HADOOP – HDFS AND YARN

Prasad M Deshpande

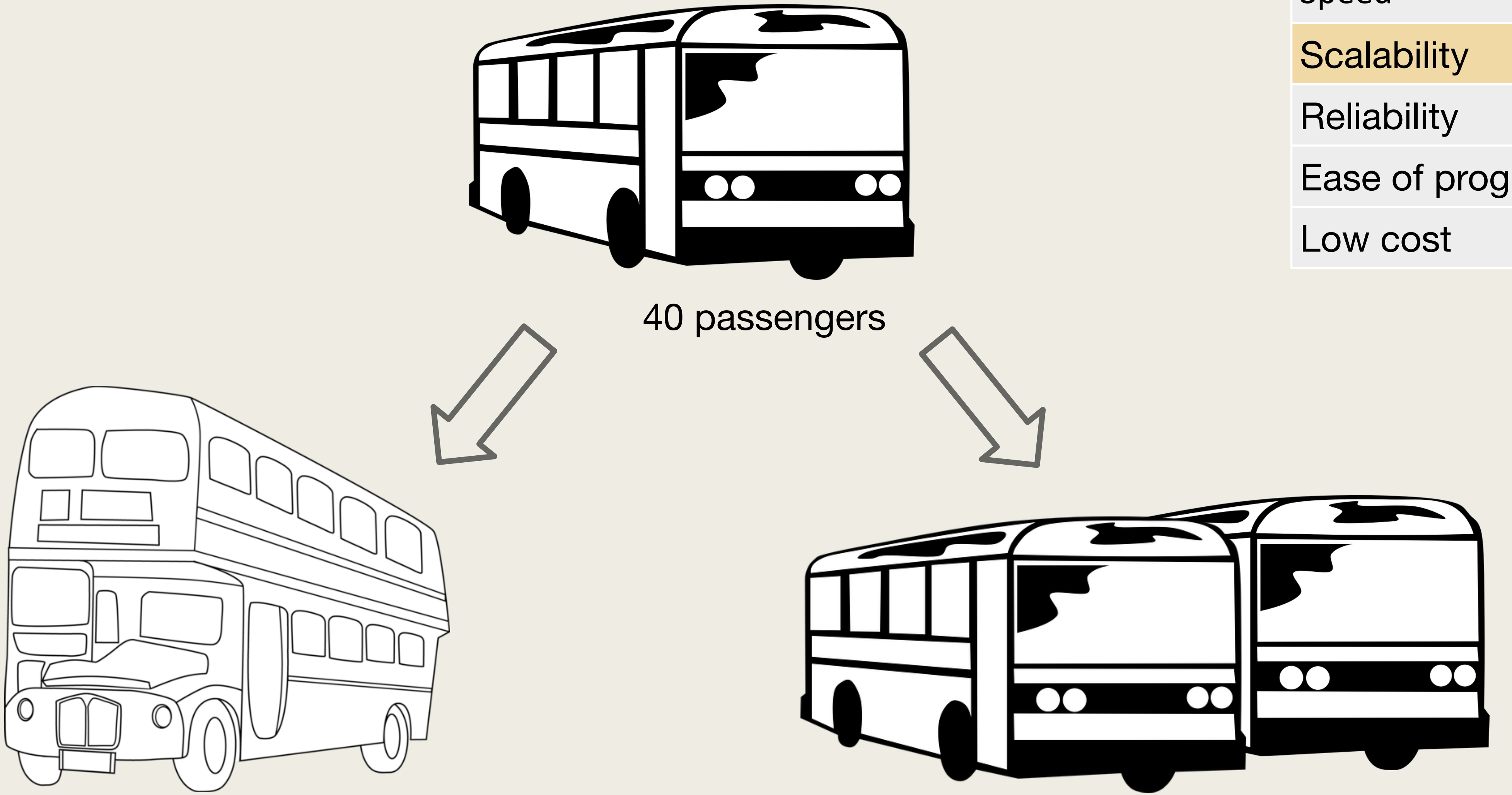
Design goals of batch processing systems

- Fast processing
  - *Data ought to be in primary storage, or even better, RAM*
- Scalable
  - *Should be able to handle growing data volumes*
- Reliable
  - *Should be able to handle failures gracefully*
- Ease of programming
  - *Right level of abstractions to help build applications*
- Low cost

➢ Need a whole ecosystem

How to scale?

Requirements
Speed
Scalability
Reliability
Ease of programming
Low cost



Ways to Scale

- To **scale horizontally** (or **scale out**) means to add more nodes to a system, such as adding a new computer to a distributed software application.

- To **scale vertically** (or **scale up**) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer.

What are the advantages and disadvantages?

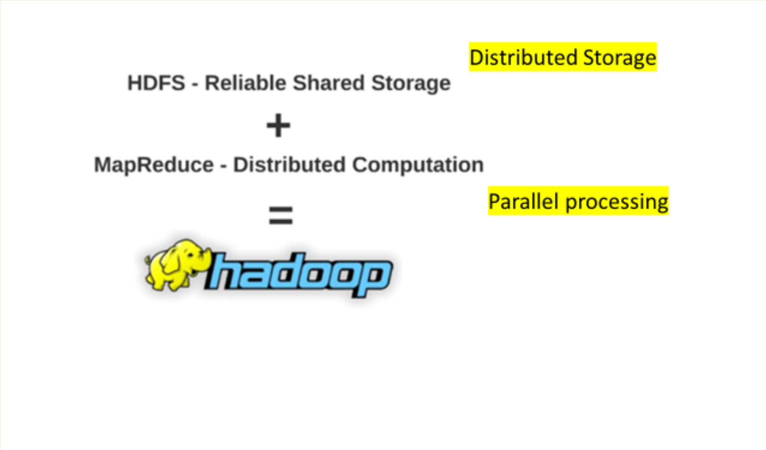
What is Hadoop?

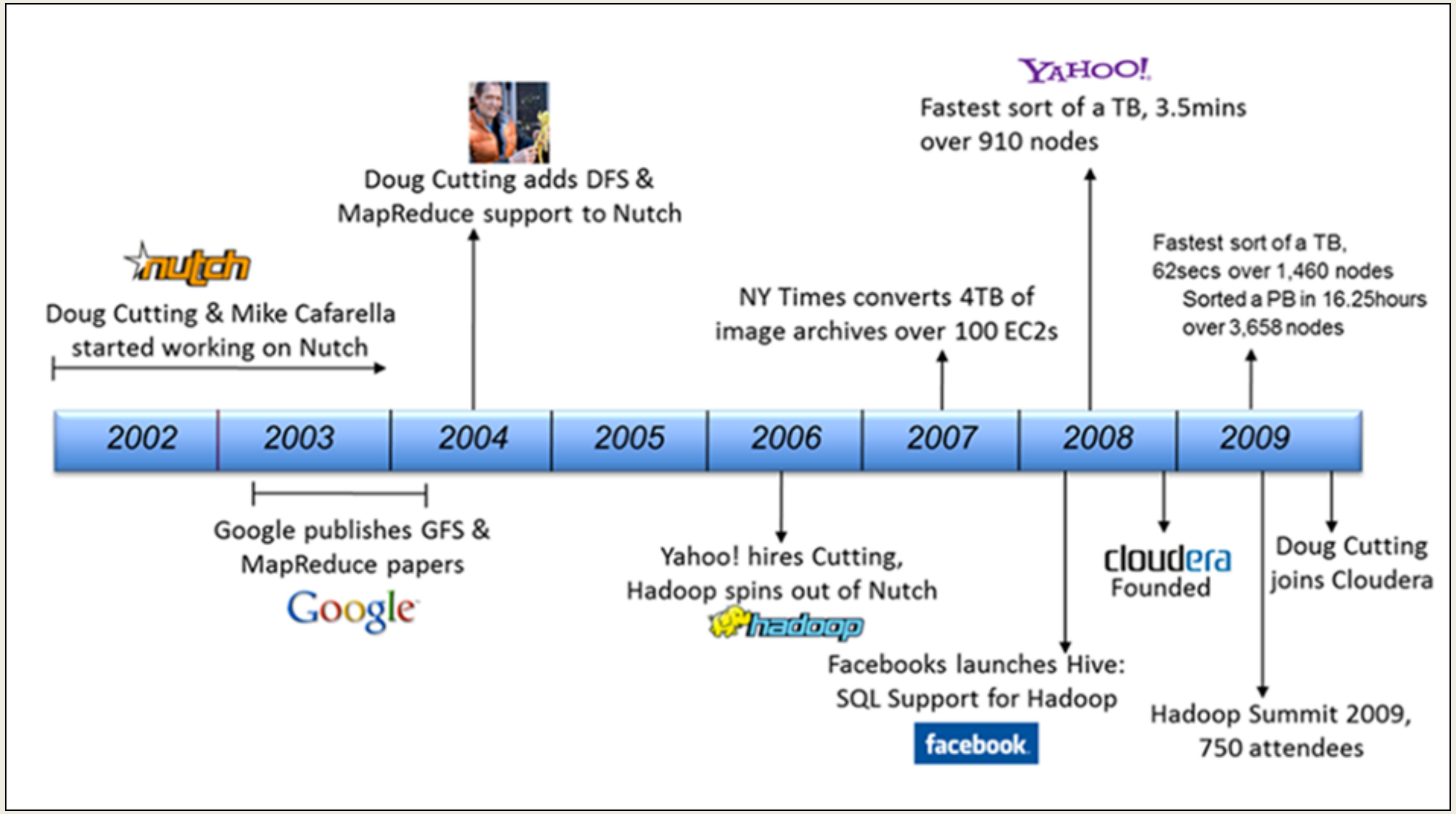
- Hadoop is an open source framework, from the Apache foundation, capable of processing large amounts of heterogeneous data sets in a distributed fashion across clusters of commodity computers and hardware using a simplified programming model.

