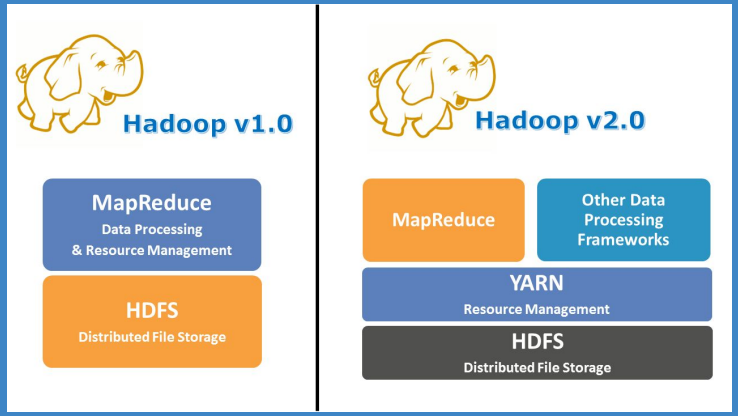


Hadoop Map Reduce and YARN
Lecture 4

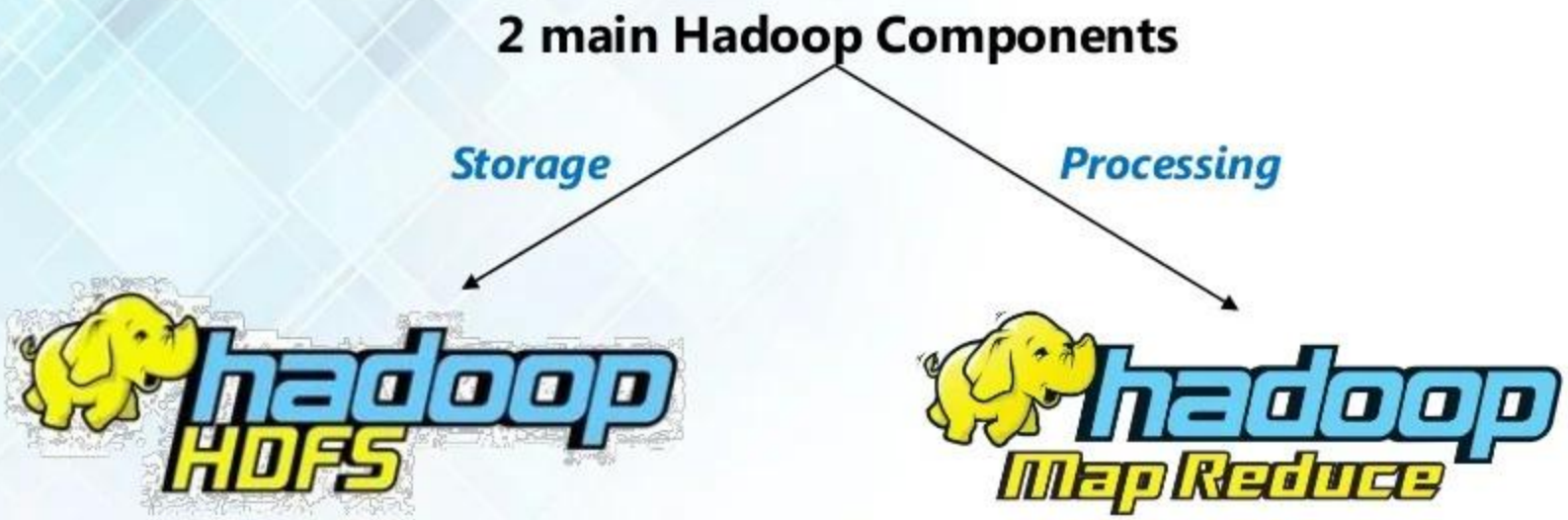
Tahmina Sultana Priya
Lecturer, Dept.of CSE
Daffodil International University

Agenda for today's Session

edure

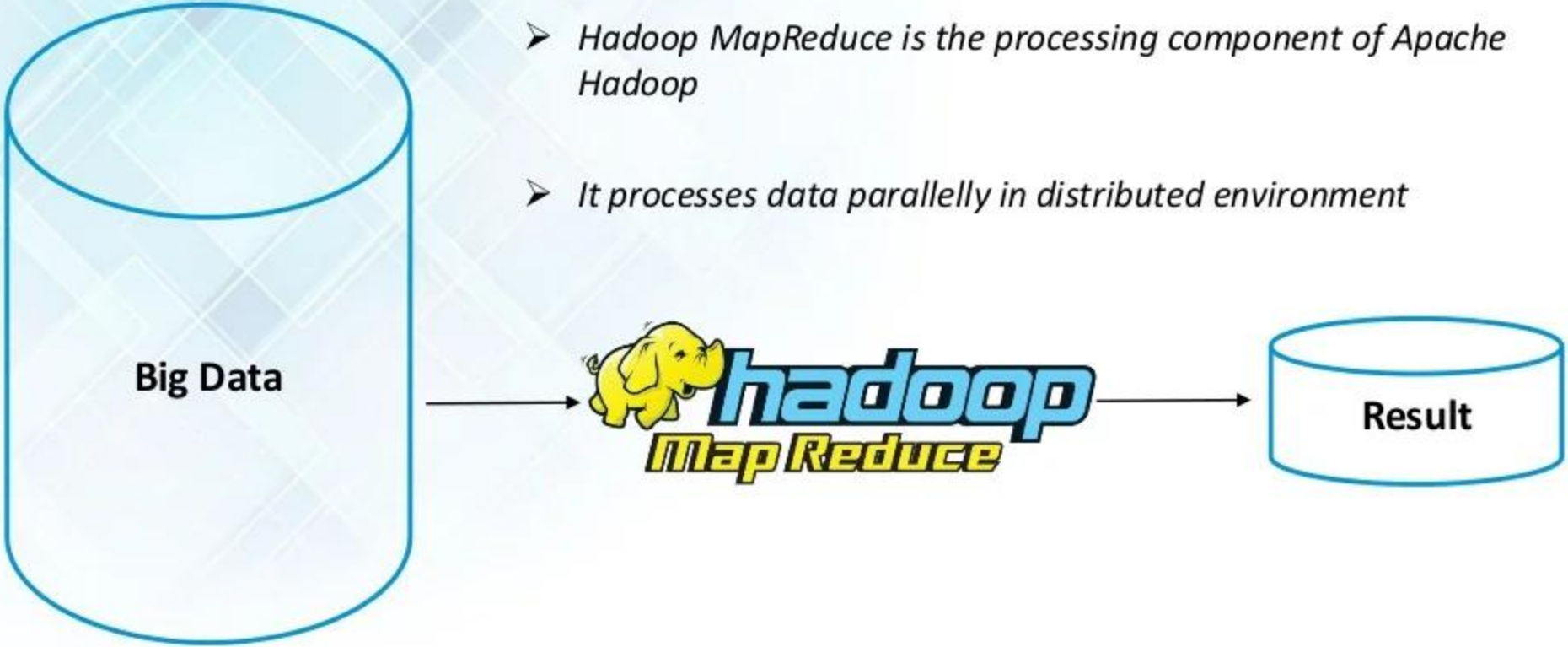
1. What is Hadoop MapReduce?
2. MapReduce In Nutshell
3. Advantages of MapReduce
4. Hadoop MapReduce Approach with an Example
5. Hadoop MapReduce/YARN Components
6. YARN With MapReduce
7. Yarn Application Workflow
8. MapReduce Program with Hands On

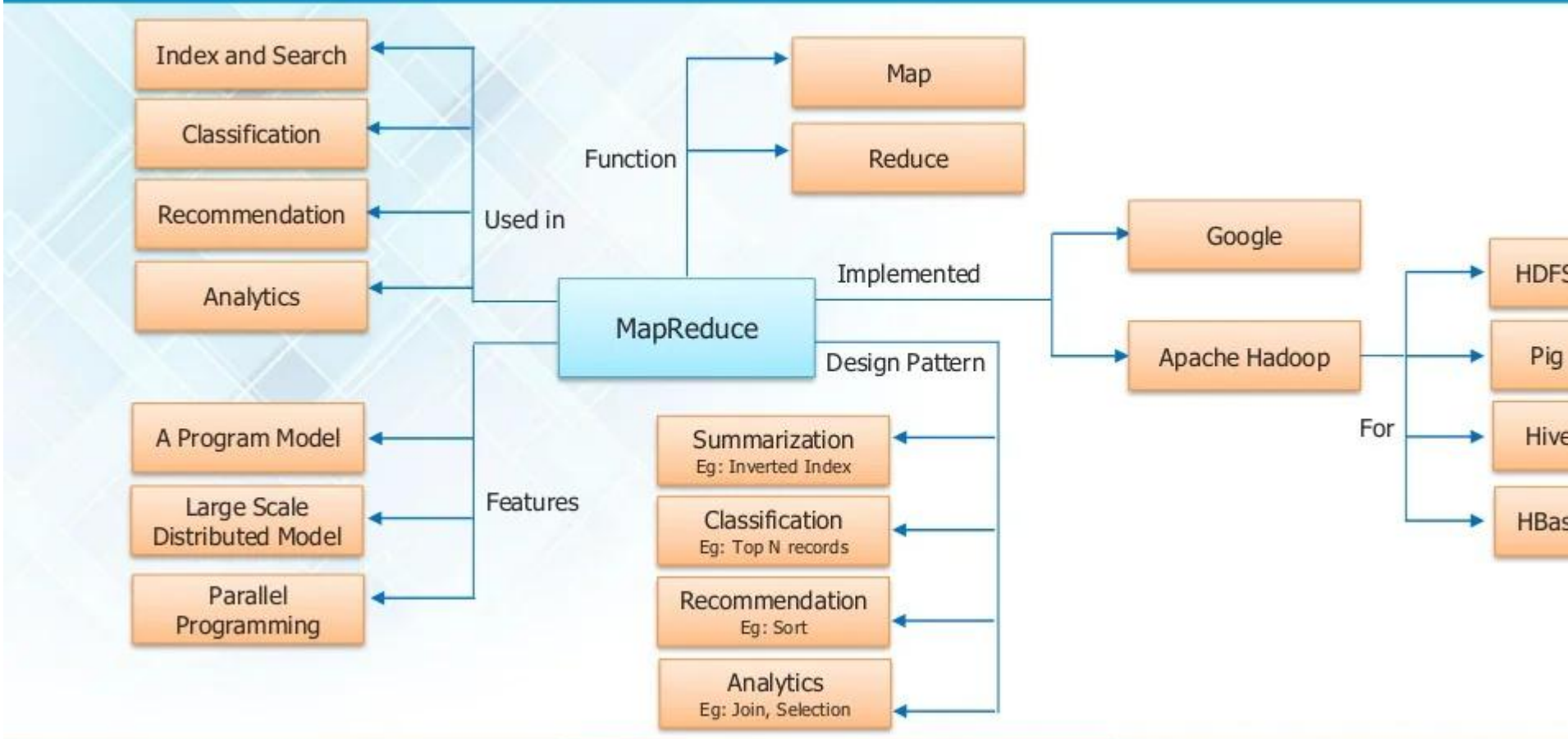
Hadoop Components



MapReduce: Data Processing Using Programming

- *Hadoop MapReduce is the processing component of Apache Hadoop*
- *It processes data parallelly in distributed environment*



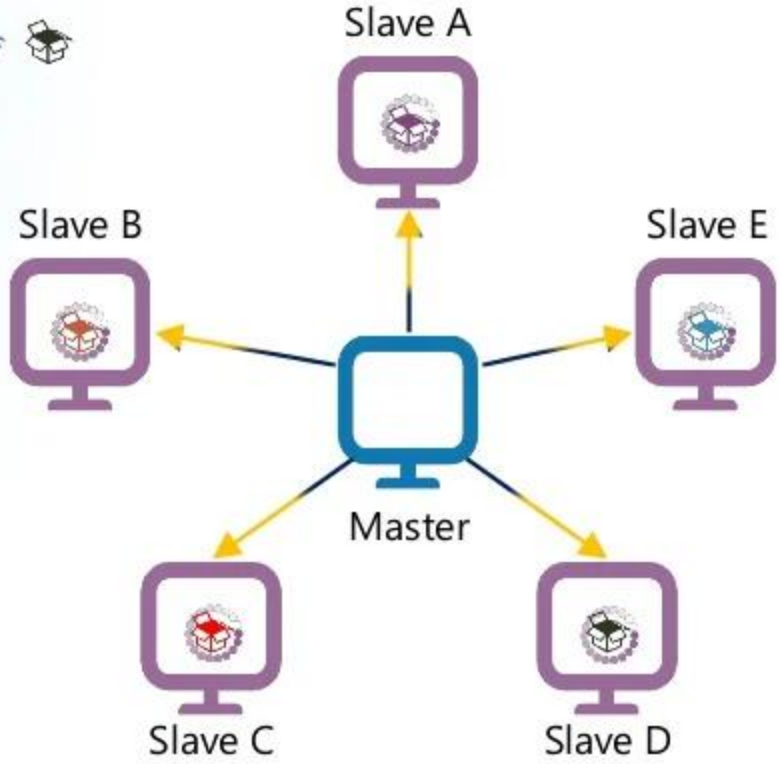

Index and Search
Classification
Recommendation
Analytics
Used in
Function
Map
Reduce
MapReduce
Implemented
Google
Apache Hadoop
For
HDFS
Pig
Hive
HBas
Design Pattern
Summarization
Eg: Inverted Index
Classification
Eg: Top N records
Recommendation
Eg: Sort
Analytics
Eg: Join, Selection
Features
A Program Model
Large Scale Distributed Model
Parallel Programming

