Reverse'em all

EECS 4413 ASSIGNMENT 1

Instructed by: Prof. Jack Jiang

Presented by: Zhongran (Julian) Deng

Sied Hoa (Heny) Tjin

Hashim Al-Helli

Randy Agyapong

David Iliaguiev

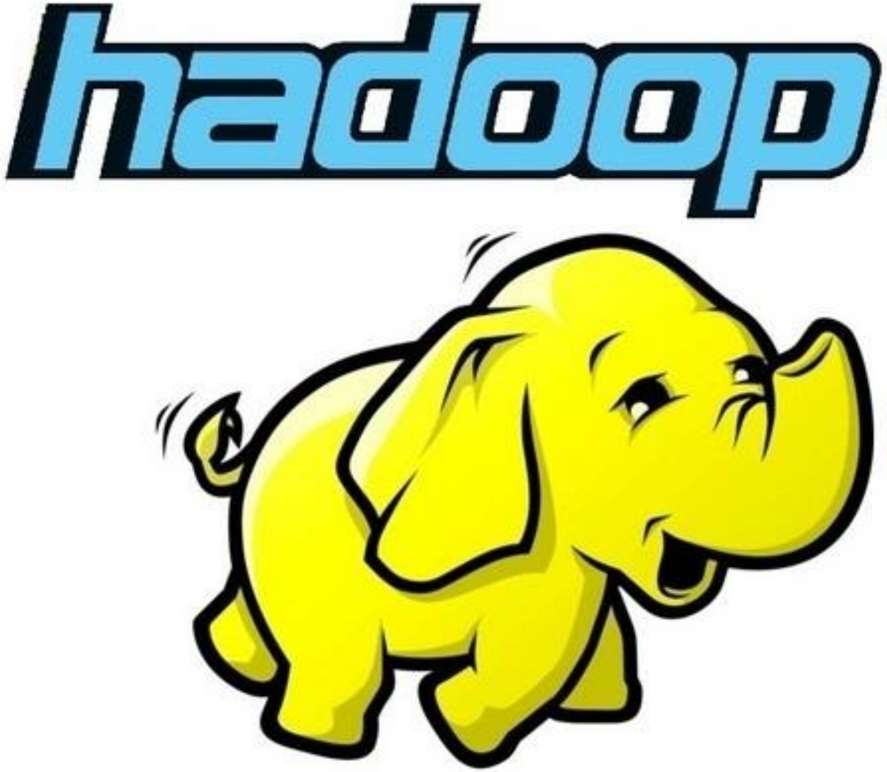
BIG DATA

- Internet generates data everyday at a petabyte scale.
- This includes structured data and unstructured data.
- Big data is a collection of large datasets that cannot be processed using traditional computing techniques.

WHAT IS HADOOP?

Feature:

- Open source
- Break through the limit of traditional database
- Support commodity computing
- High scalability



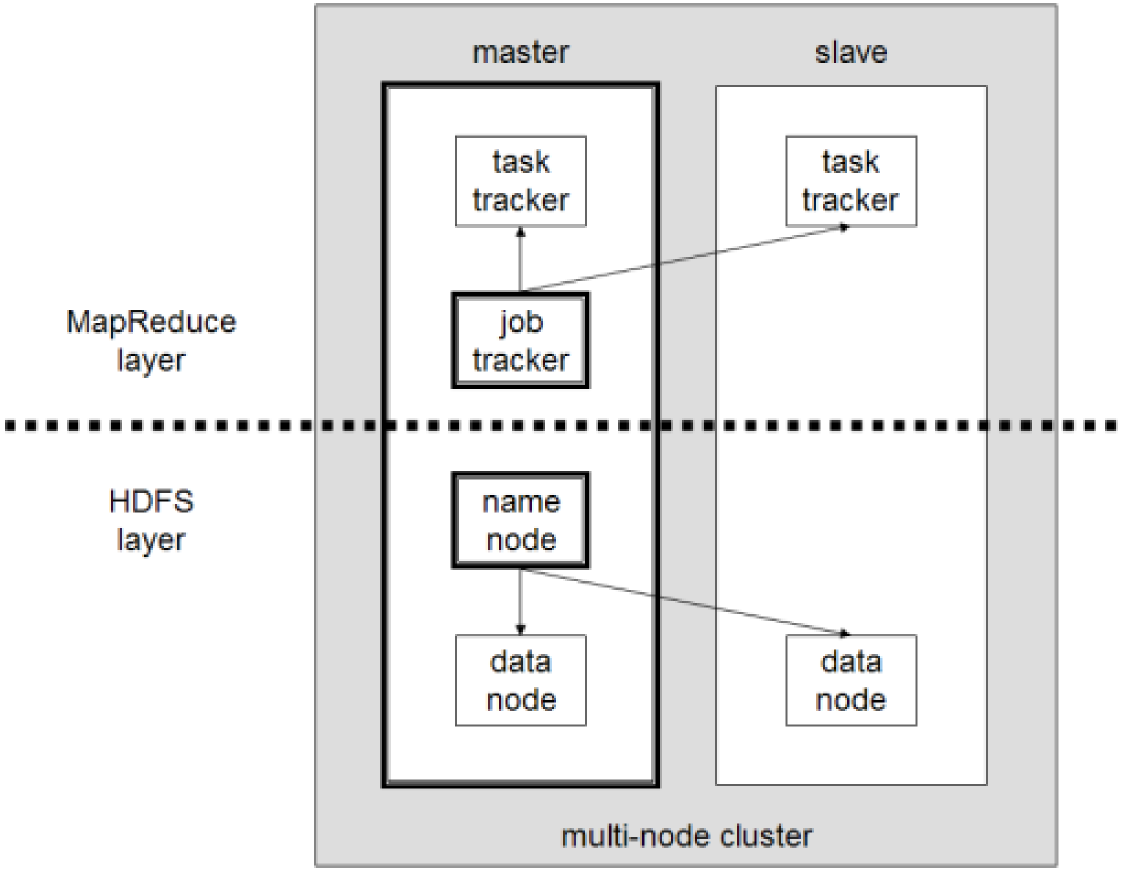
Hadoop is an open-source software framework to process and store big data.

TYPICAL ARCHITECTURE OF HADOOP

HDFS – distributed file system.

Map reduce – offline computing engine.

As versions updates, Hadoop is gaining more function modules such as Yarn which improves the performance of Hadoop.



