

ISIT312 Big Data Management

Hadoop Architecture

Dr Guoxin Su and Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Hadoop Architecture

Outline

Hadoop Distributed File System (HDFS)

NameNode metadata

DataNode and Secondary node

Yet Another Resource Negotiator (YARN)

ResourceManger

NodeManager

ApplicationMaster

HDFS: Hadoop Distributed File System

HDFS is designed for:

- Very large files
- Stream data access
- Commodity hardware

But not for:

- Low-latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

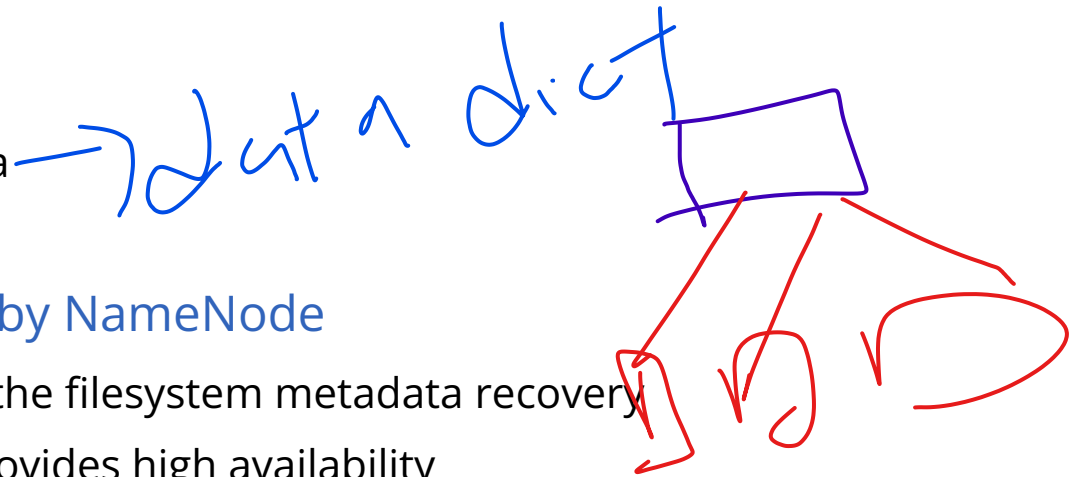
→ *Correcting*

HDFS: Hadoop Distributed File System

HDFS contains the following key components:

NameNode:

- HDFS master node process
- manages the filesystem metadata
- does not store a file itself



