

HCIA Big Data MapReduce and YARN Technical Principles

Eng. Hagar Muhammed

Lecture 3



Contents

- 1. MapReduce Overview**
2. Functions and Architectures of MapReduce
3. YARN overview
4. Functions and Architecture of YARN
5. Resource Management and Task Scheduling in YARN
6. Enhanced Features of YARN

MapReduce Overview

MapReduce is designed and developed based on the MapReduce paper published by Google. MapReduce is used for parallel and offline computing of large-scale data sets (above 1 TB).

MapReduce is a software framework and programming model used for processing huge amounts of data.

Hadoop is capable for running MapReduce programs written in various languages: Java

MapReduce Features

- **Highly abstract programming idea:** Programmers only need to describe what to do, and the execution framework will process the work accordingly.
- **Outstanding scalability:** Cluster capabilities can be improved by adding nodes.
- **High fault tolerance:** Cluster availability and fault tolerance are improved using policies such as computing or data migration.

Contents

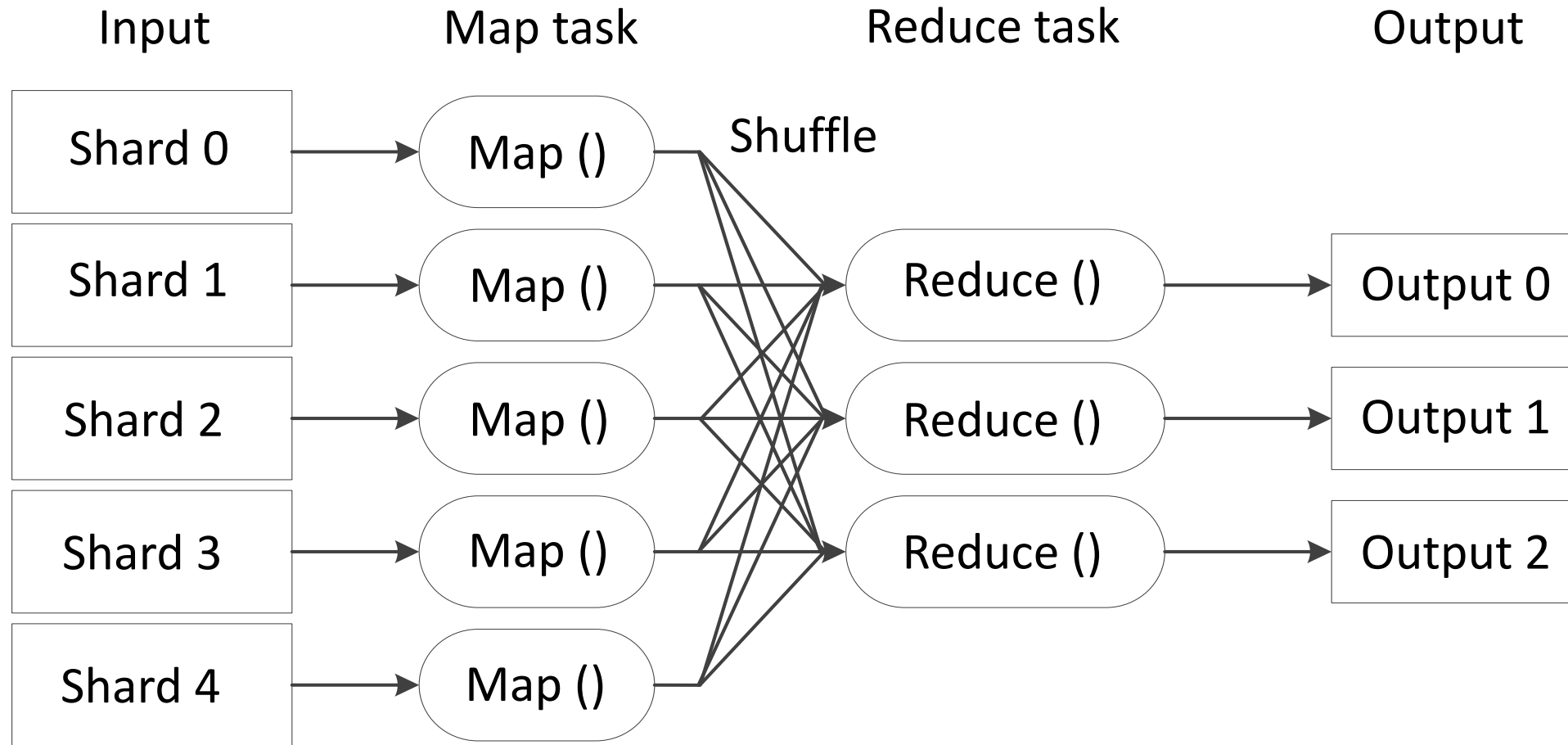
1. MapReduce Overview
- 2. Functions and Architectures of MapReduce**
3. YARN overview
4. Functions and Architecture of YARN
5. Resource Management and Task Scheduling in YARN
6. Enhanced Features of YARN