- The Hadoop framework is based closely on the following principle:

In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers.
~Grace Hopper

Hadoop Timeline



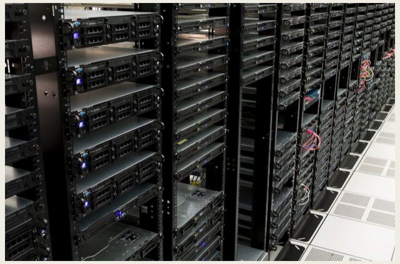


Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project.

Hardware

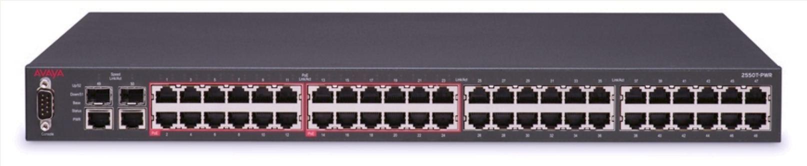
Rack

- The rack contains multiple mounting slots called bays
- A single rack can contain multiple servers stacked one above the other, consolidating network resources and minimizing the required floor space.
- The rack server configuration also simplifies cabling among network components.

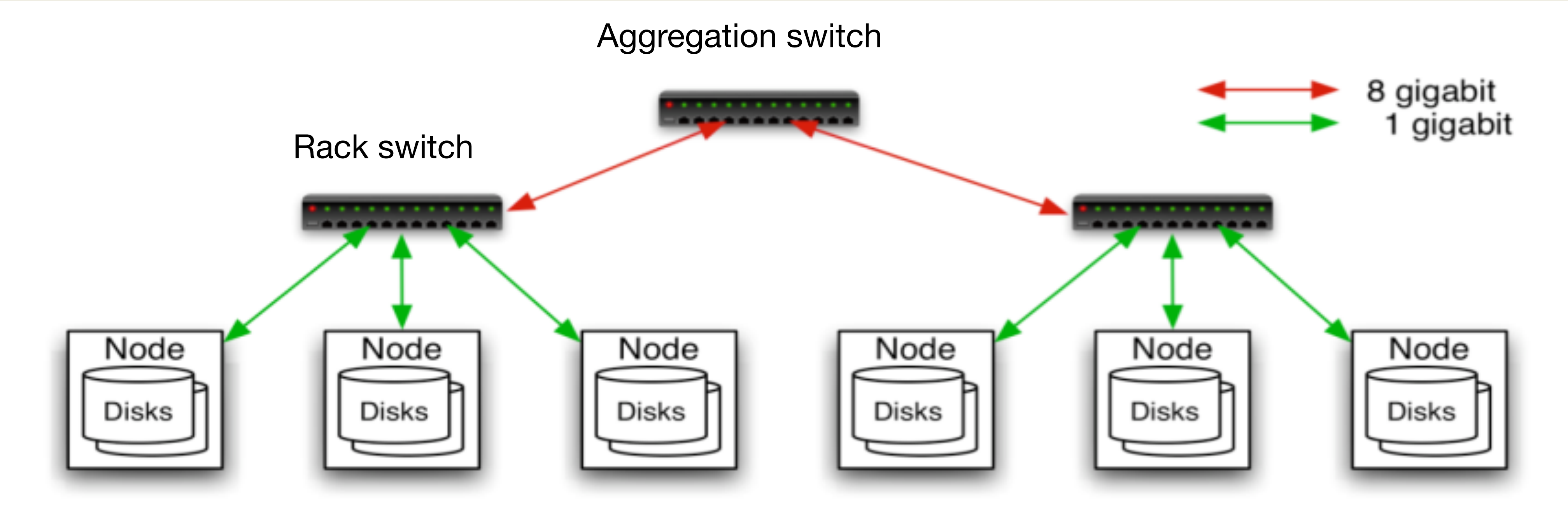


Switch

- A switch, in the context of networking is a high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN).
- Essentially, switches are the traffic cops of a simple local area network
- Switch is limited to node-to-node communication on the same network.



Hub & Spoke Hardware

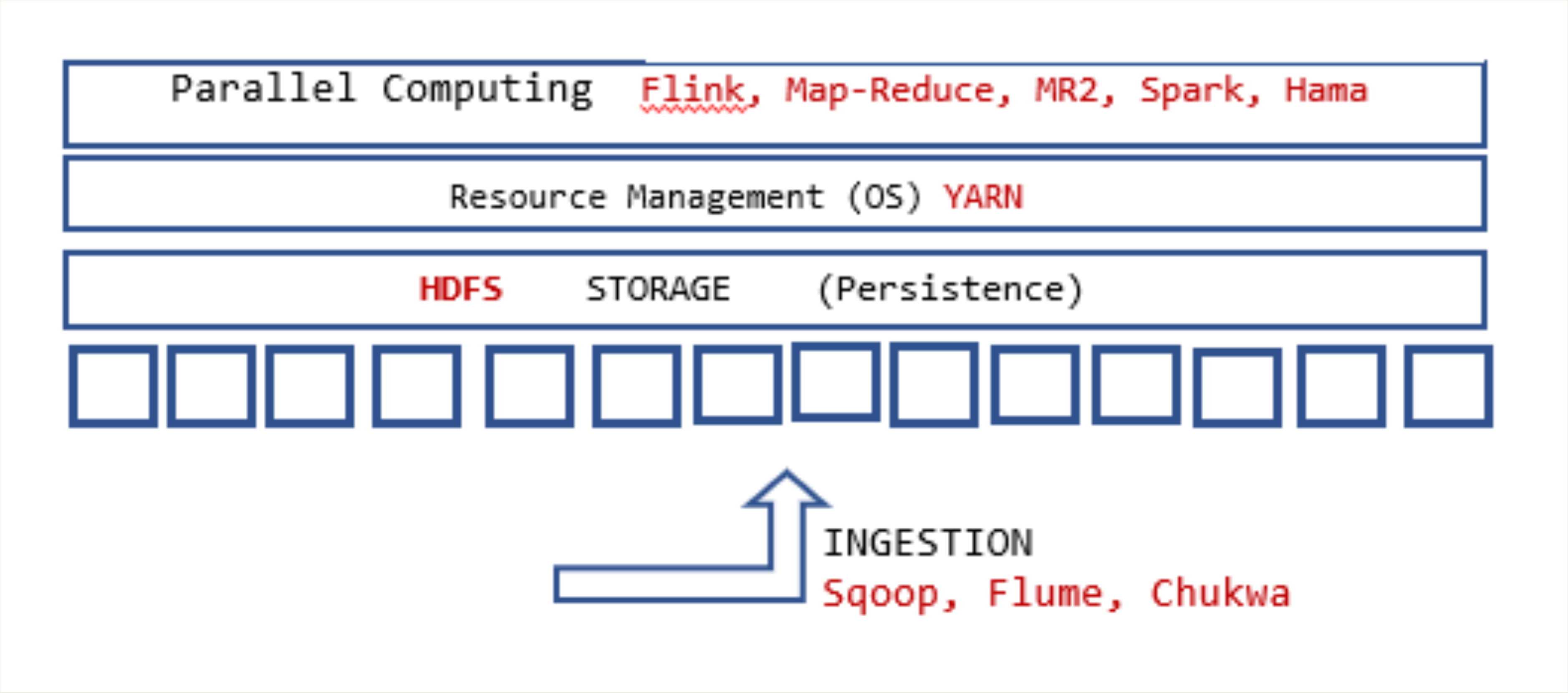


Hadoop Characteristics

Requirements
Speed
Scalability
Reliability
Ease of programming
Low cost

- Distribute data initially
  - *Let processors / nodes work on local data*
  - *Minimize data transfer over network*
  - *Replicate data multiple times for increased availability*
- Write applications at a high level
  - *Programmers should not have to worry about network programming, low level infrastructure, etc*
- Minimize talking between nodes (share-nothing)