SecondaryNameNode and Standby NameNode

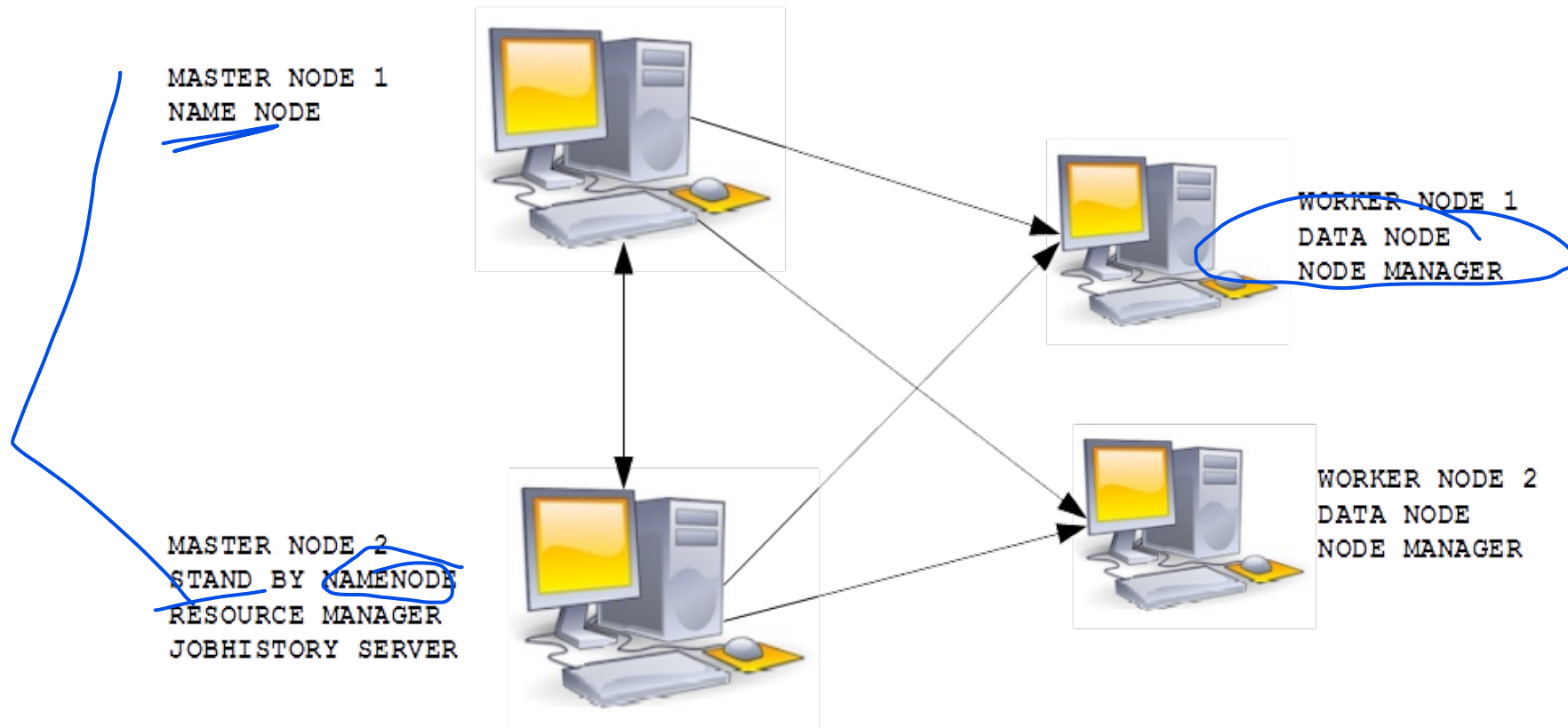
- SecondaryNameNode expedites the filesystem metadata recovery
- Standby NameNode (optional) provides high availability

DataNode

- runs HDFS slave node process
- manages block storage and access for reading or writing of data, block replication

HDFS: Hadoop Distributed File System

Architecture of HDFS



HDFS: Hadoop Distributed File System

HDFS is a virtual filesystem

- appears to a client as one file system, but the data is stored in multiple different locations
- deployed on the top of the native filesystems (such as **ext3**, **ext4** and **xfs** in Linux)