Master-slave diagram

HDFS

A distributed file system for Hadoop

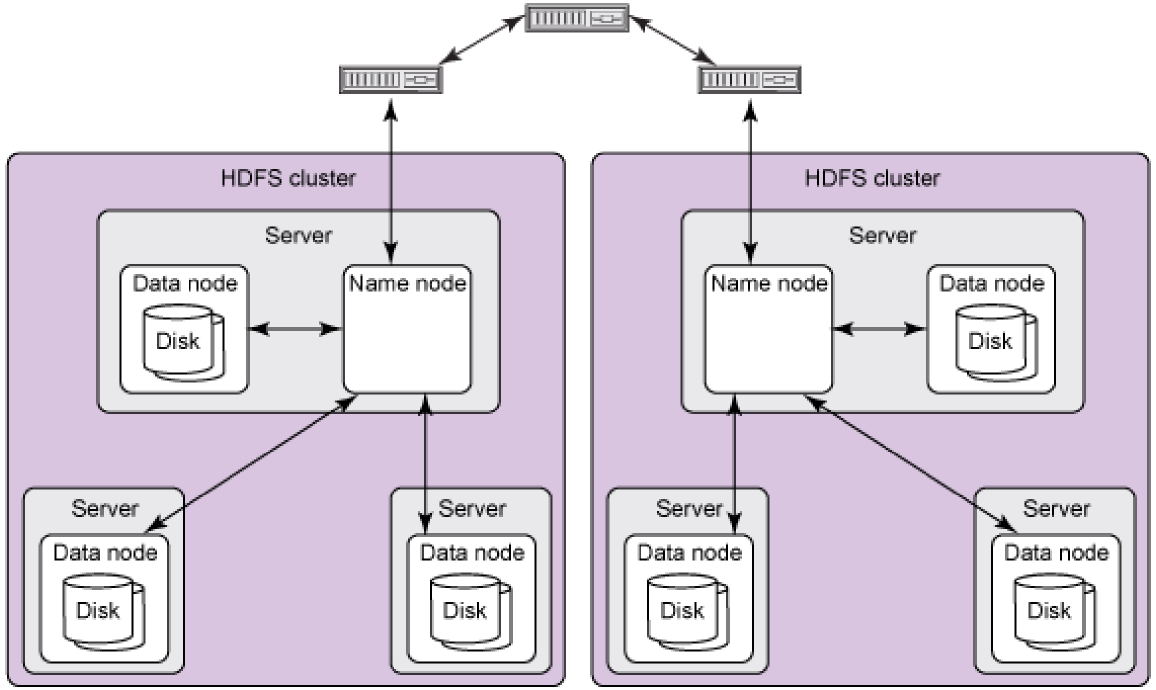
TRADITIONAL VS. HDFS

- Each block of data is small.
- Approximately 51 bytes.
- Huge amount of I/O operations while reading large data.

- Each block of data is large.
- By default 64MB and could be more.
- Reading large data sequentially after single seek operation.

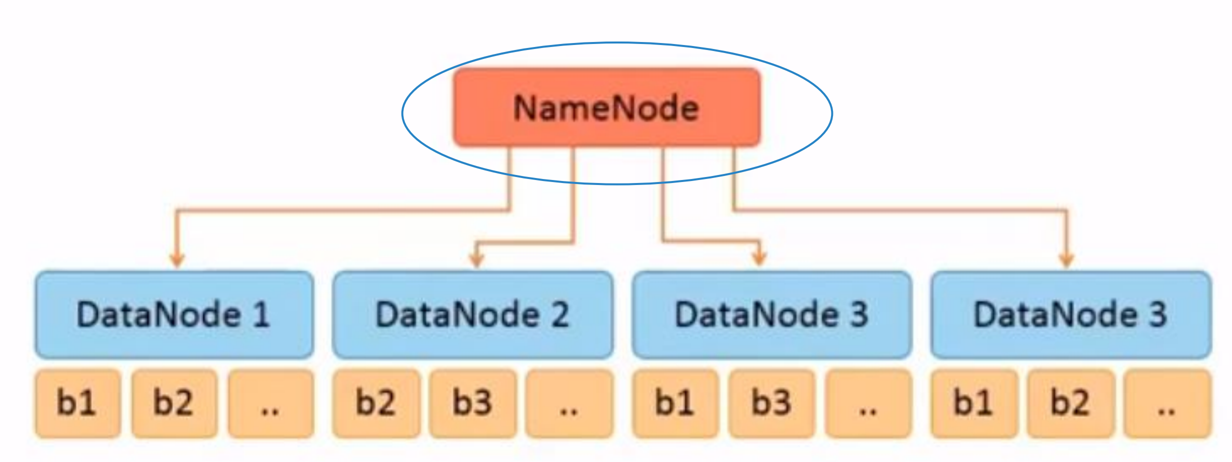
HDFS ARCHITECTURE -HADOOP 1.0

- A typical HDFS cluster contains a single name node plus a cluster of data nodes
- The server holding the name node is very crucial as there is only one existing per cluster.
- Data nodes belong to name node and it can be stored in different server.



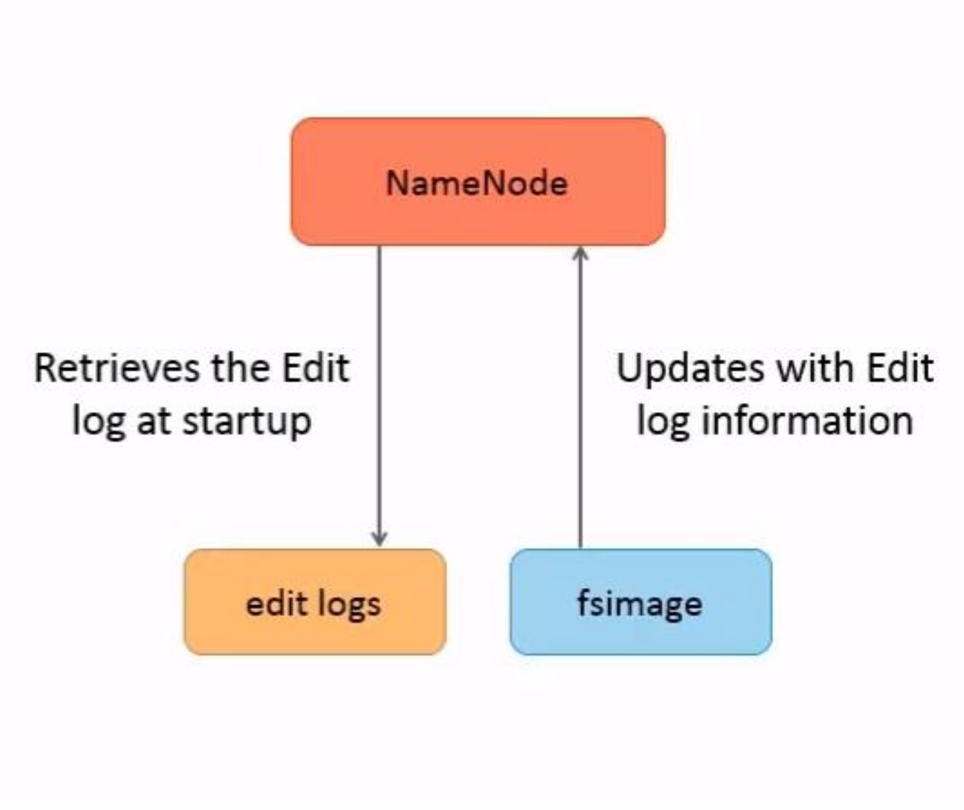
NAME NODE

- Stores metadata and information of the file system.
- Knows the data is stored in which certain data node.
- In early version, it was a single point of failure, that means, if the name node becomes unavailable ,the whole cluster goes down.



