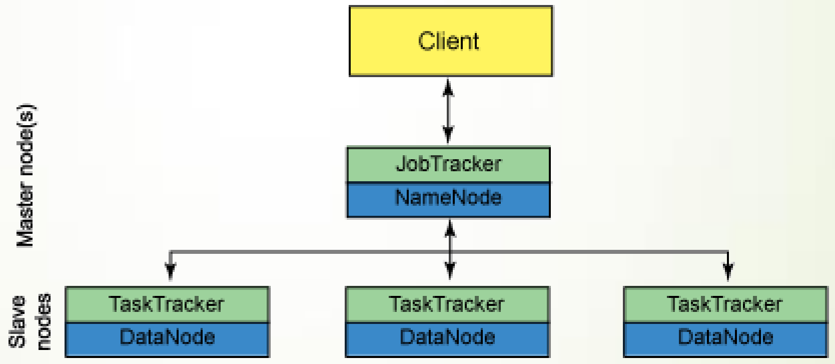

Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (Hoya)
STREAMING (Storm)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave...)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)
hadoop

- The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker and TaskTracker into separate entities. In Hadoop 2.0, the JobTracker and TaskTracker no longer exist and have been replaced by three components:
- **ResourceManager:** a scheduler that allocates available resources in the cluster amongst the competing applications.
- **NodeManager:** runs on each node in the cluster and takes direction from the ResourceManager. It is responsible for managing resources available on a single node.
- **ApplicationMaster:** an instance of a framework-specific library, an ApplicationMaster runs a specific YARN job and is responsible for negotiating resources from the ResourceManager and also working with the NodeManager to execute and monitor Containers.
- The actual data processing occurs within the Containers executed by the ApplicationMaster. A Container grants rights to an application to use a specific amount of resources (memory, cpu etc.) on a specific host. – Ref: Strata (**Rich Raposa**)

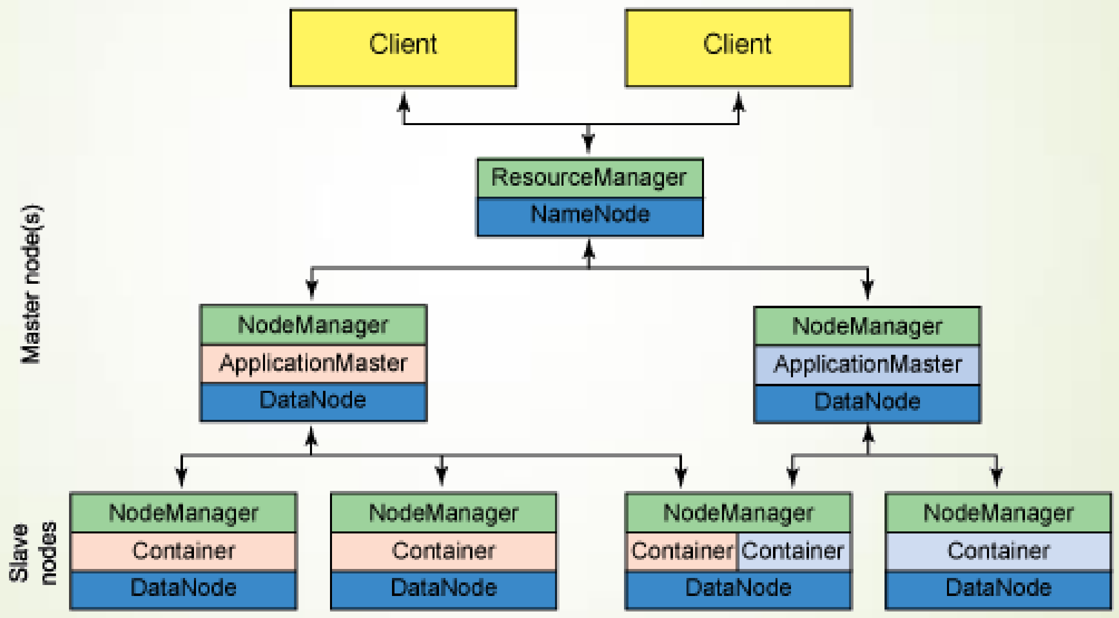
Yarn functions



MapReduce Master/slave Architecture (hadoop 1)



YARN Master/slave ARCHITECTURE (Hadoop 2)



References

- http://hadoop.apache.org/
- http://wiki.apache.org/hadoop/
- http://hadoop.apache.org/core/docs/current/hdfs_design.html
- http://wiki.apache.org/hadoop/FAQ