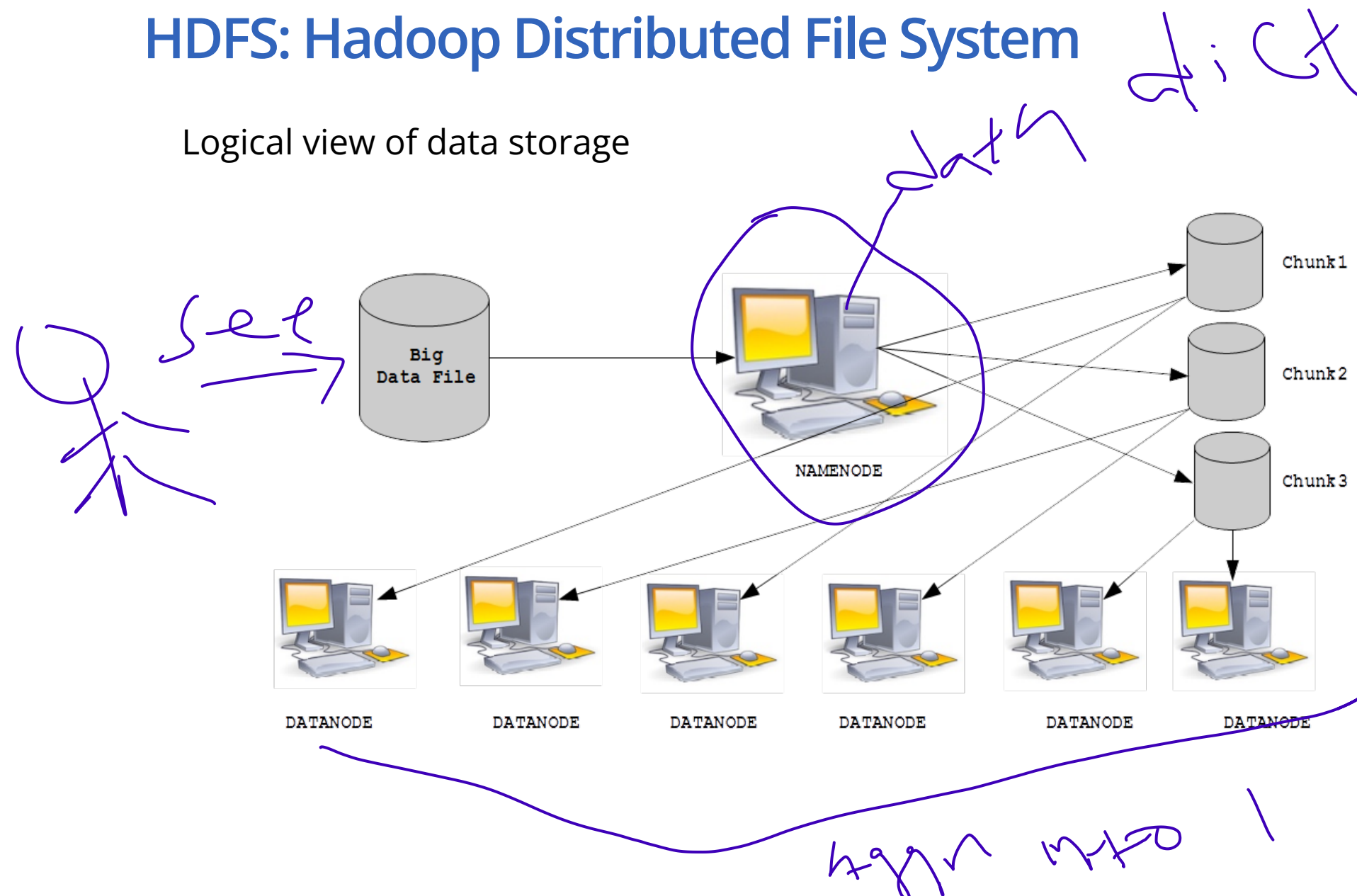
Each file in **HDFS** consists of blocks

- The size of each block defaults to 128MB but is configurable
- The default number of replicates for blocks is 3, but it is also configurable