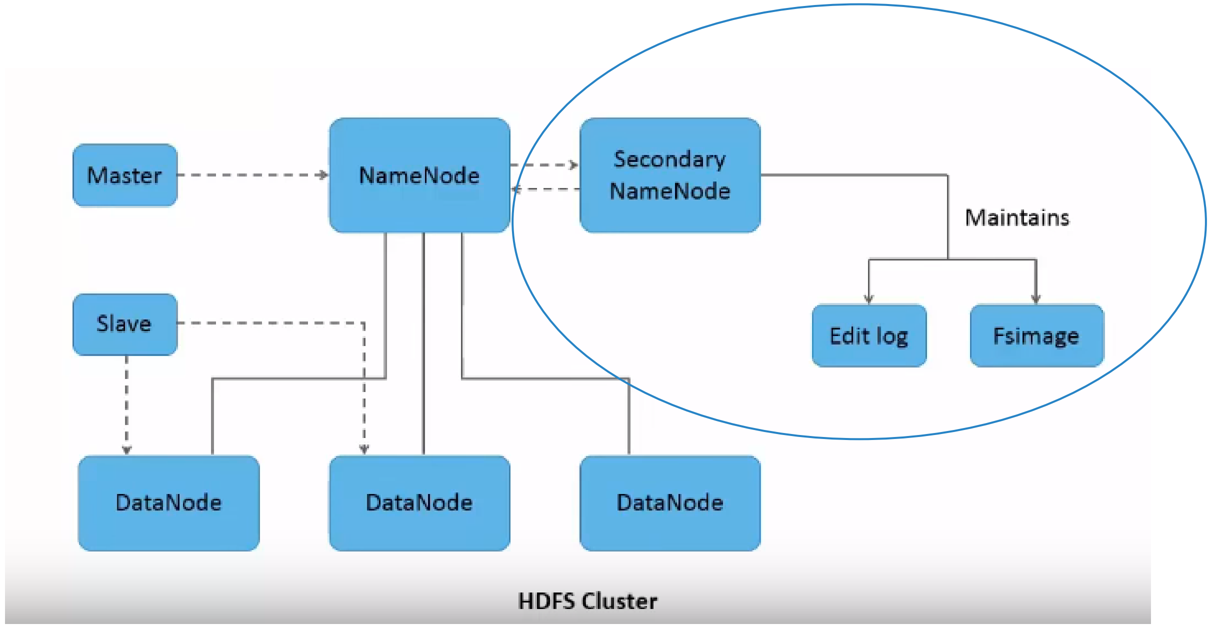
NAME NODE

- Name node holds two persistent files which are transaction log called edit logs and namespace image called fsImage.
- Edit logs records changes of metadata.
- Fsimage keeps the information of the entire file system namespace.


NameNode
Retrieves the Edit log at startup
Updates with Edit log information
edit logs
fsimage

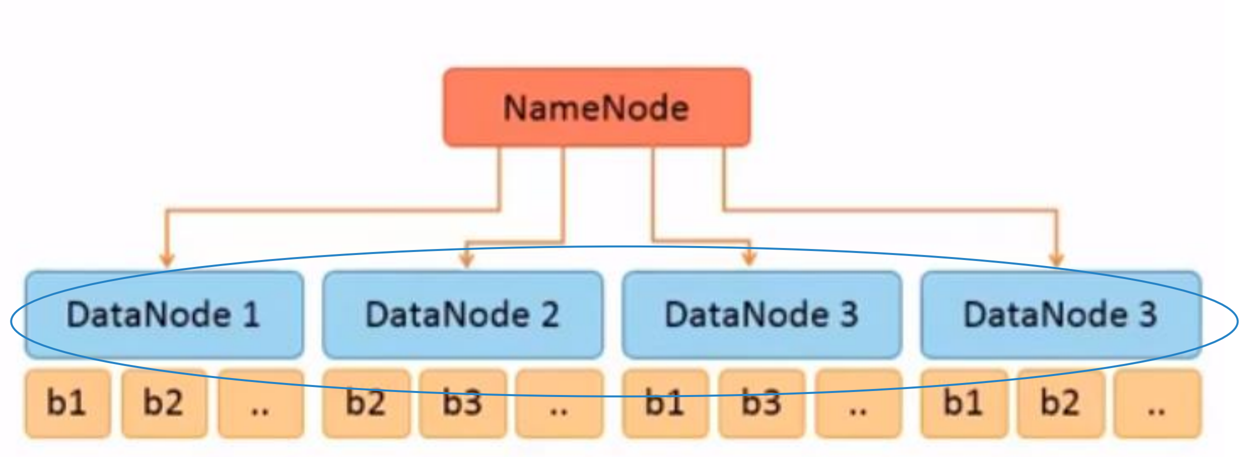
SECONDARY NAME NODE

- Not a backup daemon for name node. Can partially restore name node but can’t completely depend on it for disaster recovery.
- Maintains edit logs and fsimage.
- Helps to lower down name node’s restart time.



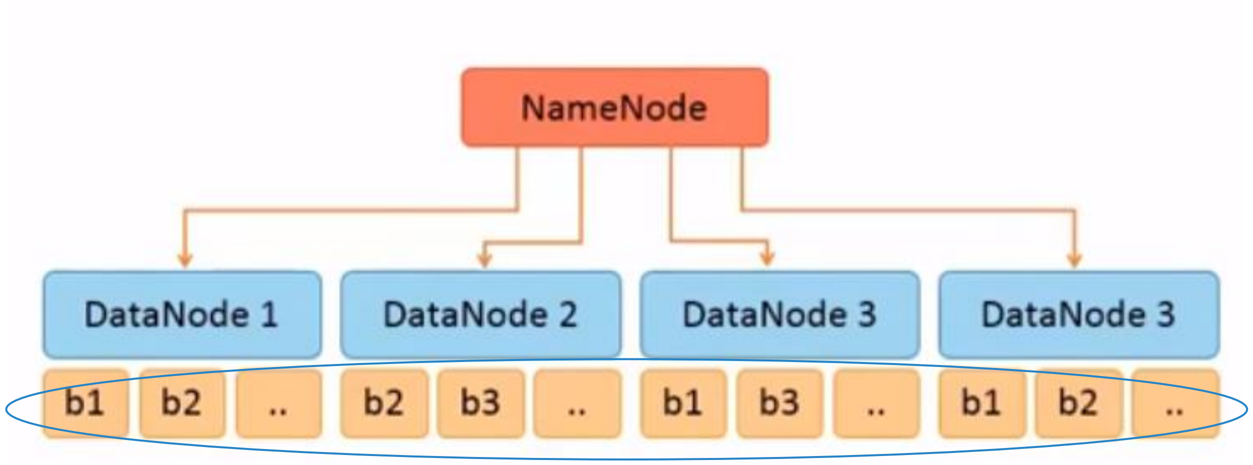
DATA NODES

- Stores and maintains data blocks.
- Responsible to store and retrieve data blocks upon request from name node or the client.
- There can be N numbers of data nodes depending on the system requirement.
- Perform operations such as read, write, block creation, deletion, replication, etc.



