CS 398 ACC
Hadoop

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim

MP1

How is it going?

- Due tomorrow at 11:59 pm.
- Latest Commit to the repo at the time will be graded.
- Last Office Hours today after the lecture until 7pm.

SSH Keys

Look for your SSH key in your Gitlab repository to access course cluster.

Please read the **Cluster Page** on the course website.

Course cluster address announced when MP2 is released

Wednesday

Wednesday Lecture Optional

- Office hours for MP2
- Troubleshooting access to the cluster

Outline

- Hadoop Overview
- Hadoop Distributed File System (HDFS) and YARN
- Hadoop Deployment Strategies
- Accessing the Course Cluster

Outline

- **Hadoop Overview**
- Hadoop Distributed File System (HDFS) and YARN
- Hadoop Deployment Strategies
- Accessing the Course Cluster

Hadoop

Open-source software for reliable, scalable, distributed computing.

Comes with:

- Hadoop Common
- Hadoop HDFS (Distributed Storage)
- Hadoop YARN (Resource Manager / Task Scheduler)
- Hadoop MapReduce (MapReduce Programming Model)

What Hadoop does for you

Hadoop abstracts:

- Parallelization
- Scheduling
- Resource Management
- Inter-machine communication
- Handling software/hardware failures
- Etc.

Reasoning: Extract common cloud computing primitives to build reliable cloud applications faster, and more reliably

Outline

- Hadoop Overview
- **Hadoop Distributed File System (HDFS) and YARN**
- Hadoop Deployment Strategies
- Accessing the Course Cluster

HDFS

What problem are we trying to solve?

- Distributing / replicating data to a multi-node cluster
- Replicating data intelligently, minimizing bandwidth
- Keeping replicas in multiple racks / geographic regions

HDFS is the Data Management Layer of Hadoop.

- Scalable, fault-tolerant, cost-efficient, data storage

Assumptions and Goals in Design

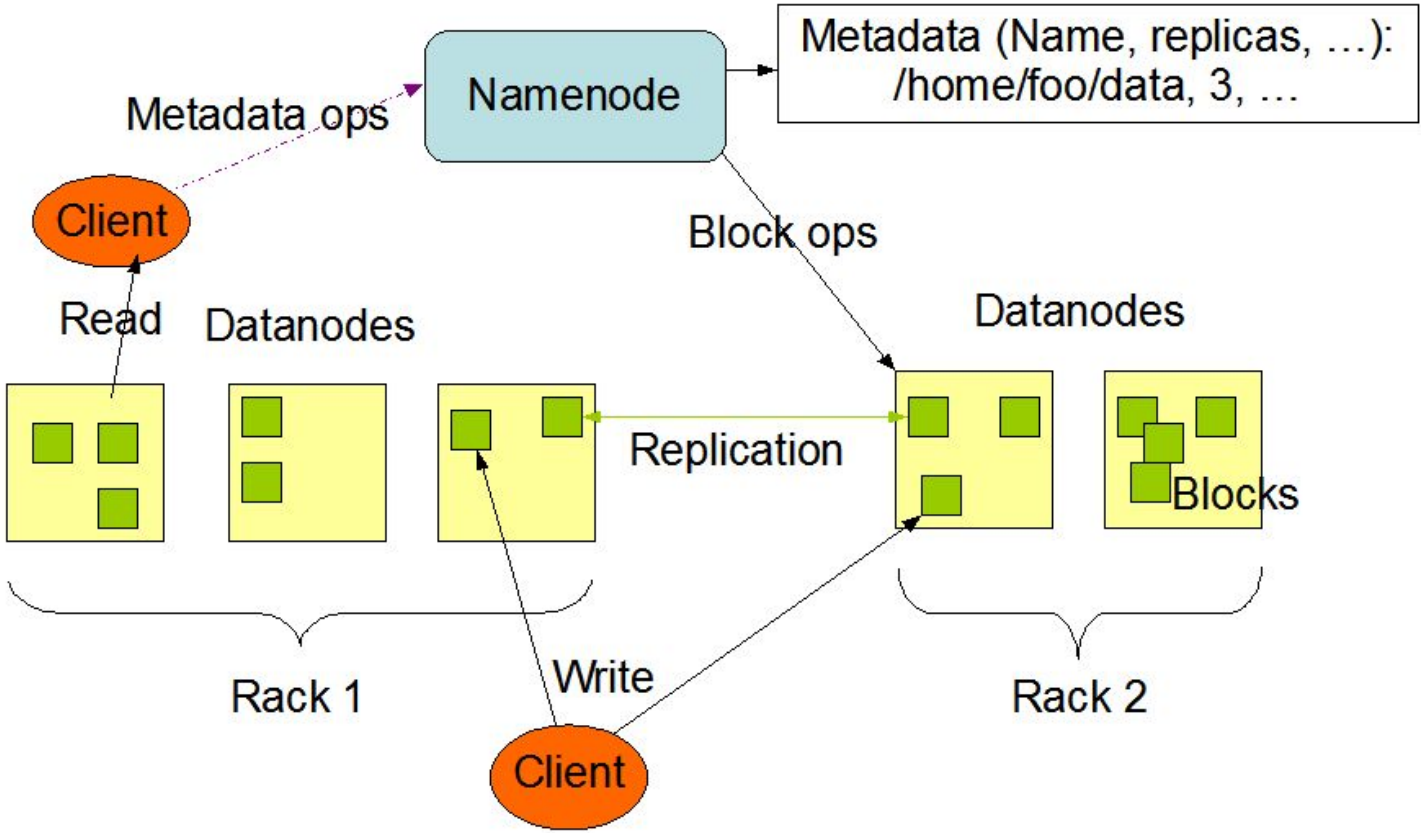
HDFS assumes there will be:

- Hardware Failure
- Large Datasets

HDFS aims for:

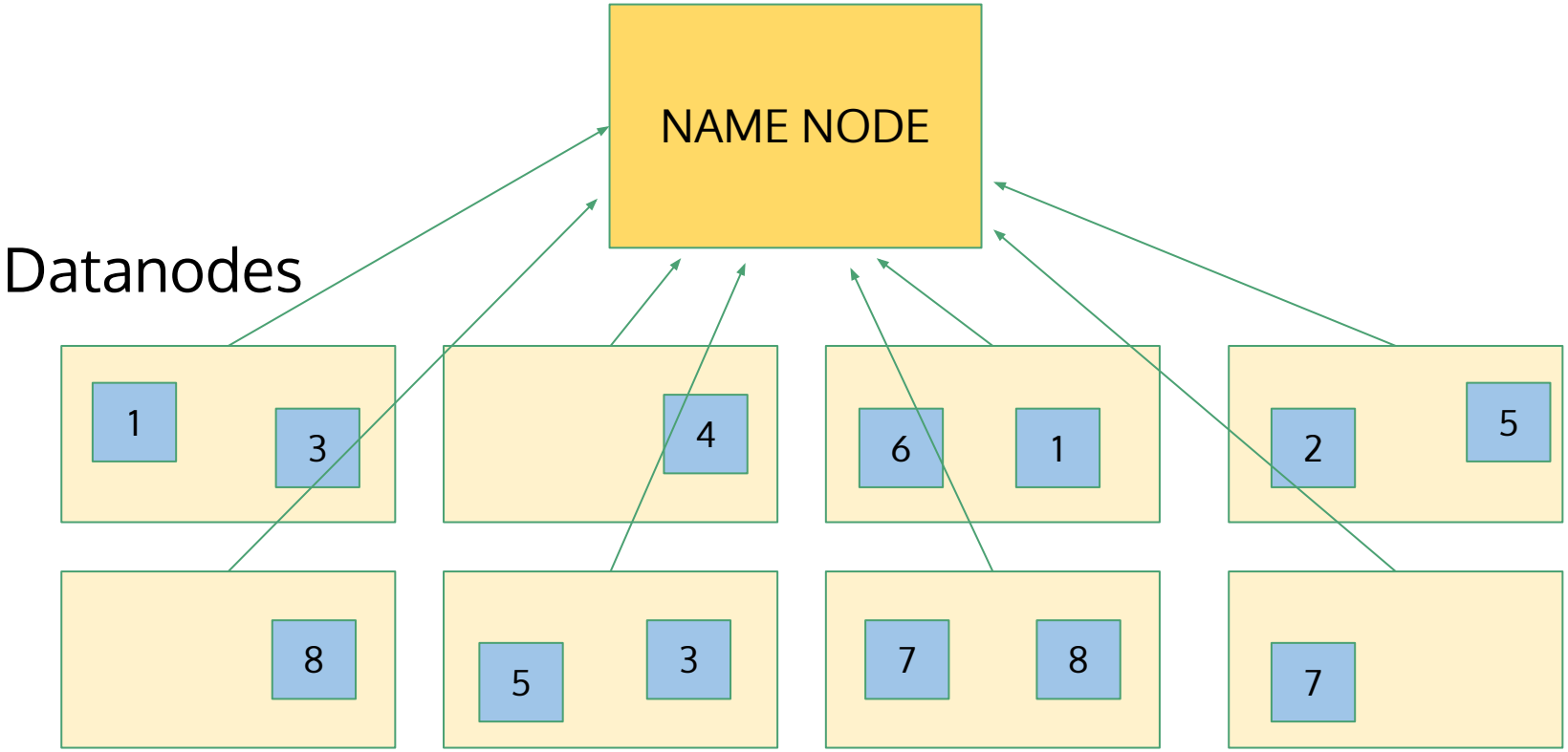
- Fault-Tolerance
- Portability Across Heterogeneous Hardware and Software Platforms
- Optimized sequential operations
- Streaming Data Access