more than 128 MB
128 MB
in 3rd file

HDFS: Hadoop Distributed File System

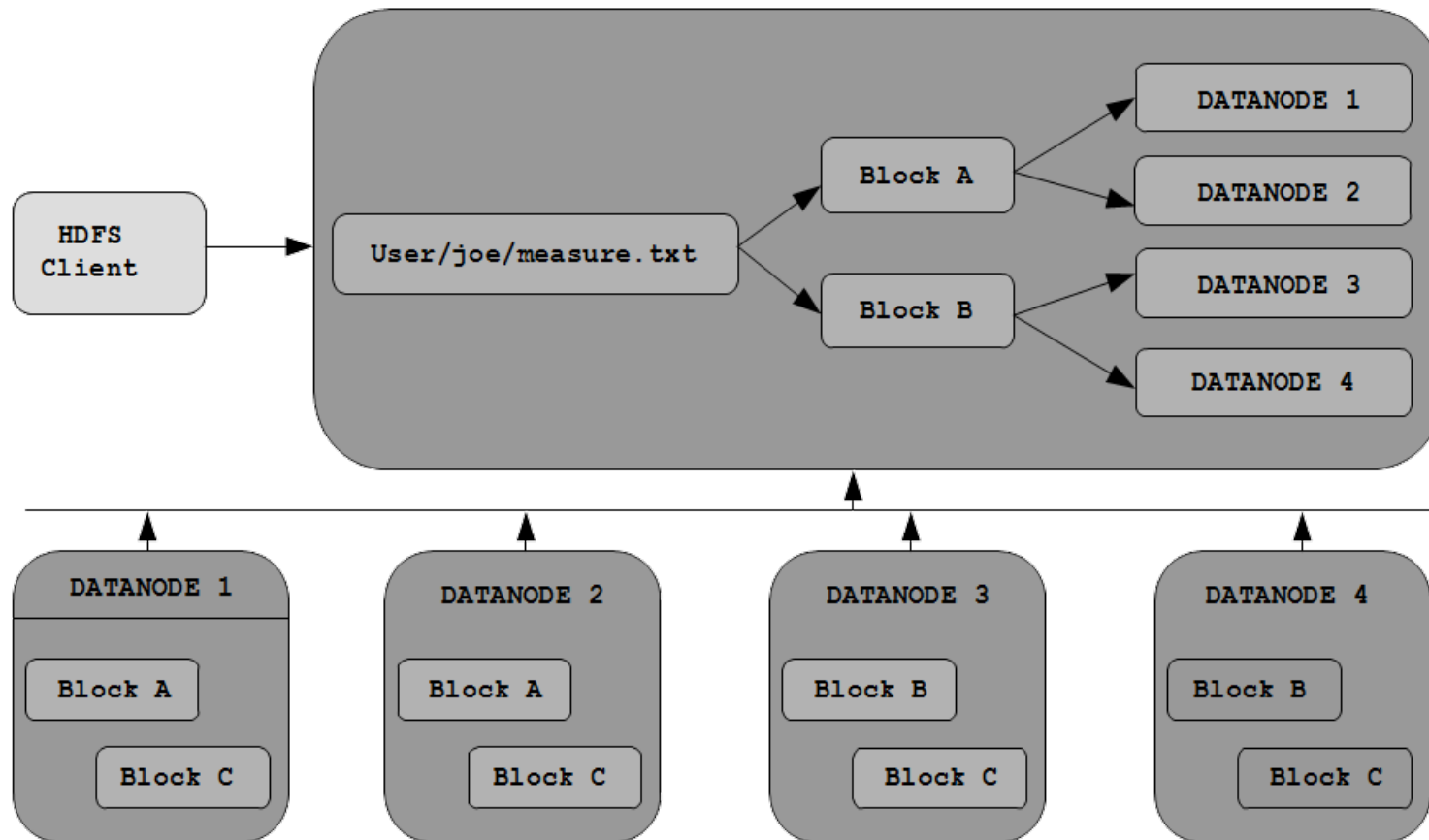
Logical view of data storage



[TOP](#)

HDFS: Hadoop Distributed File System

Physical implementation of data file storage



Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

[NodeManager](#)

[ApplicationMaster](#)

NameNode Metadata

data dict

NameNode stores the metadata of the files in HDFS

object	block_id	seq	locations	ACL	Checksum
/data/file.txt	blk_00123	1	[node1,node2,node3]	-rwxrwxrwx	8743b52063..
/data/file.txt	blk_00124	2	[node2,node3,node4]	-rwxrwxrwx	cd84097a65..
/data/file.txt	blk_00125	3	[node2,node4,node5]	-rwxrwxrwx	d1633f5c74..