Eco-system



Parallel Computing Flink, Map-Reduce, MR2, Spark, Hama
Resource Management (OS) YARN
HDFS STORAGE (Persistence)
INGESTION
Sqoop, Flume, Chukwa

HDFS design goals

Designed for

- Very large files (petabytes)
- Write once, read many times
- Append only
- Fault tolerance

Not good for

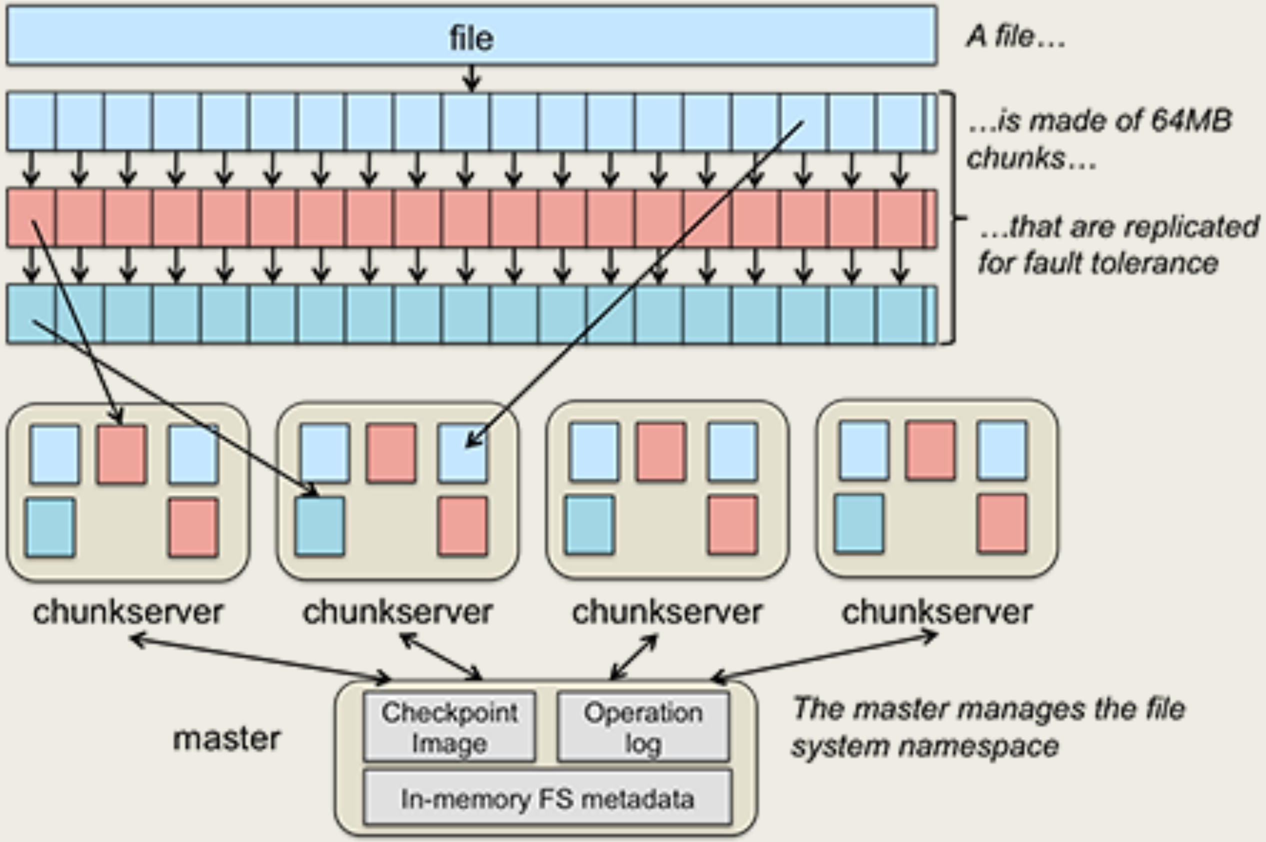
- Low latency data access
- Small files
- Update intensive workloads

HDFS is an open-source implementation of Google file system (GFS)

Design decisions

- Break files in very large blocks (128 MB)
  - *compare with 1024 bytes in a Linux file system*
- Fault tolerance
  - *Replicate the blocks (x3)*
  - *Replica placement*
  - *Periodic status - heartbeat and block report messages*
- Two types of nodes – NameNode and DataNode
- How many blocks and of what size would a file of size 600 MB have?

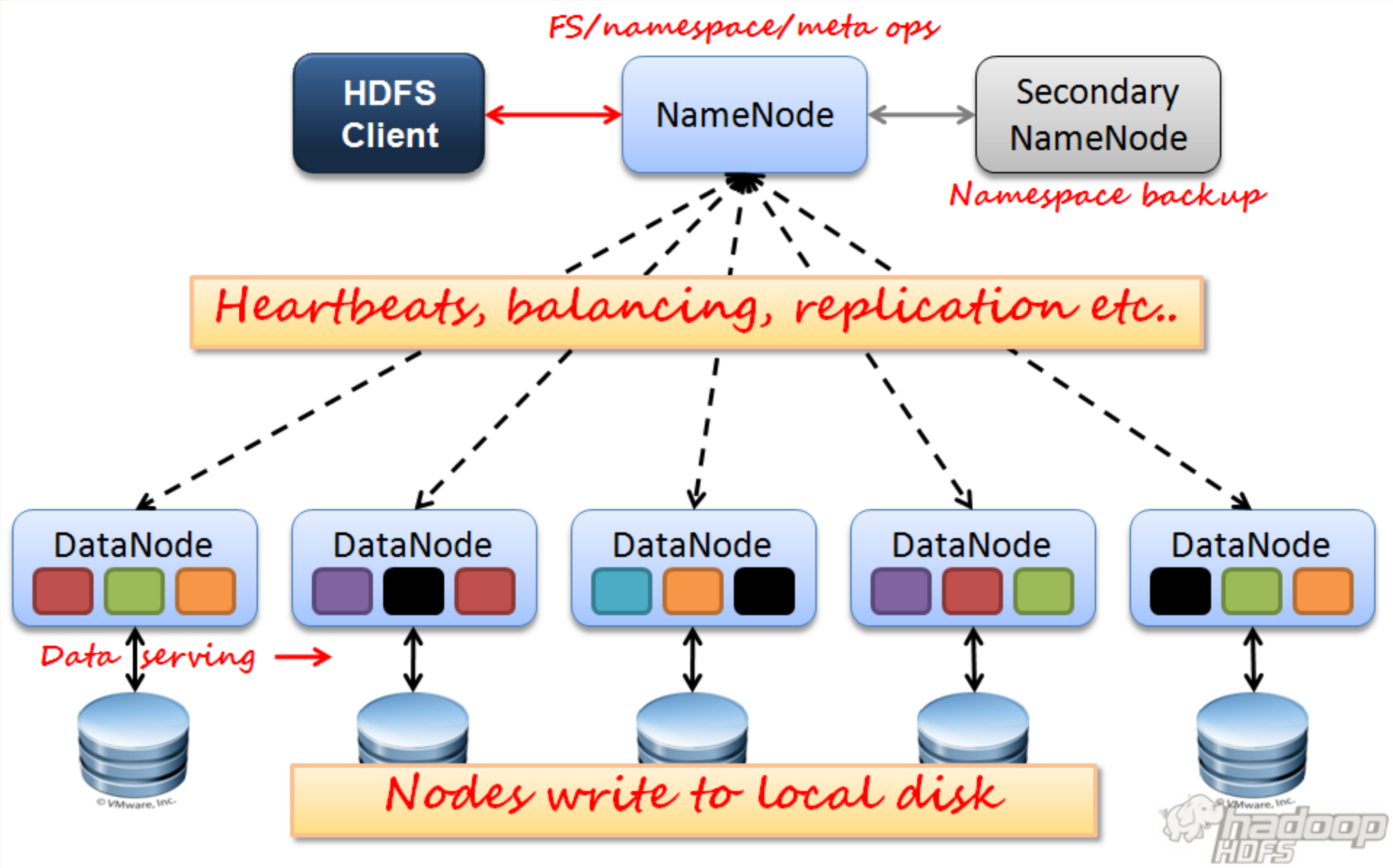
Background: Google File System (GFS)



