Hadoop: Conceptual Architecture

Team Space Jam

Introduction

- Open source software
- Designed to store and process large amounts of data
- Delegates tasks across servers allowing many devices to run concurrently
- Several main parts
    - HDFS
    - MapReduce
    - YARN

HDFS (Hadoop Distributed File System)

- NameNode: Runs on a master node that tracks and directs storage of the cluster
- DataNode: Runs on slave nodes, which are the majority of the machines in a cluster
- Client Machine: They load data into the cluster, submit MapReduce jobs and view the results of the job once complete.

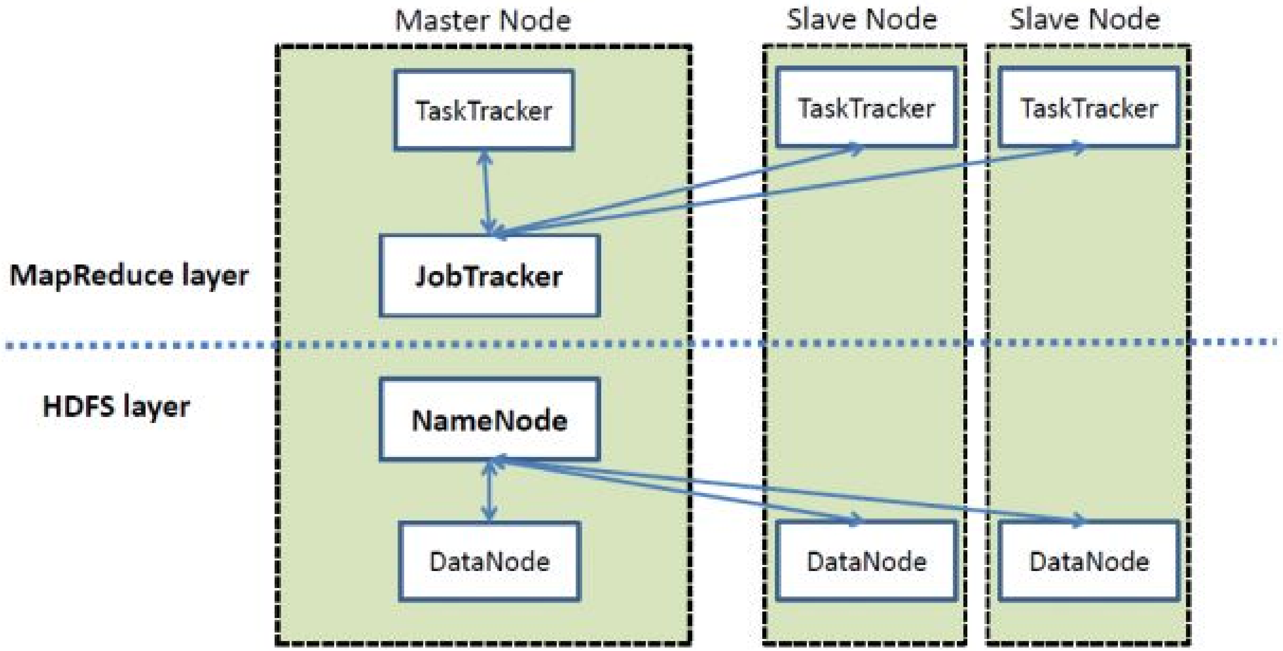
MapReduce

- Process the large amount of data Hadoop stores in HDFS
- Once a task has been created it is spread across multiple nodes and run simultaneously
- Delegation of tasks is handled by two daemons
  - JobTracker
    - Oversees how MapReduce jobs are split up into tasks and divided among nodes within the cluster
  - TaskTracker
    - The TaskTracker accepts tasks from the JobTracker, performs the work and alerts the JobTracker once it is done. TaskTrackers and DataNodes are located on the same nodes to improve performance

YARN (Yet Another Resource Negotiator)

- Updated way of handling the delegation of resources of MapReduce jobs
- Takes the place of JobTracker and TaskTracker
- Introduced in Hadoop 2.0