NameNode functions:

- Maintain the metadata pertaining to the file system (e.g., the file hierarchy and the block locations for each file)
- Manage user access to the data files
- Map the data blocks to the DataNodes in the cluster
- Perform file system operations (e.g., opening and closing the files and directories)
- Provide registration services and periodic heartbeats for DataNodes

Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

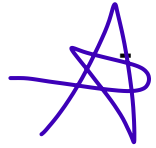
[NodeManager](#)

[ApplicationMaster](#)

DataNode and Secondary node

DataNode functions:

- Provide the block storage by storing blocks on the local file system
- Fulfil the read/write requests
- Replicating data across the cluster
- Keeping in touch with the **NameNode** by sending periodic block reports and heartbeats



A heartbeat confirms the **DataNode** is alive and healthy, and a block report shows the blocks being managed by the **DataNode**

Secondary NameNode and Standby NameNode functions:

- Without a **NameNode**, there is no way to know to which files the blocks stored on the **DataNodes** correspond to
- In essence, all files in **HDFS** are lost
- **Secondary NameNode** periodically backups the metadata in the (primary) **NameNode**, which is usually for recovery
- **Standby NameNode** is a hot node that running together with the (primary) **NameNode** in the cluster, facilitating high-availability

Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

[NodeManager](#)

[ApplicationMaster](#)

Yet Another Resource Negotiator (YARN) →

YARN: the core subsystem in Hadoop responsible for governing, allocating, and managing the finite distributed processing resources available on a Hadoop cluster

- introduced in Hadoop 2 to improve the **MapReduce** implementation, but general enough to support other distributed computing paradigms

YARN provides its core services via two types of long-running daemons:

- A **ResourceManager** (one per cluster) to manage the use of resources across the cluster, and
- **NodeManagers** running on all the **nodes in the cluster to launch and monitor containers**

hardware resources
not the software

Yet Another Resource Negotiator (YARN)

Architecture of YARN

A **client** is the program that submits jobs to the cluster

- May also be the gateway machine that the client program runs on

A **job**, also called an **application**, contains one or more tasks

- A task in a MapReduce job can be either a **mapper** and a **reducer task**

Each **mapper** and **reducer** task runs within a **container**

- **Containers** are logical constructs that **represent a specific amount of memory** and other resources, such as **processing cores (CPU)**
- For example, a **container** can represent 2GB memory and 2 processing cores
- **Containers** may also refer to the running environment of an application



Cloud computing

Yet Another Resource Negotiator (YARN)

Architecture of YARN

ResourceManager: YARN's daemon running on a master node

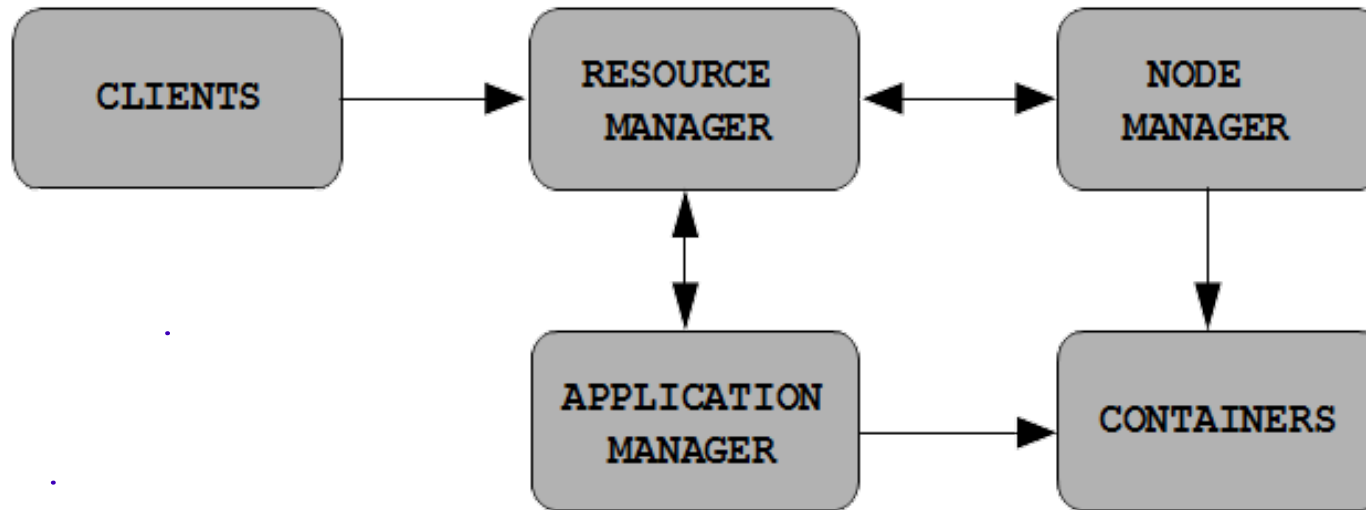
- ResourceManager is responsible for granting cluster computing resources to applications running on the cluster
- Resources are granted the items of containers

