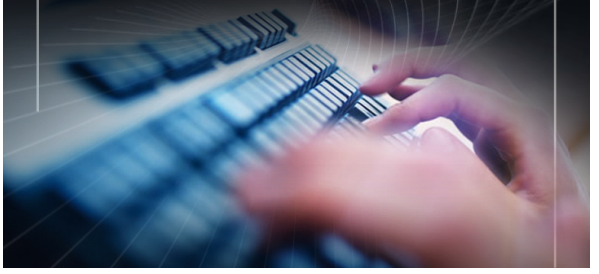
MAPREDUCE & YARN

Outline

- MapReduce continued
- YARN
- MapReduce at work

MAPREDUCE CONTINUED



Network topology and Hadoop

- Proximity of nodes: bandwith is the key
  - Limiting factors is transfer rate between node: bandwidth is a scarce resource

- BANDWIDTH IS THE MEASURE OF DISTANCE
- BUT… bandwidth cannot be measured easily

- Bandwidth become less for:
  - Processes on the same node
  - Different nodes on the same rack
  - Nodes on different racks in the same data center
  - Nodes in different data centers

Example

Data on DataCenter i, in Rack j, in Node k

Distance (Di/Rj/Nk,Dl/Rm/Nn)

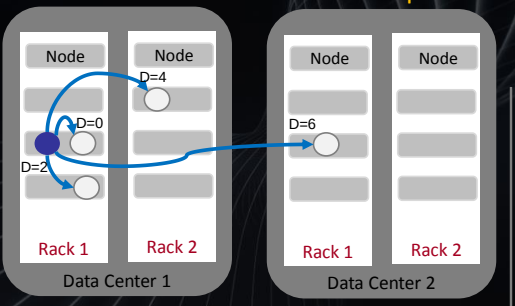
D(D1/R1/N1, D1/R1/N1) = 0

D(D1/R1/N1, D1/R1/N2) = 2

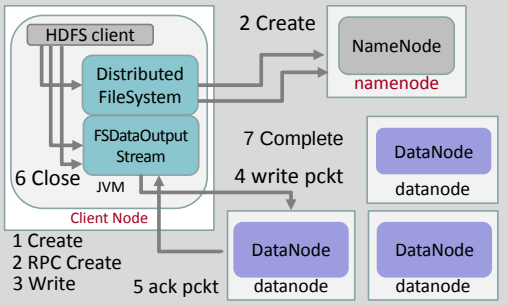
D(D1/R1/N1, D1/R2/N3) = 4

D(D1/R1/N1, D2/R3/N4) = 6

Network distance in Hadoop



Anatomy of a file write



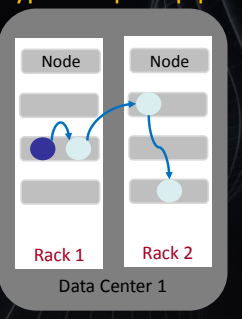
Replica placement

- Storage of replicas of original data must be managed properly
- Trade-off between reliability and write/read bandwidth

Replica strategy

- 1st on the same node as the client
- 2nd on a different rack from the first
- 3rd on a different node of the same rack of the 2nd
- Further replicas randomly

Typical replica pipe



Review questions

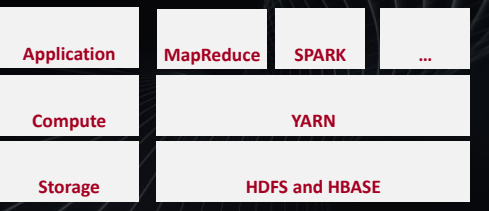
- Identify the mathematical formula used by Hadoop to find the network distance
- Why is the network distance so important in Hadoop?

Review questions

- How does the name node define the data nodes in which to store replicas of the original data?

YARN

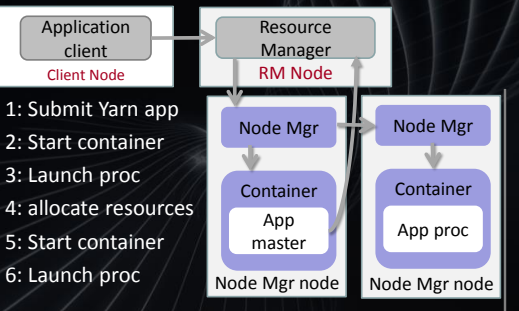
Yet Another Resource Negotiator



YARN Components

- Resource manager
- Node managers
- Containers

Anatomy of a YARN Run



Interprocess communication

- YARN itself does not provide any communication mechanisms between client, master, process
- App specific (e.g. RPC)