High Level Architecture of Hadoop

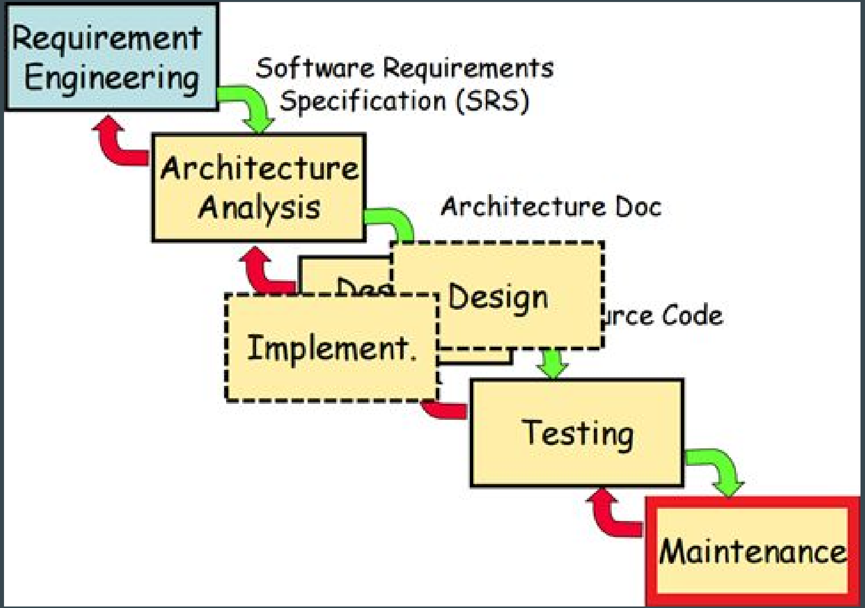


How does the system evolve

The system evolves just like any other – updates

- Observe change logs, update notes
- Users and product testers submit bugs
- Bugs are addressed, features are added

  Competitors have their own file system (Microsoft, Google)
  - Borrow ideas, improve on their weak points



From "Introduction Overview" in class

Notable Version 1.0 Features

- File Security
- Disk-Failure procedures
- Hbase
- Real time read/write access
- Hadoop-client, hadoop-minicluster
  - Client installation, and testing
- ‘run-as-user’ ‘non-secure’ mode support

Hadoop Version 2.0

- YARN aka ‘next gen MapReduce’
  - The core that allows multiple data processing
  - SQL real-time streaming, batch processing
- HDFS Federation
  - Improves on the original HDFS
  - Separates namespace and storage, multiple namespaces
- Multi-resource scheduling
- HDFS snapshots
- Microsoft Windows support
- OfflineImageViewer

Hadoop Version 3.0

- The latest version to be released to the public
- Alpha phases come before Beta, generally not a stable release (stick to 2.X for reliability)
- Alpha phases are useful for gathering as much user input as possible to address bugs for next release
- “The intention is the “release early, release often” to quickly iterate on feedback collected from downstream users”