- The MapReduce calculation process can be divided into two phases: Map and Reduce. The output in the Map phase is the input in the Reduce phase.
- MapReduce can be understood as a process of summarizing a large amount of disordered data based on certain features and processing the data to obtain the final result.

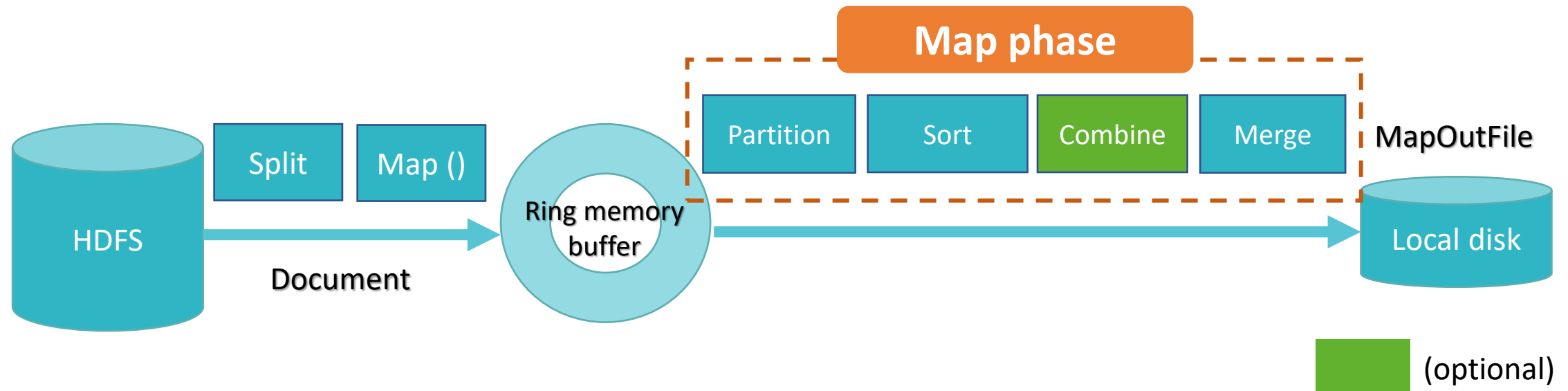
MapReduce Process

- In the Map phase, keys and values (features) are extracted from each piece of uncorrelated data.
- In the Reduce phase, data is organized by keys followed by several values. These values are correlated. Then, further processing may be performed to obtain a result.