Differences from traditional file system

- Write once, read many times
- Append only
- Large block size

HDFS



FS/namespace/meta ops
HDFS Client
NameNode
Secondary NameNode
Namespace backup
Heartbeats, balancing, replication etc..
DataNode
DataNode
DataNode
DataNode
DataNode
Data serving
Nodes write to local disk

NameNode

- Metadata in Memory
  - *The entire metadata is in main memory*
- Types of metadata
  - *List of files*
  - *List of Blocks for each file*
  - *List of Data Nodes for each block*
  - *File attributes, e.g. creation time, replication factor*
- A Transaction Log
  - *Records file creations, file deletions etc*

Block Replica 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 replicas
- Policy is pluggable

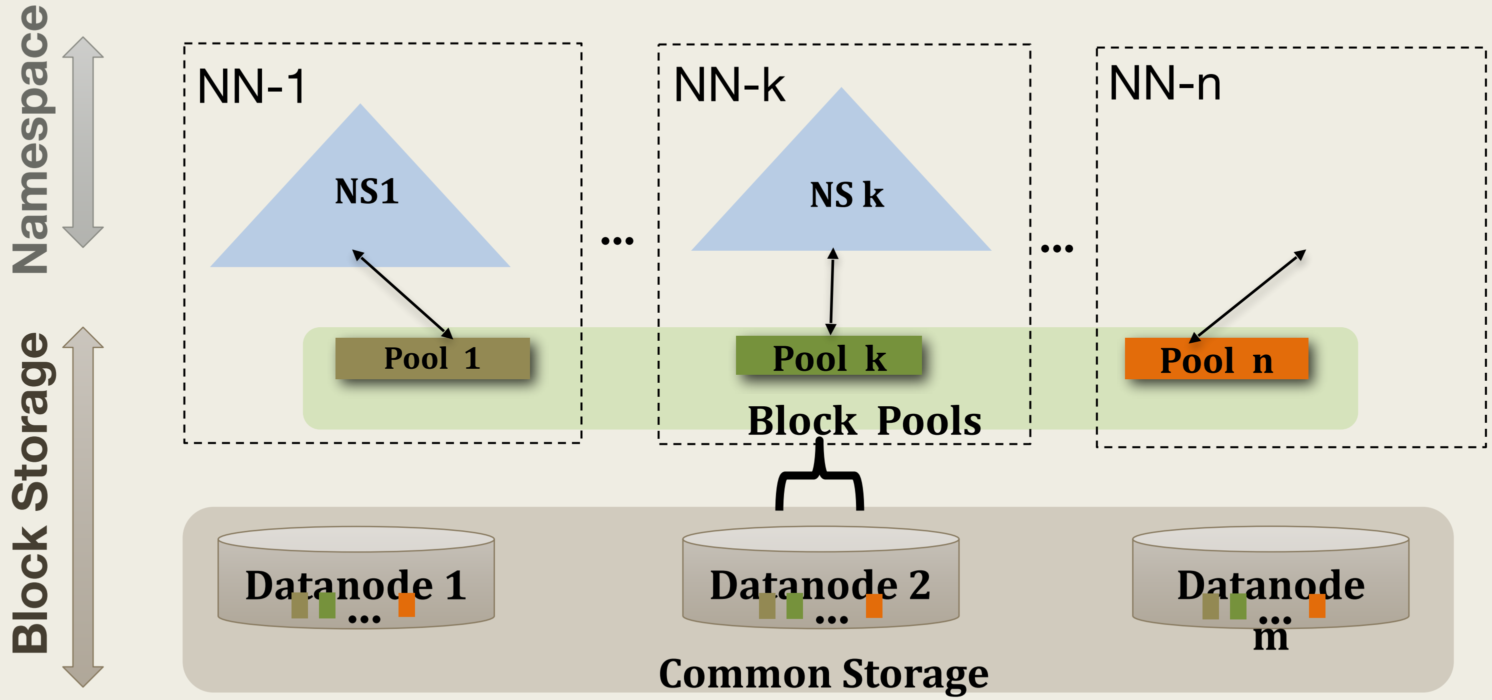
Heartbeat and Rebalancing

- Heart beats
  - *Data Nodes send heart beat to the Name Node*
  - *Once every 3 seconds*
  - *Name Node uses heartbeats to detect Data Node failure*
- Rebalancing: % disk full on Data Nodes should be similar
  - *Usually run when new Data Nodes are added*
  - *Cluster is online when Rebalancer is active*
  - *Rebalancer is throttled to avoid network congestion*
  - *Command line tool*

- Any problems you foresee wrt the design we have seen so far?

- Memory requirement
  - *Rule of thumb – 1000 MB per Million Blocks of file storage*
- Example:
  - *How many blocks in a 200 node cluster with each node having 24 TB Disk?*
  - *200 * 24,000,000 MB / [128MB * 3] ~ 12 million blocks*
  - *What is the memory requirement?*

HDFS 2.0: Name Node Federation Elaborated

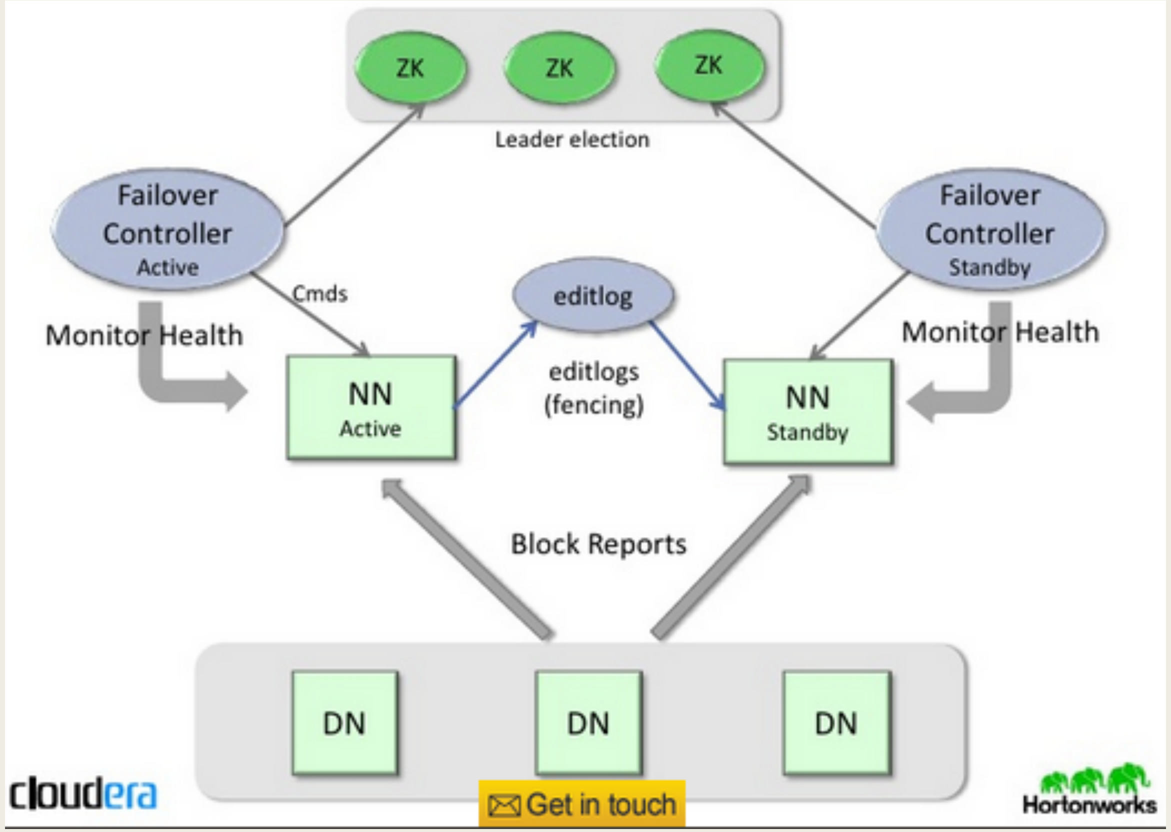


- Multiple **independent** Namenodes and Namespace Volumes in a cluster
  - *Namespace Volume = Namespace + Block Pool*
- Block Storage as generic storage service
  - *Set of blocks for a Namespace Volume is called a* ***Block Pool***
  - *DNs store blocks for all the Namespace Volumes – no partitioning*

Failure scenarios

- Data node failure
- Name node failure

HDFS 2.0: High Availability Elaborated



Reference: Hadoop Summit 2012 | HDFS High Availability talk

Failover

- In order to provide a fast failover, it is also necessary that the Standby node have up-to-date information regarding the location of blocks in the cluster.
- In order to achieve this, the DataNodes are configured with the location of both NameNodes, and send block location information and heartbeats to both.

