



JOHNS HOPKINS
CAREY BUSINESS SCHOOL

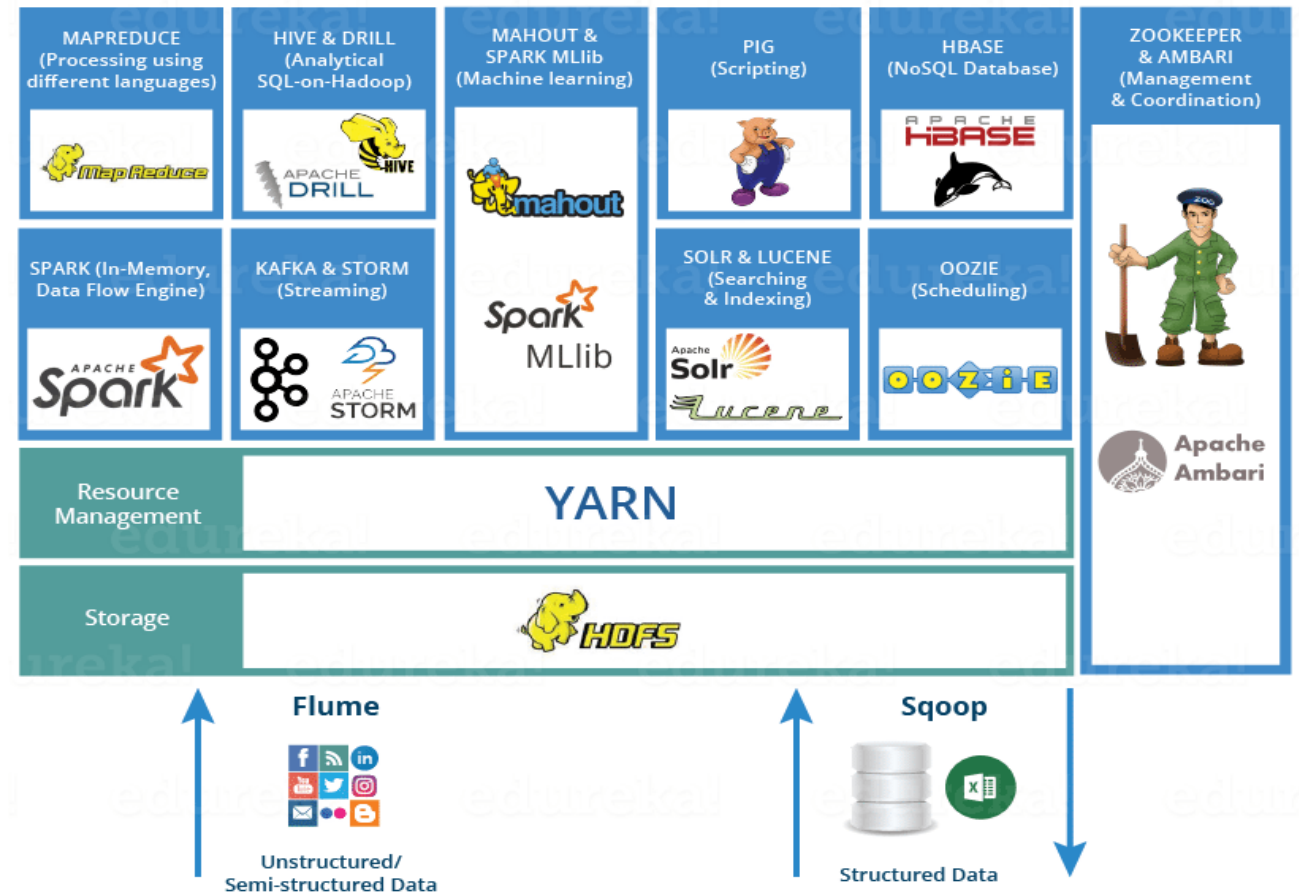
HDFS and YARN

BU.330.740 Large Scale Computing on the Cloud

Minghong Xu, PhD.
Associate Professor

Hadoop Overview

- » An open source framework for writing and running distributed applications that process large amount of data
- » Google: first to publicize MapReduce for scaled data processing
- » Doug Cutting: develop the first version of Hadoop





Key Components

- » Distributed File System: HDFS
 - Most low-level knowledge of this course
- » Operating System: YARN
- » Original Distributed Processing Engine: MapReduce
- » Improved MapReduce: Spark
- » Distributed Query Language: Hive
- » Distributed Scripting Language: Apache Pig



Distributed File System



Hadoop Distributed File System

» **Problem 1:** *Data is too big to store on one machine.*

» **Solution:** Store the data on multiple machines!

