Cluster Architecture
Where do I store block 1?

data nodes 03, 05, 08

client node

name node

data nodes 01, 02, 04, 06, 07

data node 01

data node 02

data node 03

data node 04

data node 05

data node 06

data node 07

data node 08

data node 09

data node 10

data node 11

data node 12

rack 1

rack 2

rack 3

blk 1 (03, 05, 08)

blk 1 (03, 05, 08)

blk 1 (03, 05, 08)

blk 1 (03, 05, 08)
MapReduce architecture

1. MR v1
2. MR v2 (YARN)
MR (v1) Architecture

1: run MR program
2: get new job ID
3: copy job resources
4: submit job
5: initialize job
6: retrieve input splits
7: heartbeat (returns task)
8: retrieve job resources
9: launch child JVM
10: run map or reduce task

Client JVM

TaskTracker

Child JVM

HDFS
Shortcomings of MR v1

1. Jobtracker is a **SPOF**!

2. Jobtracker **limits scalability**
   - 1. max. cluster size: **4000 nodes** (Yahoo)

3. Jobtracker responsible for both **scheduling** and **tracking**
1. YARN (yet another resource negotiator)

2. separates responsibilities

   1. Scheduling: **ResourceManager**

   2. Tracking: **Application Masters**
1. YARN (yet another resource negotiator)

2. separates responsibilities

   1. Scheduling: **Resource Manager**

   2. Tracking: **Application Masters**

   restores scalability
MR (v2) Architecture

1: run

2: get new application ID

4: submit applications

3: copy job resources

7: retrieve input splits

10: retrieve job resources

11: run

5a: start container

5b: launch

6: initialize job

8: allocate resources

9a: start container

9b: launch

1: run

2: get new application ID

4: submit applications

3: copy job resources

7: retrieve input splits

10: retrieve job resources

11: run

5a: start container

5b: launch

6: initialize job

8: allocate resources

9a: start container

9b: launch

HDFS

ResourceManager

NodeManager

MR program

job

client JVM

task JVM

map or red task

YARN child

NodeManager

MR-AppMaster

Node Manager

resource manager node

node manager node

other node manager nodes
1. Manages resources such as **memory**

2. Sub-component: **scheduler**
   
   1. **capacity** or **fair**
   
   2. Allocates **containers** for AM & tasks
MR v2: Resource Manager

1. manages resources such as **memory**

2. sub-component: **scheduler**
   
   1. **capacity** or **fair**
   
   2. allocates **containers** for AM & tasks

allows for heterogenous hardware; no more “slots”
MR v2: Application Master

1. manages application lifecycle
   1. **task** coordination: request containers from RM
   2. **monitoring** (via heartbeat)
   3. **counter** management

2. per-application
MR v2: Node manager

1. Responsible for actual **execution** of tasks

2. Tracks task **progress**
MR v2: Misc

1. Job history server
2. Shuffle service
3. co-existence w/
   1. other Hadoop versions
   2. non-MR apps (e.g., Impala)