What Will the Future Hold

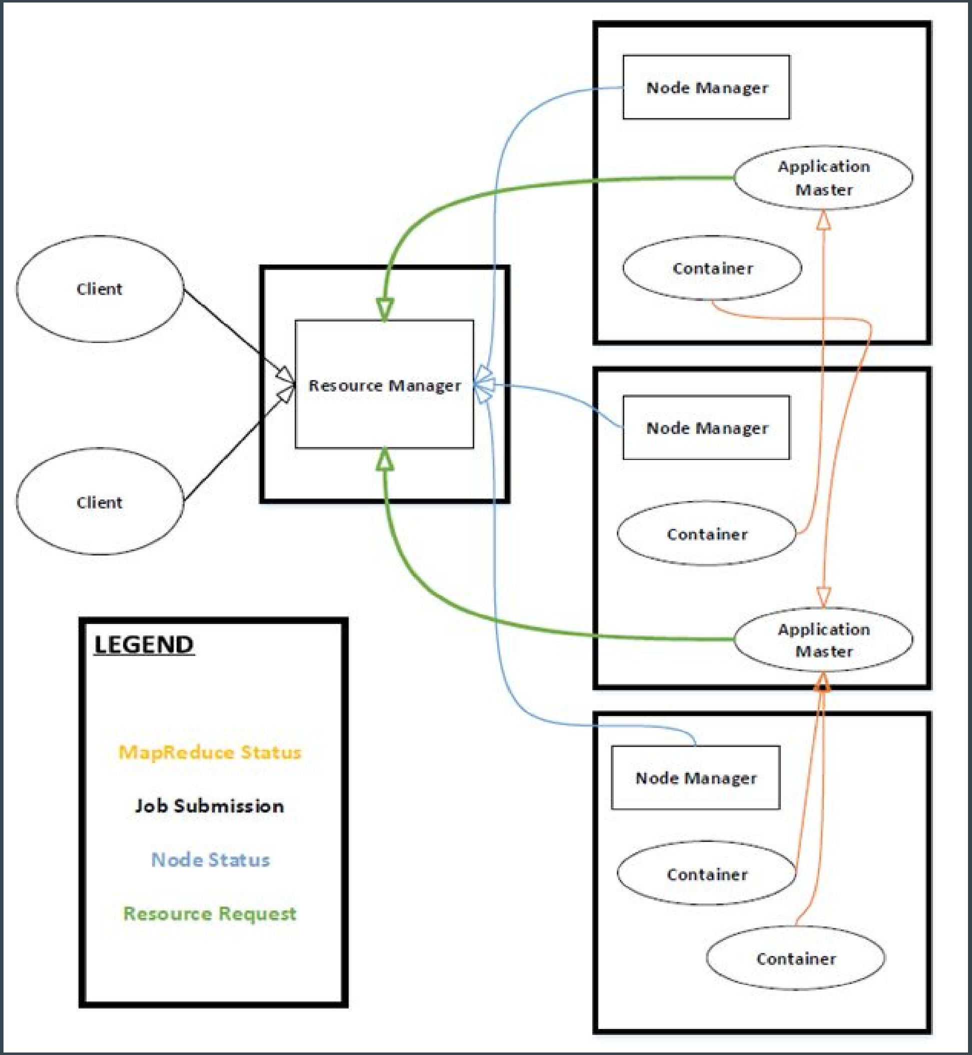
- The information world is expanding (well except physical chips)
  - Data clusters, websites, everything!
  - Always a challenge to handle larger clusters faster
  - Flexible data storage
    - More convenient for users
    - “Stuff everything into a big box and let us deal with it”
- Other search engines pose competition
  - Google, Microsoft’s Bing
- Hadoop’s humble beginning as a simple search engine might someday become the next big system to index, store, search, and sort through large data