» **Problem 2:** *Very high end machines are too expensive*

» **Solution:** Run on commodity hardware!

» **Problem 3:** *Commodity hardware can fail*

» **Solution:** Software is intelligent enough to handle hardware failure!

» **Problem 4:** *What happens to the data if the machine storing the data fails?*

» **Solution:** Replicate the data!



Solutions (Cont.)

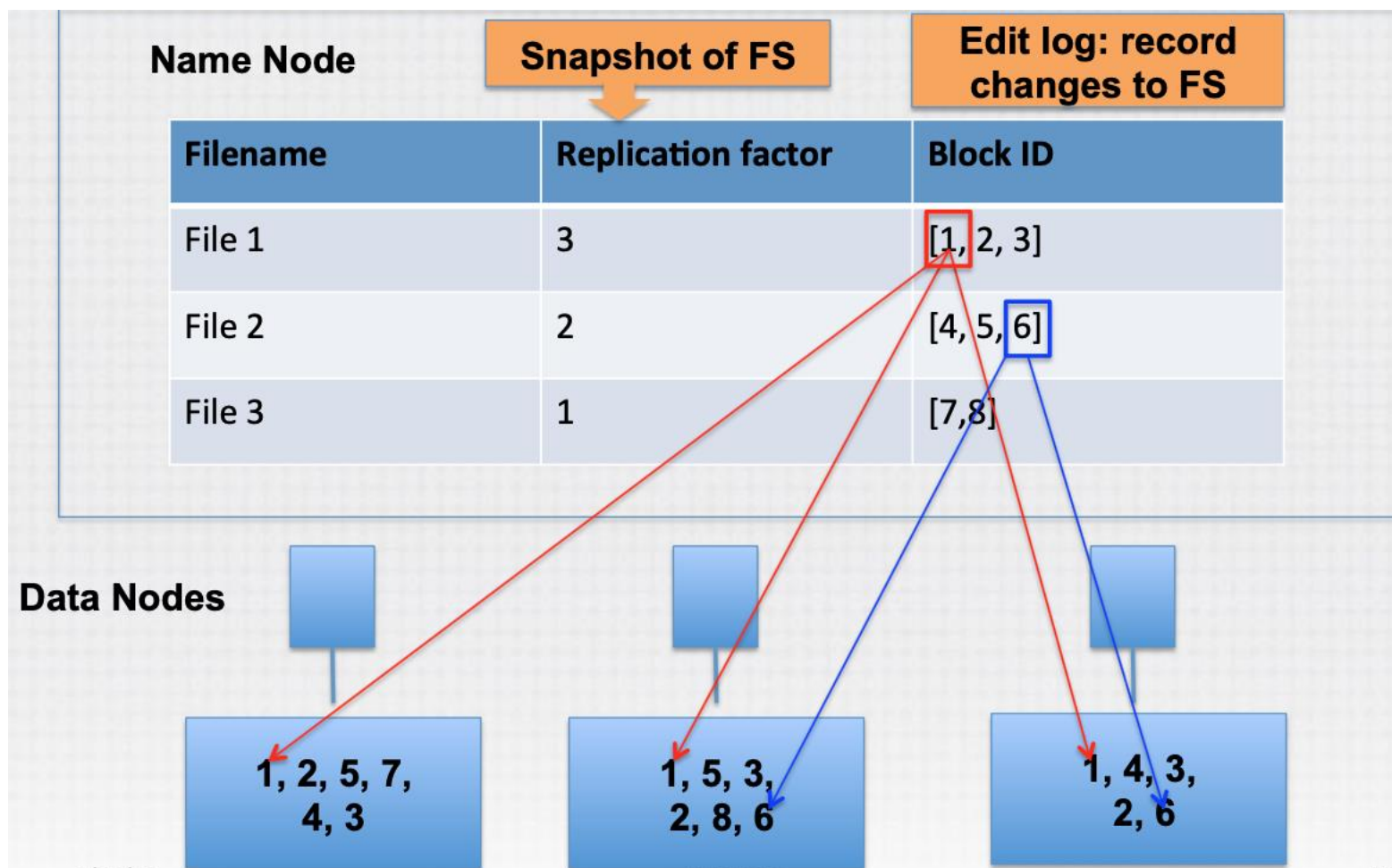
- » **Problem 5:** *How can distributed machines organize the data in a coordinated way?*
- » Insights from your own team work experience...
 - *How to assign tasks to different workers in an efficient way?*
 - *What happens if tasks fail?*
 - *How do workers exchange results?*
 - *How to synchronize distributed tasks allocated to different workers?*
- » **Solution:** primary-secondary structure