HDFS Architecture



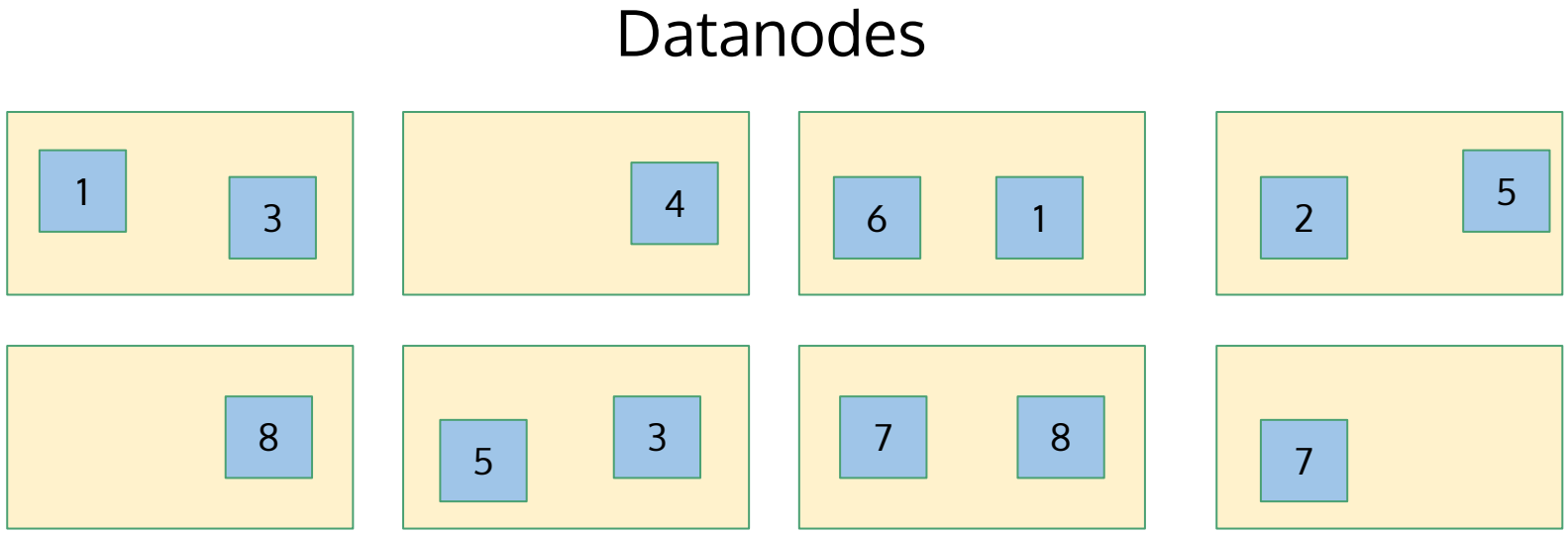
https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