Resource requests

- Memory, CPU, bandwidth
- Locality: specify on which node/rack containers must run

- Locality constraints important for using bandwidth efficiently
- Resources can be requested dynamically: Upfront vs Phased

Scheduling in YARN

- Objective: allocate resources over time according to specific policies
  - FIFO
  - Capacity
  - Fair Scheduler

FIFO

- Benefits: simple, no configuration needed
- Drawbacks: not suitable for shared clusters (hungry apps will starve the rest)

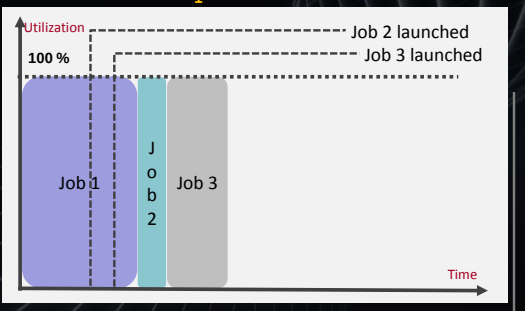
Capacity

- Dedicated queues are given a certain capacity slot
- Queues can be further divided hierarchically
- Queue elasticity possible

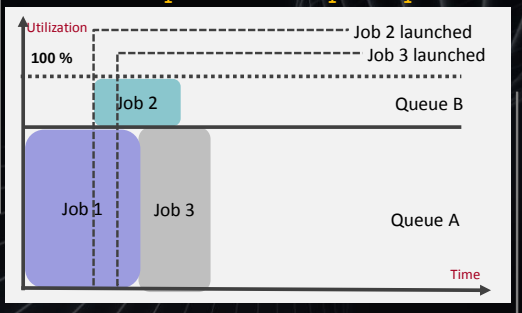
Fair Scheduler

- All running apps get the same share of resources
- Hierarchical queues possible
- Each queue can have weight and a specific internal scheduling policy

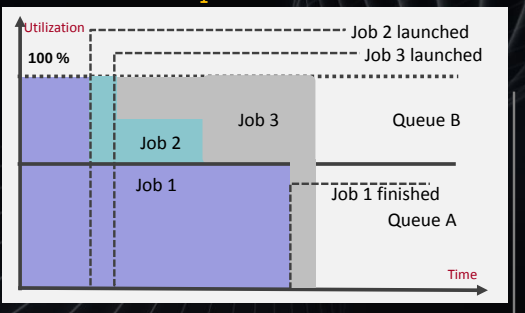
Comparison: FIFO



Comparison: Capacity



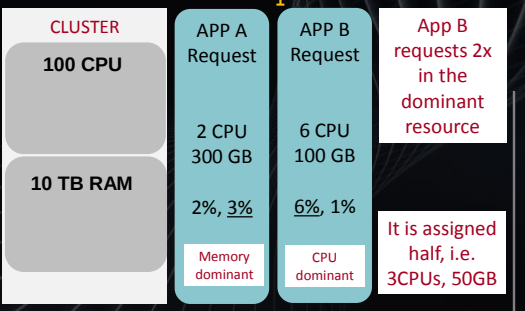
Comparison: Fair



Dominant Resource Fairness

- Balancing usage of multiple resource types is an issue
- YARN can address it looking at each user's dominant resource respect to cluster usage

Example



Review questions

- Which are YARN basic components?
- How does YARN run an application?
- Can a YARN application do resource requests while it is running?

Review questions

- Which are the types of scheduling possible in YARN?
- Identify benefits and drawbacks of each of them

Review questions

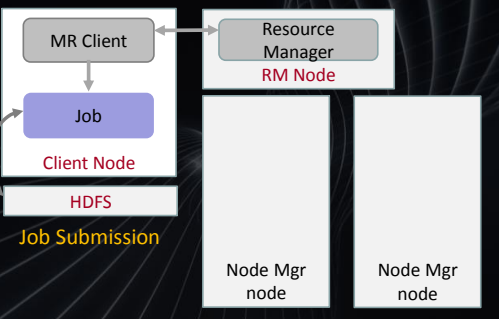
- Describe how YARN addressess balacing multiple resources using Dominant Resource Fairness