HDFS Primary-Secondary Architecture

- » Single NameNode
 - Sometimes a backup: secondary NameNode
- » Many (Thousands) DataNodes
- » Files are split into fixed sized blocks and stored on data nodes
- » Data blocks are replicated for fault tolerance and fast access
 - By default: 3

Illustration



Adapt from K. Zhang's notes, Spring 2019



NameNode

- » Manages file system namespace, and file metadata
- » Mapping file to list of blocks
- » Mapping of datanode to list of blocks
- » Monitor datanode health
- » Replicate missing blocks

DataNode

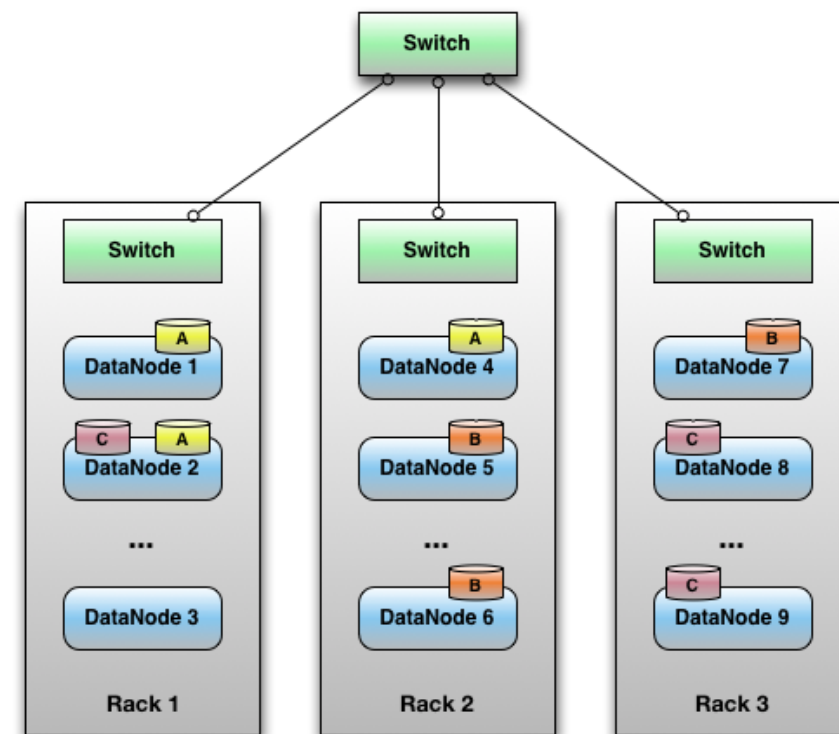


- » Handle block storage & block integrity
- » Periodically send reports to NameNode
- » Clients access the blocks directly from data nodes *Imp.*

- » *Q: Why not access blocks through NN?*
- » Reasons:
 - Prevent NN from being the bottleneck of the cluster
 - Allow HDFS to scale to large number of concurrent clients
 - Spread the data traffic across the cluster

Data Replicate

- » **Frist replica** is put on one node in the local rack
- » **Second one** is put on a node in a different (remote) rack
- » **Third one** is on a different node in the same remote rack
- » **Additional** replicas are randomly placed
- » Objectives: load balancing, fast access, fault tolerance.