HDFS - Data Replication

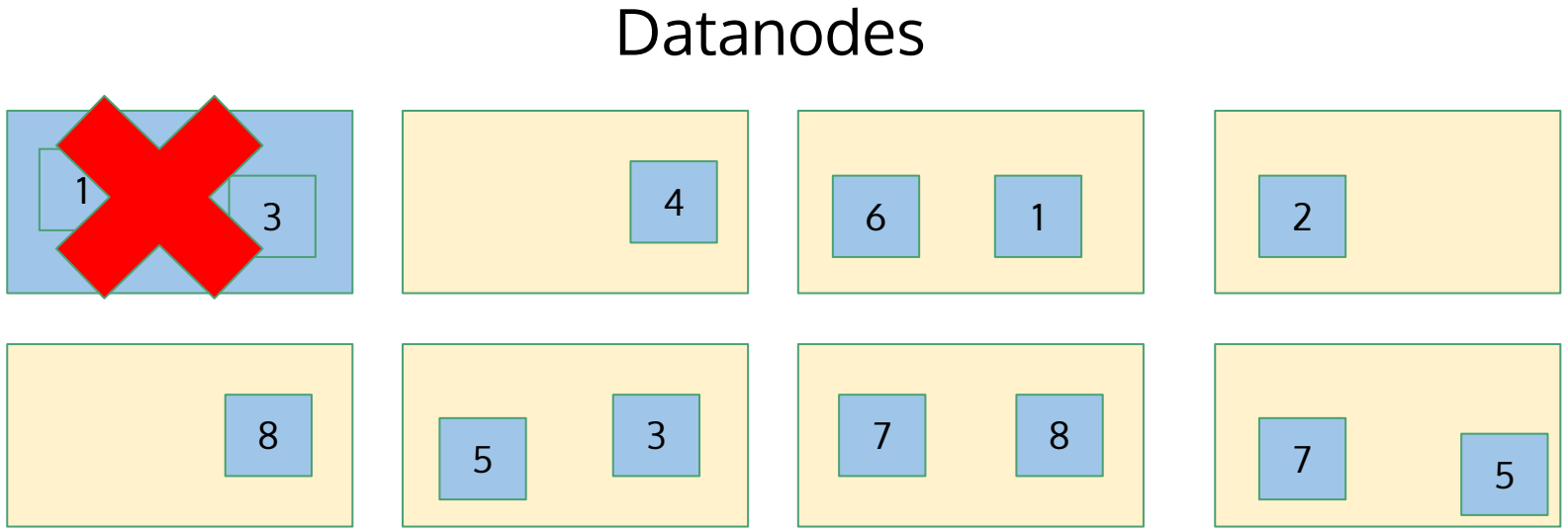


HDFS - Data Replication

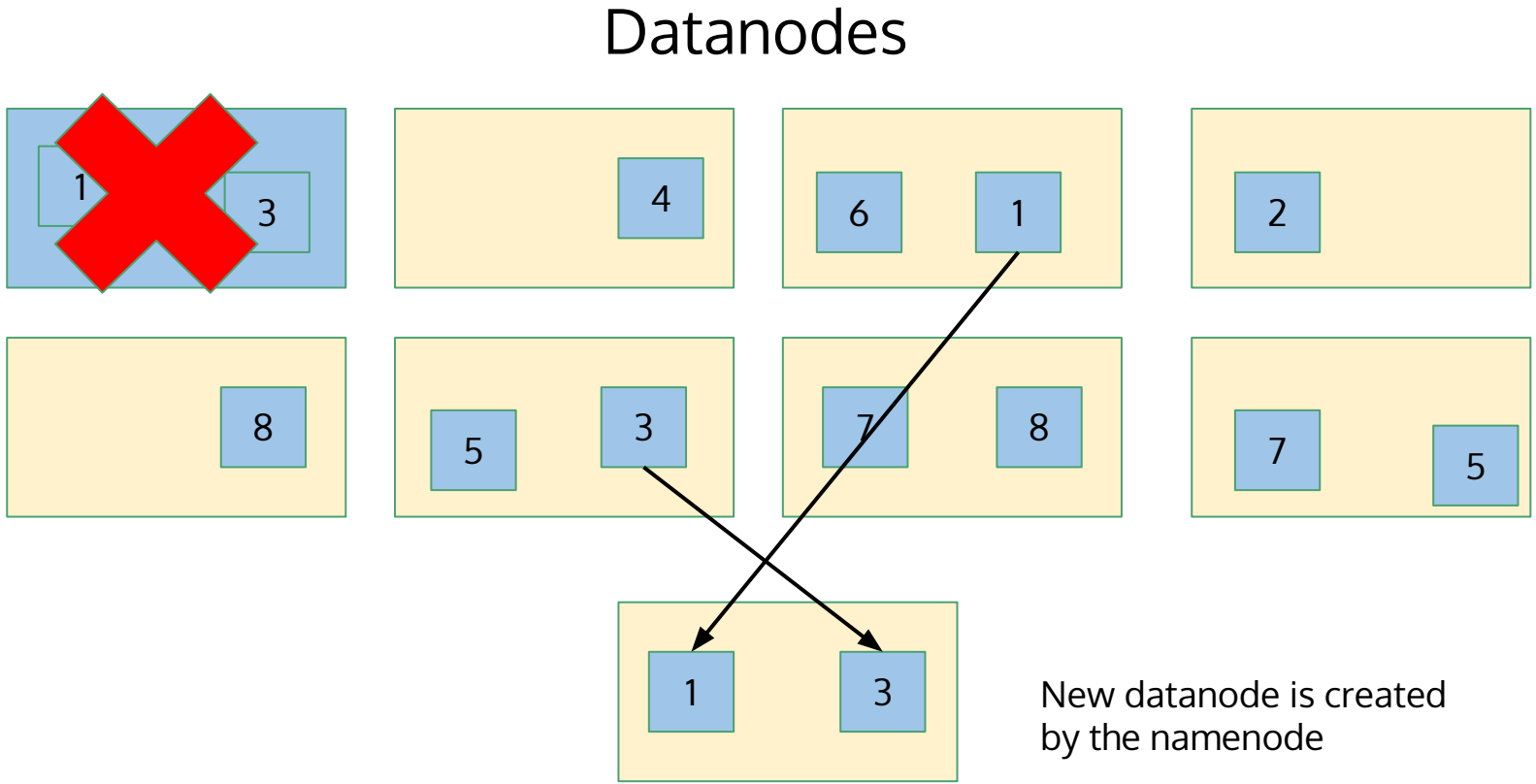
HDFS can reliably store very large files across machines.
The blocks of files are replicated for fault tolerance.



HDFS - Data Replication



HDFS - Data Replication



HDFS - Data Replication

- How do we tell HDFS to replicate files?
  - Set replication factor for file/directory
    - “hdfs dfs -setrep <num_replicas> /path/to/file”
  - Hadoop will increase/decrease number of replicas accordingly
  - Maximum achievable replication is equal to number of datanodes in the cluster

YARN - Yet Another Resource Negotiator

A framework for **job scheduling** and **cluster resource management.** *“How a job gets a container”*

- Underneath Hadoop 2.x+
- Negotiates resources and schedules jobs
- Treats each server as a collection of **containers**.
  - Each container can run different types of tasks and might be different sizes.
- Users can submit any type of application supported by YARN.

YARN - Yet Another Resource Negotiator

A framework for **job scheduling** and **cluster resource management.** *“How a job gets a container”*

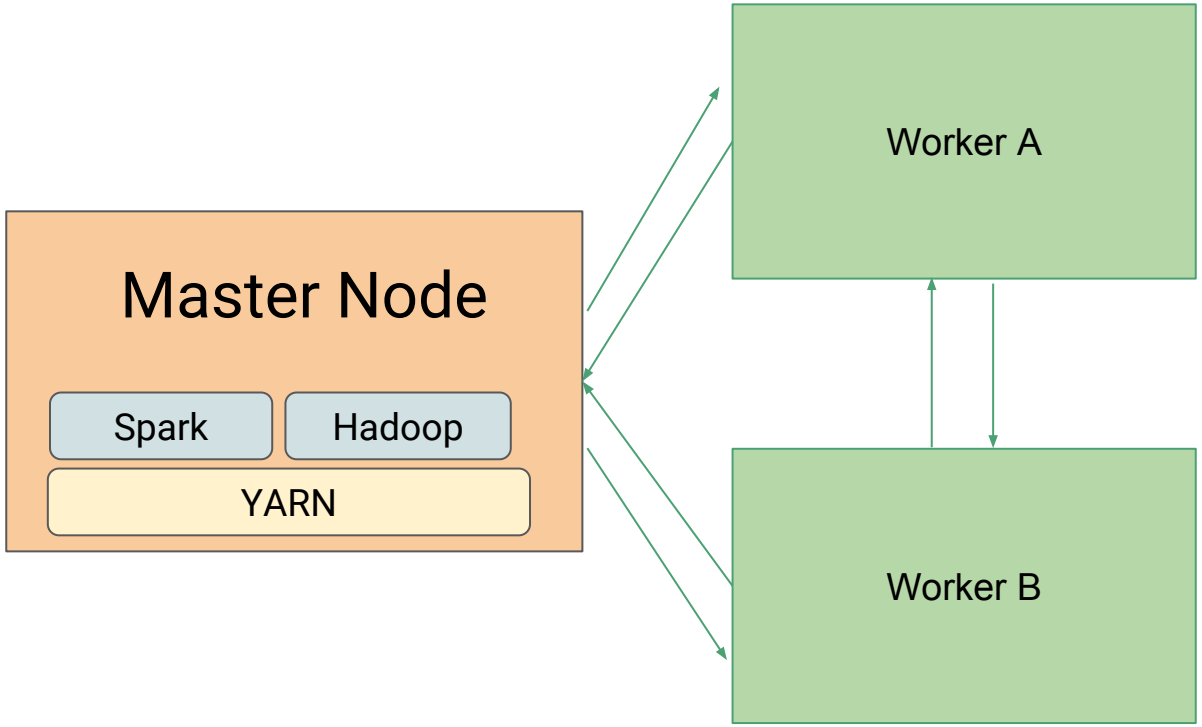
- Not just for MapReduce!
  - Used by Spark, Tez, Pig, and others

YARN

Three Main Components:

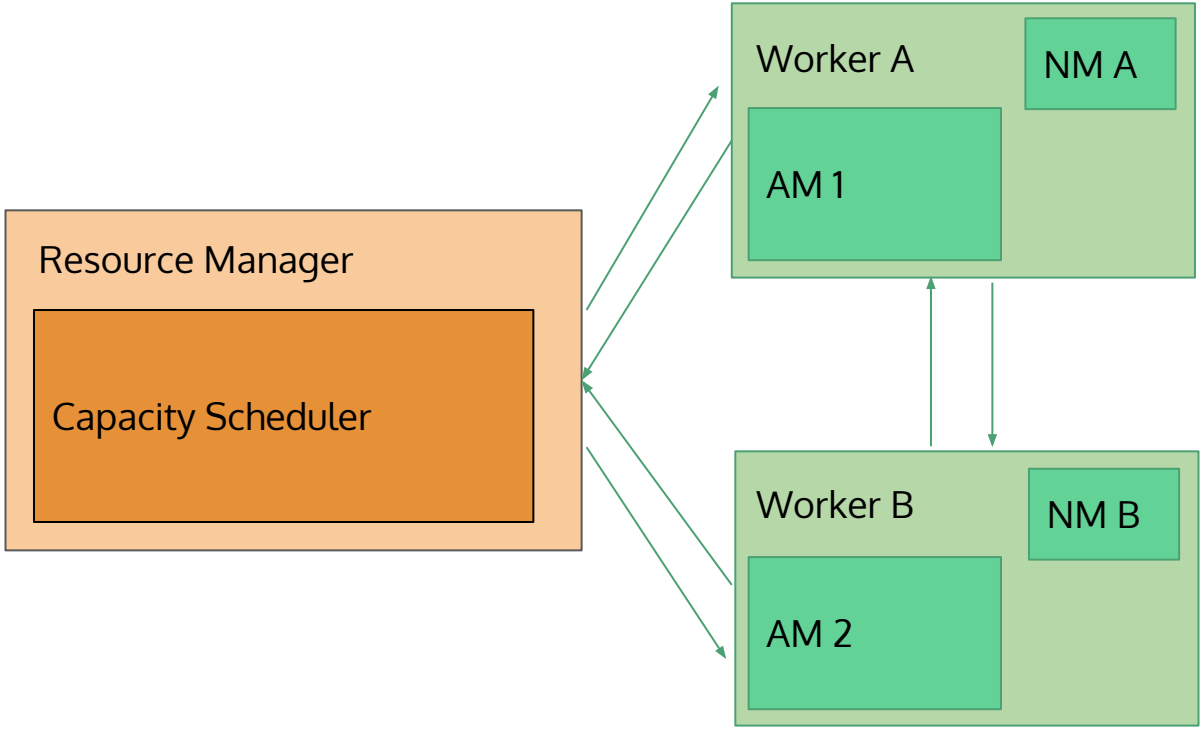
- Global **Resource Manager (RM)**
  - Keeps track of live NodeMangers and available resources.
  - Allocates available resources to appropriate applications.
- Per-Server **Node Manager (NM)**
  - Provides computational resources in forms of containers.
  - Manages processes running in containers.
- Per-Application **Application Master (AM)**
  - Coordinates the execution of all tasks within its application.
  - Asks for appropriate resource containers to run applications/tasks.