2 Biggest Advantages of MapReduce

Advantage 1: Parallel Processing

Data →

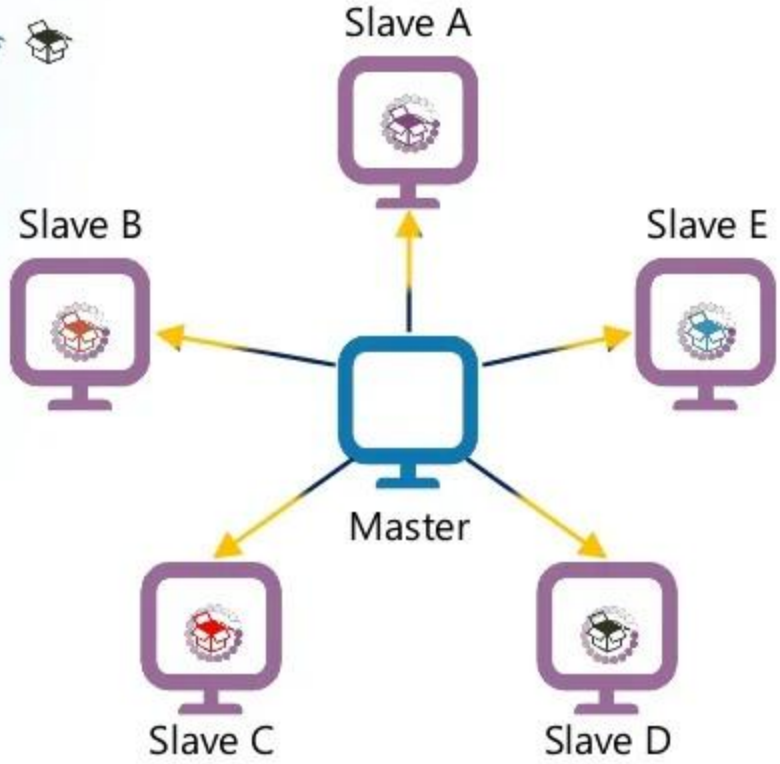
- Data is processed in parallel
- Processing becomes fast



Advantage 2: Data Locality - Processing to Storage

Data →

- Moving Data to processing is very costly
- In MapReduce, we move processing to Data



Traditional vs MapReduce Way

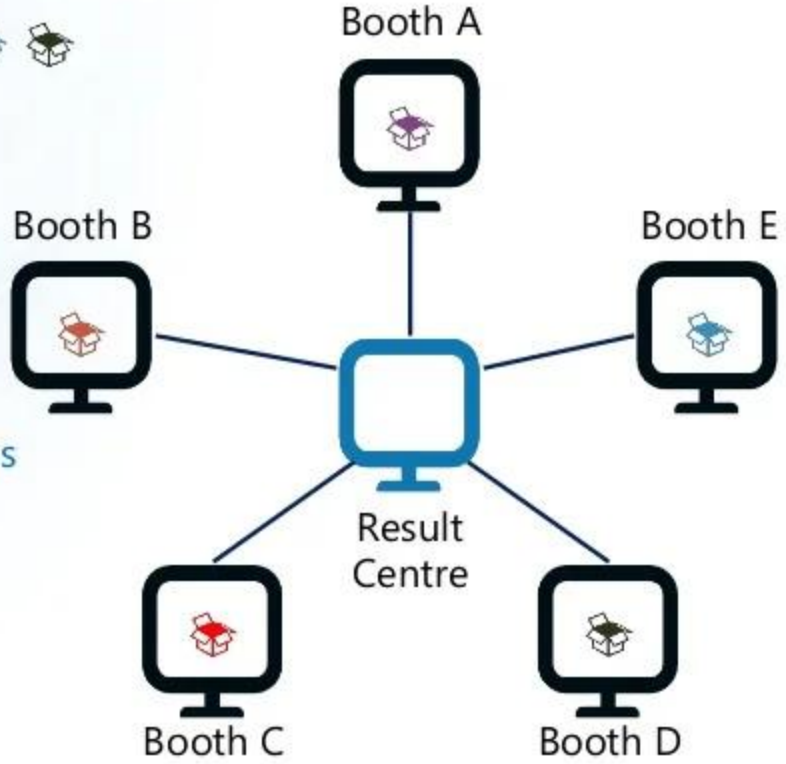
Election Votes Counting

edure

Data →

Election Votes Casting

- Votes is stored at different Booths
- Result Centre has the details of all the Booths



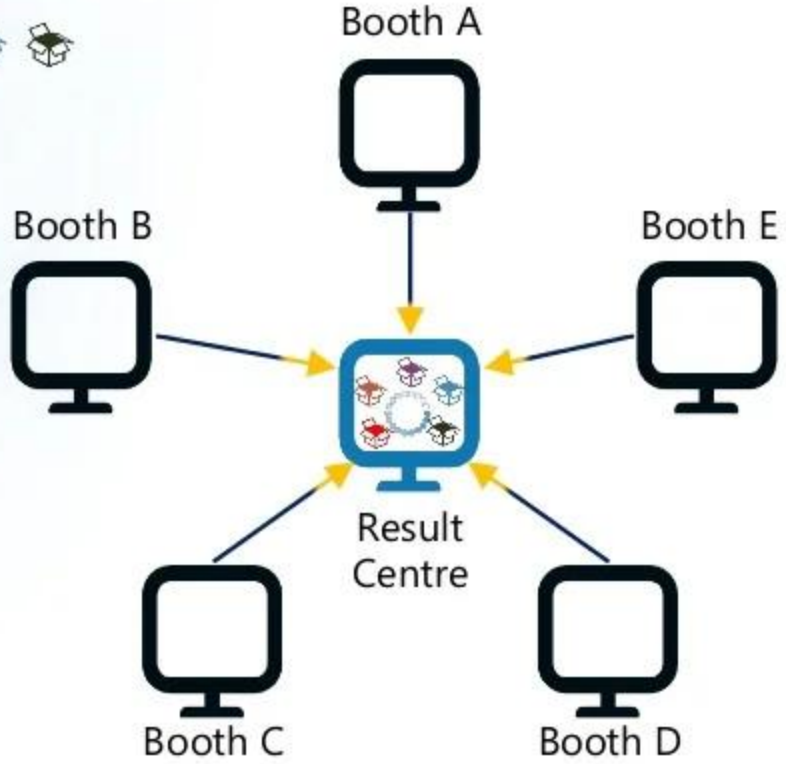
Election Votes Counting – Traditional Way

edure

Data →

Counting – Traditional Approach

- Votes are moved to Result Centre for counting
- Moving all the votes to Centre is costly
- Result Centre is over-burdened
- Counting takes time

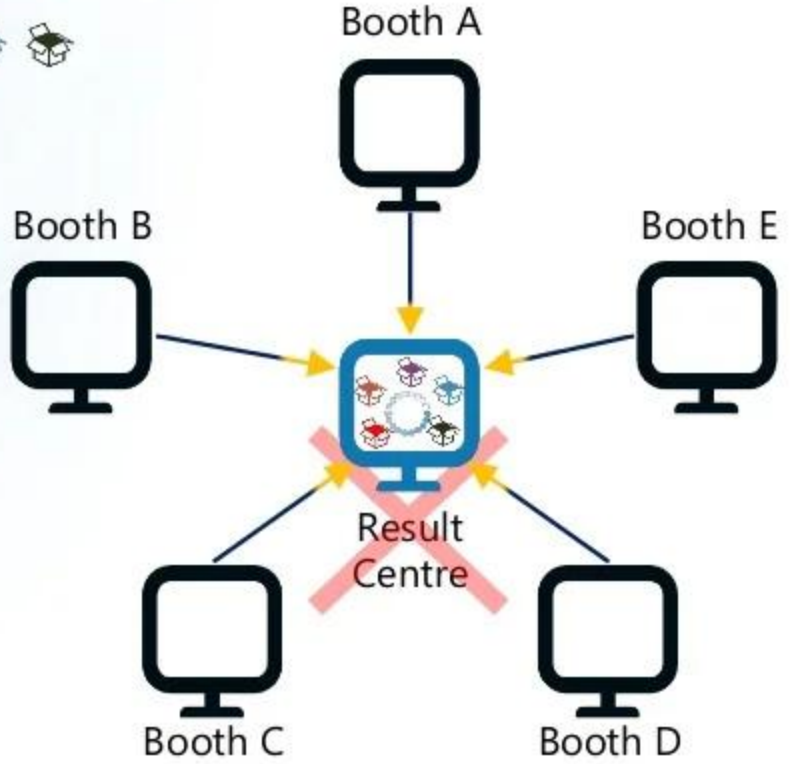


Hadoop MapReduce To the Rescue!