HDFS Network Topology

- » The critical resource in HDFS is **bandwidth**, distance is defined based on that
- » Measuring bandwidths between any pair of nodes is too complex
- » **Basic Idea:**
 - Processes on the same node
 - Different nodes on the same rack
 - Nodes on different racks in the same data center (cluster)
 - Nodes in different data centers

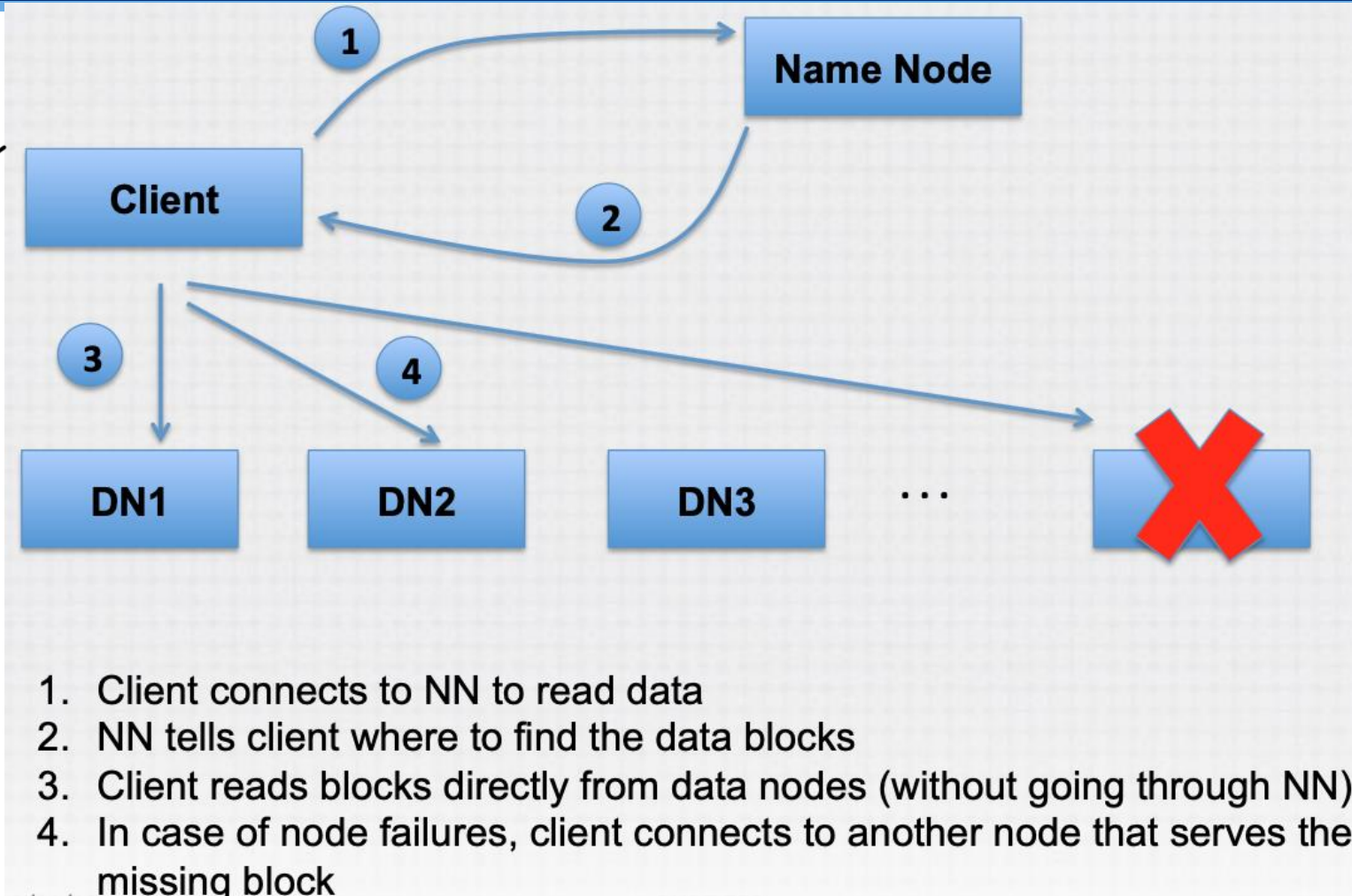


**Bandwidth
becomes less**

Data Read