MapReduce Working Process

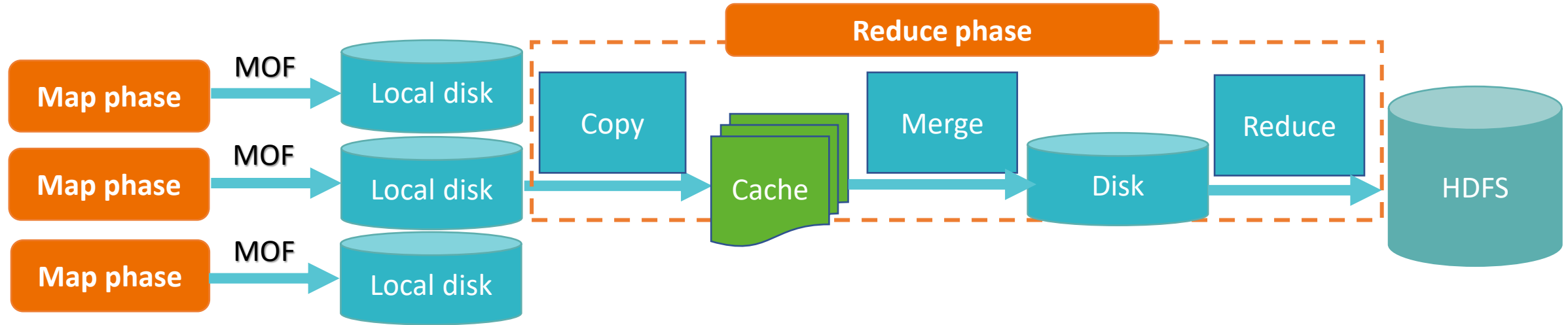


- Before jobs are submitted, the files to be processed are split. By default, the MapReduce framework regards a block as a split. Client applications can redefine the mapping between blocks and splits.
- In the Map phase, data is stored in a ring memory buffer. When the data in the buffer reaches about 80%, the data in the buffer needs to be written to the local disk.



Reduce Phase

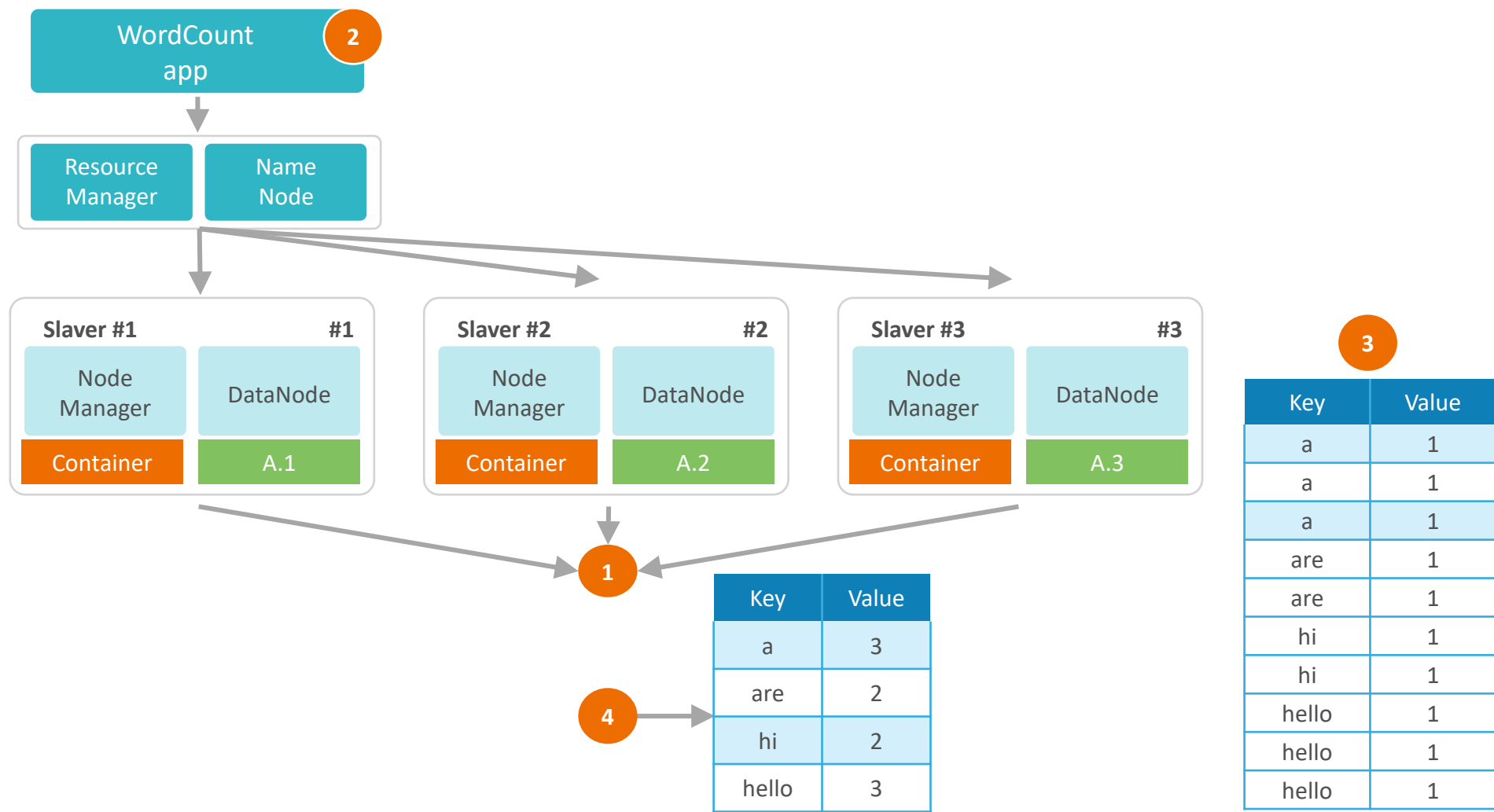
- MOF files need to be sorted. If Reduce tasks receive little data, the data is directly stored in the buffer. As the number of files in the buffer increases, the MapReduce background thread merges the files into one large file. Many intermediate files are generated during the merge. The last merge result is directly output to the Reduce function defined by the user.