edure

Data →

✕ Hadoop MapReduce Doesn't Follow This Approach

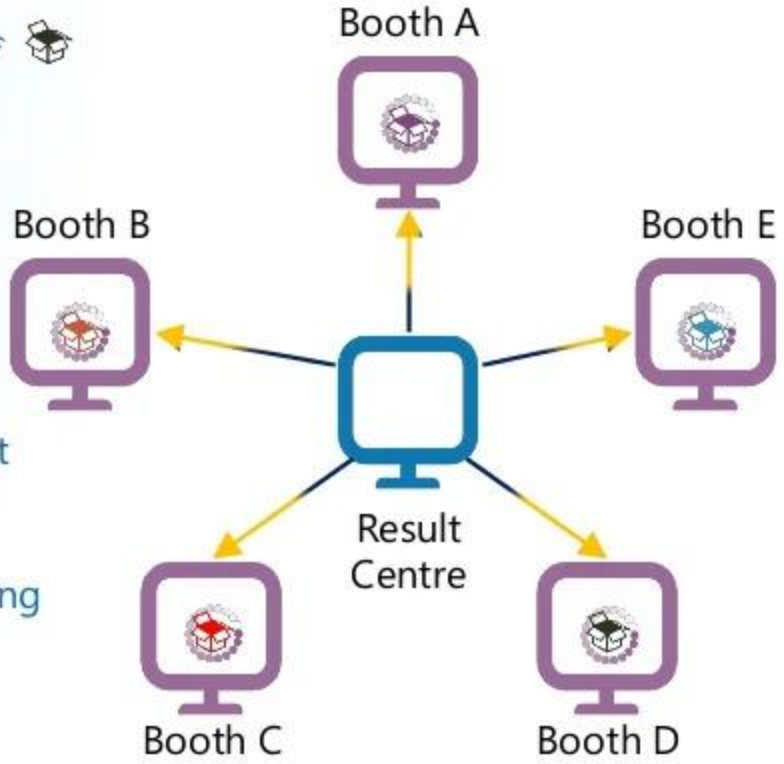


Election Votes Counting – MapReduce Way

Votes→

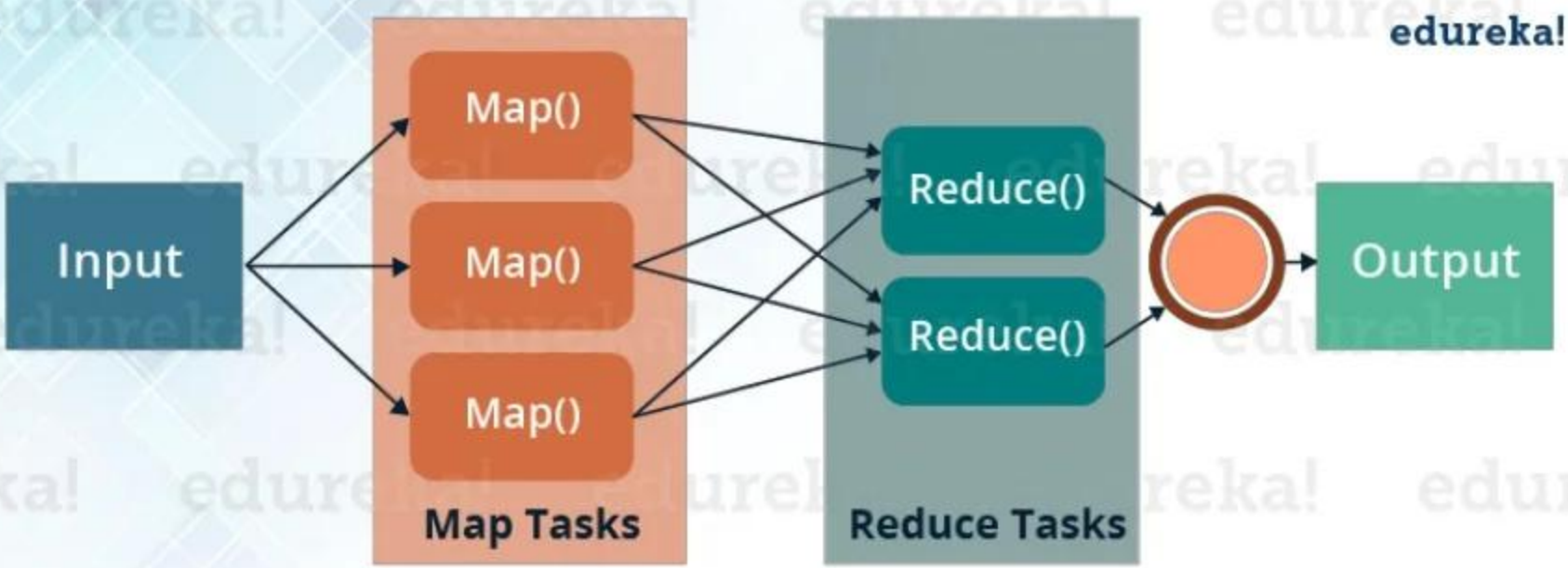
Counting – MapReduce Approach

- Votes are counted at individual booths
- Booth-wise results are sent back to the result centre
- Final Result is declared easily and quickly using this way

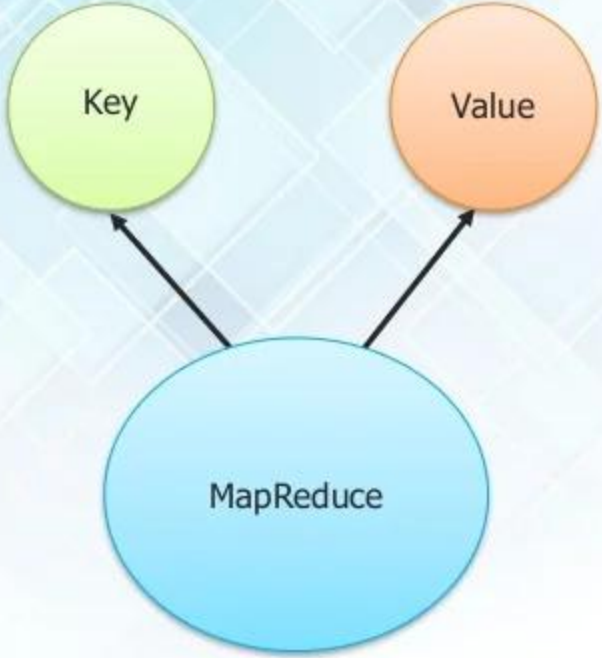


MapReduce In Detail

MapReduce Way



Anatomy of a MapReduce Program



Map:

(K1, V1) → List (K2, V2)

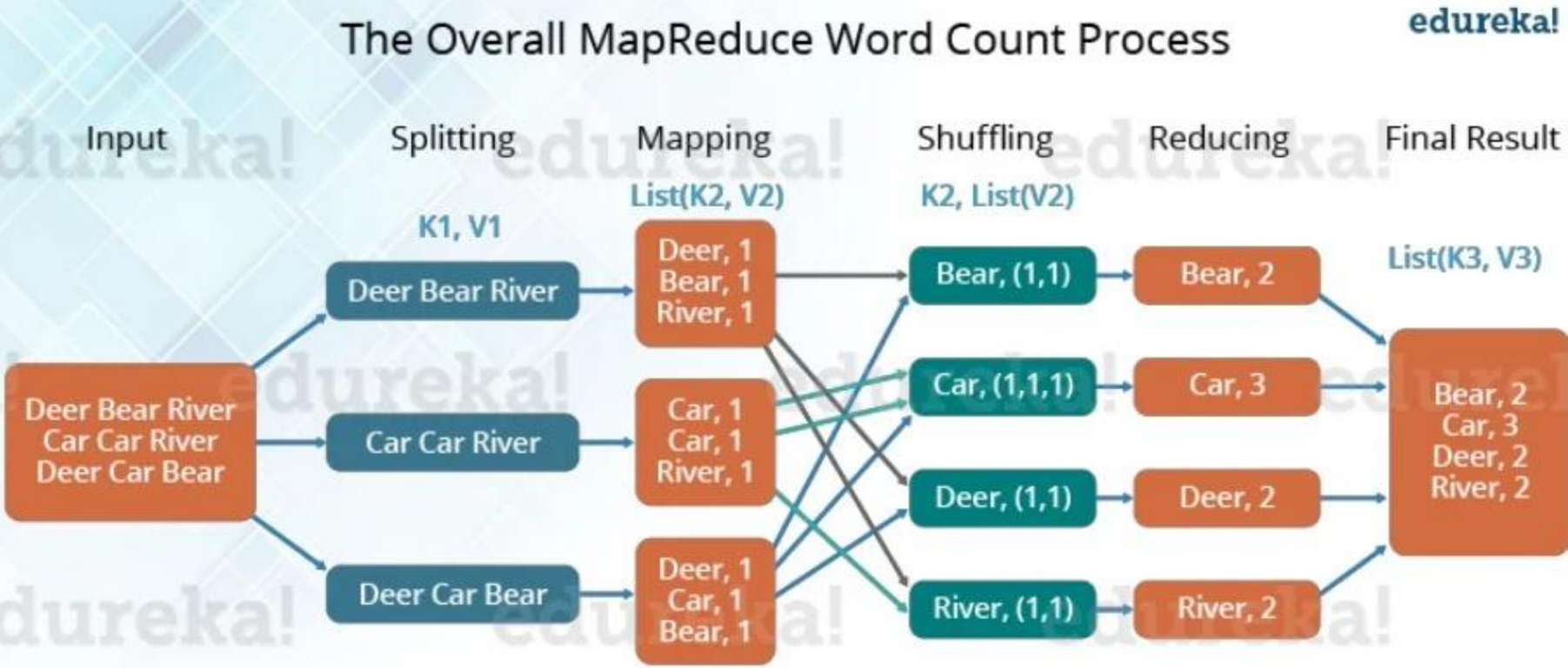
Reduce:

(K2, list (V2)) → List (K3, V3)

Let us take an example to understand

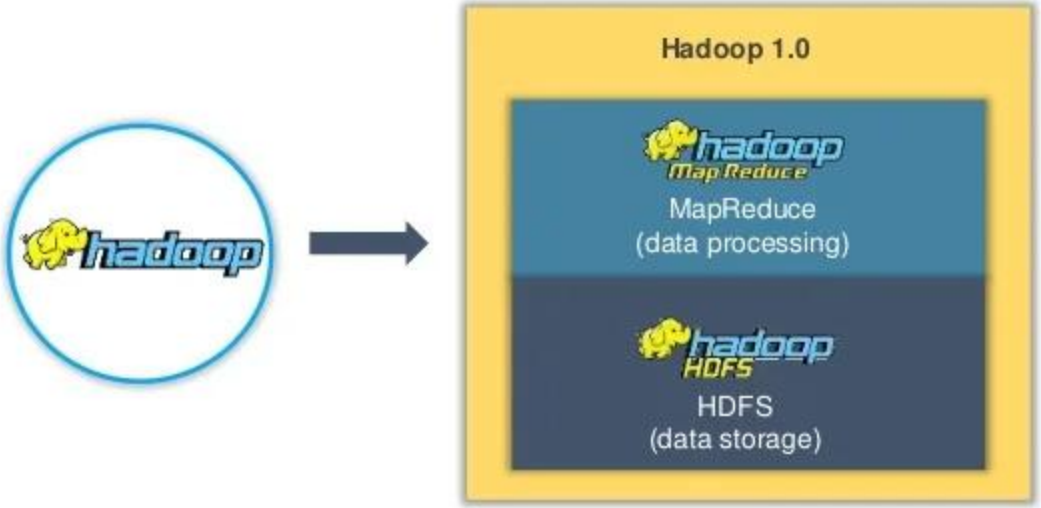
MapReduce Way

MapReduce Way – Word Count Process



Hadoop 1.0 (MR 1)

Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)

MapReduce consisted of
Job Tracker and Task Tracker

Job Tracker

Allocated resources, performed scheduling and monitored jobs

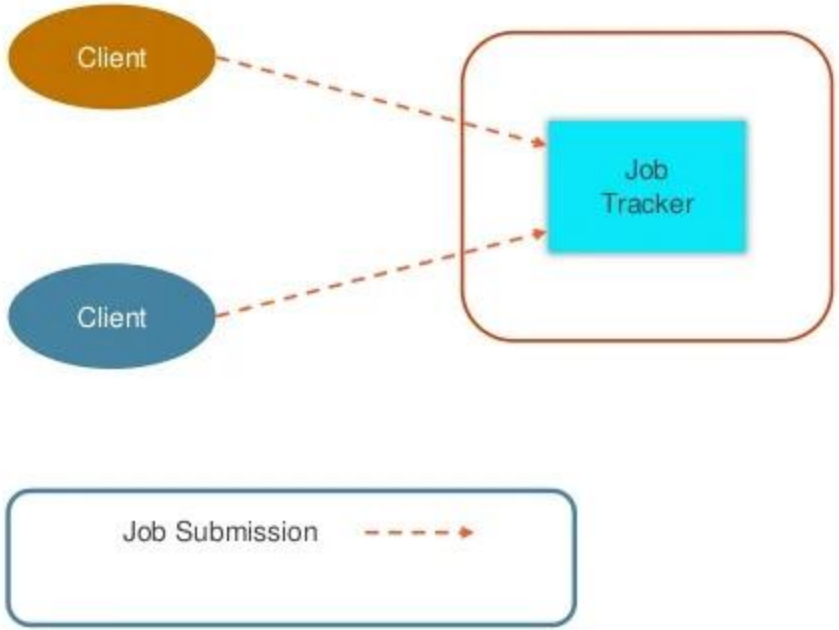
Assigned map and reduce tasks to jobs running on Task Trackers