NodeManager: YARN's daemon running on a slave node.

- NodeManager manages containers on a slave node
- ApplicationMaster: the first container allocated by the ResourceManager to run on a NodeManager for each application

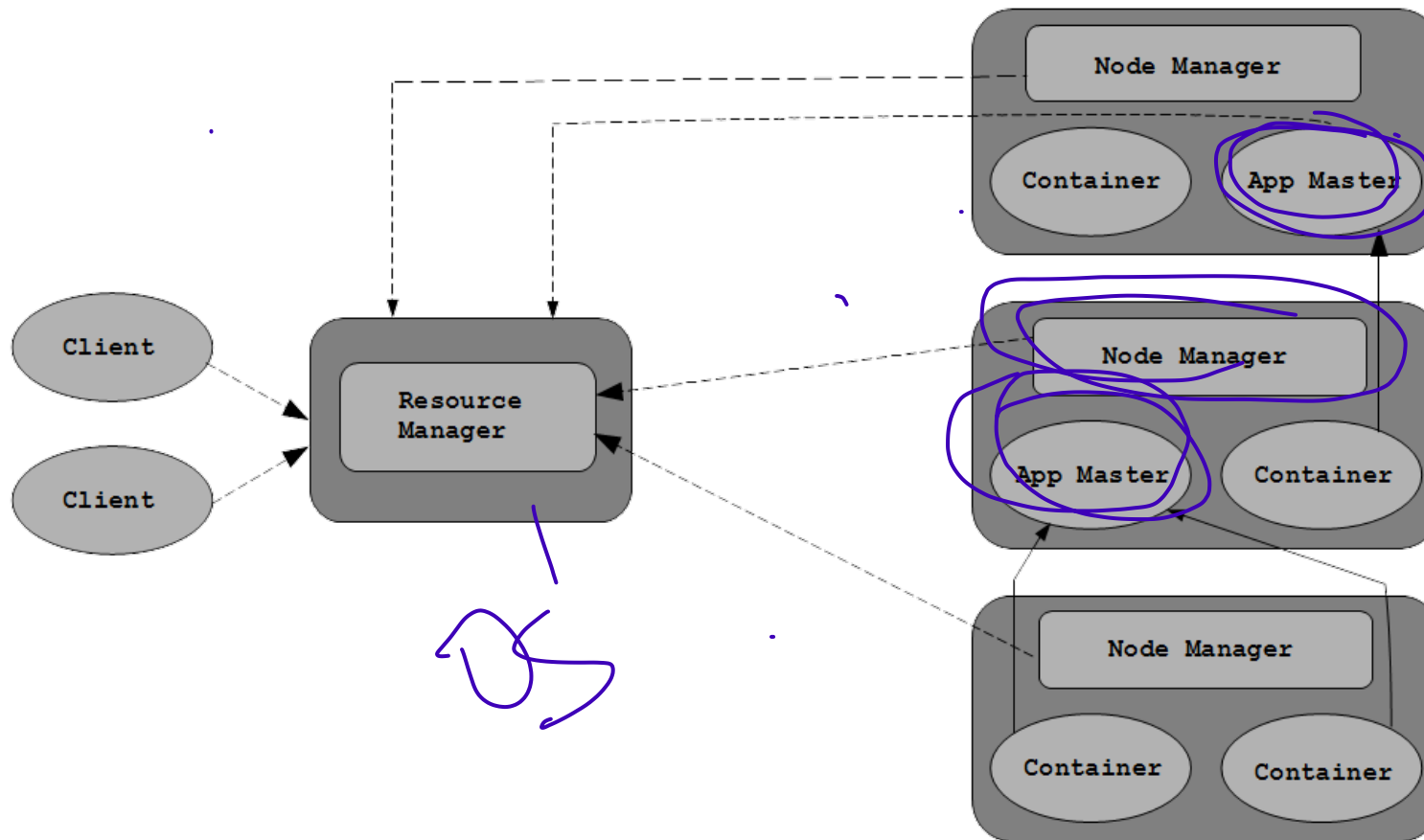
Yet Another Resource Negotiator (YARN)