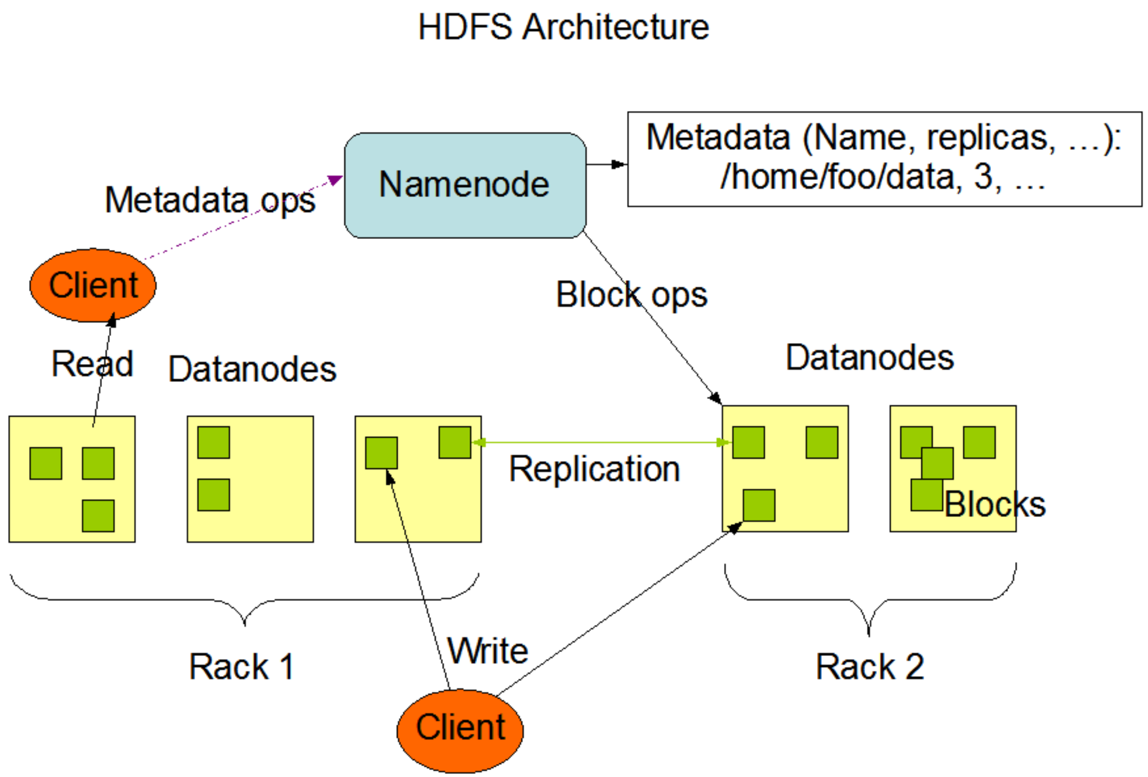
DATA BLOCK

- Each file is split into one or more blocks stored and replicated in data nodes.
- By default, each block is 64MB.
- The size can be increased to 128MB or bigger.



DATA BLOCK REPLICATIO N

- The data blocks are replicated for fault tolerance.
- The data blocks are distributed in data node system within the cluster and thus ensures the replica of data is maintained.
- Rack-awareness.



INTERACTION OF EACH COMPONENTS

- All HDFS communication protocols build on the TCP/IP protocol.
- Client communicate with the name node using a proprietary RPC (Remote Procedure Call)-based protocol.
- Each data node serves up blocks of data over the networking using block protocol specific to HDFS.

WEAKNESS OF HDFS IN HADOOP 1.0

- Single point of failure. Once name node is unavailable, the whole cluster goes down.

HDFS ARCHITECTURE- HADOOP 2.0+

- Running two name nodes.
- The back-up name node (or checkpoint node) is in the same cluster.
- Each name node is configured as Active/Passive. Only one name node is in Active state.

MAPREDUCE

A distributed processing framework for Hadoop

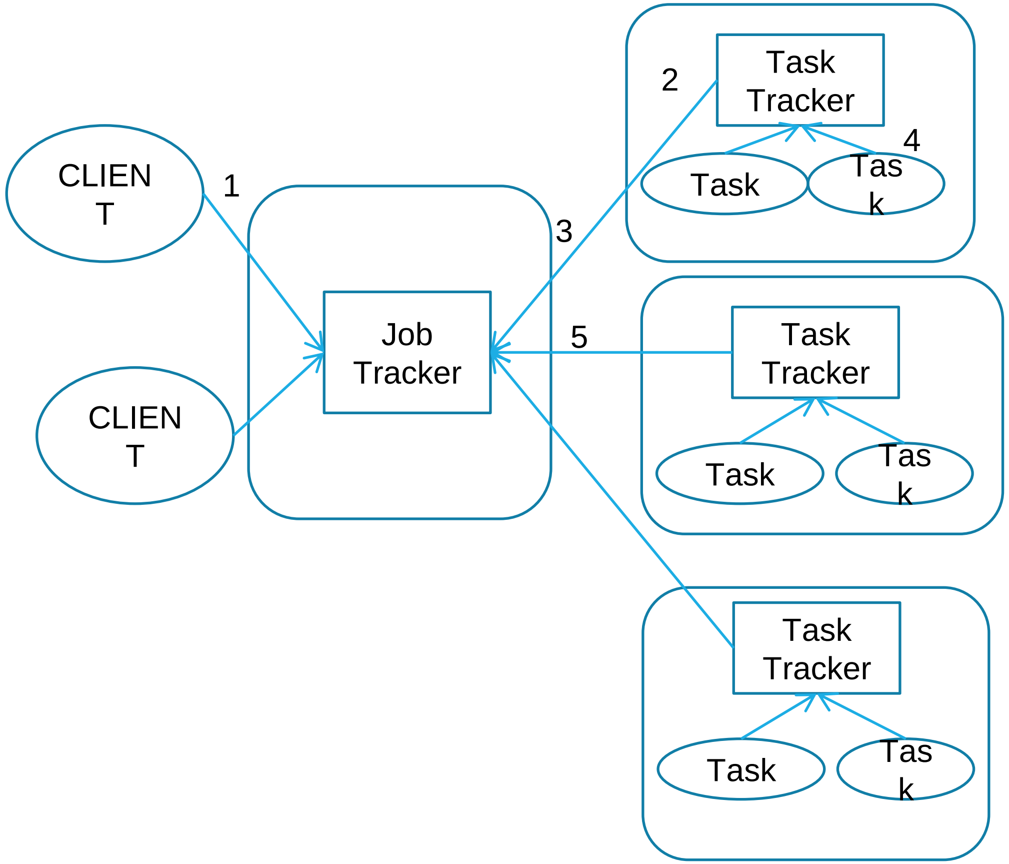
MAPREDUCE ENGINE IN HADOOP VER. 1

Client submit MapReduce Jobs to JobTracker.