Data Flow & Control

Moving Computation is Cheaper than Moving Data

- YARN
- HDFS
- MapReduce

Sequence of Events **YARN**



ResourceManager:
Schedule Resources

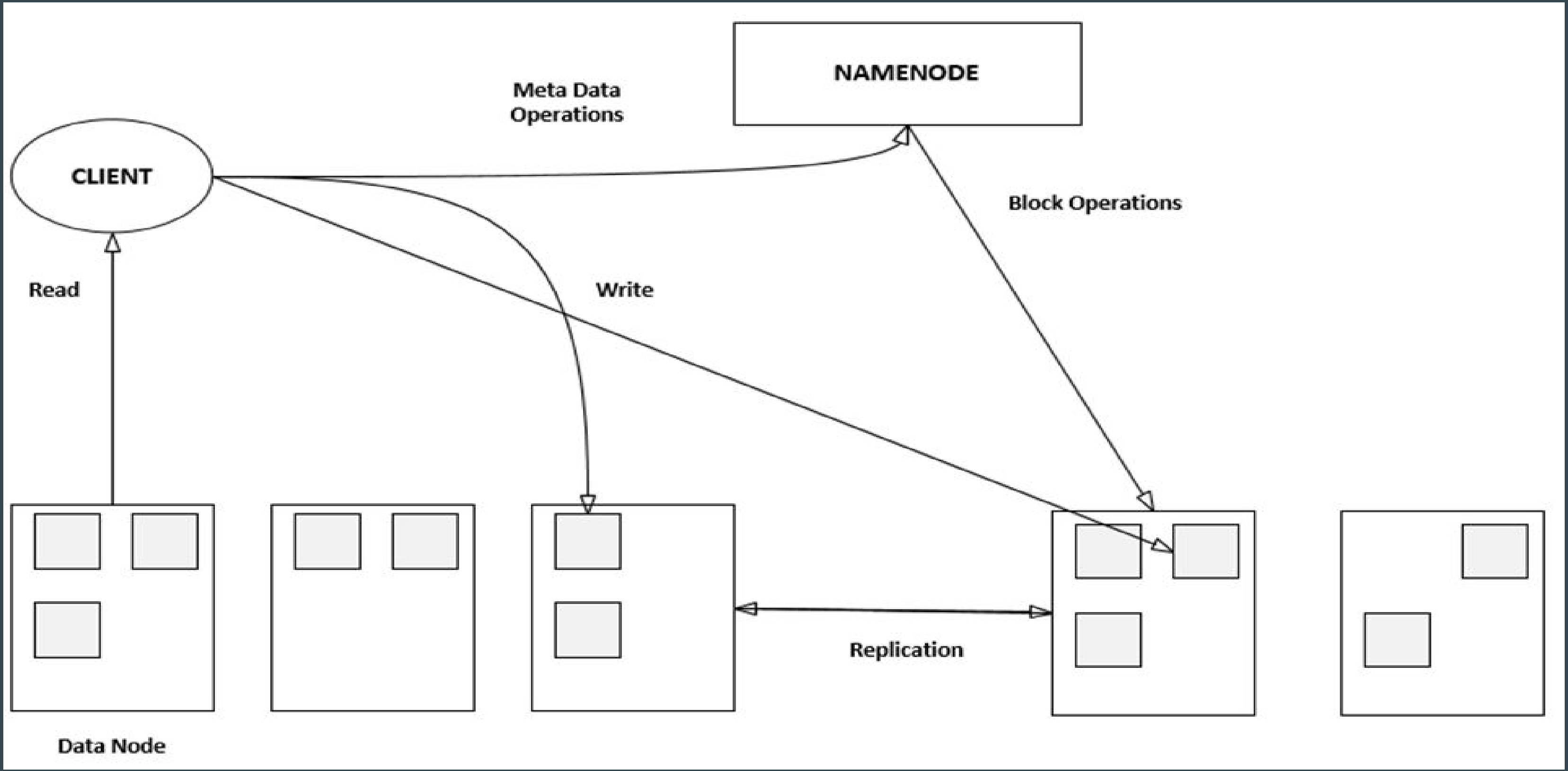
NodeManager:
Monitor Resource Usage & Launch Containers