Architecture of YARN



Yet Another Resource Negotiator (YARN)

Architecture of YARN



[TOP](#)

Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

[NodeManager](#)

[ApplicationMaster](#)

ResourceManager

There is one **ResourceManager** per cluster, which consists of two key components: **Scheduler** and **ApplicationManager**

Key functions of **ResourceManager**:

- Creates the first container for an application to run **ApplicationMaster** for that application
- Tracks the heartbeats from **NodeManagers** to manage **DataNodes**
- Runs **Scheduler** to determine resource allocation among the clusters
- Manages cluster level security
- Manages the resource requests from **ApplicationMasters**
- Monitors the status of **ApplicationMasters** and restarts that container upon its failure
- Deallocates the containers when the application completes or after they expire

The role of **ResourceManager** is pure management and scheduler

It does not perform any actual data processing, for example the **Map** and **Reduce** functions in a **MapReduce** application

Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

[NodeManager](#)

[ApplicationMaster](#)

NodeManager

Each **DataNode** runs a **NodeManager** daemon for performing **YARN** functions

Main functions of a **NodeManager** daemon:

- Communicates with **ResourceManager** through health heartbeats and container status notifications.
- Registers and starts the application processes
- Launches both **ApplicationMaster** and the rest of an application's resource containers (that is, the map and reduce tasks that run in the containers) on request from **ApplicationMaster**
- Oversees the lifecycle of the application containers
- Monitors, manages and provides information regarding the resource consumption (CPU/memory) by the containers
- Tracks the health of **DataNode**
- Provides auxiliary services to **YARN** applications, such as services used by the MapReduce framework for its shuffle and sort operations



Hadoop Architecture

Outline

[Hadoop Distributed File System \(HDFS\)](#)

[NameNode metadata](#)

[DataNode and Secondary node](#)

[Yet Another Resource Negotiator \(YARN\)](#)

[ResourceManger](#)

[NodeManager](#)

[ApplicationMaster](#)

ApplicationMaster

For each **YARN** application, there is a dedicated **ApplicationMaster**

Functions of **ApplicationMaster**:

- Managing task scheduling and execution
- Allocating resources locally for the application's tasks

ApplicationMaster is running within a container

ApplicationMaster's existence is associated with the running application

When an application is completed, its **ApplicationMaster** no longer exists

Once created, **ApplicationMaster** is in charge of requesting resources with **ResourceManager** to run the application

The resource request are very specific, for example:

- the file blocks needed to process the job,
- the amount of the resource, in terms of the number of containers to create for the application,
- the size of the containers, etc.

Summary

Terminologies

For convenience, we use the **names of HDFS and YARN processes** to refer to both the **hosts** and the **daemons** running on the corresponding hosts

For example, **RecourseManager** refers to both **a master node** and the **RecourseManager daemon** on that master node; **DataNode** refers to both **a slave node** and the **DataNode daemon** on that slave node.

Hadoop is a leading platform for big data

Hadoop consists of a storage layer (**HDFS**), a coordination and management layer (**YARN**) and a processing layer (e.g., **MapReduce**)

HDFS and **YARN** have key services (daemons)

MapReduce is a fundamental computing model (i.e., batch processing) for big data

Next: Interaction with **Hadoop** and "dive" into the **MapReduce** framework