Read from

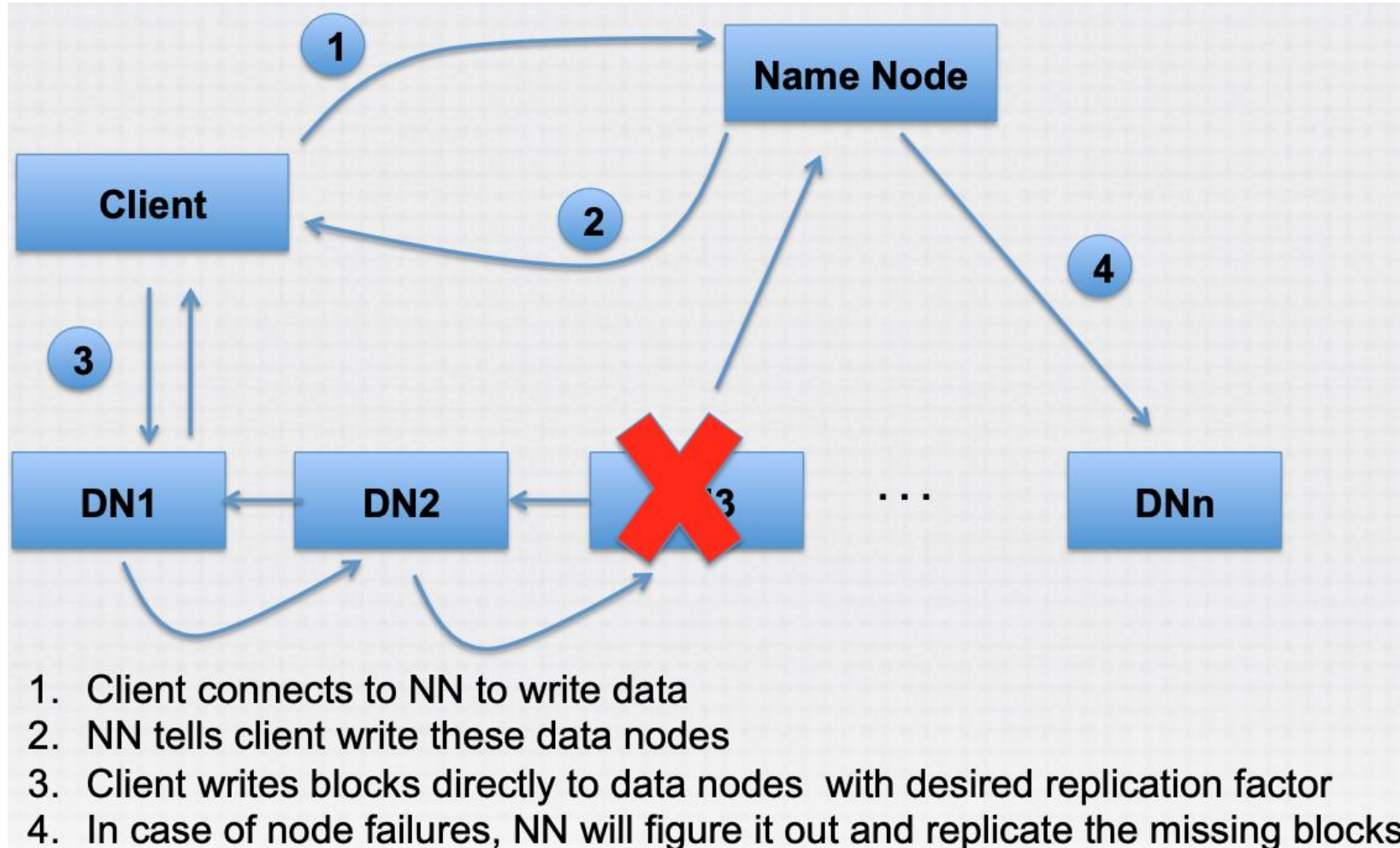


Adapt from K. Zhang's notes, Spring 2019

Data Write



Write flow

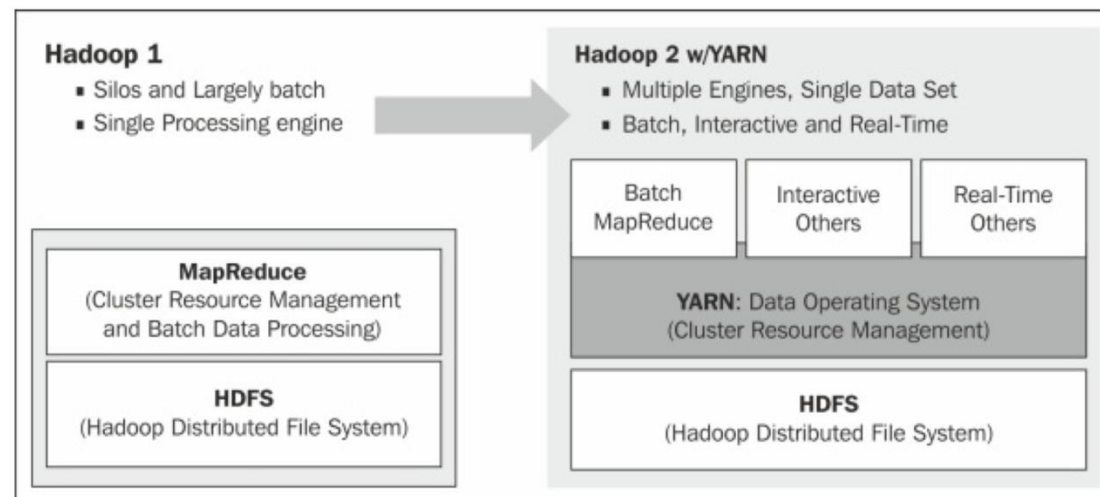




Yet Another Resource Negotiator

What is YARN?

- » Cluster resource management system for Hadoop
- » Introduced in Hadoop 2 to improve MapReduce implementation
- » Connect between high level applications (Spark, HBase) and low level Hadoop environment
- » Large-scale, distributed operating system for big data applications



Components



» Resource Manager

- 1 per cluster
- Track resource in a cluster, schedule applications → *resource allocation.*
- Single point of failure, but can be restored in case of failures

» Application Master

- Run in a separate process on a datanode
- One instance per application
- Send status and resource needs to RM
- Can run lightweight task on the same node