YARN inside the course cluster



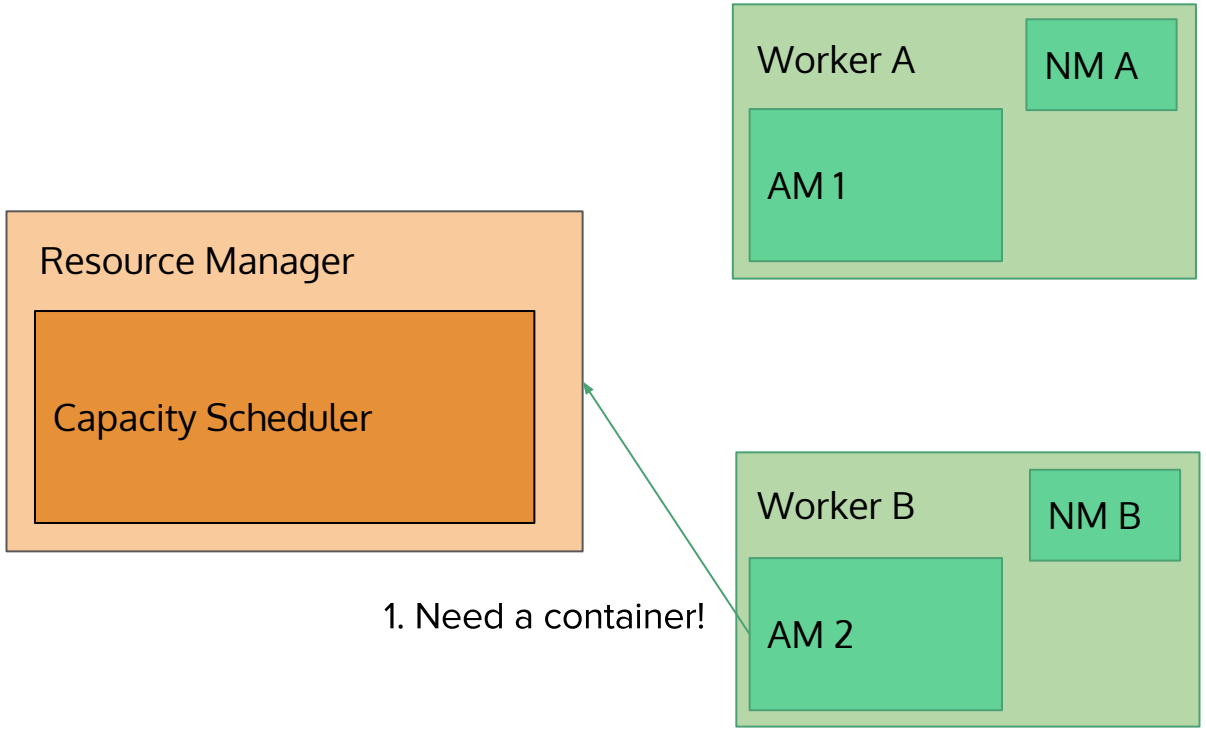
YARN inside the course cluster

AM = Application Master
NM = Node Manager



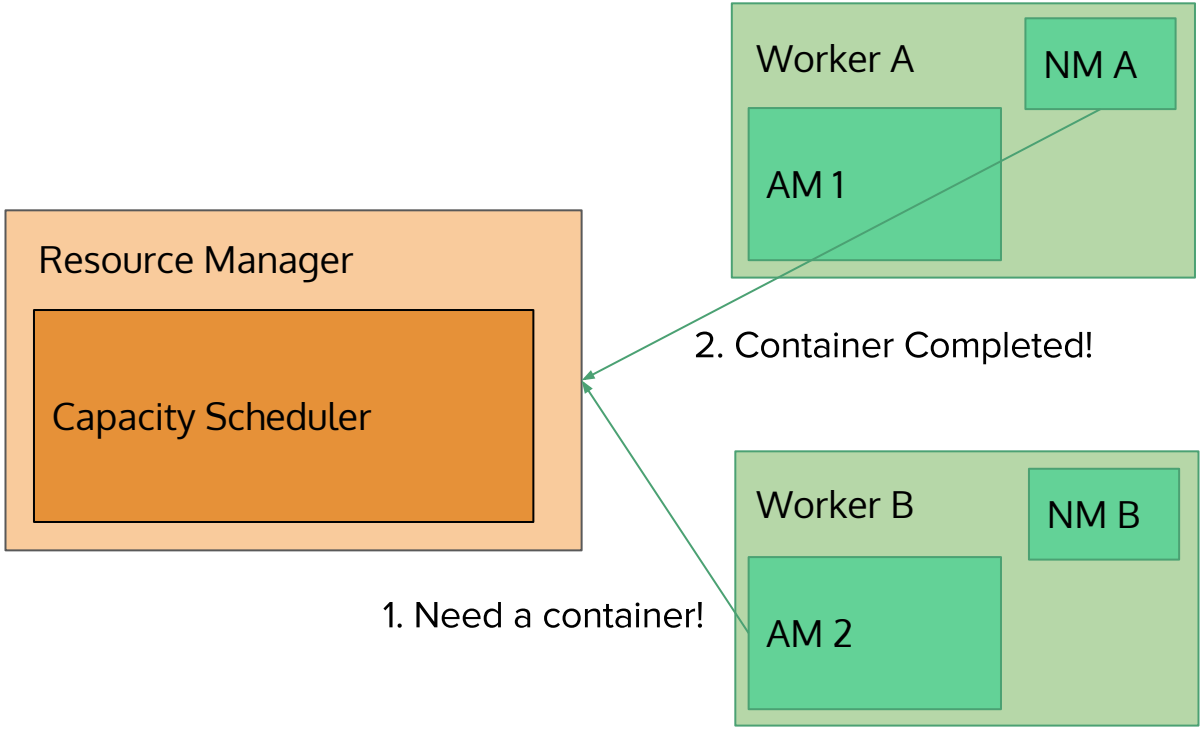
YARN inside the course cluster

AM = Application Master
NM = Node Manager



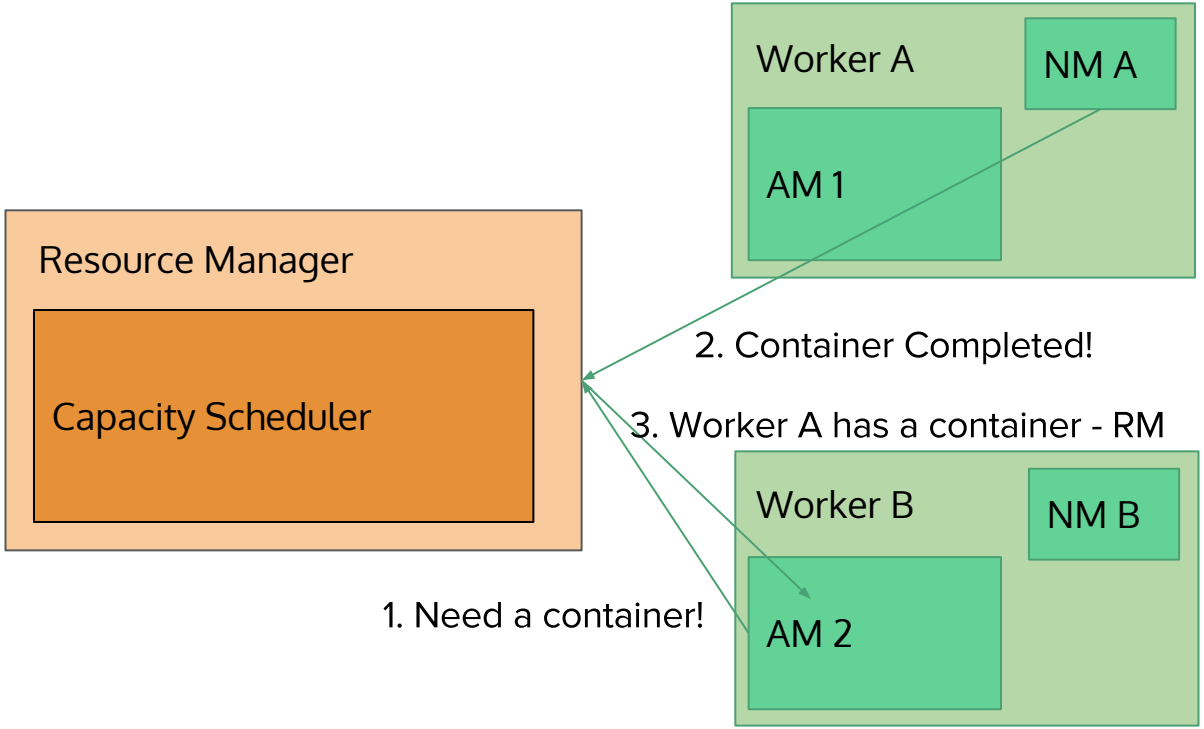
YARN inside the course cluster

AM = Application Master
NM = Node Manager



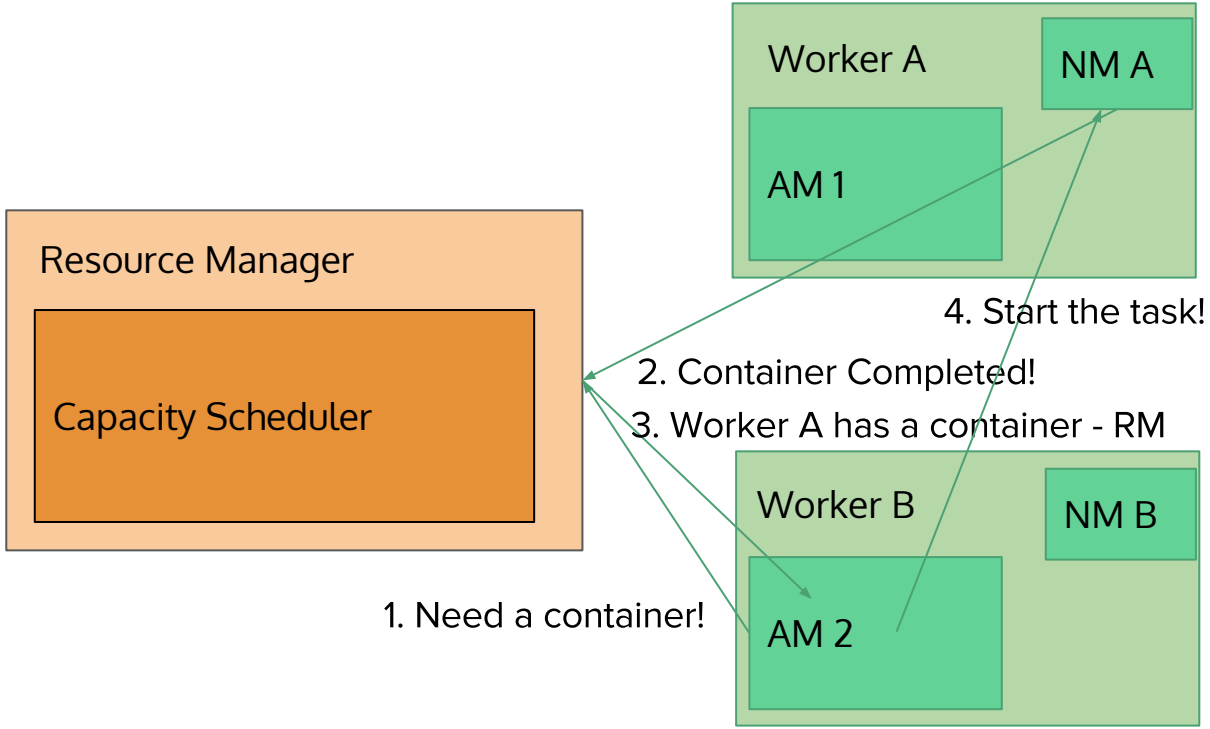