AppMaster:
Responsible for Negotiating Allocation of Resource Containers

High Level Architecture
HDFS

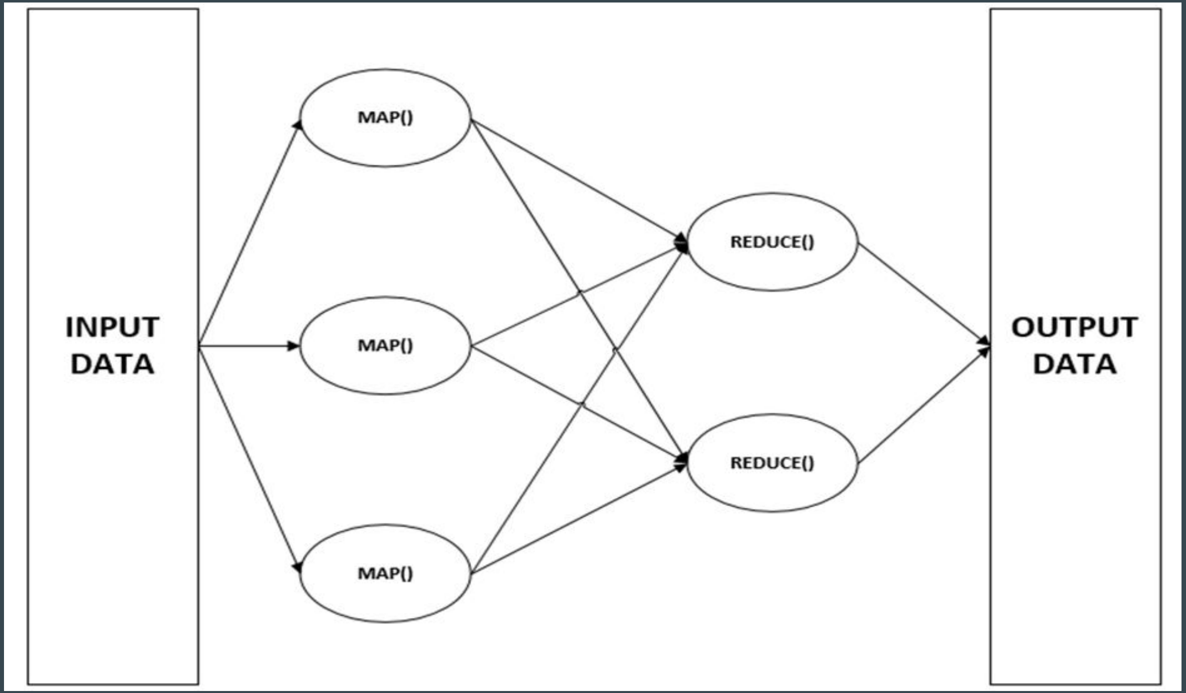
- NameNode is failure tolerant as a result of the implementation of data block replication as the practice of representing particular data blocks in one or more nodes.
- Users access the filesystem using the Client
- When an application reads a file, the Client asks NameNode for a list of DataNodes that hosts replicas of the blocks of the file.
- The DataNode is contacted directly by the Client and the transfer of the desired block is requested.
- When the Client receives a writing request, it asks the NameNode to choose a DataNode to host replicas of the block.
- A pipeline is then created from node-to-node and the data is sent. Once the block is full, the process continues through to the next block.

High Level Architecture **HDFS**



Overview of **MapReduce**

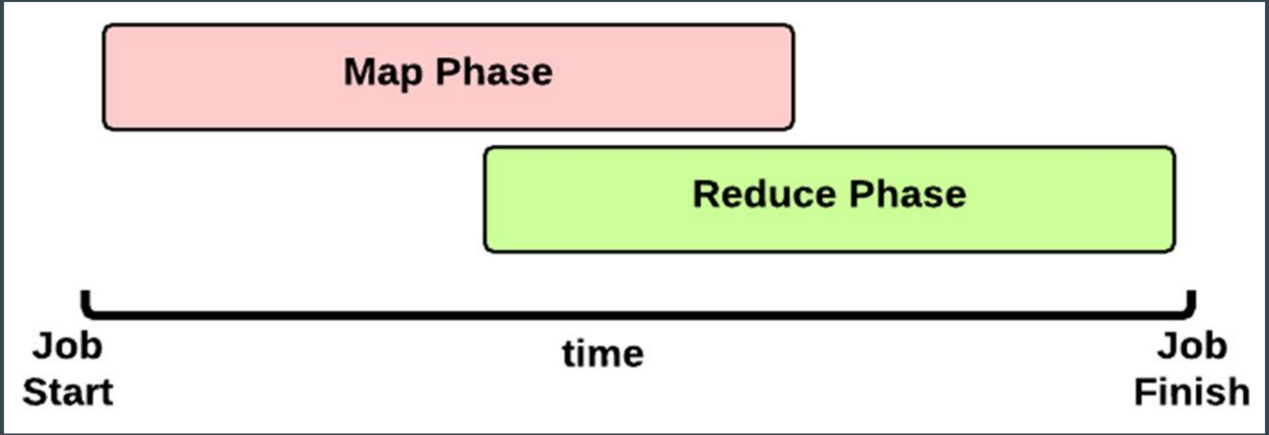
- A job in MapReduce splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner.
- The output of the maps is sorted and used as input for the reduce task.