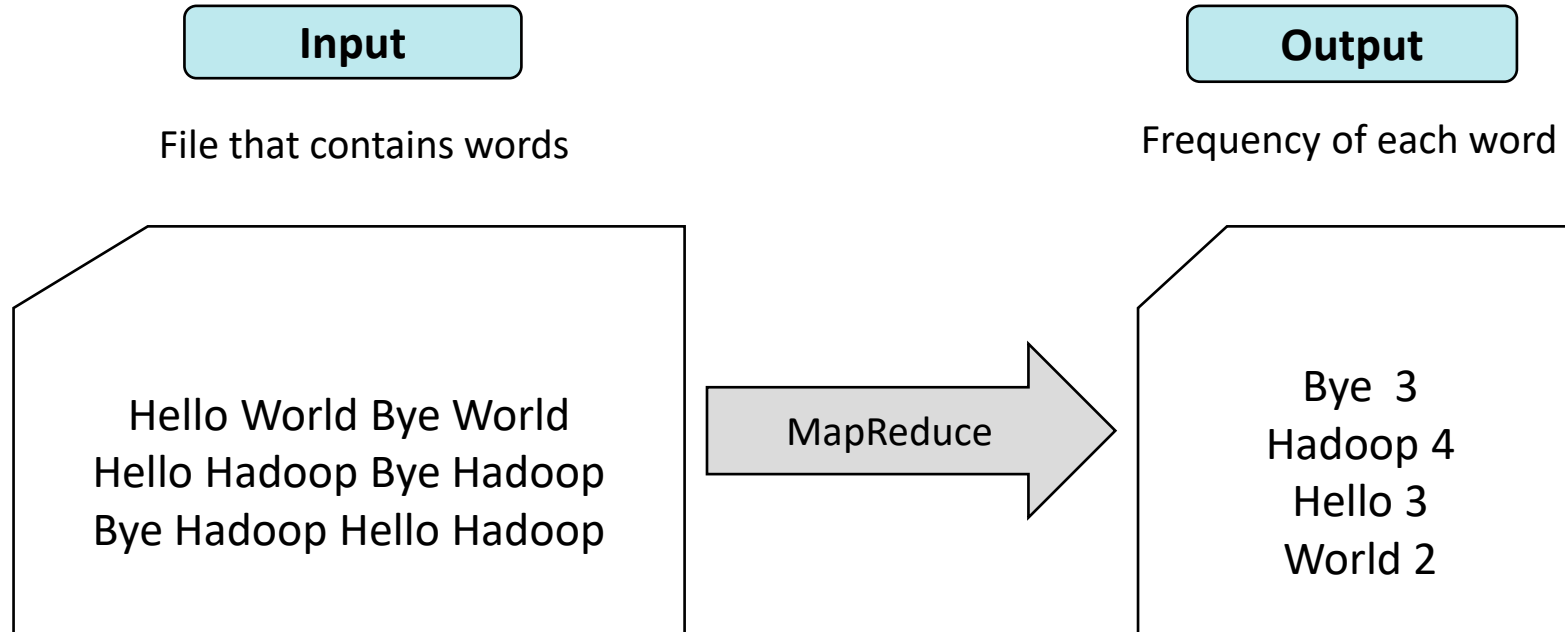
Shuffle Process

- Shuffle is a process of transferring intermediate data between the Map and Reduce phases, including obtaining MOF files by Reduce tasks from Map tasks as well as sorting and merging MOF files.