Zookeeper

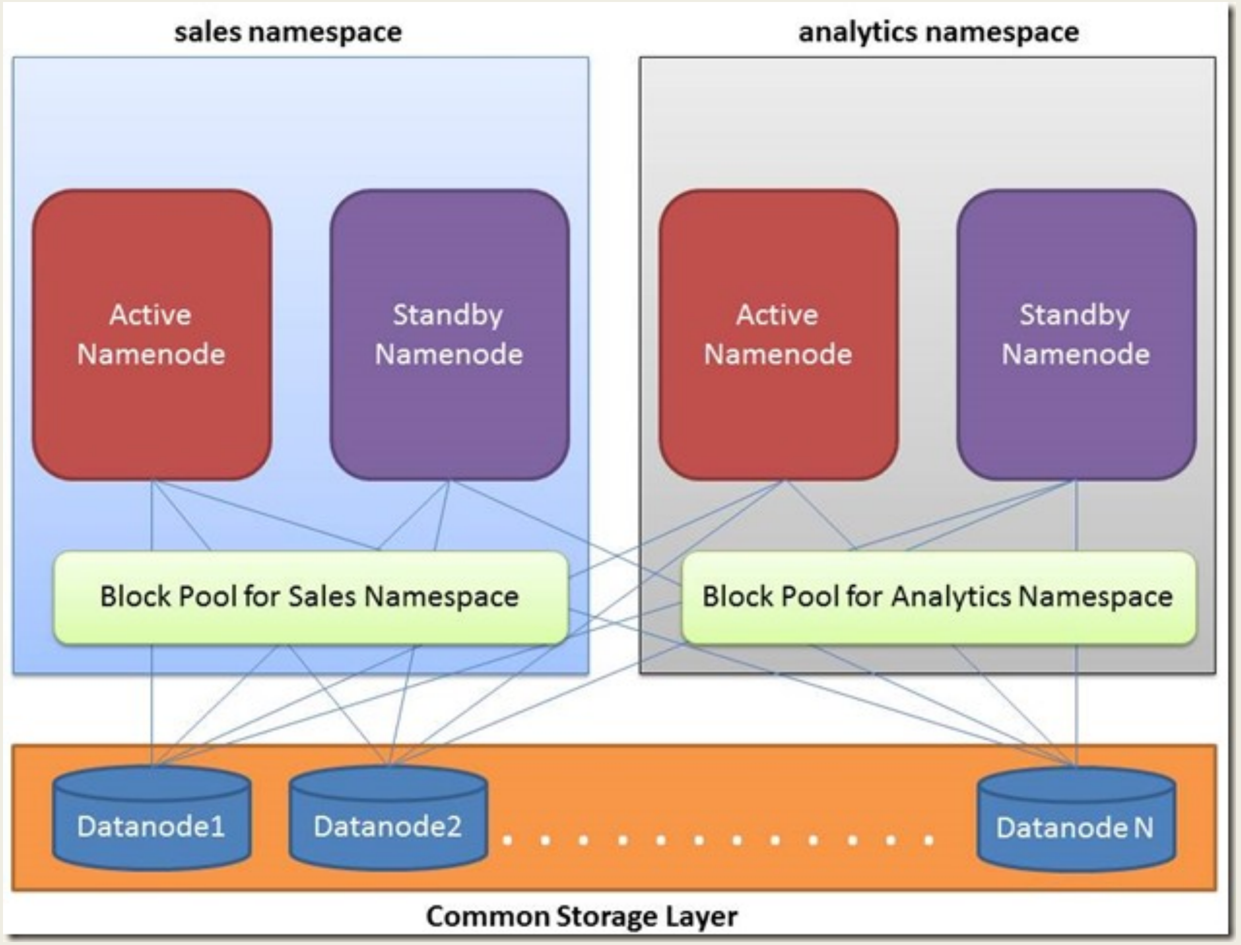
The ZKFailoverController (ZKFC) is a ZooKeeper client that also monitors and manages the state of the NameNode.

Each of the hosts that run a NameNode also run a ZKFC.

The ZKFC is responsible for **Health monitoring** of Namenode

- the ZKFC contacts its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds promptly with a healthy status, the ZKFC considers the NameNode healthy.
- If the NameNode has crashed, frozen, or otherwise entered an unhealthy state, the health monitor marks it as unhealthy.

HDFS 2.0: High Availability, Federated



http://pe-kay.blogspot.in/2016/02/configuring-federated-hdfs-cluster-with.html

HDFS interface – CLI

- Similar to other file system commands
- E.g.
  - *Hadoop fs -ls hdfs://localhost/user/tom*
  - *hadoop fs -copyFromLocal input/docs/quangle.txt hdfs://localhost/user/tom/quangle.txt*
  - *hadoop fs -copyToLocal quangle.txt quangle.copy.txt*