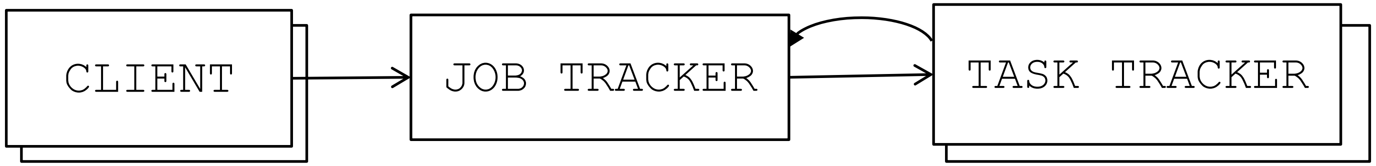
Job Tracker push jobs to TakTrackers

JobTracker report progress to Task Tracker.

Job Tracker support only Map jobs and Reduce Jobs

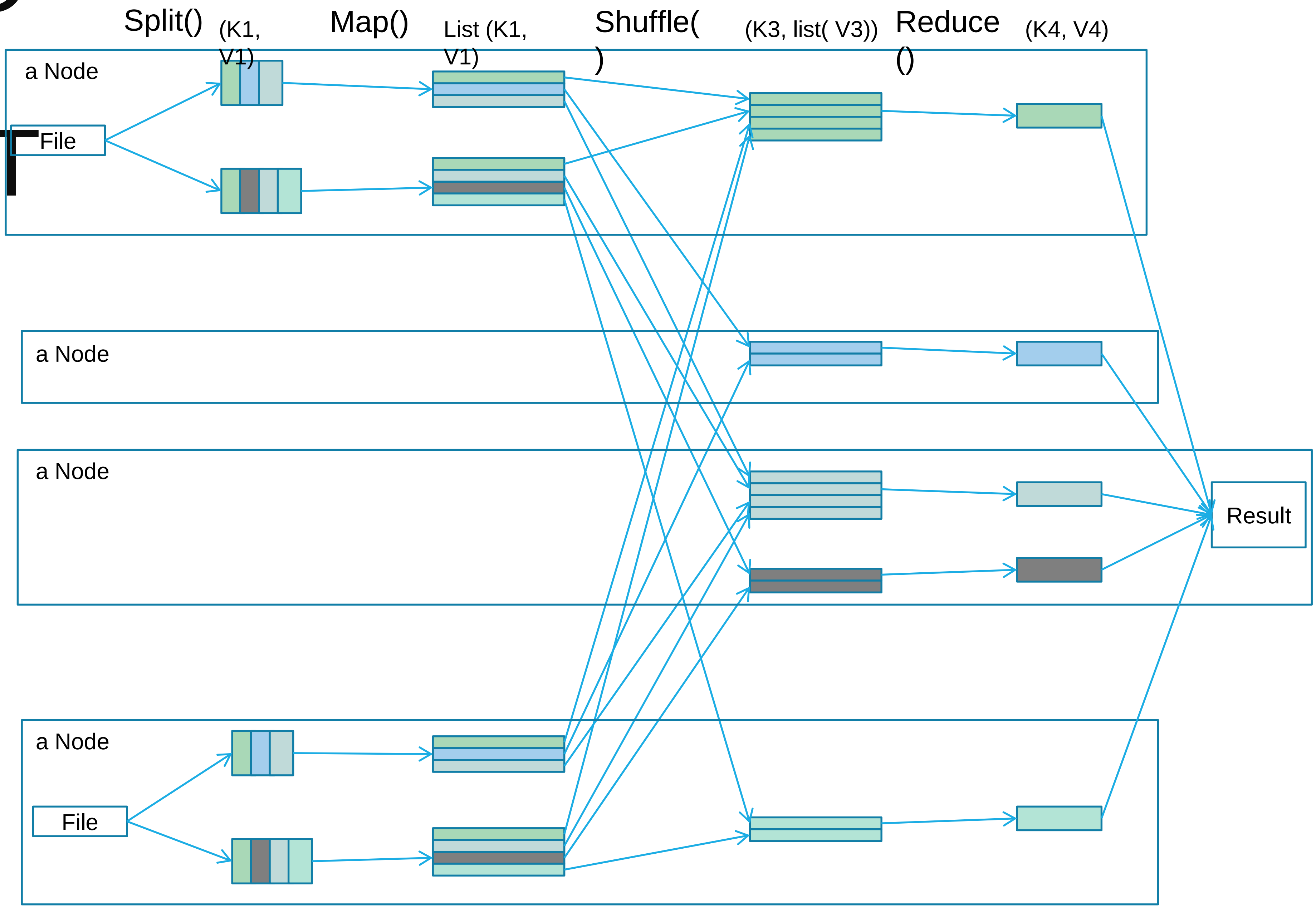


MAPREDUCE ENGINE IN HADOOP VER. 1



1. Client submit MapReduce Jobs.
2. Job Tracker will push work out (to available task-tracker nodes)
3. Every task tracker node will spawn JVM (Java Virtual Machine) process
4. At a set frequency, the task tracker will send signal to Job Tracker to indicate its ‘liveliness’

MAPREDUCE ALGORITHM



SHARE NOTHING ARCHITECTURE IN HADOOP MAPREDUCE

- Each node is independent of other nodes in the system
- No share resources that can become bottlenecks
- Lack of shared data: each node is processing distinct subset of data, hence no need to manage access to shared data

Advantages:

1. Easily manage workflow (associated with transparent process feature)
2. Scalable: No shared resources, hence addition of nodes adss resources to the system and does not add further contention. As input data increases, just need to apply more nodes (linear scalability)
3. Fault tolerant:
   a. each node/server is independent, hence no single points of failure
   b. Failed process in one node can be restarted on other node.
   c. The system can support multiple failures, depending number of data replication. (by default it is set to 3 replication)
   d. Hence hardware failure will only slow down process, but not entirely crash the whole job
   e. Inputs are immutable. Hence result can be recalculated

ISSUES WITH MAPREDUCE IN VERSION 1

- Scalability:
  - Max cluster size is fixed (4000 nodes)
  - Max concurrent tasks is fixed (40,000 processes)
  - Coarse synchronization in Job Tracker limited scalability
- Availability: If job tracker fails, all queued and running jobs are killed
- Resource Utilization: Fixed/Static allocation of resources for map and reduce process results in low resource utilization
- Ability to support for Alternate Programming paradigms and Services:
  - Iterative application is not efficient in MR (10 times slower)
  - Non-MapReduce Applications are needed