Concurrency

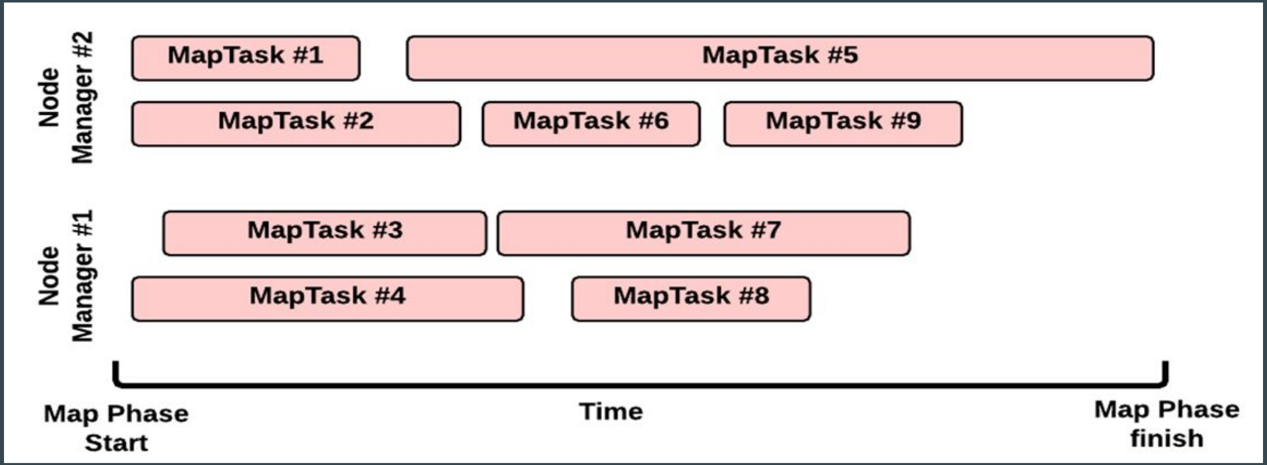
- Hadoop Map/Reduce framework uses a master/slave architecture
- Single master jobtracker which orders multiple slave tasktrackers allowing parallel execution of Map/Reduce tasks
- Provides ability to handle large amounts of data
- Hadoop Distributed File System allows storage of the large data sets

Concurrency



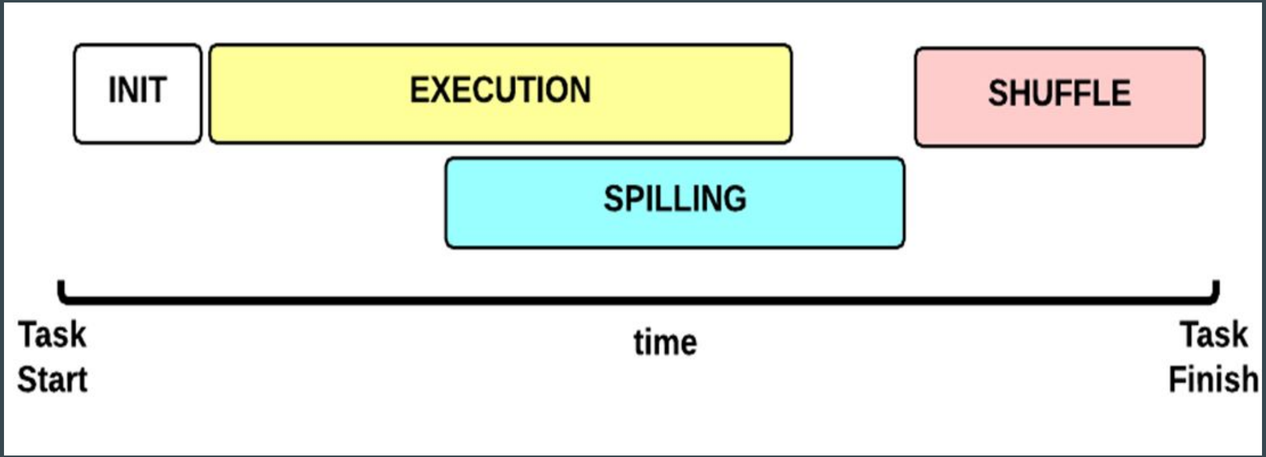
- Map phase pipes intermediate data to reduce phase
- Map Phase overlaps Reduce Phase
- Able to split input data while reducing mapped data

Concurrency



- Multiple node managers
- Multiple tasks running at once per node manager
- Amount of concurrent tasks depends on node manager capacity

Concurrency



- Map Task execution
- As in-memory buffer fills, spilling phase commences
- Final shuffle phase occurs when execution and spilling are completed

What are the implications for division of responsibilities among participating developers?