HDFS Interface – Java FileSystem

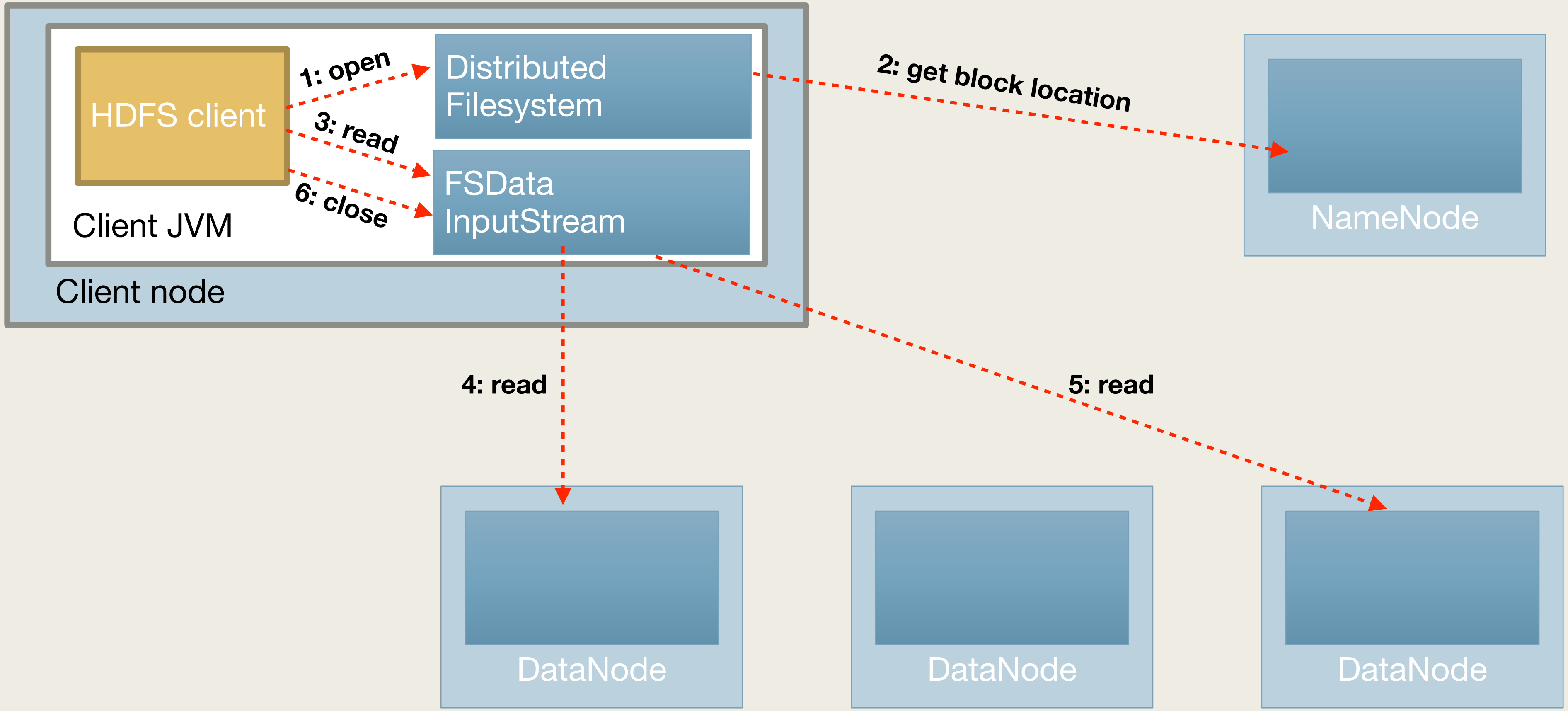
Reading Data

- public static FileSystem get(URI uri, Configuration conf) throws IOException
- public FSDataInputStream open(Path f) throws IOException
- public class FSDataInputStream extends DataInputStream implements Seekable, PositionedReadable

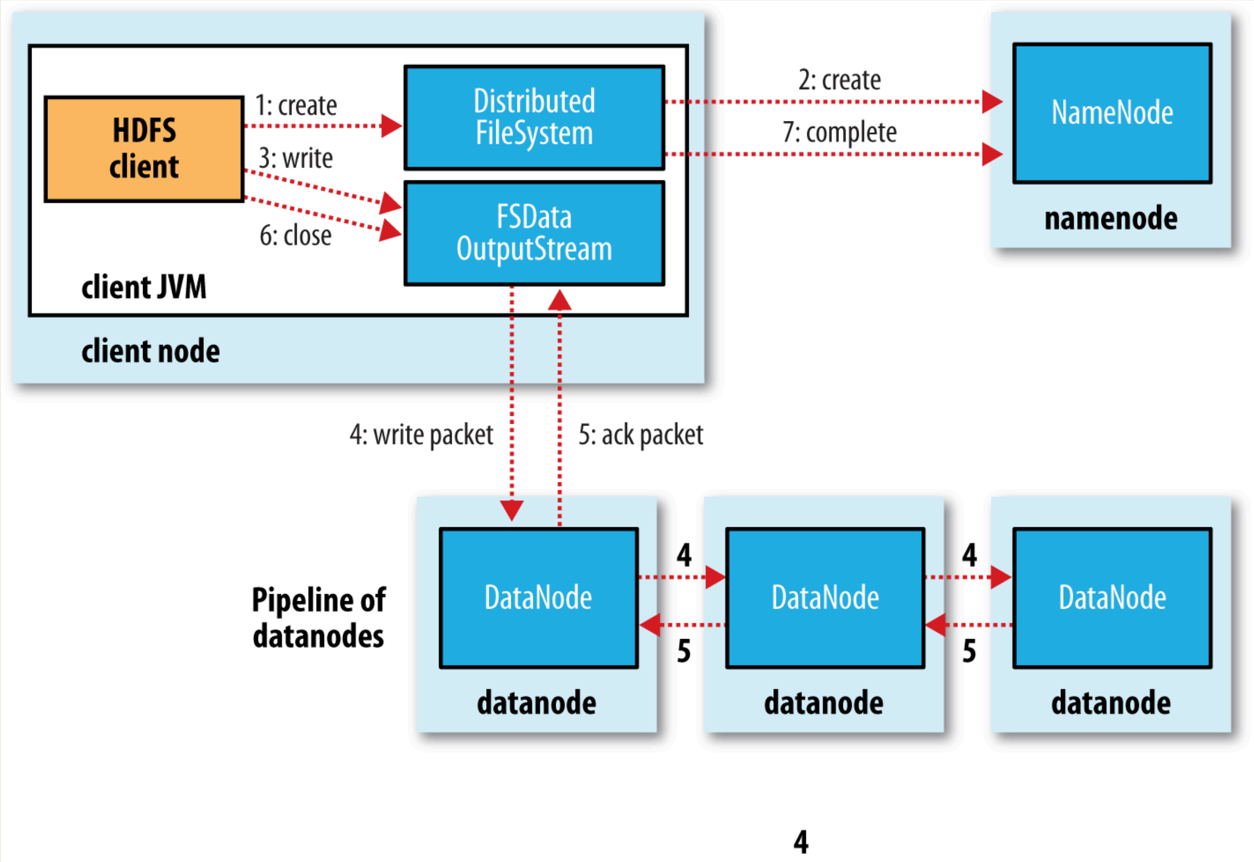
Writing Data

- public FSDataOutputStream create(Path f) throws IOException
- public FSDataOutputStream append(Path f) throws IOException
- public class FSDataOutputStream extends DataOutputStream implements Syncable