YARN

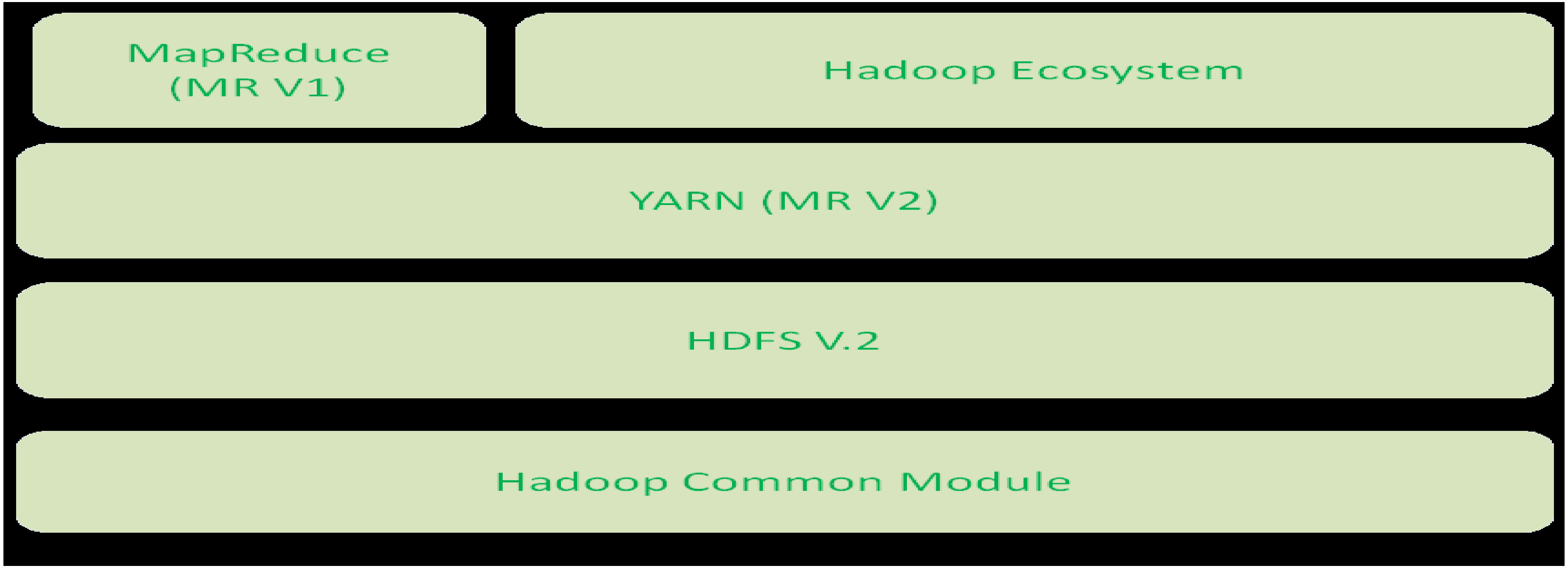
A resource management platform to improve performance

HADOOP 2.0- YARN

- **YARN framework and Next Generation MapReduce (MRv2):** YARN provides better resource management in Hadoop, resulting in improved cluster efficiency and application performance. It’s a pure scheduler.

- **MapReduce Becomes User library or one of the application that runs inside YARN for cluster services.**

HADOOP 2 ARCHITECTURE



MapReduce (MR V1)

Hadoop Ecosystem

YARN (MR V2)

HDFS V.2

Hadoop Common Module

- The introduction of YARN as an alternative to MapReduce is a huge improvement to the product.
- HDFS was also modified to overcome problems of release 1.0

YARN (YET ANOTHER RESOURCE NEGOTIATOR)

- Provides all resources and resource management for the cluster.

- YARN separates MapReduce's job tracker and task tracker capabilities into separate entities, enabling Hadoop to support more varied processing approaches and a broader array of applications.

YARN COMPONENTS

- Resource Manager
- Application Master
- Node Manager

RESOURCE MANAGER

- Resource manager and node manager form the basis for managing applications in distributed manner

- the responsibility of the resource manager is to distribute the available resources to the applications using its scheduler, the scheduler performs the allocations according to constraints which can be user limits or queue capacities limits

- the scheduling is done based on the resource requirements of the applications

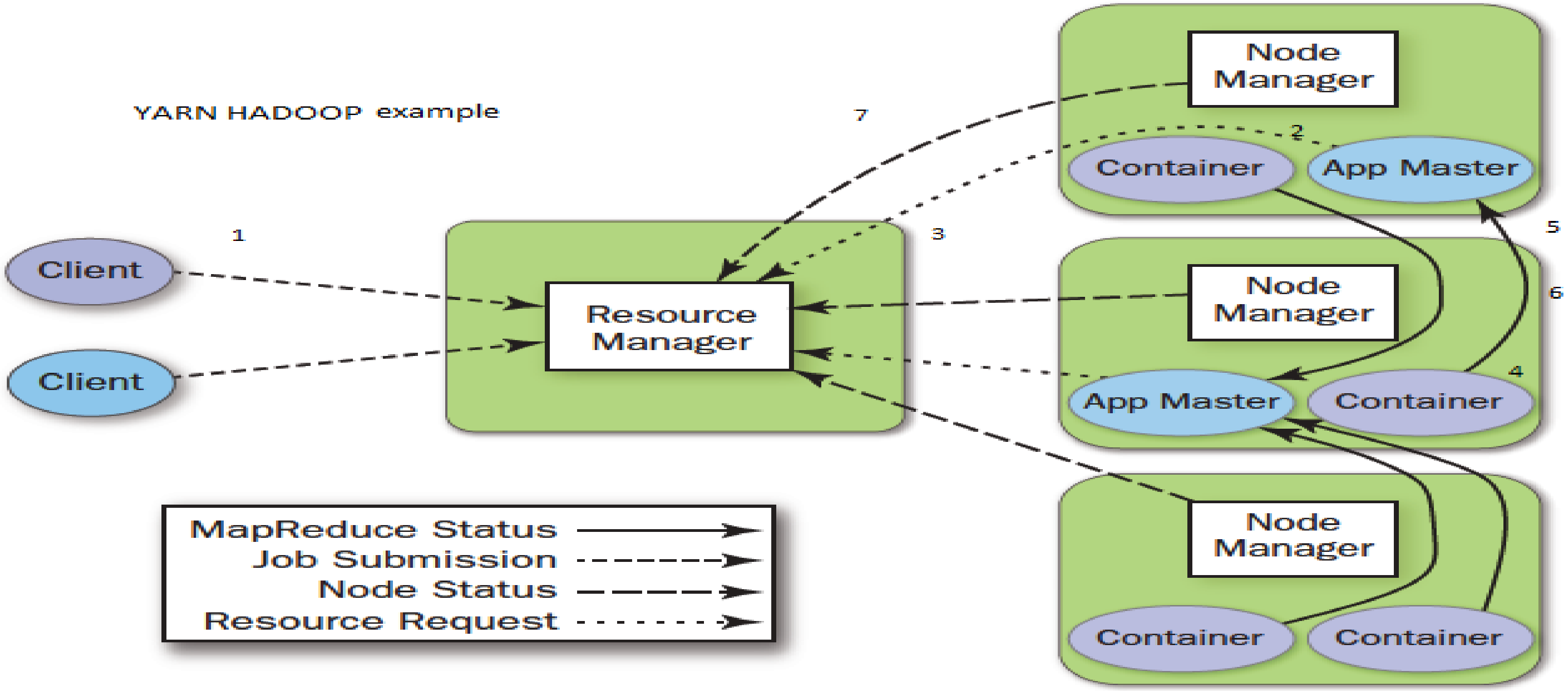
APPLICATION MASTER

- it negotiates resources from resource manager and works with node managers to execute and monitors the components tasks.