↓
manages the lifecycle of applications →

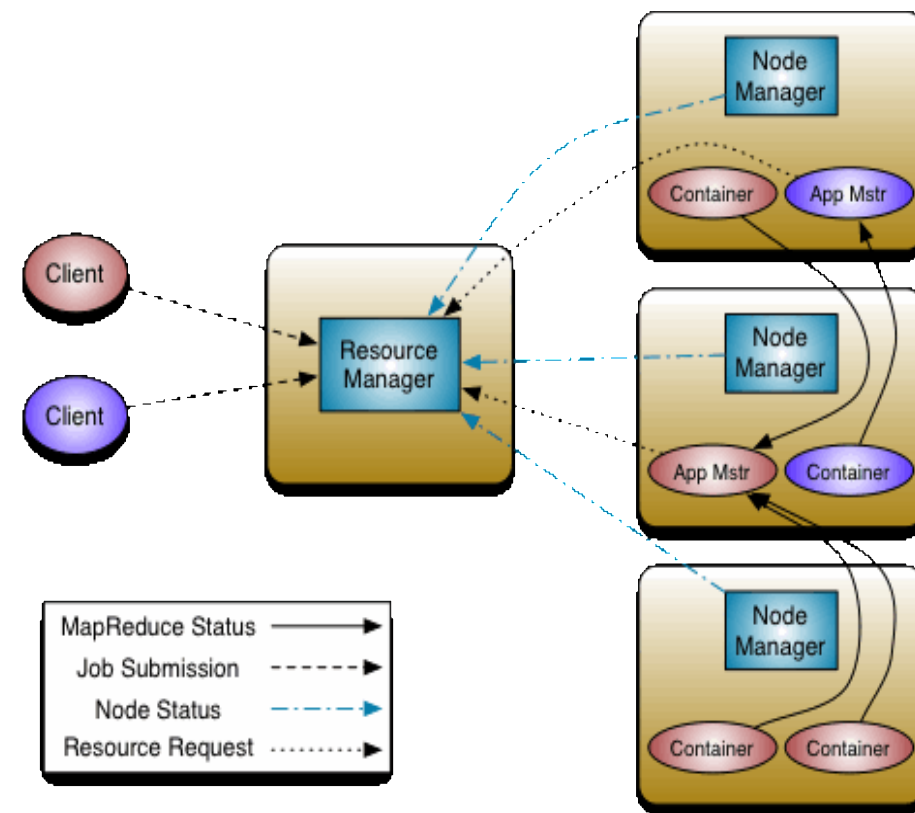
» Node Manager

- 1 per node ✓
- Monitor node resources such as CPU, Memory, Disk space, Network etc. ✓
- Collect log data, report to Resource Manager ✓

coordinates with Resource Manager to request resources.

Steps to run Yarn application

- » Client make request to Resource Manager to run application
- » Resource Manager request Node Manager to allocate container for creating Application Master instance on available node
 - Container: basic unit of hardware allocation
- » When Application Master instance already run, it sends request (heartbeat, resource needs) to Resource Manager

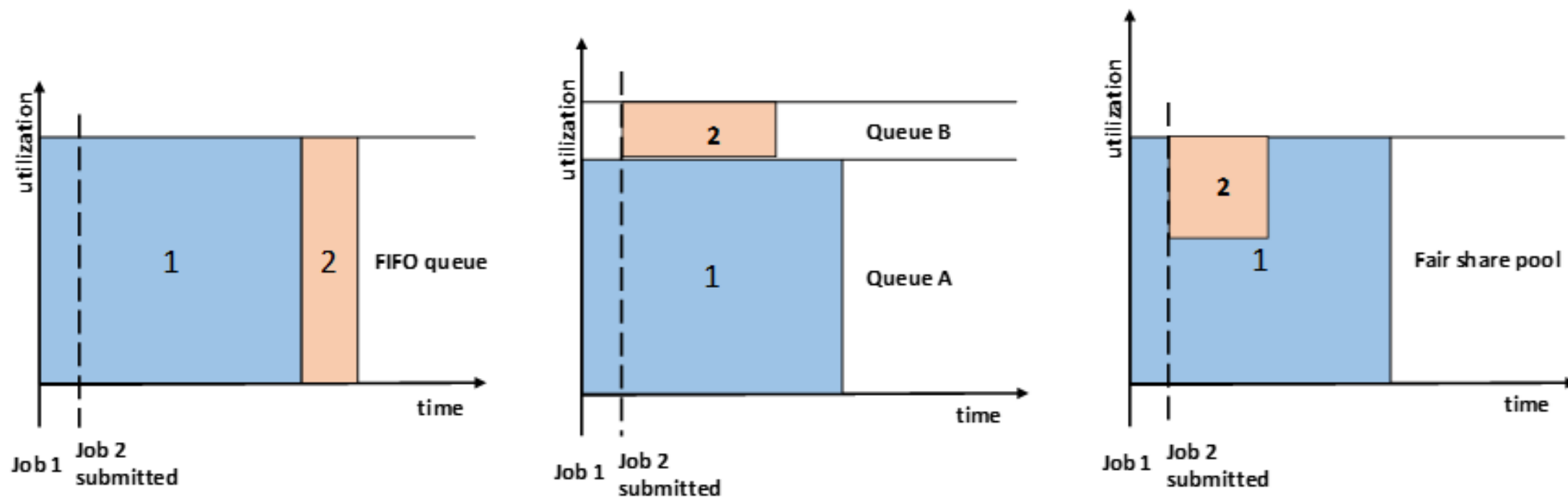




Scheduling in Yarn

- » Important task since Hadoop cluster is shared between many users and tasks; which to run first?
- » Hadoop 1: FIFO scheduler with fixed cpu, memory, disk count
- » Hadoop 2: allow Capacity and Fair schedulers with dynamic cpu, memory and disk count
- » Yarn supports all three

Illustrations



(a) FIFO scheduler

(b) Capacity Scheduler

(c) Fair Scheduler

cluster is divided into subgroups
Figure 1: YARN Schedulers' cluster utilization vs. time
as per capacity & parallel
scheduling happens.

↓
equitable
share of resources