File read flow

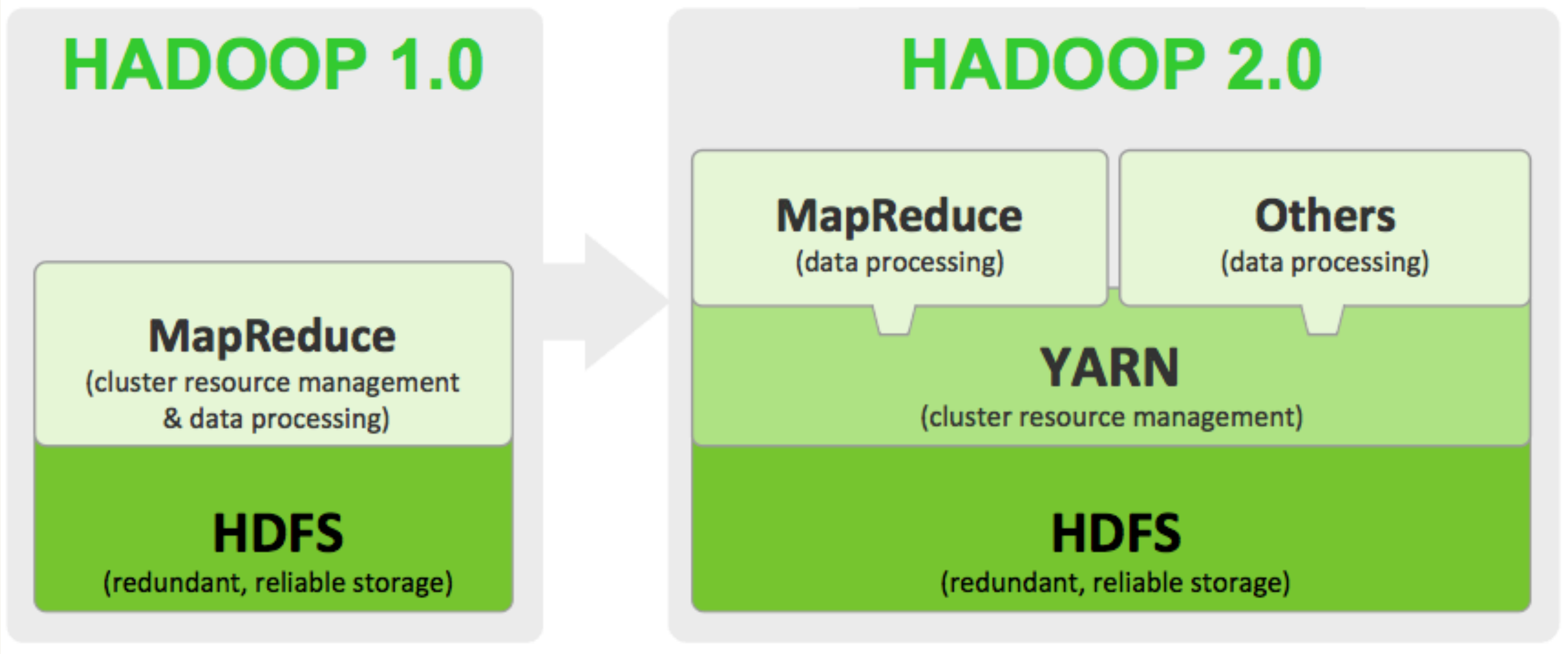


File write flow



YARN – Resource Manager

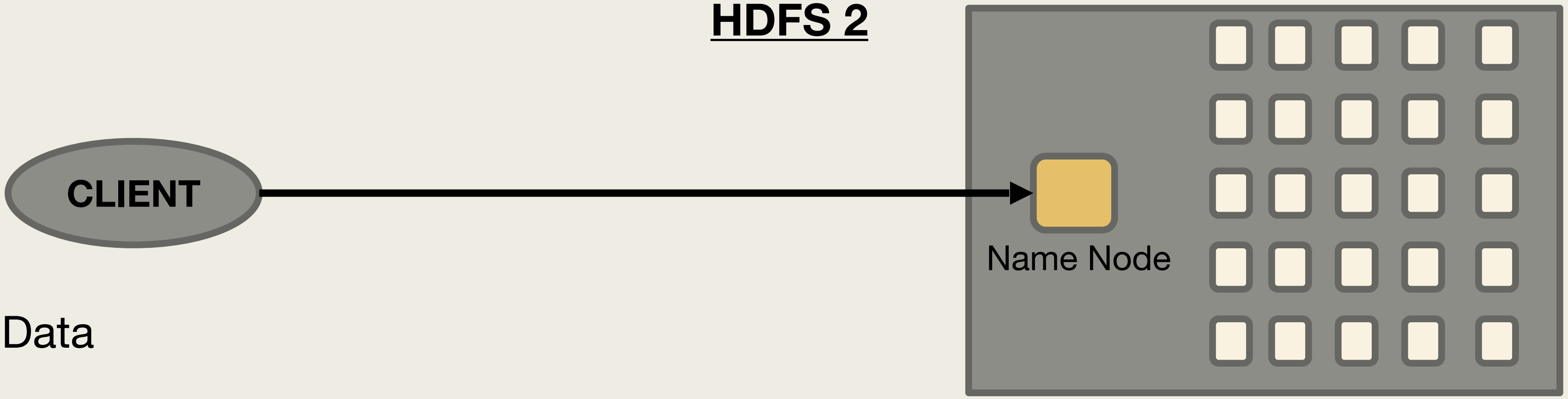
- Yet Another Resource Negotiator
- Replaces Job Tracker and Task Tracker architecture in MR1
- YARN designed to scale up to 10,000 nodes and 100,000 tasks.
- Under YARN, there is no distinction between resources available for maps and resources available for reduces – all resources are available for both; the notion of slots has been discarded
- Resources are now configured/allocated in terms of amounts of memory (in megabytes) and CPU



Enables non-MapReduce tasks to work within a Hadoop installation

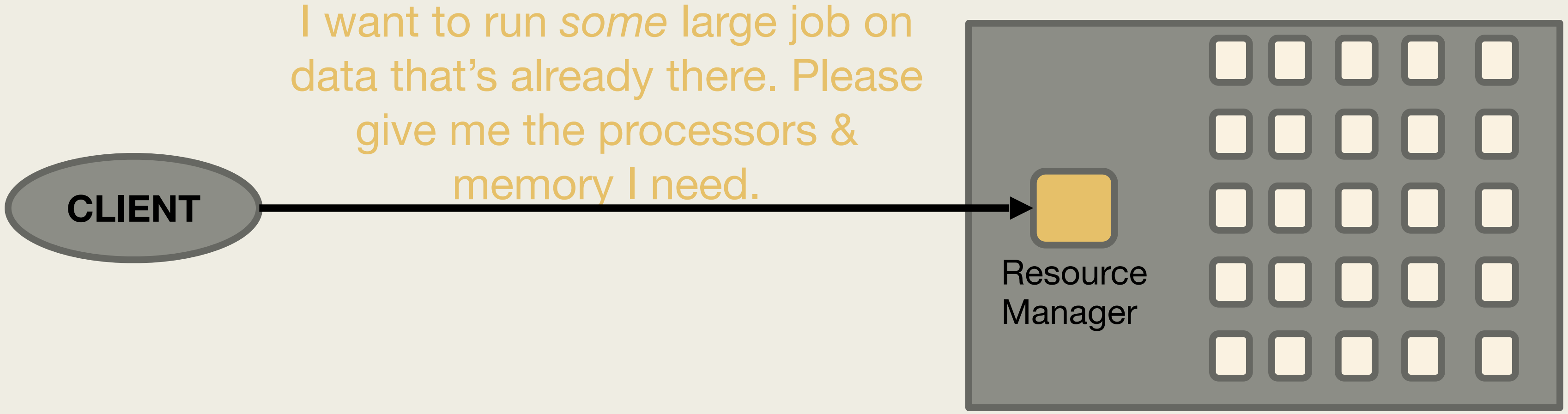
Data and compute distribution

HDFS 2



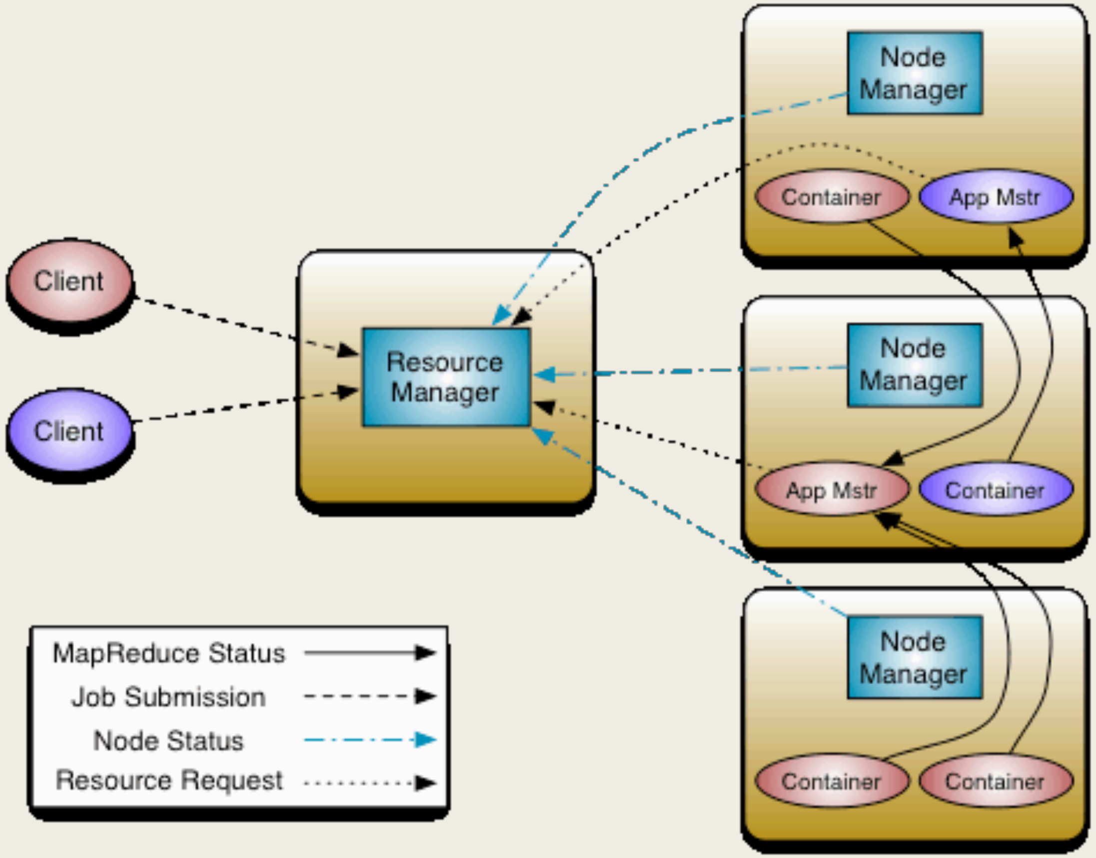
Distributed Data

YARN



Distributed Processing

YARN Architecture



Resource Manager

- acts as the sole arbitrator of cluster resources.
- ultimate authority that arbitrates resources among all the applications.
- responsible for optimizing cluster utilization

Node Manager

- per-machine slave [daemon]
- responsible for launching the applications’ containers,
- monitoring their resource usage (CPU, memory, disk, network)
- reports to the ResourceManager
- provides logging and other auxiliary services

Application Master

- per application
- negotiates appropriate resource containers with RM
- works with the NMs to execute and monitor the containers and their resource consumption
- monitors progress

- The YARN system (RM and NM) needs to protect itself from faulty or malicious AMs and resources granted to them

Container

- Container is the resource allocation [container ID, Node Manager], which is the successful result of the RM granting a specific ResourceRequest from AM

- RM responds to a resource request by granting a container, which satisfies the requirements laid out by the AM in the initial ResourceRequest.