NODE MANAGER

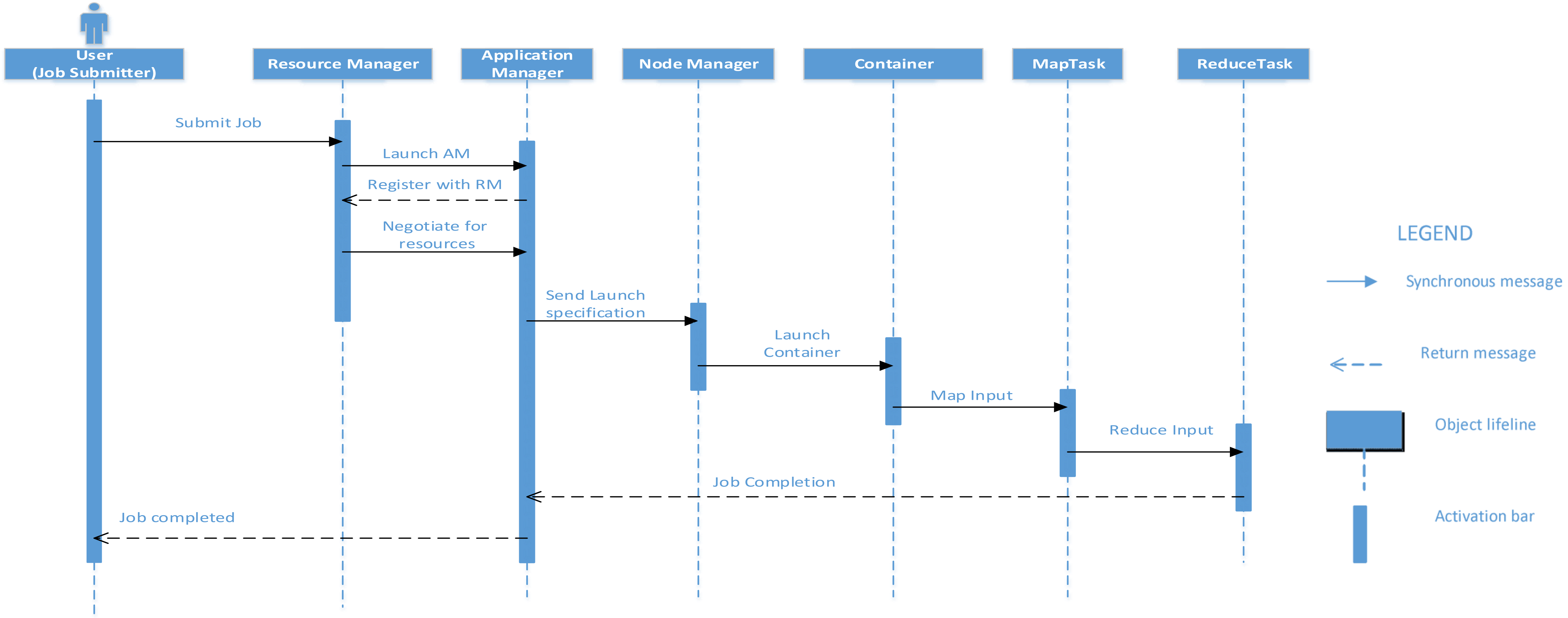
- monitors resource usage of cpu, memory, disk, and network and report back to the resource manager

SIMPLE EXAMPLE OF YARN WITH DIAGRAM

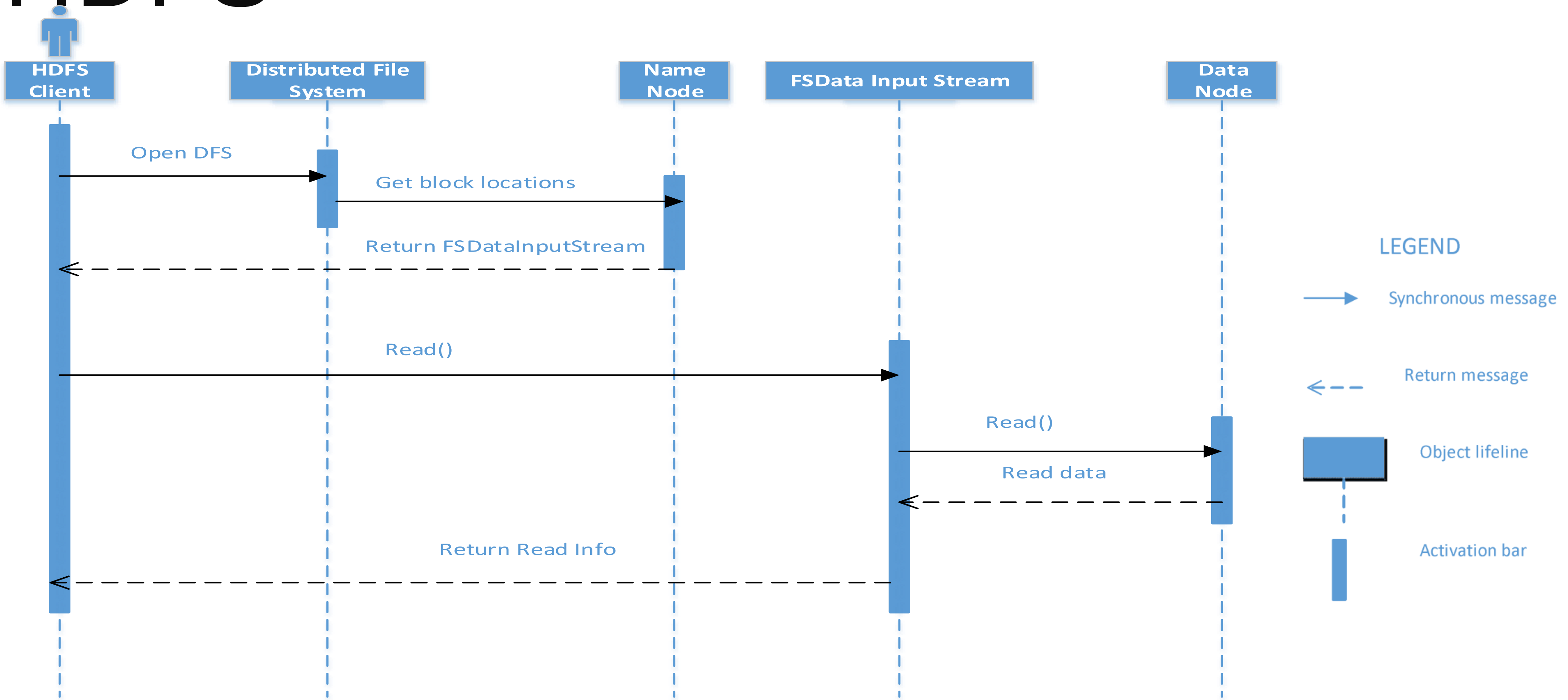


SEQUENCE DIAGRAM

SEQUENCE DIAGRAM OF YARN AND MAPREDUCE



SEQUENCE DIAGRAM OF HDFS



HADOOP ECOSYSTEM

APACHE HADOOP APP DEVELOPMENT

Hadoop has many ways for developers to create apps to interact with the system:

- HTTP Browser
- Client Local File System
- FS Shell
- Java applications
- C applications