YARN inside the course cluster

AM = Application Master
NM = Node Manager



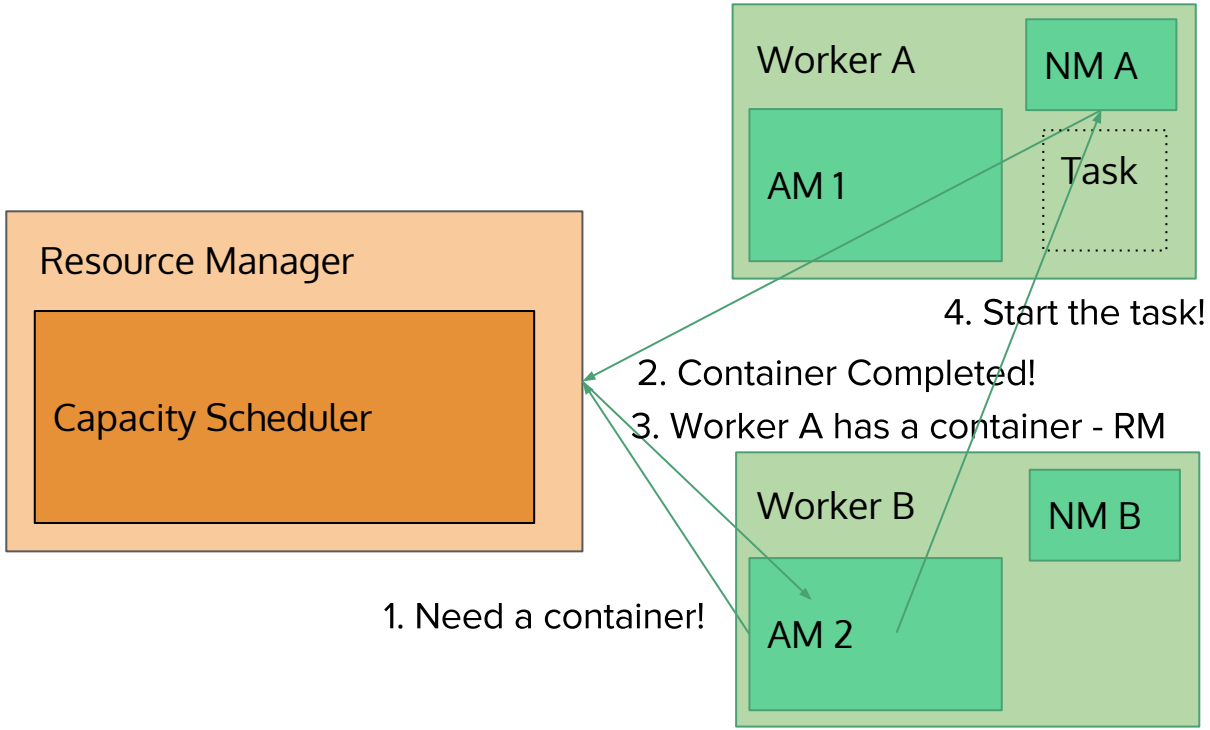
YARN inside the course cluster

AM = Application Master
NM = Node Manager



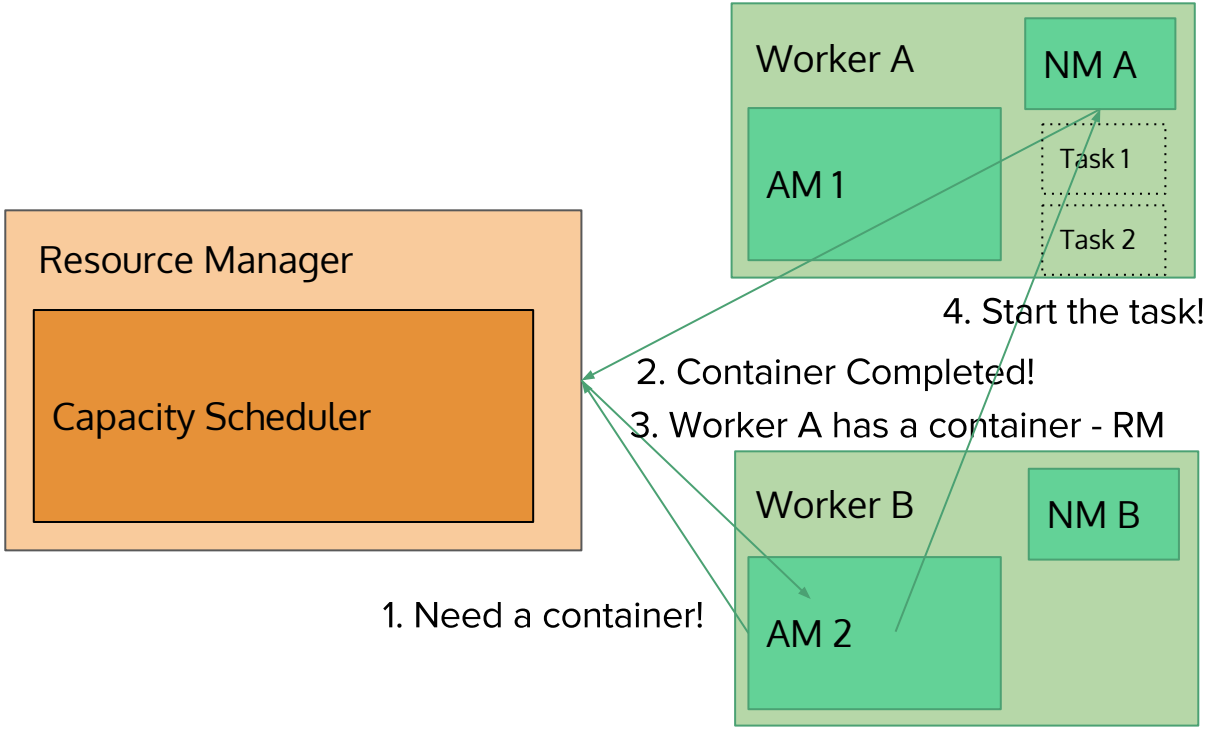
YARN inside the course cluster

AM = Application Master
NM = Node Manager



YARN inside the course cluster

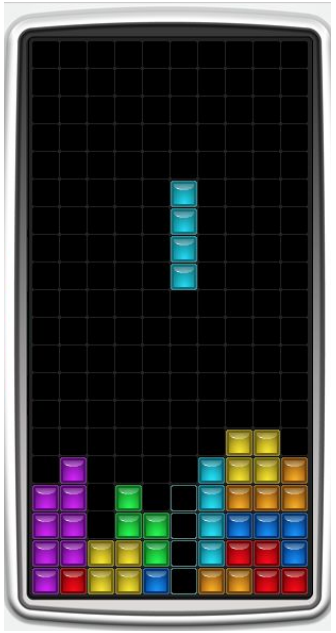
AM = Application Master
NM = Node Manager