Roles and Responsibility

- Users – Use and provide feedback on the product.
- Contributors – Volunteers contributing time, code, documentation, or other.
- Committers – Responsible for technical management.
- Release Manager – A committer who produces a Release Candidate

Project Management Committee

- Decides and approves what products are released.
- Resource Maintenance. (code, mailing list, websites)
- Represent the project.
- Resolve license disputes.
- Voting for new members.
- Maintaining project guidelines and bylaws.

Decision Making

•Voting - Performed through the project development mailing list. Committers mail the following based on their decisions.

–+1 Yes

–-1 No (Counts as veto)

–+0 Yes, but cannot help

–-0 No, but not concerned enough to prevent it

Actions

- Change of Code
- Product Release
- Replacement of Codebase
- New Branch Committer, Committer, PMC member.
- Removal of Branch Committer, Committer, PMC member
- Bylaw Modification

Approvals

- Consensus Approval – Three +1 and no vetoes
- Lazy Consensus – No -1’s
- Lazy Majority – Three +1 and more +1 then -1
- Lazy 2/3 Majority – Three +1 and Two times as many +1 as -1

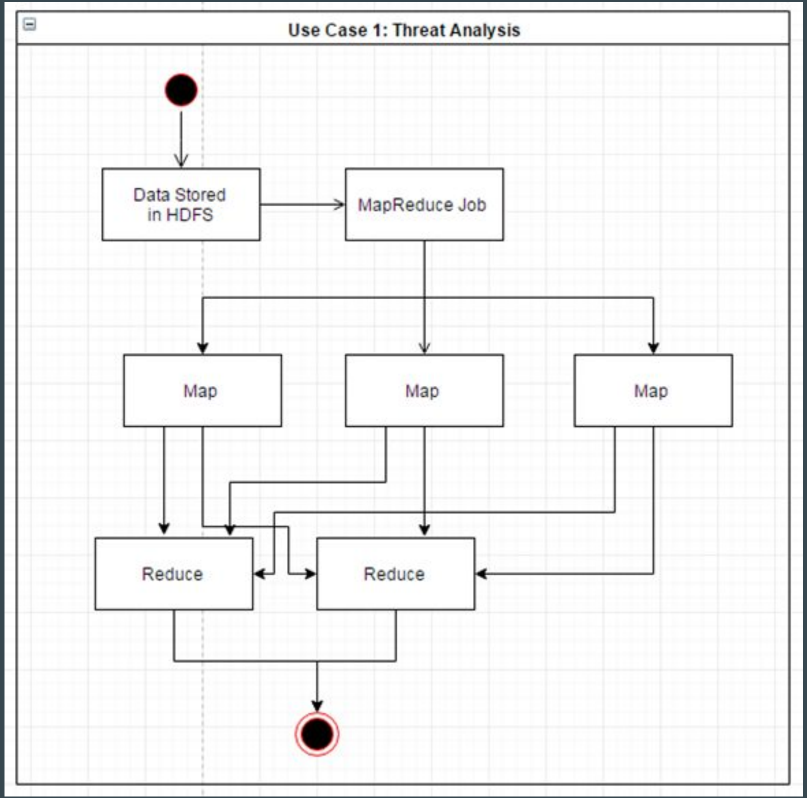
Use Case 1: Threat Analysis Scenario

- Online criminals write malicious viruses.
- People rely on software services to protect them from viruses/malware.
- Thus, an antivirus software is created to protect users from such threats.
- The antivirus software computes a signature for each virus it is aware of.

Use Case 1: Threat Analysis Scenario

- Virus/Malware signature database grew enormously over the past few decades.
- Antivirus company must now compare possible threats to the huge database of malware/virus signatures.