Task Tracker

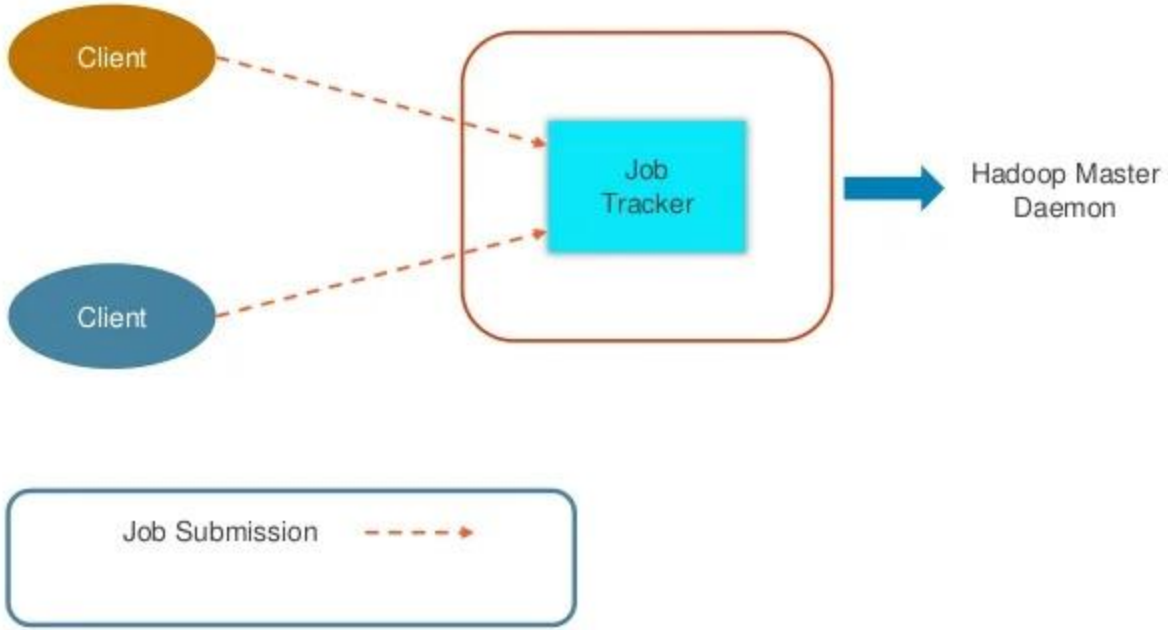
Task Trackers processed the jobs

Task Trackers reported their progress to the Job Tracker

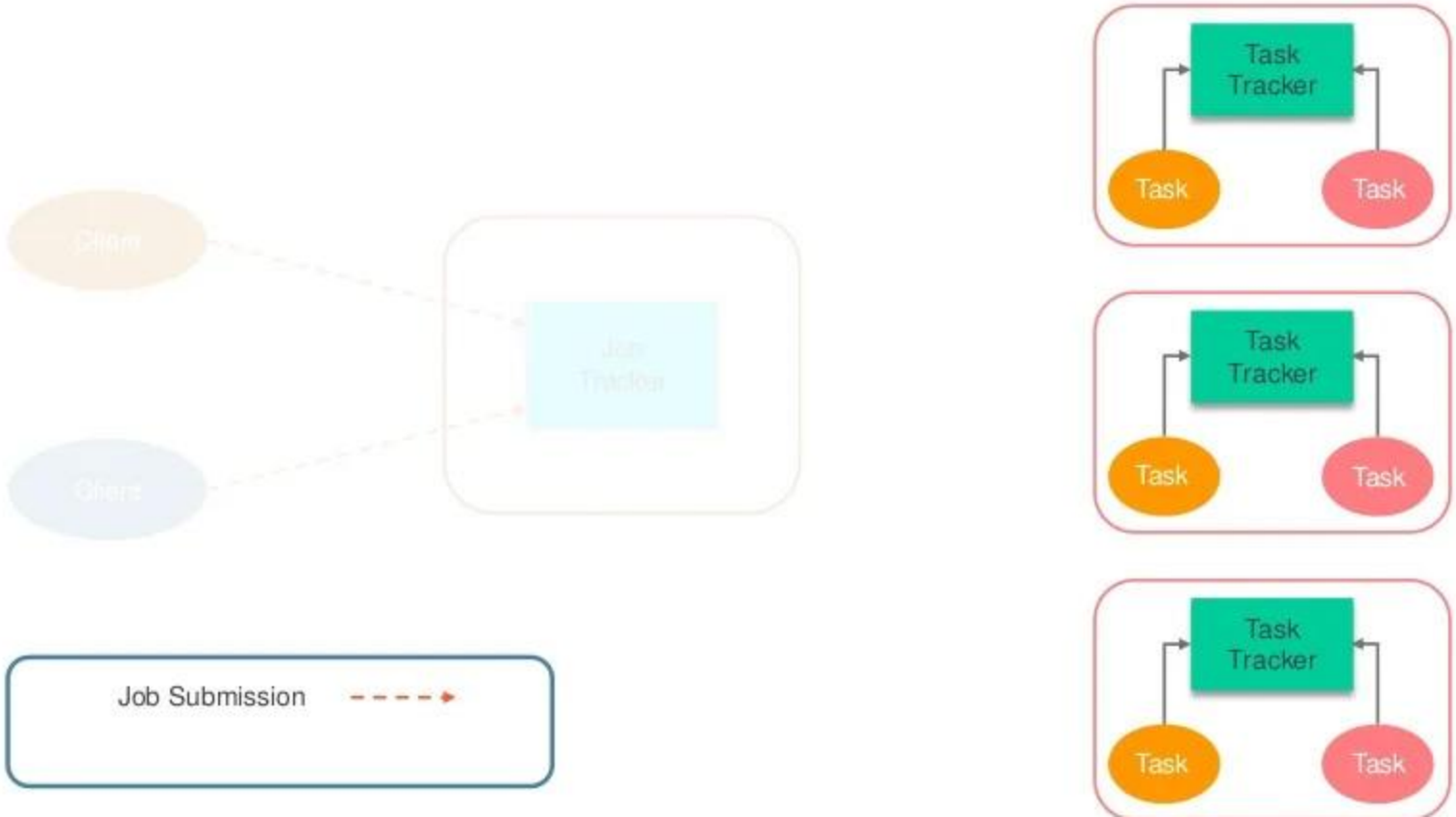
Hadoop 1.0 (MR 1)



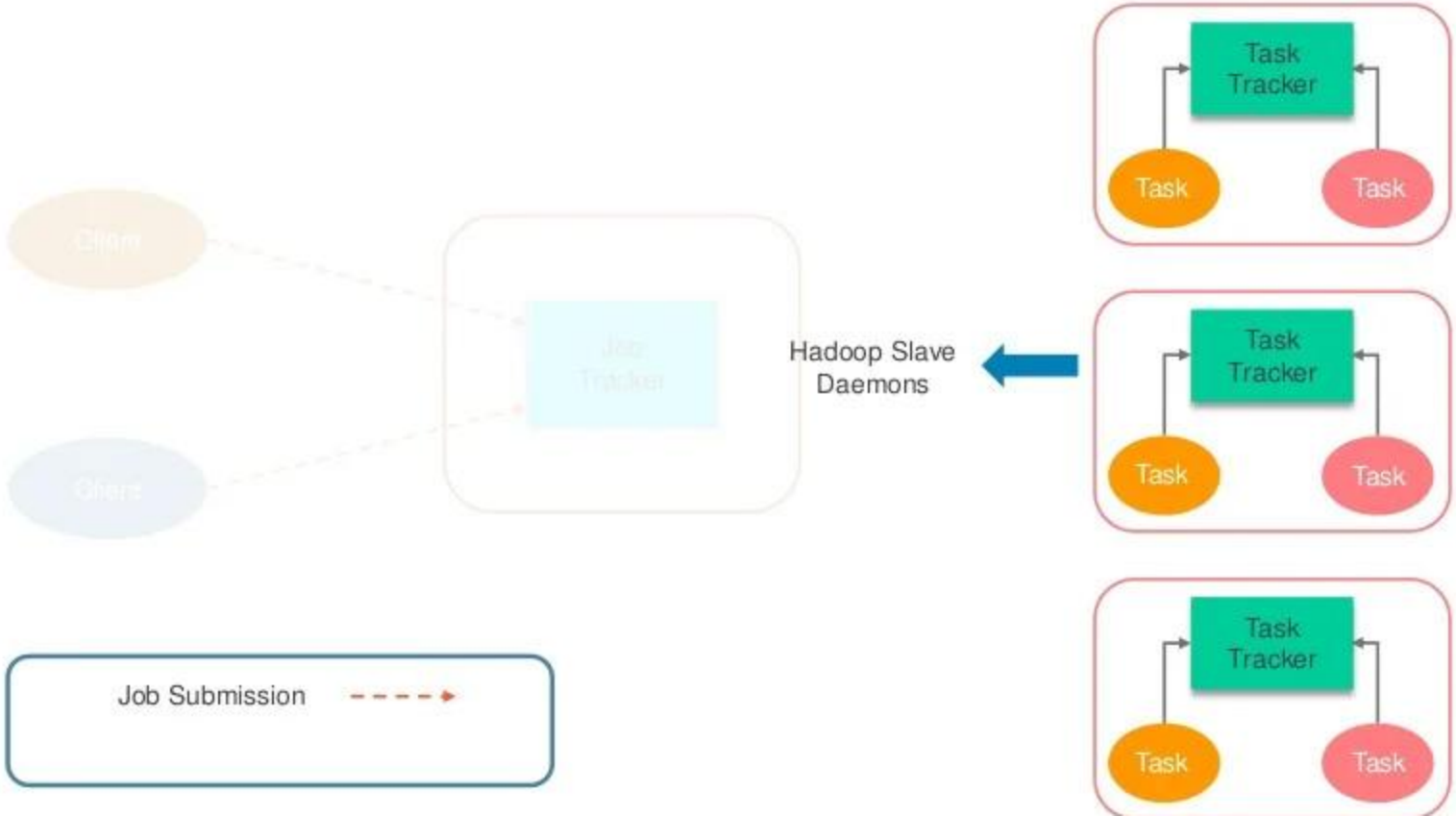
Hadoop 1.0 (MR 1)



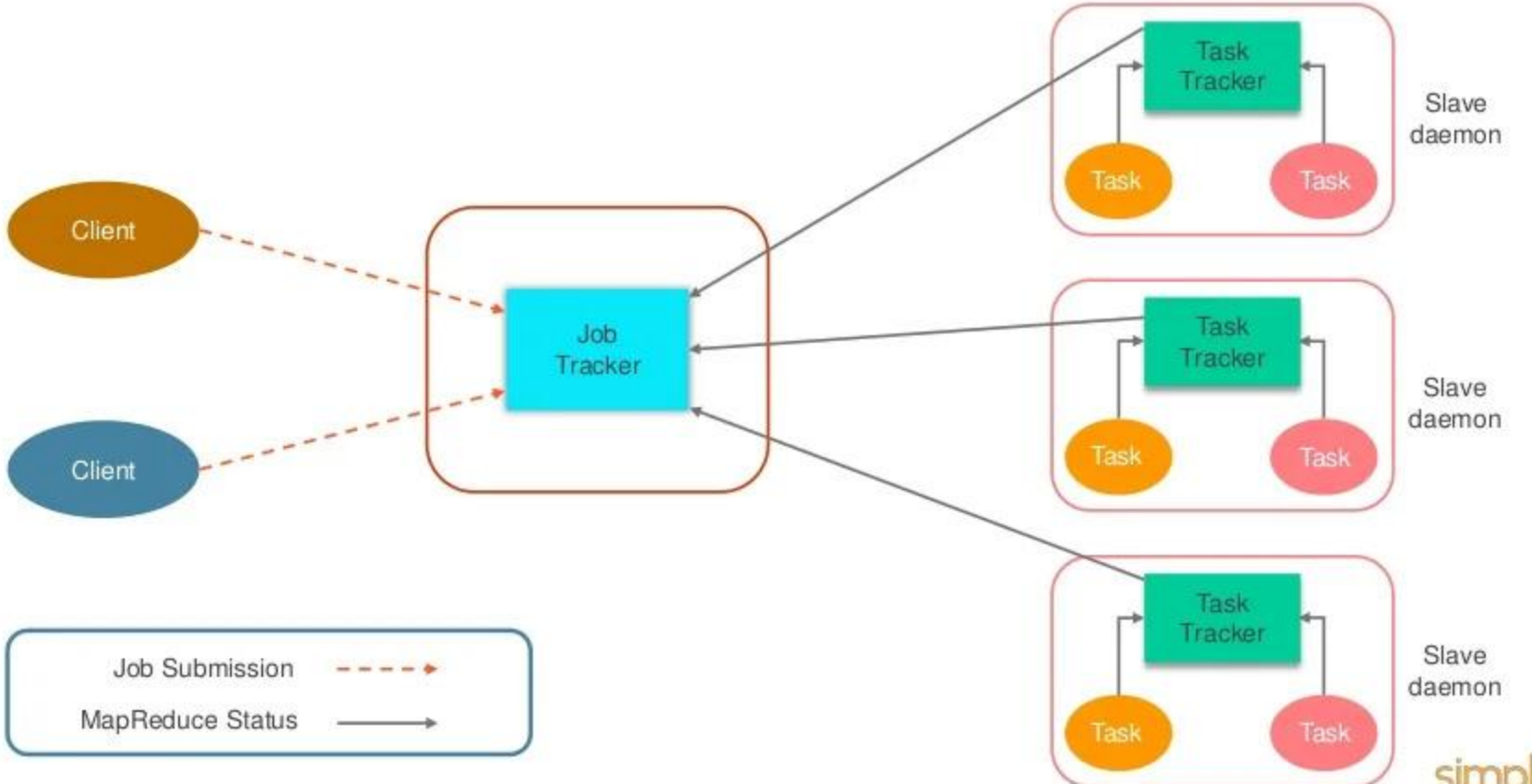
Hadoop 1.0 (MR 1)