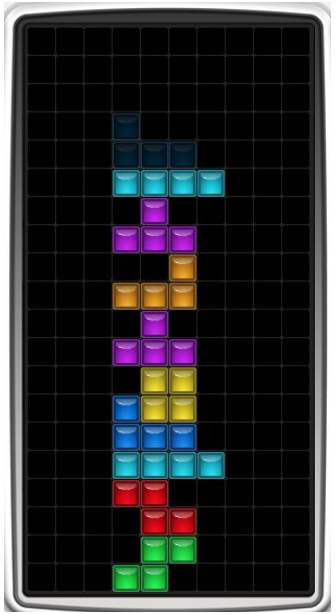
Why is it so important?

- Performance
- Latency
- Cost Efficiency

With a scheduler



Without a scheduler



Other Scheduler Considerations

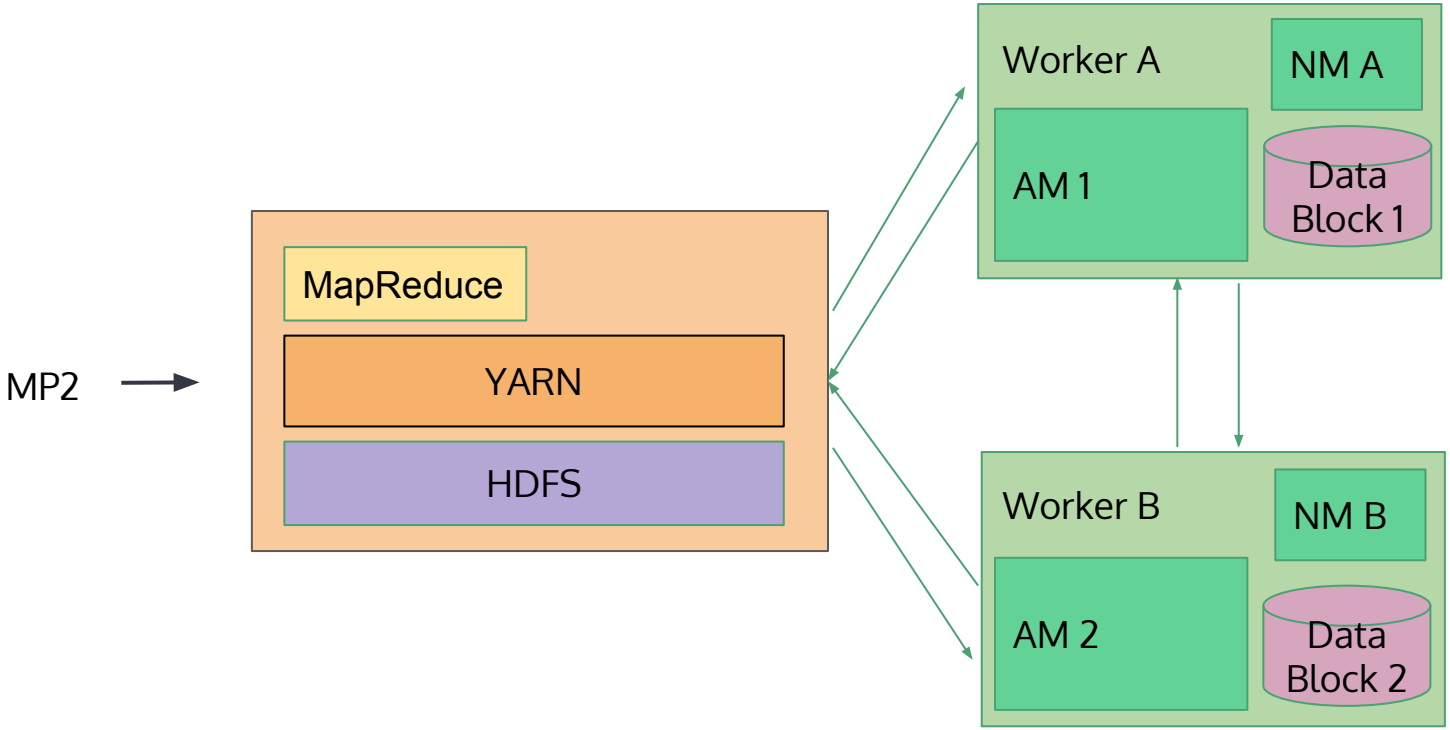
- Data Locality
  - Many worker nodes are also Data nodes
  - Try to schedule jobs on workers that *already* have a copy of the input data
  - Why? Then we don’t have to transfer data to the worker on startup