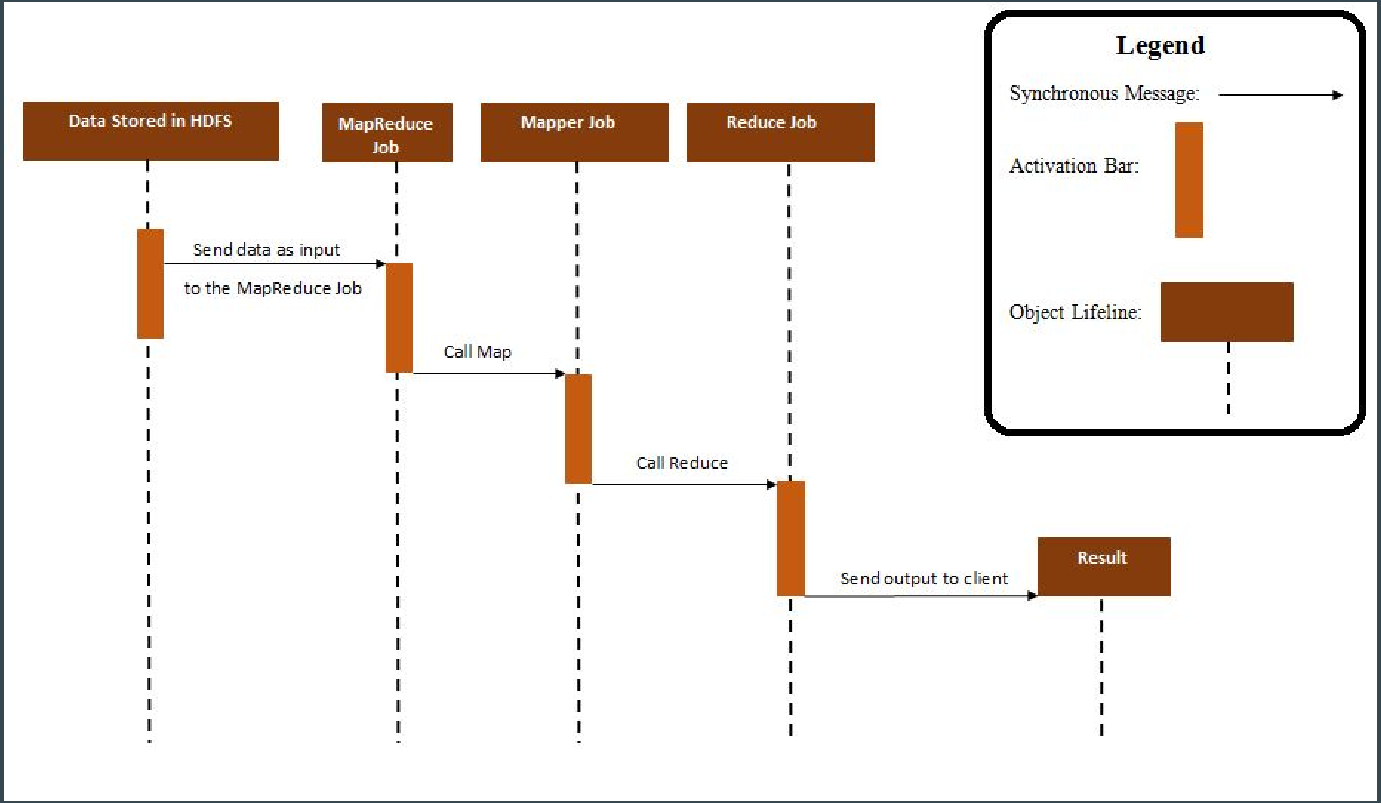
Hadoop Solution: State Diagram



Use Case 2: Search Quality Scenario

- Search Engines must return relevant information to its user.
- As time passes on, massive amounts of information is being added; making it much more difficult when processing the data.
- The search engine must be able to process files and extract keywords.
- Previous user activity and personal information must also be taken into account.

Hadoop Solution: Sequence Diagram



Conclusion

Questions or Concerns