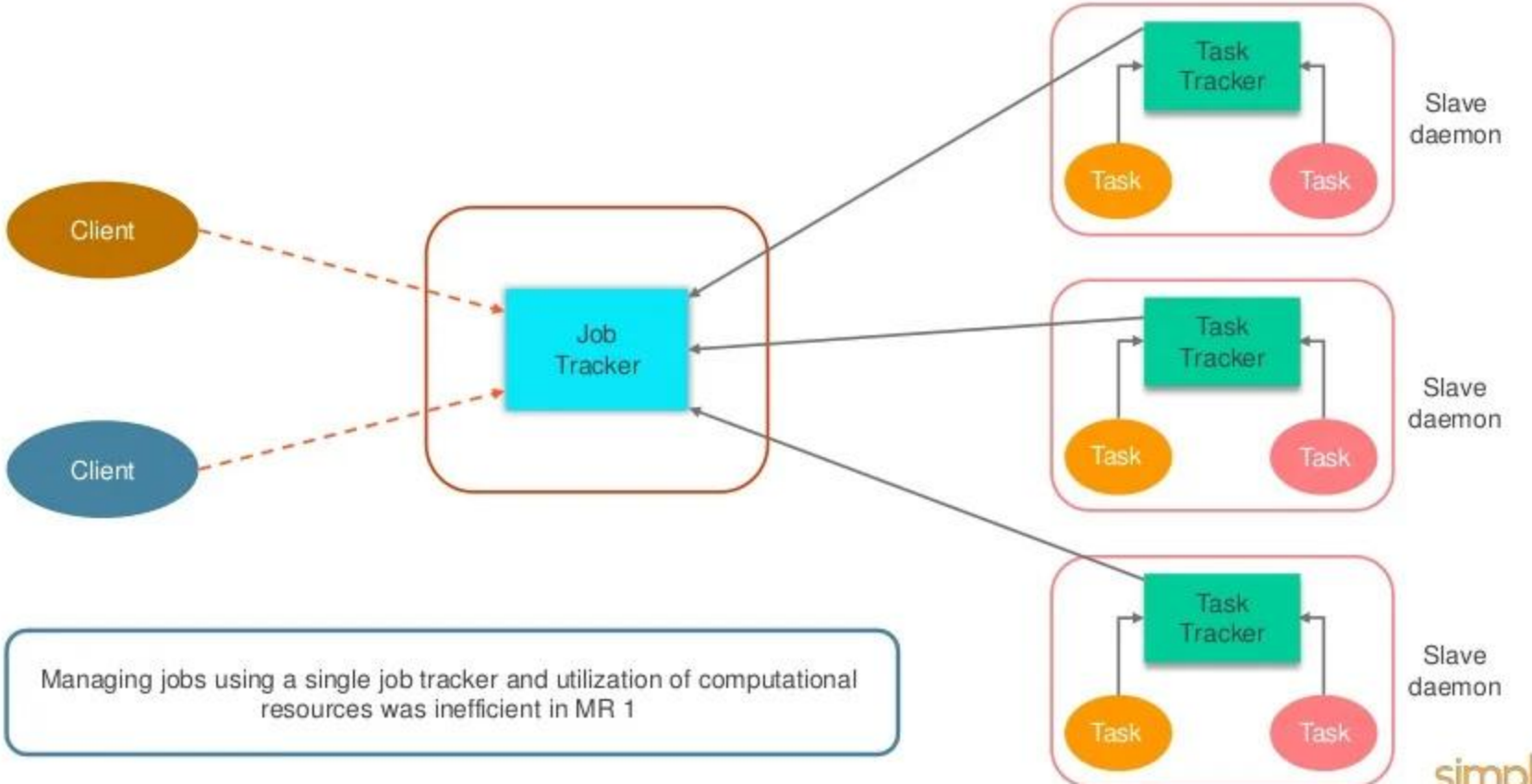
Hadoop 1.0 (MR 1)



Hadoop 1.0 (MR 1)

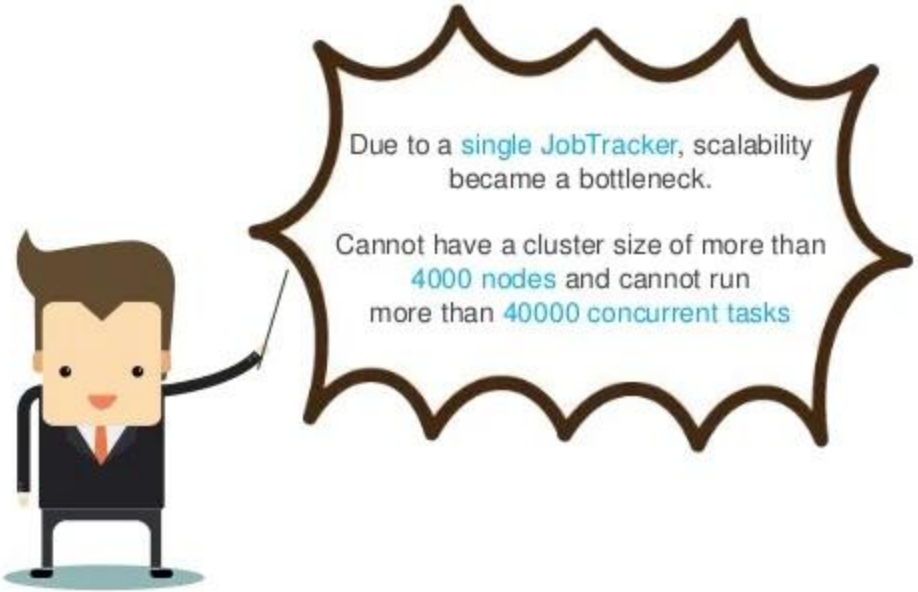


Hadoop 1.0 (MR 1)



Limitations of Hadoop 1.0 (MR 1)

1 Scalability



Limitations of Hadoop 1.0 (MR 1)

2 Availability issue

JobTracker is single point of failure. Any failure kills all queued and running jobs. Jobs need to be resubmitted by users

Limitations of Hadoop 1.0 (MR 1)

3 Resource Utilization

Limitations of Hadoop 1.0 (MR 1)

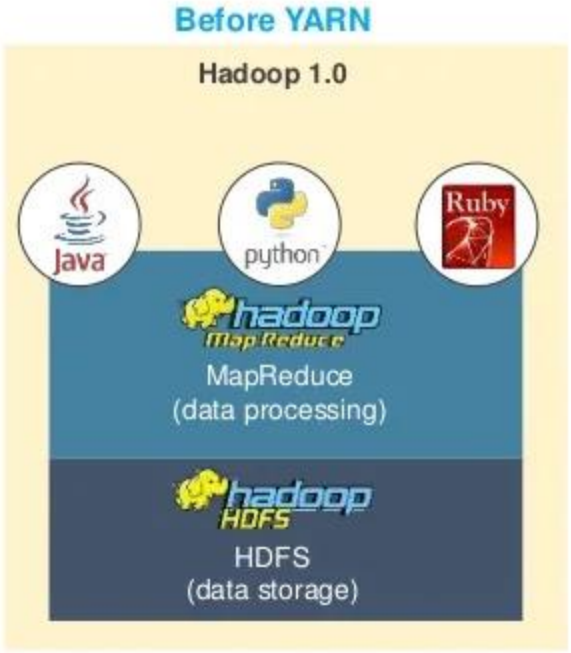
4 Limitations in running non-MapReduce applications

Problem in performing real-time analysis and running Ad-hoc query as MapReduce is batch driven

MapReduce Using Yarn

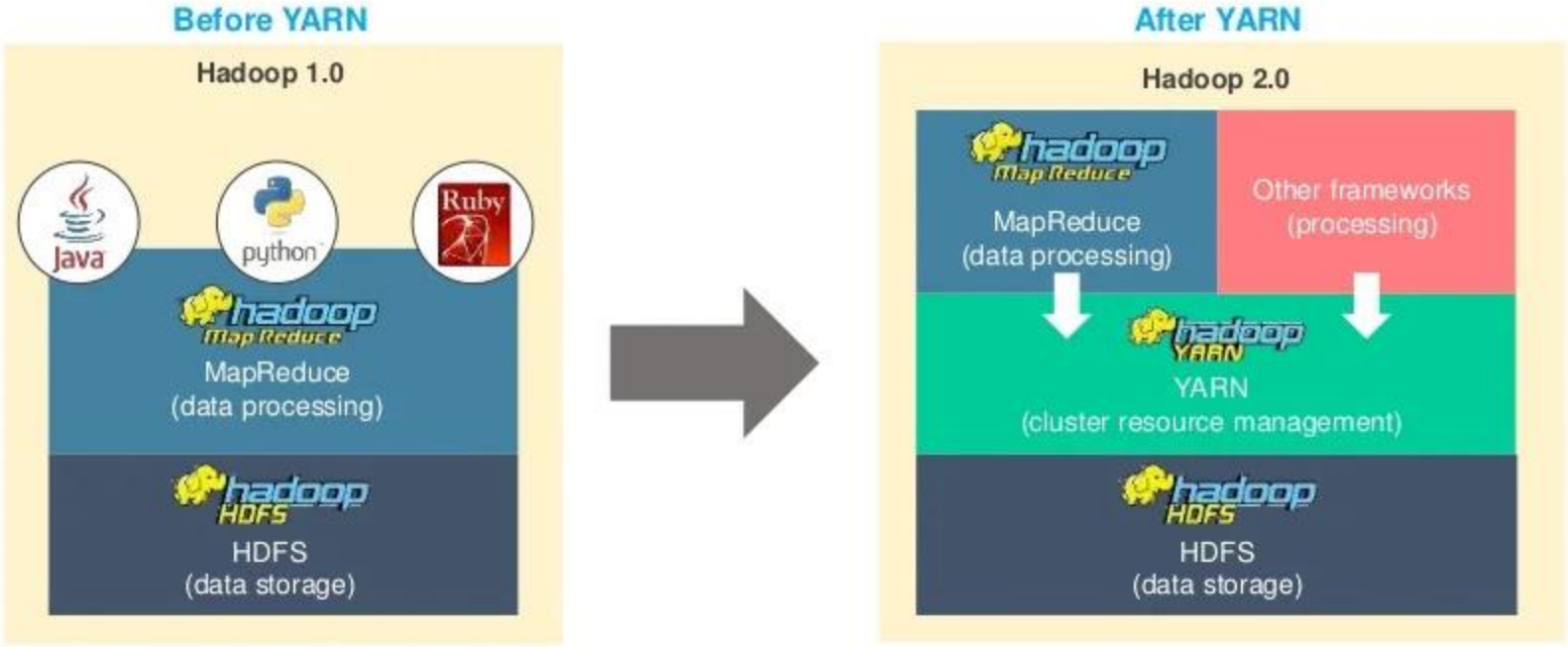
Need for YARN

Need for YARN



Designed to run MapReduce jobs only and had issues in scalability, resource utilization, etc.

Need for YARN



Designed to run MapReduce jobs only and had issues in scalability, resource utilization, etc.

YARN solved those issues and users could work on multiple processing models along with MapReduce

Solution - Hadoop 2.0 (YARN)

Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Solution - Hadoop 2.0 (YARN)

Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Compatibility

Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues

Solution - Hadoop 2.0 (YARN)

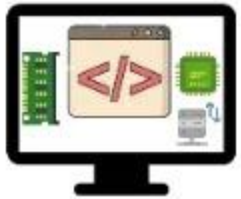
Scalability

Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks

Compatibility

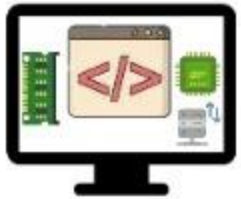
Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues

Resource utilization



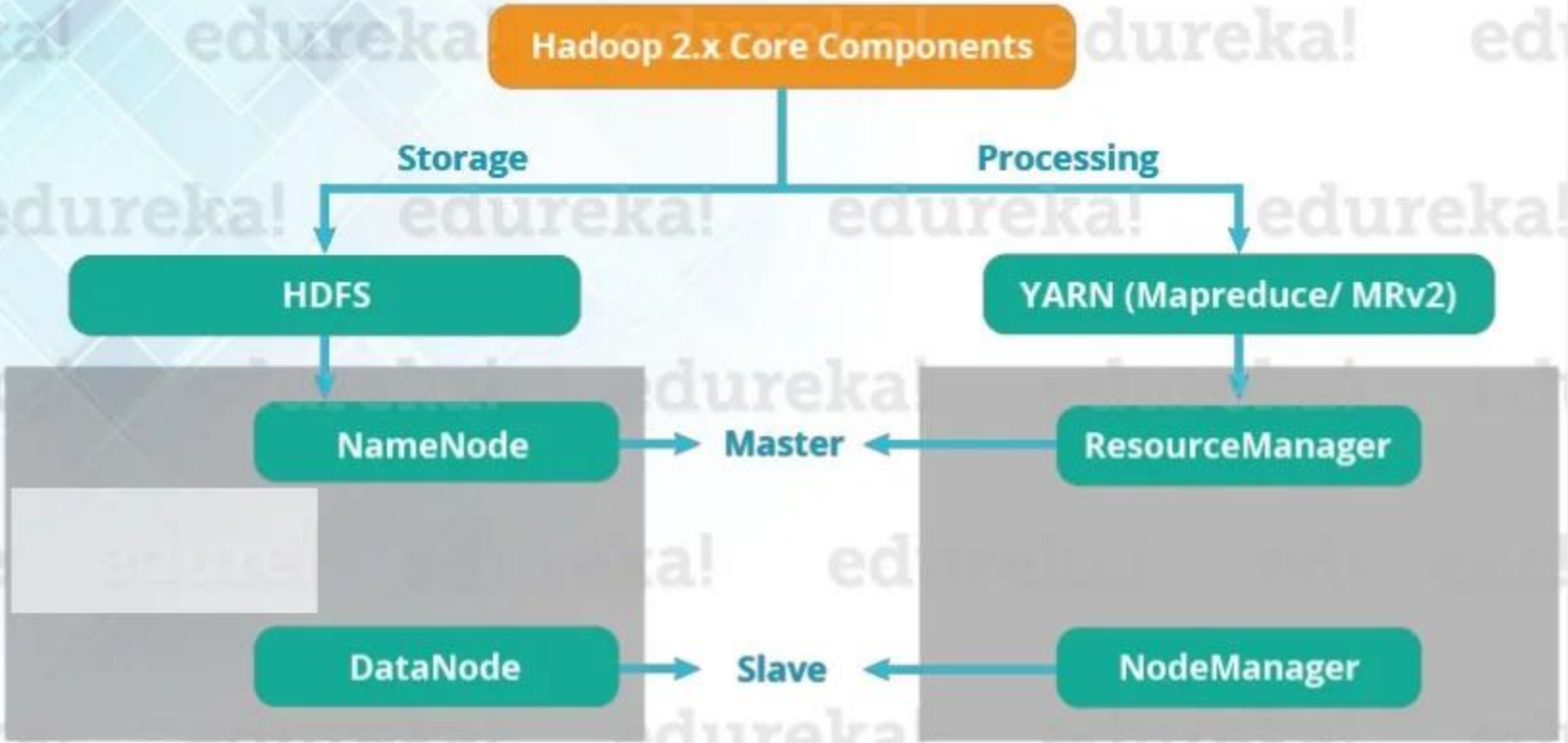
Allows dynamic allocation of cluster resources to improve resource utilization

Solution - Hadoop 2.0 (YARN)

Scalability	Compatibility	Resource utilization	Multitenancy
			
Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks	Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues	Allows dynamic allocation of cluster resources to improve resource utilization	Can use open-source and propriety data access engines and perform real-time analysis and running ad-hoc query

YARN – Moving beyond MapReduce

Hadoop 2.x Daemons



Hadoop 2.x MapReduce Yarn Components

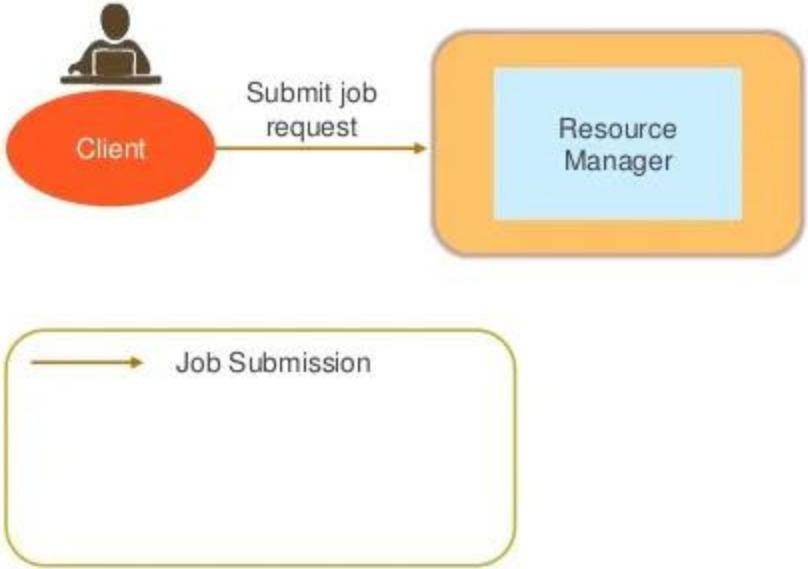
edure

- Client
  - Submits a MapReduce Job
- Resource Manager
  - Cluster Level resource manager
  - Long Life, High Quality Hardware
- Node Manager
  - One per Data Node
  - Monitors resources on Data Node
- Job History Server
  - Maintains information about submitted MapReduce jobs after their ApplicationMaster terminates
- ApplicationMaster
  - One per application
  - Short life
  - Coordinates and Manages MapReduce Jobs
  - Negotiates with Resource Manager to schedule tasks
  - The tasks are started by NodeManager(s)
- Container
  - Created by NM when requested
  - Allocates certain amount of resources (memory, CPU etc.) on a slave node