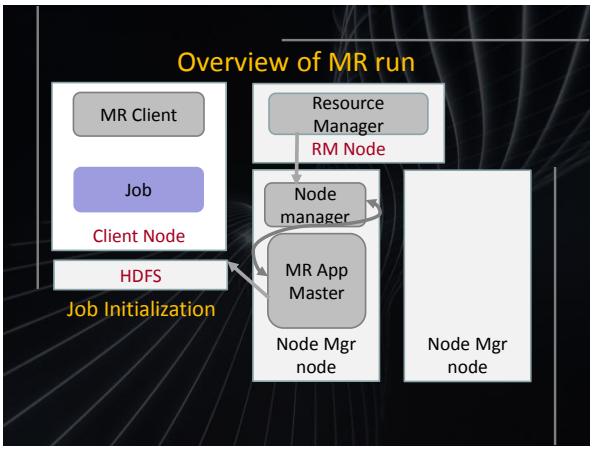
HOW MAPREDUCE WORKS

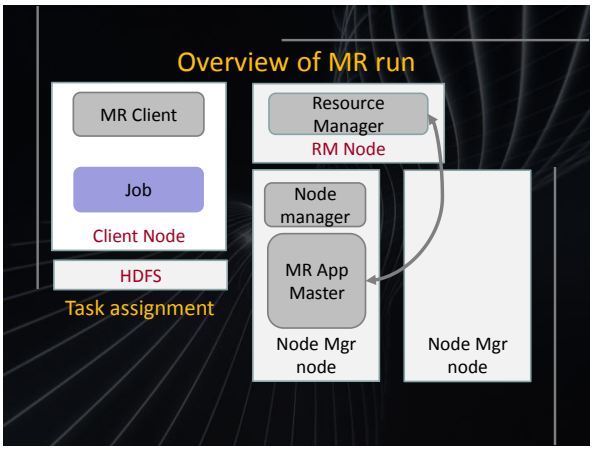
Running a MR job

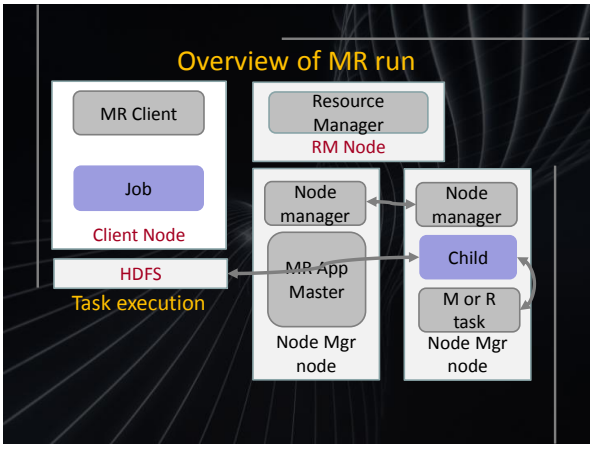
▪ The client	▪ submits job
▪ Resource manager	▪ allocation of resources
▪ Node managers	▪ launch and control containers
▪ App master	▪ co-ordination
▪ HDFS	▪ sharing job files

Overview of MR run









Job submission

- The submit() method creates a JobSubmitter instance and calls submitJobInternal()

- waitForCompletion() polls each second and reports status to the console:
  - When job completed ok, the job counters are displayed
  - If not ok, error is logged to the console

Job submission

- Ask resource manager for a new AppId
- Checks output specifications
- Computes input splits

- Copies resources to the shared filesystem
- Submits the job by calling submitApplication() on the resource manager

Job Initialization

- The ResourceManager calls the YARN scheduler
- The YS allocates a container and launches the AppMaster in it

- The AppMaster is initializated with bookkeeping objects
- The AppMaster retrieves the input splits computed in the client
- Creates a Map for each split

Task assignement

- Uberization?
- If no, the AppMaster requests containers for all map and reduce tasks to the resource Manager

- Map Tasks requests prior than Reduce
- Data locality constraints taken into account

Task execution

- The AppMAster starts a container in each node

- Before running the task, all necessary resources are collected (job config, JAR, etc.)
- Each child process runs in a separate JVM (to avoid crashes of the node manager)

Review questions

- Which are the 5 entities involved in a MapReduce job Run?
- In your opinion, what can happen if an ApplicationMaster fails?

SUMMARY QUESTION

- Identify the mathematical formula used by Hadoop to find the network distance
- Why is the network distance so important in Hadoop?

- How does the name node define the data nodes in which to store replicas of the original data?

- Which are YARN basic components?
- How does YARN run an application?
- Can a YARN application do resource requests while it is running?

- Which are the types of scheduling possible in YARN?
- Identify benefits and drawbacks of each of them

- Describe how YARN addressess balacing multiple resources using Dominant Resource Fairness

- Which are the 5 entities involved in a MapReduce job Run?
- In your opinion, what can happen if an ApplicationMaster fails?