Fault Tolerance

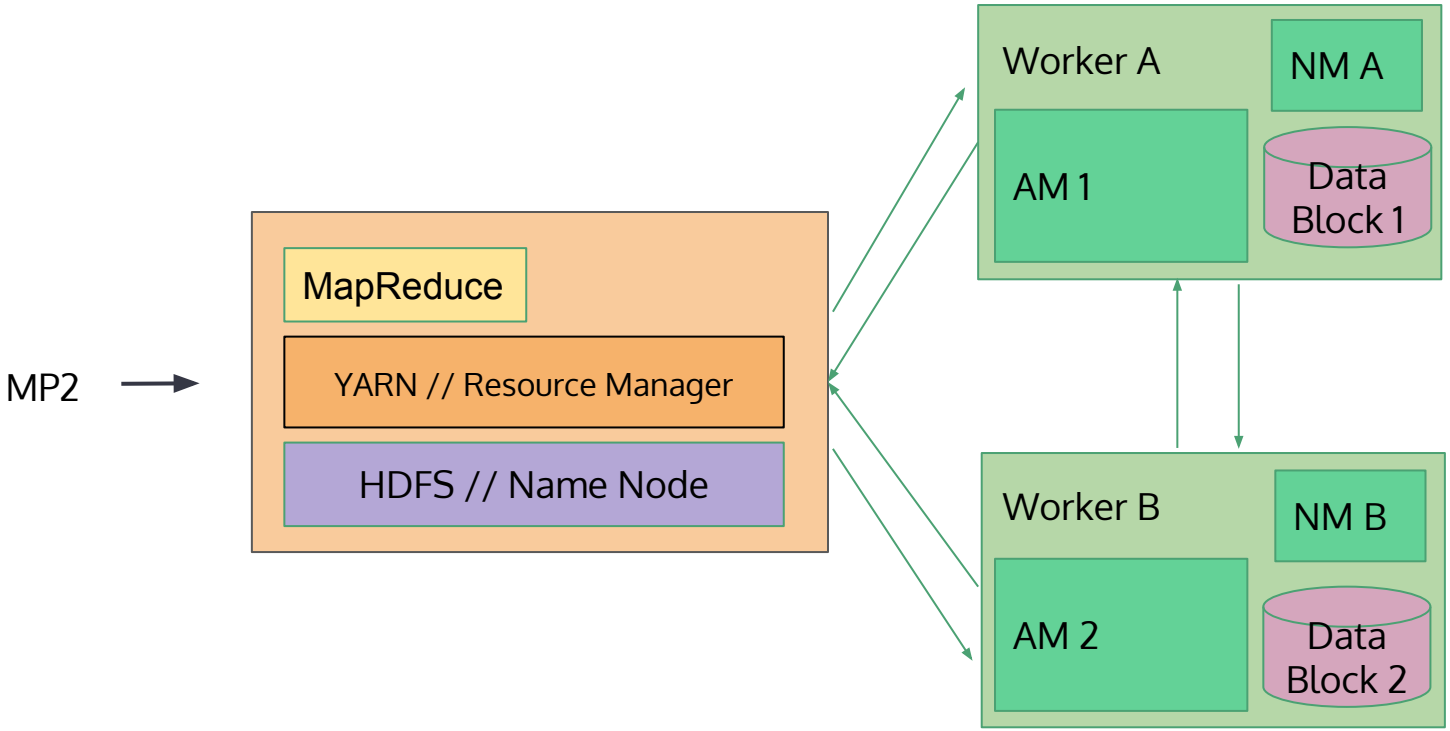
```
RM = Resource Manager
AM = Application Master
NM = Node Manager
```

- Node Failure
  - **No heartbeat from NM to RM**
    - RM let all AMs know
  - **NM fails to keep track of each task in the node**
    - If task fails while in-progress, mark the task as idle and restart them
  - **No heartbeat from AM to RM**
    - RM restarts AM, syncs up with its running tasks
- Resource Manage Failure (on Master)
  - Secondary RM gets spun up.
  - Use last checkpoint and sync up.

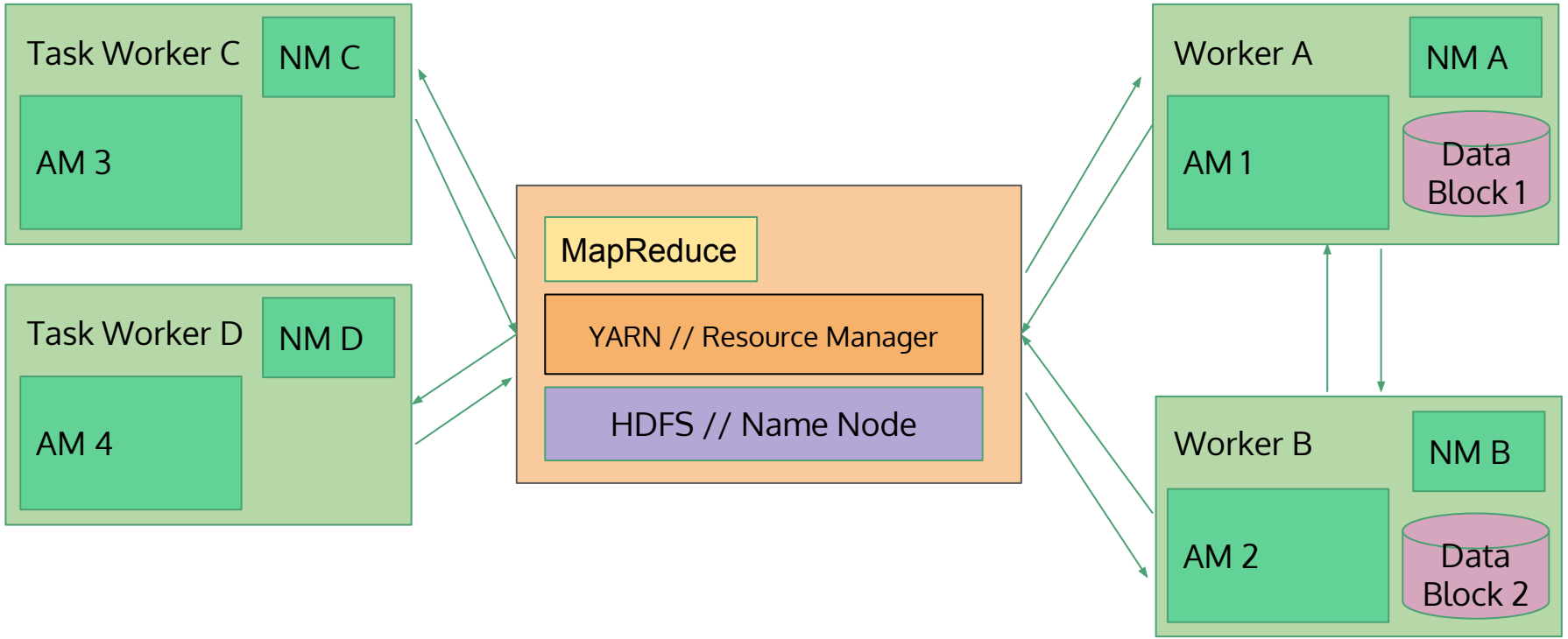
How does it look as a whole?



How does it look as a whole?



How does it look as a whole?