ACCESS

- Developers can create websites for users to browse files of HDFS instances with an HTTP browser
- Using NFS gateway HDF can be mounted as part of the client's local file system
- FS Shell is provided so that clients can access the HDFS through commands, this also allows for scripting languages to be used (i.e. Python)

DEVELOPMENT

Hadoop provides various Java APIs to interact with the system

- HDFS provides a FileSystem Java API
- Master REST API for MapReduce
  - Jobs API and Tasks API
- YARN web services REST API
- Hadoop Auth API
- CredentialProvider API

C language wrapper for Java and REST API is also available

IMPLICATIONS FOR DEVELOPERS

Use of OO Language like Java to create applications for use with Hadoop allows for:

- Modularity
- Maintainability
- Use of OODP(Object-Oriented Design Patterns)

The Layers of Hadoop (HDFS, MapReduce, YARN) allows for a division of resposibilities

LAYERS OF HADOOP

Developers can distribute work amongst themselves to operate independently and combine to create stable and robust software, groups focused on:

- Access to HDFS can combine modules developed using the FileSystem Java API
- Security can develop modules using Hadoop Auth API and CredentialProvider API
- Status of application, jobs and tasks can develop modules using MapReduce REST APIs, they can also,
- Split up resource management and job/task scheduling using the YARN REST APIs

HADOOP ECOSYSTEM

- Core components
- Hadoop Database
- MapReduce Query Tools
- Data import and export
- Workflow automation
- Administration
- YARN Application FrameWorks
- Other

HADOOP ECOSYSTEM- CORE COMPONENTS

Core components

○HDFS

○YARN

○MAPREDUCE

HADOOP ECOSYSTEM- HADOOP DATABASE

- Apache HCatalog : table and storage management service for data created using Hadoop. The table abstraction removes the need for user to know where data is stored.
- Apache Hbase: is Hadoop database, designed for hosting large tables with billions of rows and columns, its non-relational database

HADOOP ECOSYSTEM- MAPREDUCE QUERY TOOLS

Apache Pig: high level language enables programmer to write complex MapReduce transformation using simple scripts.

Apache Hive: provides database query interface to Hadoop

HADOOP ECOSYSTEM- DATA IMPORT AND EXPORT

Apache sqoop: tools for transferring large data between HDFS and relational database

Apache Flume: service for collecting, aggregating, and moving large amount of log data.

Apache Avro: is serialization format that makes data exchange possible between programs written in any language.

HADOOP ECOSYSTEM- WORKFLOW AUTOMATION

Apache Oozie: service for scheduling apache Hadoop jobs.

Apache Falcon: enables automation of data movement and processing for replication operations and some others. Falcon triggers a job start when data changes or new data becomes available.

HADOOP ECOSYSTEM-ADMINISTRATION

Apache Ambari: web based tool for managing and monitoring Apache Hadoop Clusters using GUI (Graphical User Interface).

HADOOP ECOSYSTEM- YARN APPLICATION FRAMEWORKS

Applications written specifically for the YARN environment. Like MapReduce, Apache Giraph (for graph processing), Spark (for memory processing), and others.

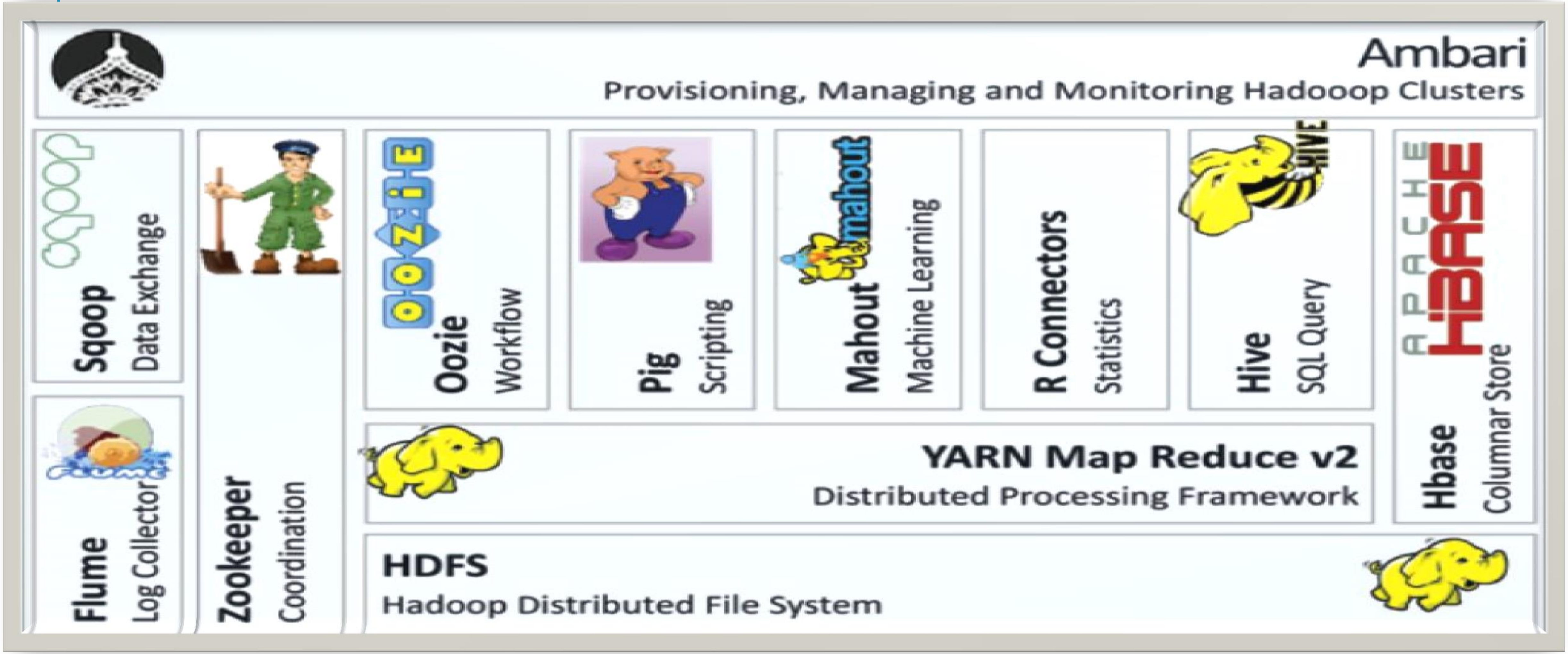
HADOOP ECOSYSTEM-OTHERS

Apache ZooKeeper

1. service for maintaining configuration, health, and status elements on and between the nodes.
2. It maintains common objects needed in the cluster environment such as configurations information, naming space and so on.
3. Provides application reliability if one application master dies zookeeper will spawn new application master to resume the tasks.

Apache Mahout: machine learning library implements many approaches to machine learning.

HADOOP ECOSYSTEM



Ambari
Provisioning, Managing and Monitoring Hadooop Clusters
sqoop
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
OOZIE
Oozie
Workflow
Pig
Scripting
mahout
Mahout
Machine Learning
R Connectors
Statistics
HIVE
Hive
SQL Query
APACHE HBASE
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System