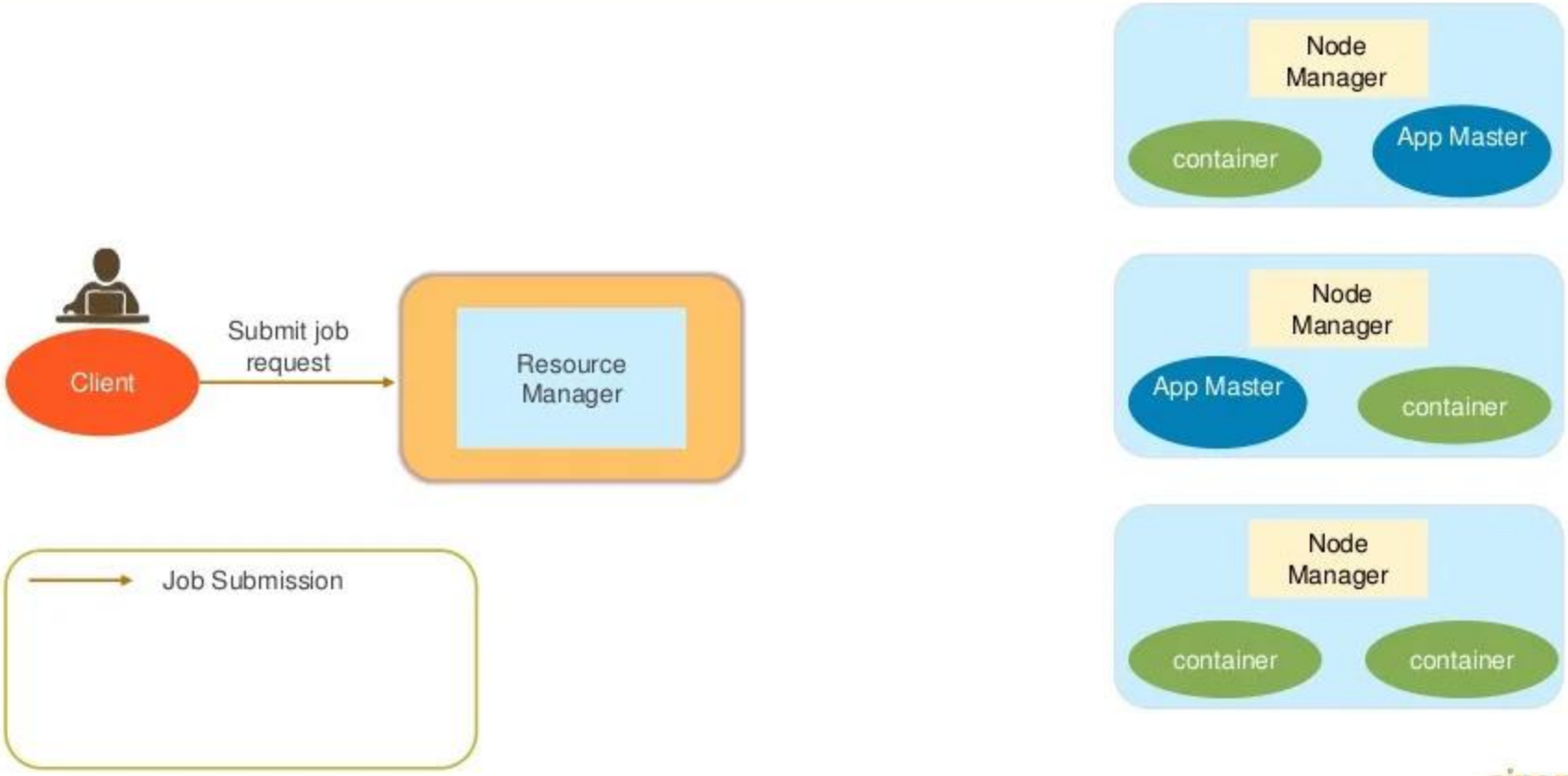
YARN Architecture

YARN Architecture

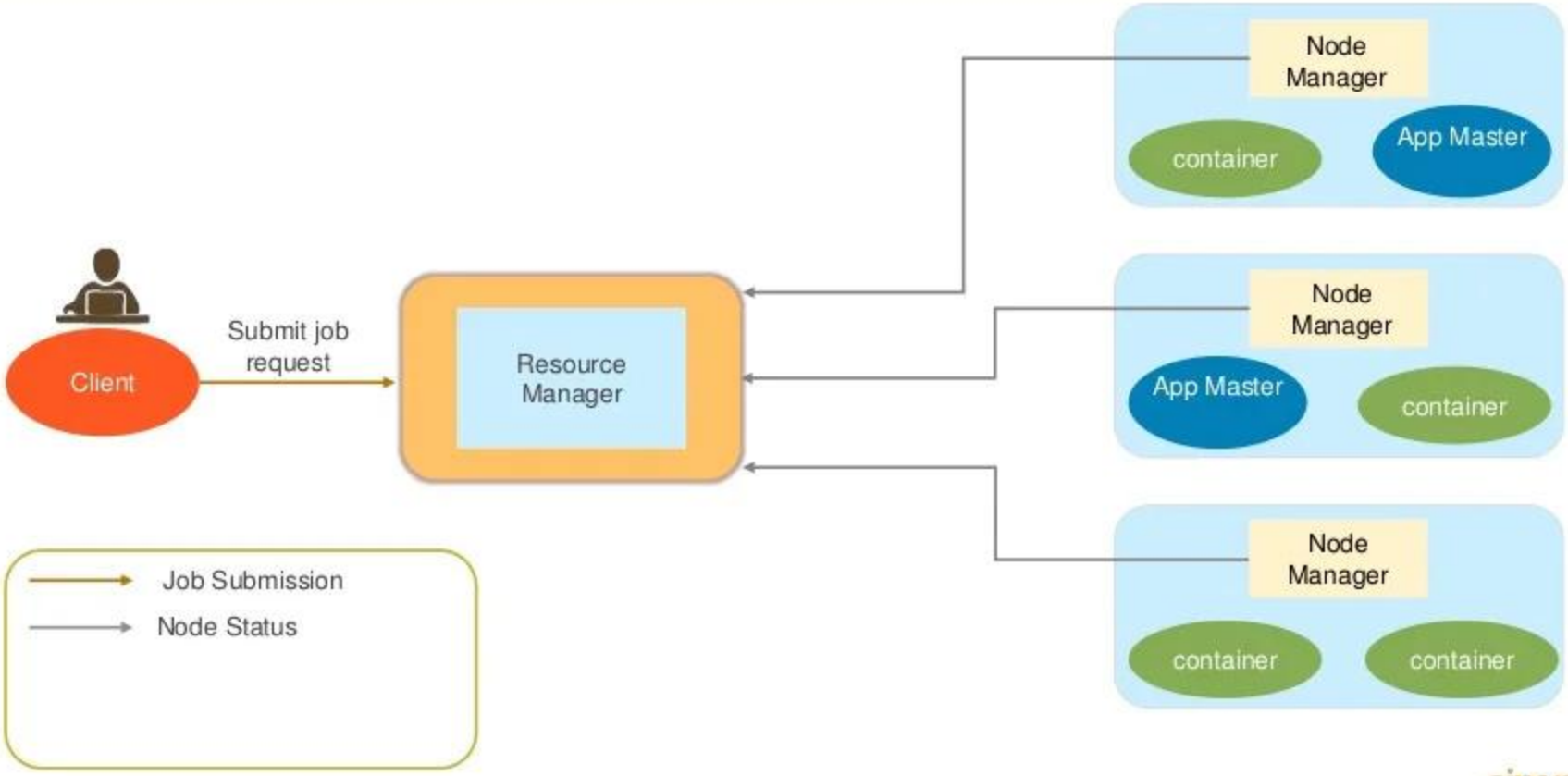
YARN Architecture



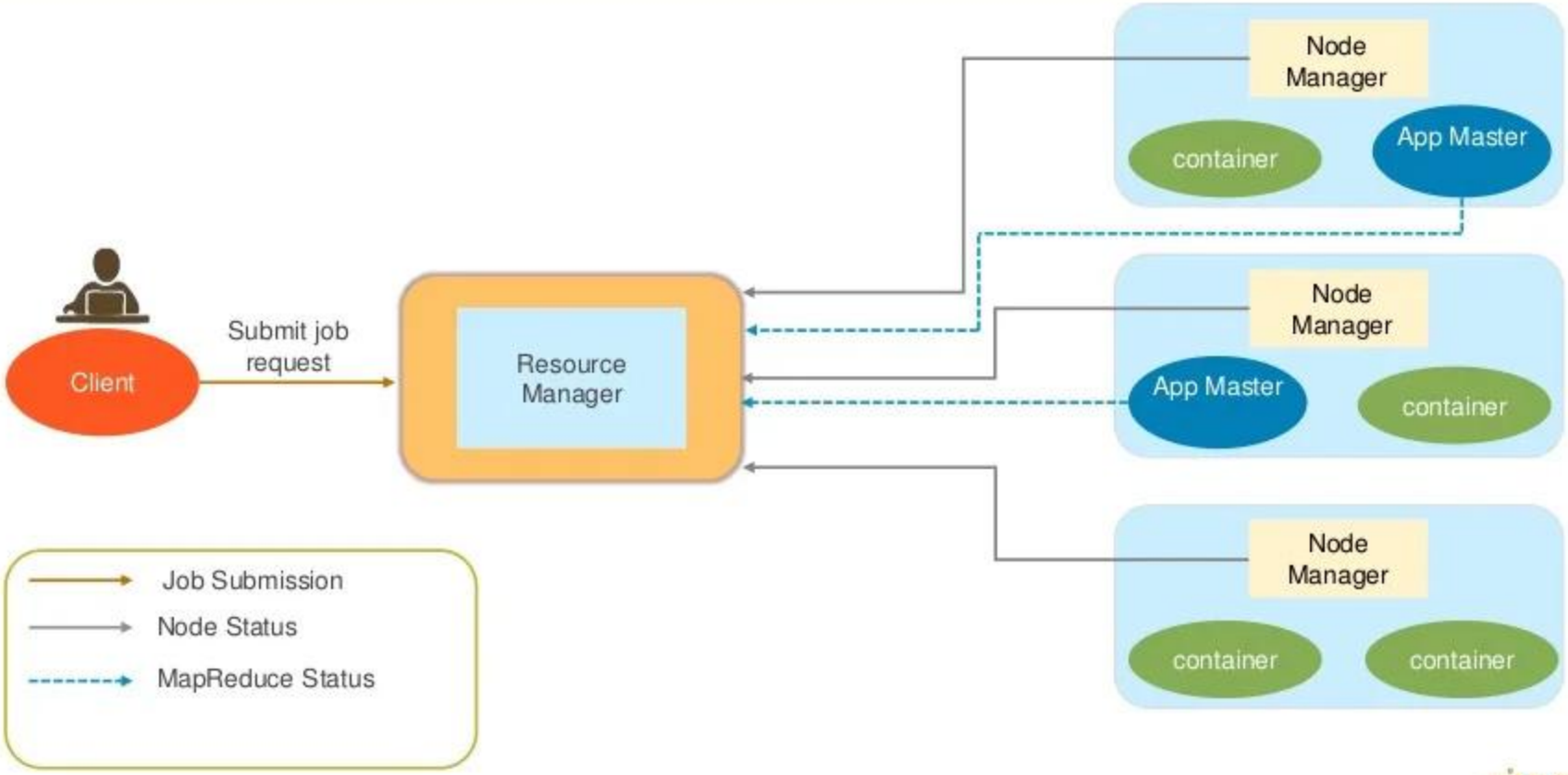
YARN Architecture



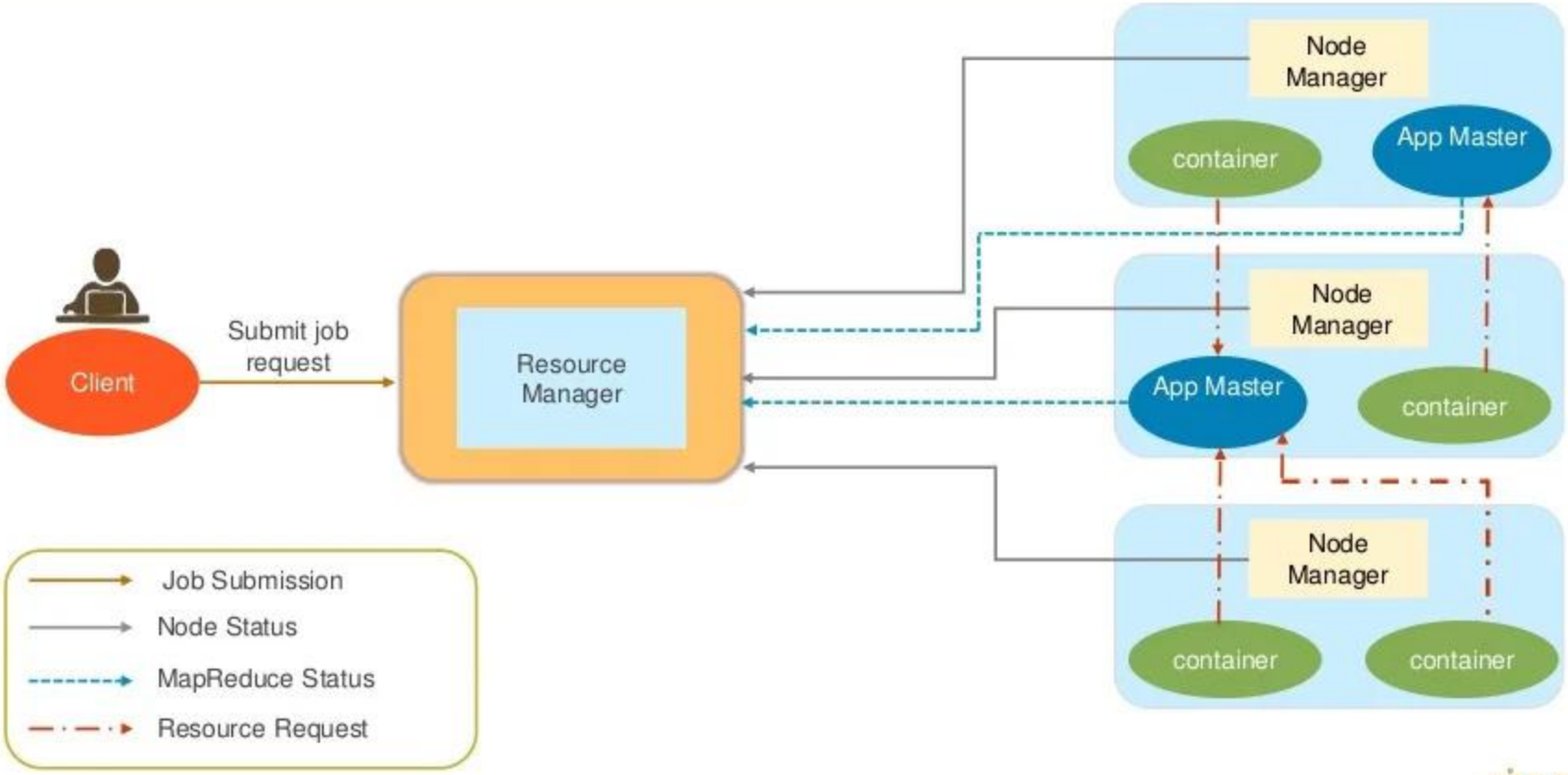
YARN Architecture



YARN Architecture

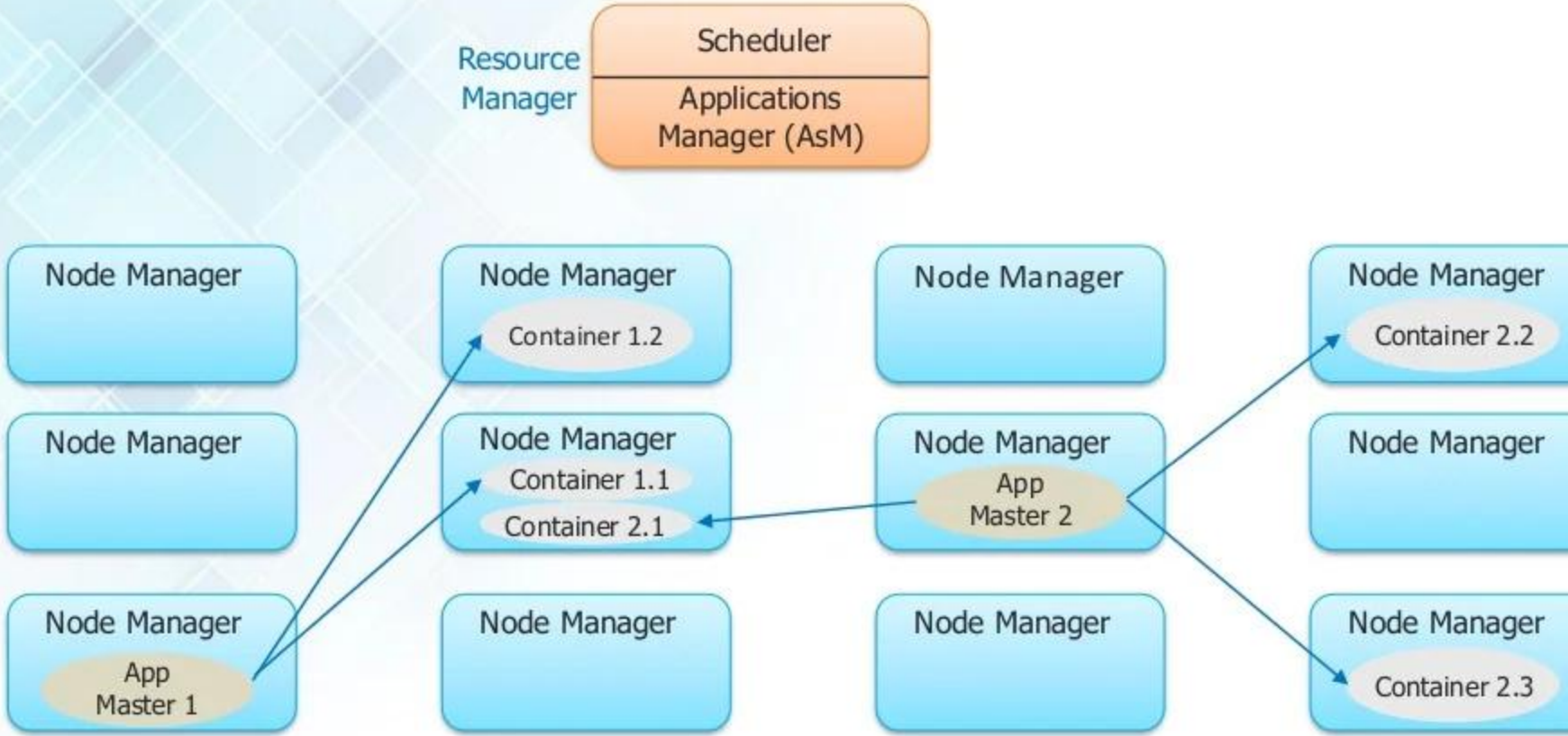


YARN Architecture



YARN Application Workflow in MapReduce

YARN Workflow

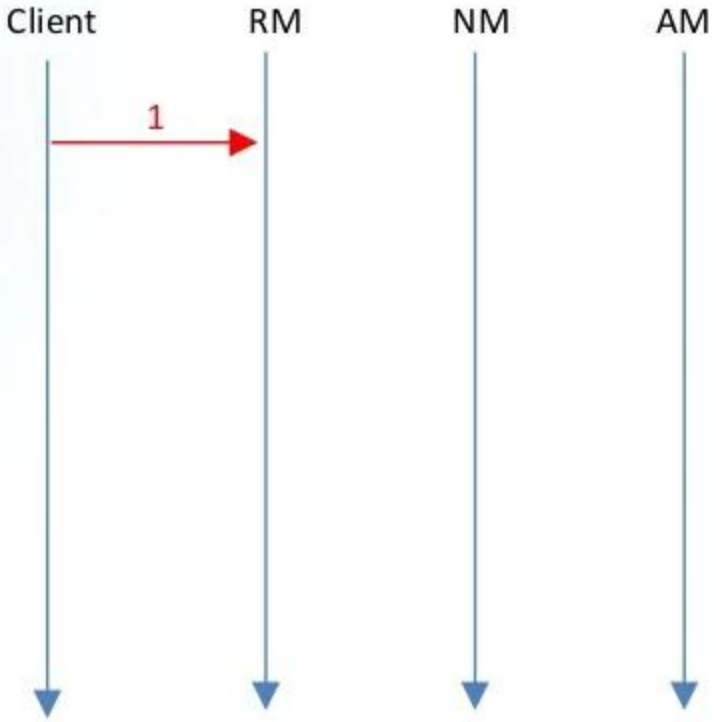


Application Workflow

edure

→ Execution Sequence :