Outline

- Hadoop Overview
- Hadoop Distributed File System (HDFS)
- **Hadoop Deployment Strategies**
- Accessing the Course Cluster

Submitting Hadoop jobs

- Input/Output **must** reside in HDFS
- Program code can live in the normal filesystem
  - Programs can be copied to all nodes upon job startup, because they're small

Submitting a streaming job to Hadoop:

```
$ hadoop jar $STREAMING_JAR \
    -files mapper.py,reducer.py
    -mapper mapper.py -reducer reducer.py \
    -input /shared/my_cool_dataset -output my_job_out
```

Hadoop Jobs on course cluster

- ~~We~~ MRJob made things a bit easier for you:

```
$ python my_mr_job.py -r hadoop
    --output-dir my_job_out   # HDFS directory
    --no-output               # Suppress output via STDOUT
    hdfs:///data/my_cool_dataset     # HDFS path for input
```

Hadoop Jobs

- Where is your output stored? In HDFS
  - Check your output directory for files named part-XXXXX
- Hadoop Job Counters
  - Displayed after job completes
  - Can be viewed during job in the Hadoop web interface
  - Information about: HDFS usage, MapReduce records and I/O

Common Pitfalls

- Make sure HDFS output directory doesn't already exist
  - Remove directories with "hdfs dfs -rm -r /path/to/directory"

- "My job isn't starting" / "My job is stalled"
  - There may be others in front of you in the queue
  - Check the Hadoop web interface (instructions in the MP2 doc)
  - Or, run "yarn application -list" to see if other jobs are running

- "My job crashed with a 'subprocess failed' error"
  - This is almost always an error in *your* code
  - If your program crashes for *any line* of input, this can crash the whole job
  - Solution: Test locally where you can get better logging

Outline

- Hadoop Overview
- Hadoop Distributed File System (HDFS)
- Hadoop Deployment Strategies
- **Accessing the Course Cluster**

Course Cluster Warnings

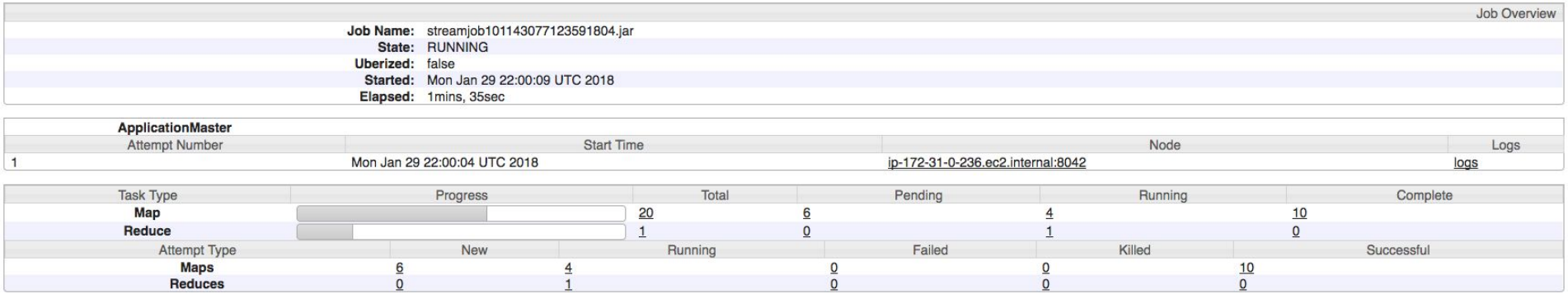
- Do not share your AWS credentials with anyone.
- Do not do other course work on the course cluster.
- Do not run any other side projects with the course cluster.
- Do not use excessive resources or break the cluster.

We monitor the cluster usages closely, and the instructors reserve the **right to revoke your cluster privilege** if deemed necessary.

Hadoop Web Interfaces

- Check MP2 documentation
- Very useful/informative for viewing real-time progress of your jobs

Hadoop Web Interfaces



Job Overview

Job Name:	streamjob10114307712359180 4.jar
State:	RUNNING
Uberized:	false
Started:	Mon Jan 29 22:00:09 UTC 2018
Elapsed:	1mins, 35sec

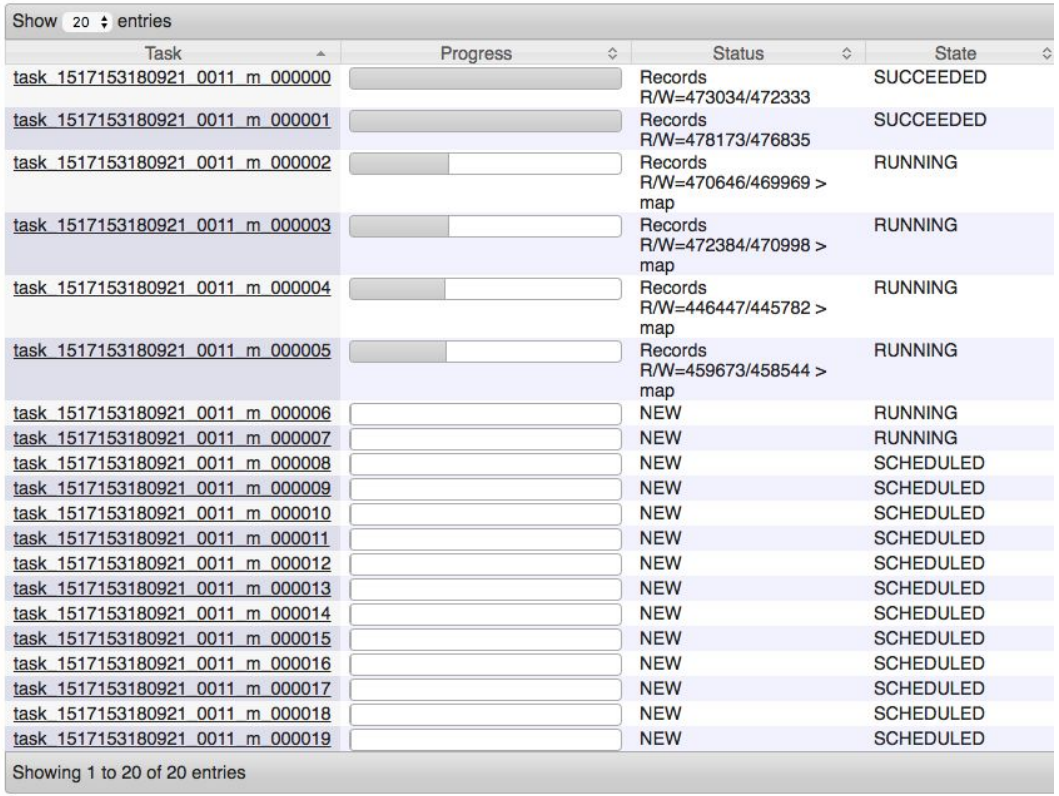
ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Mon Jan 29 22:00:04 UTC 2018	ip-172-31-0-236.ec2.internal:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map		20	6	4	10
Reduce		1	0	1	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	6	4	0	0	10
Reduces	0	1	0	0	0

Hadoop Web Interfaces



Show 20 entries

Task	Progress	Status	State
task_1517153180921_0011_m_000000		Records R/W=473034/472333	SUCCEEDED
task_1517153180921_0011_m_000001		Records R/W=478173/476835	SUCCEEDED
task_1517153180921_0011_m_000002		Records R/W=470646/469969 > map	RUNNING
task_1517153180921_0011_m_000003		Records R/W=472384/470998 > map	RUNNING
task_1517153180921_0011_m_000004		Records R/W=446447/445782 > map	RUNNING
task_1517153180921_0011_m_000005		Records R/W=459673/458544 > map	RUNNING
task_1517153180921_0011_m_000006		NEW	RUNNING
task_1517153180921_0011_m_000007		NEW	RUNNING
task_1517153180921_0011_m_000008		NEW	SCHEDULED
task_1517153180921_0011_m_000009		NEW	SCHEDULED
task_1517153180921_0011_m_000010		NEW	SCHEDULED
task_1517153180921_0011_m_000011		NEW	SCHEDULED
task_1517153180921_0011_m_000012		NEW	SCHEDULED
task_1517153180921_0011_m_000013		NEW	SCHEDULED
task_1517153180921_0011_m_000014		NEW	SCHEDULED
task_1517153180921_0011_m_000015		NEW	SCHEDULED
task_1517153180921_0011_m_000016		NEW	SCHEDULED
task_1517153180921_0011_m_000017		NEW	SCHEDULED
task_1517153180921_0011_m_000018		NEW	SCHEDULED
task_1517153180921_0011_m_000019		NEW	SCHEDULED

Showing 1 to 20 of 20 entries

MP 2

Read the MP Documentation Carefully

We won’t answer Piazza questions if the answers are in the doc.

MP 2

Due in next Tuesday (2/6) at 11:59pm

“Hadoop MR with real DataSets on a Hadoop cluster “

Please start early!

Cluster can get crowded near the deadline.

> Check Piazza for Q&A and Announcements