- A ResourceRequest format:

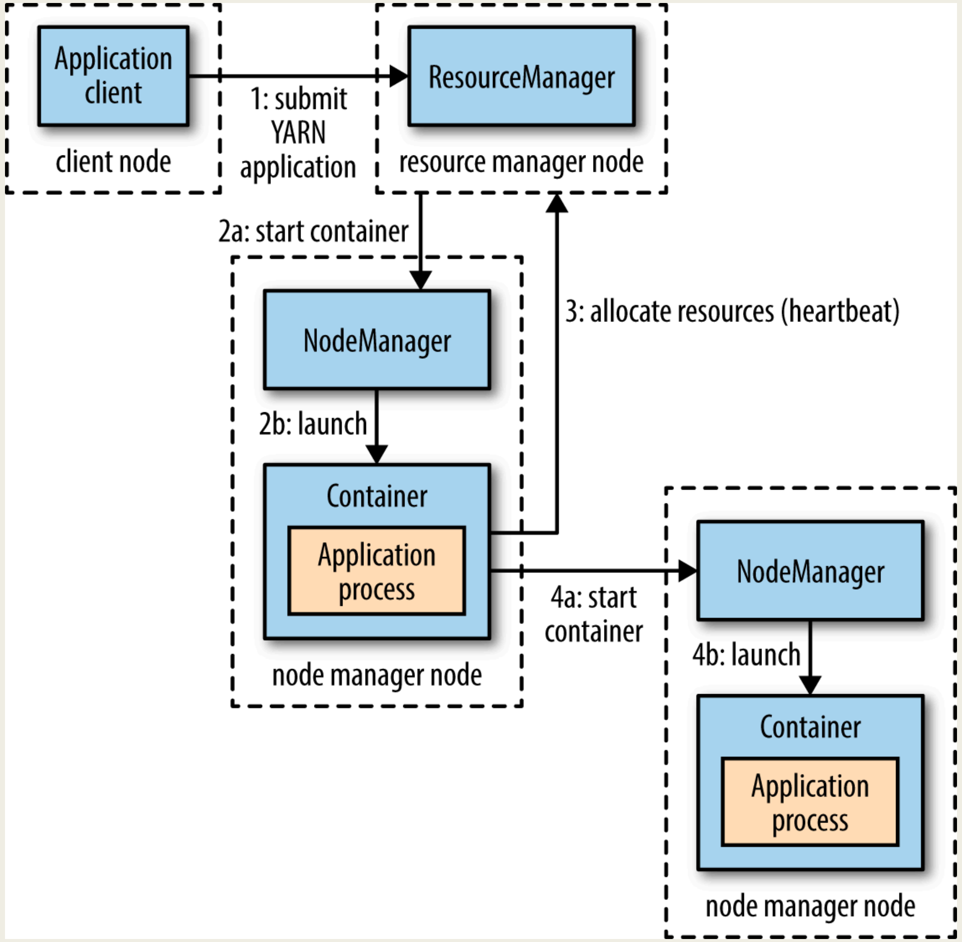
<resource-name, priority, resource-requirement, number-of-containers>

  – *Resource requirements: CPU and Memory*

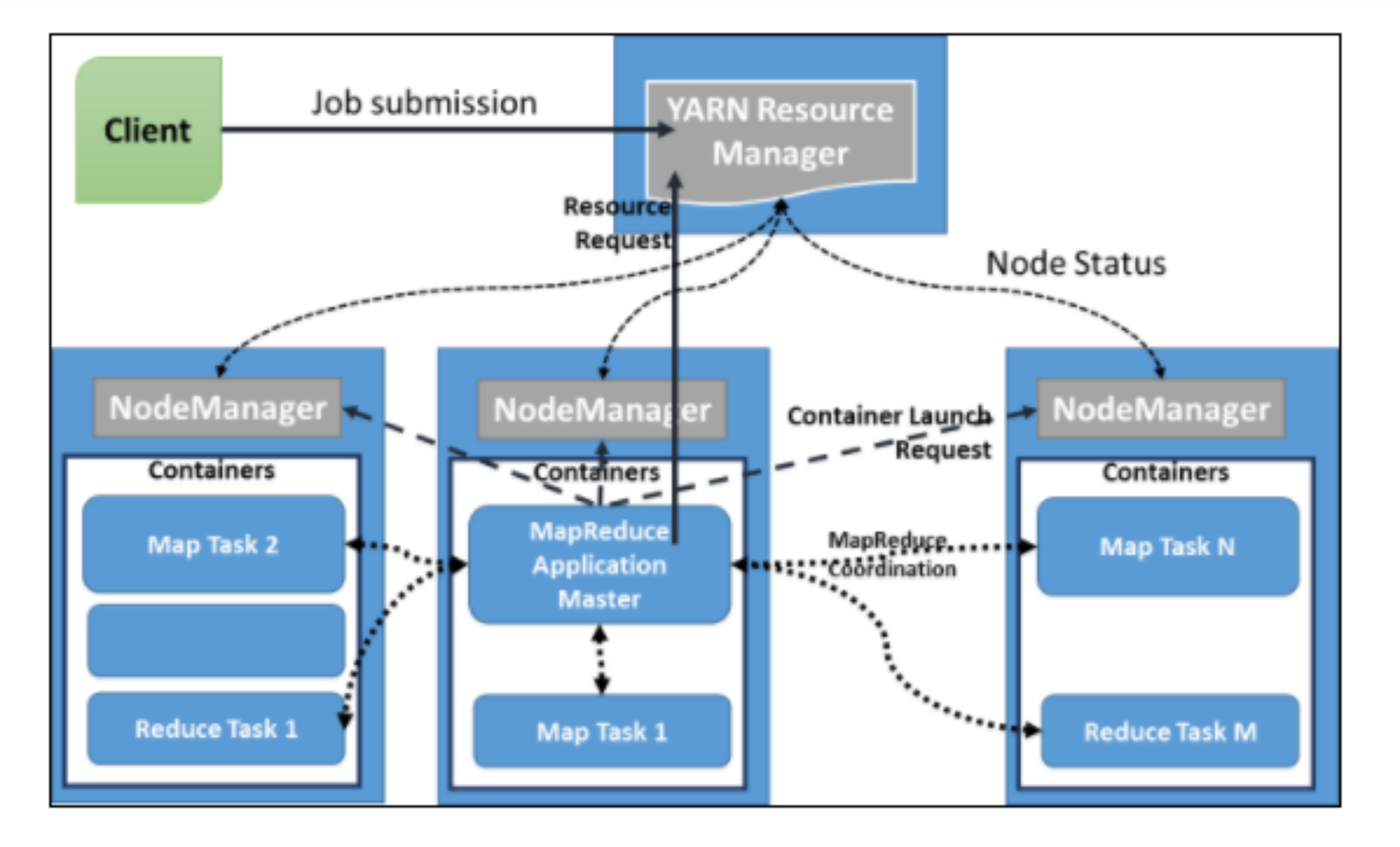
Launching the tasks

- AM presents the Container to the NM managing the host, on which the container was allocated, to use the resources for launching its tasks
- The Container launch specification API is platform agnostic and contains:
  - *Command line to launch the process within the container.*
  - *Environment variables*
  - *Local resources necessary on the machine prior to launch, such as jars, shared-objects, auxiliary data files etc*

Sequence of actions



MR on YARN


Client
Job submission
YARN Resource Manager
Resource Request
Node Status
NodeManager
Containers
Map Task 2
Reduce Task 1
NodeManager
Containers
MapReduce Application Master
Map Task 1
Container Launch Request
MapReduce Coordination
NodeManager
Containers
Map Task N
Reduce Task M

Reading list

- Hadoop The Definitive Guide, Tom White (https://grut-computing.com/HadoopBook.pdf)
  - *Chapter 3 – The design of HDFS, HDFS Concepts, The Java Interface (details can be skipped), Data Flow*
  - *Chapter 4, except Scheduling in YARN*