1. Client submits an application

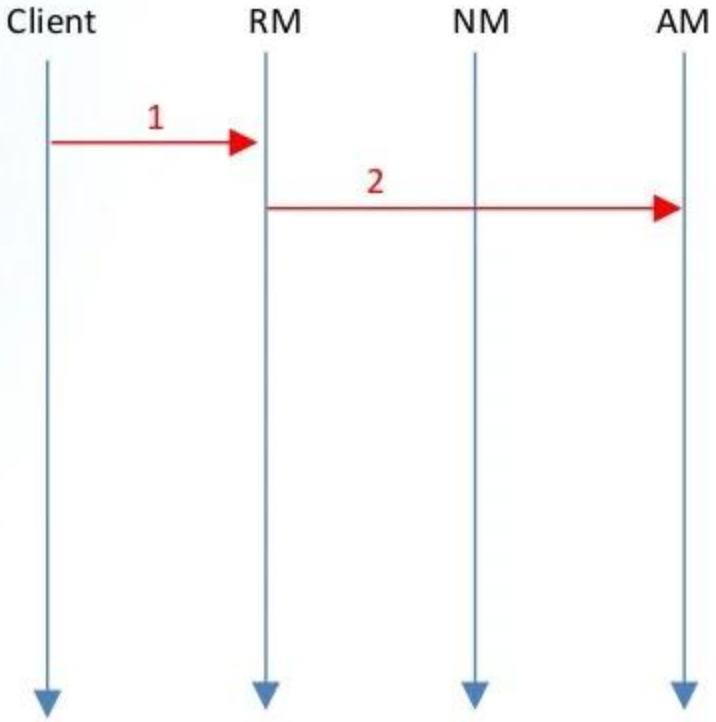


Application Workflow

edure

→Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM

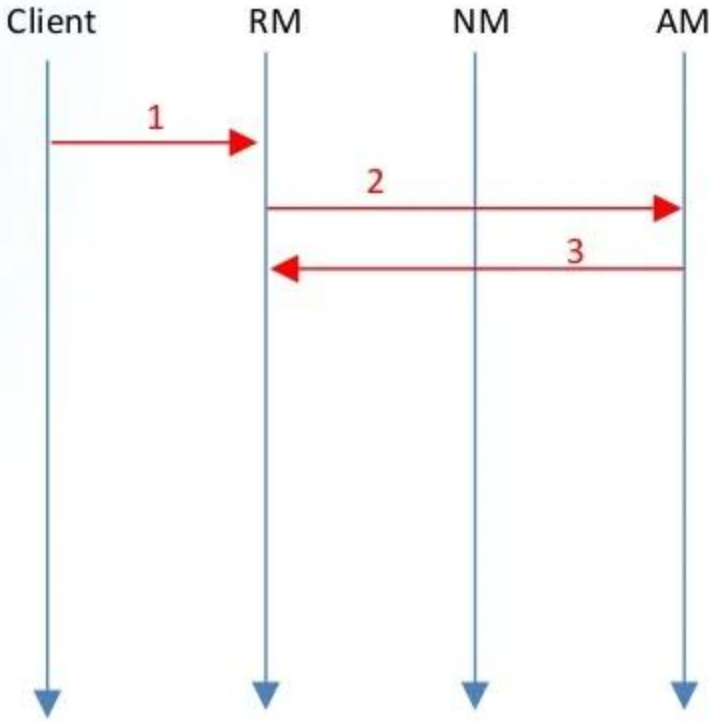


Application Workflow

edure

→ Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM

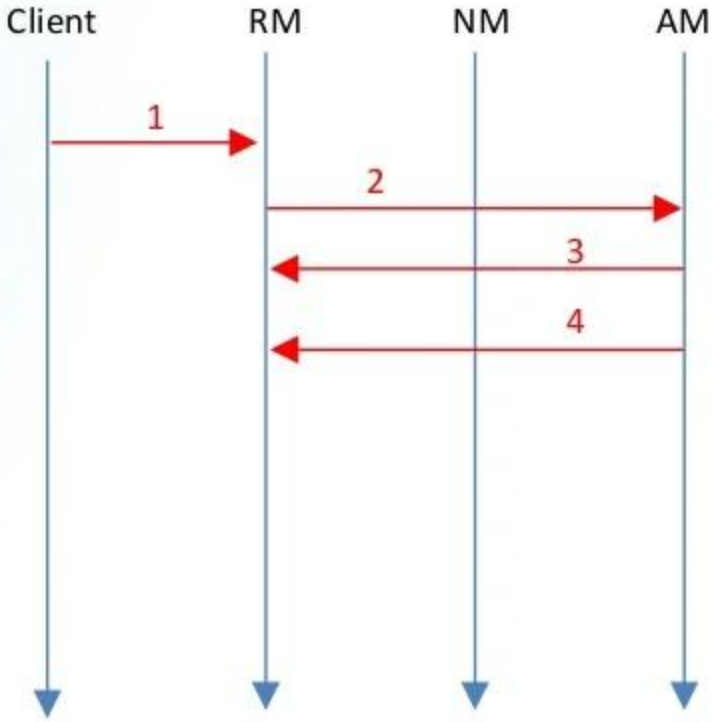


Application Workflow

edure

→ Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM

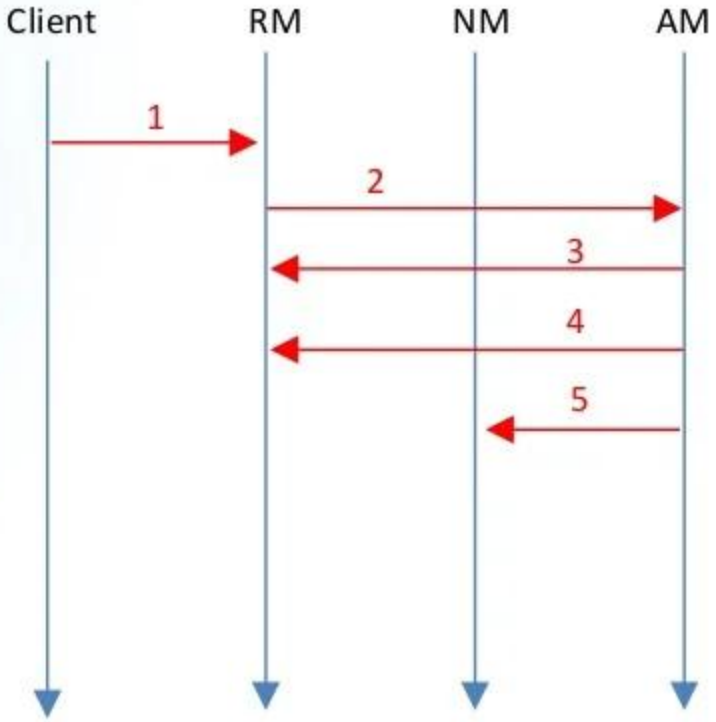


Application Workflow

edure

→Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM
5. AM notifies NM to launch containers

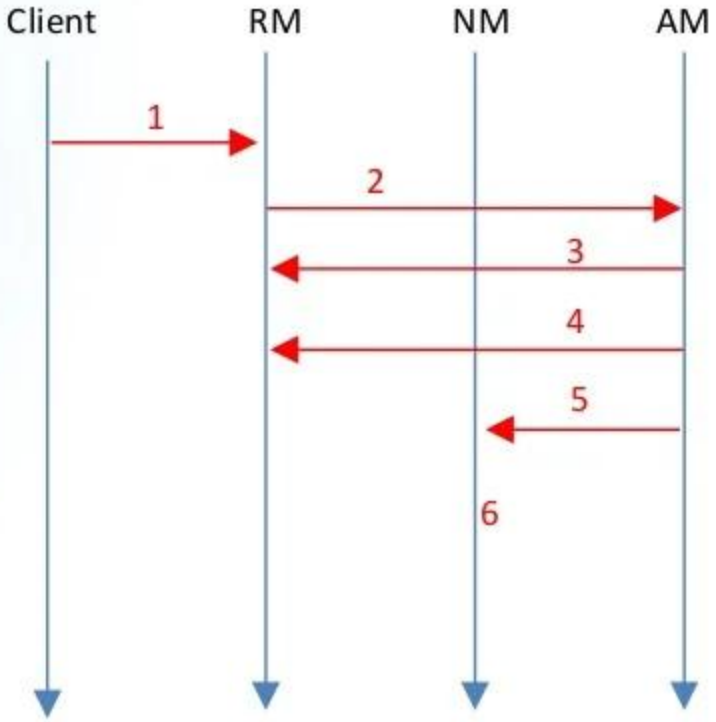


Application Workflow

edure

→ Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM
5. AM notifies NM to launch containers
6. Application code is executed in container

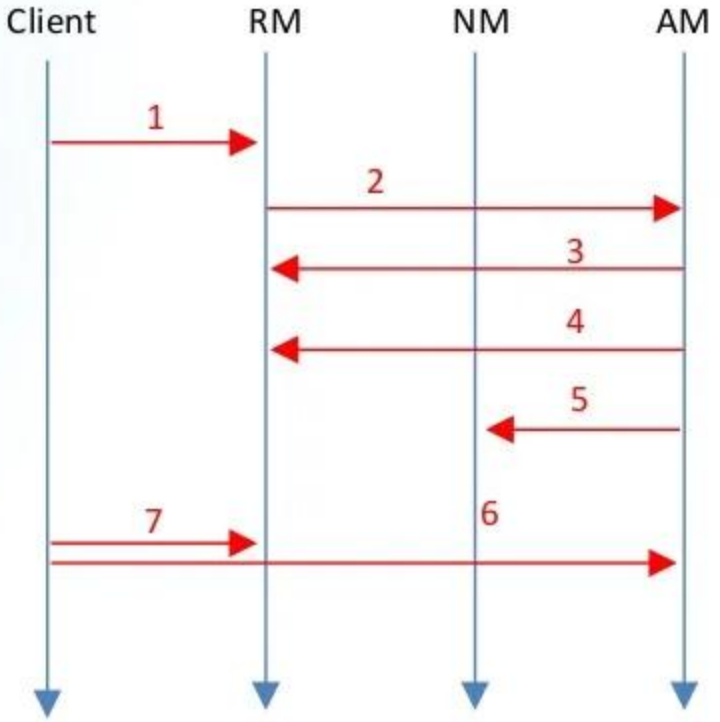


Application Workflow

edure

→ Execution Sequence :

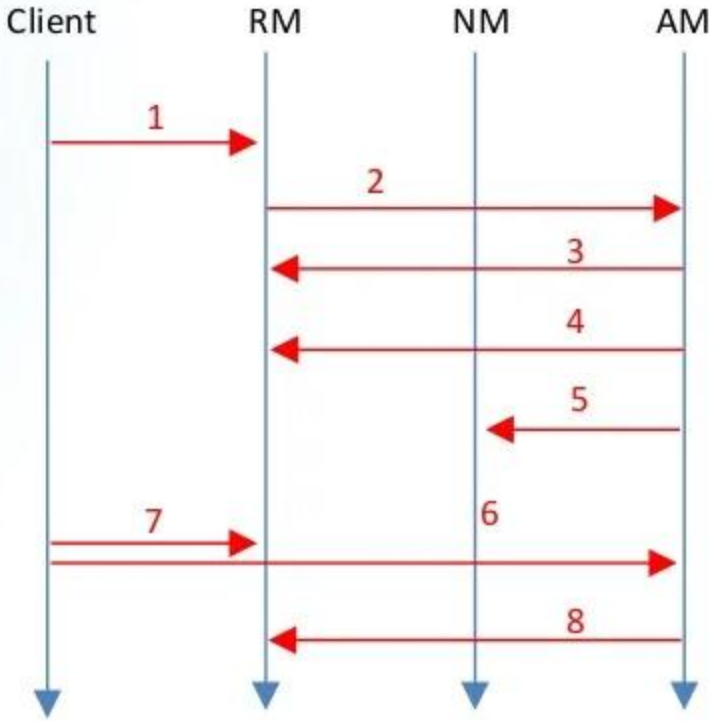
1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM
5. AM notifies NM to launch containers
6. Application code is executed in container
7. Client contacts RM/AM to monitor application's status



Application Workflow

→ Execution Sequence :

1. Client submits an application
2. RM allocates a container to start AM
3. AM registers with RM
4. AM asks containers from RM
5. AM notifies NM to launch containers
6. Application code is executed in container
7. Client contacts RM/AM to monitor application's status
8. AM unregisters with RM



Thank You

Slide content from
Edureka and Simplilearn