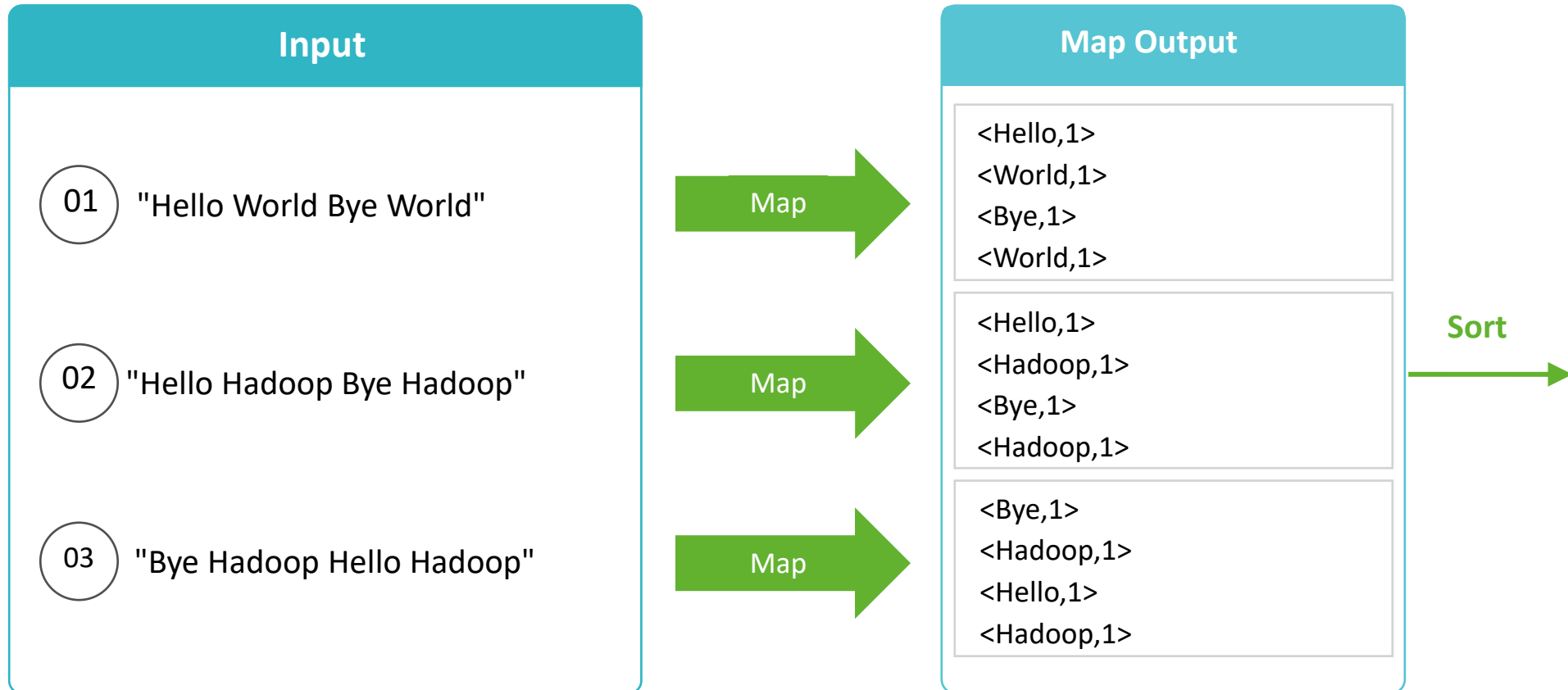
Example WordCount



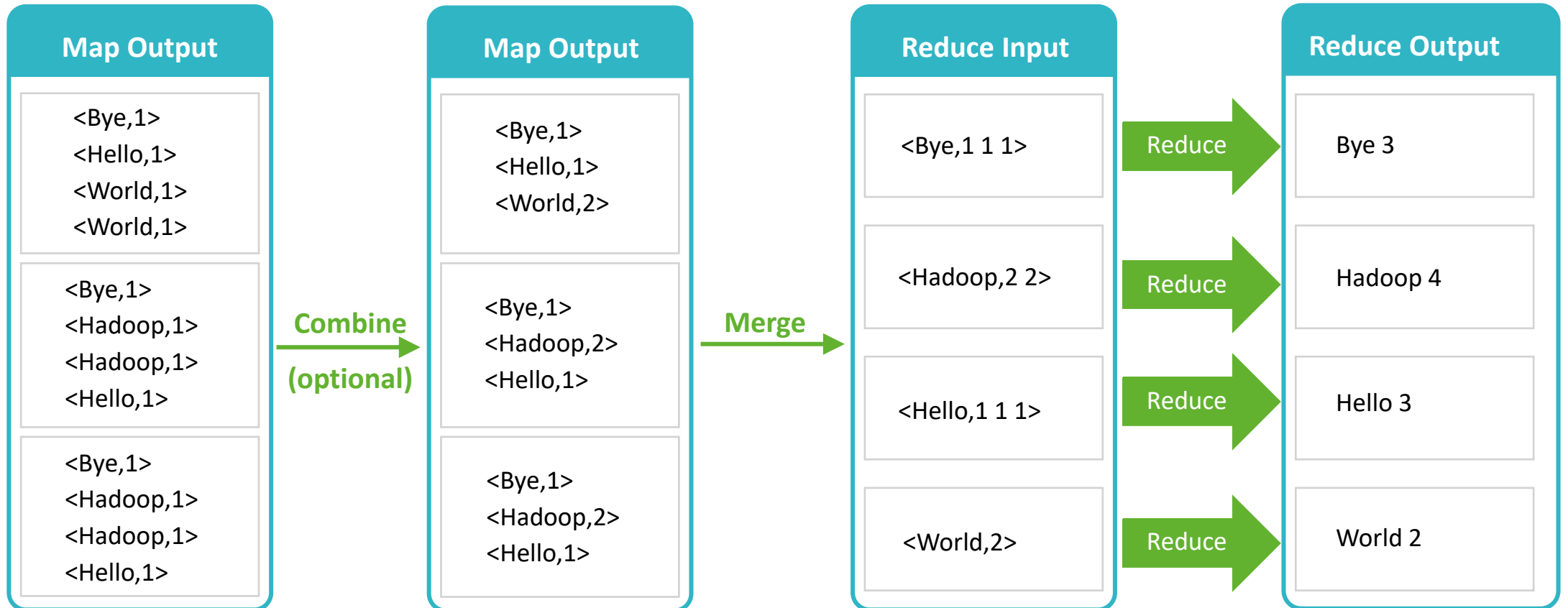
Functions of WordCount



Map Process of WordCount



Reduce Process of WordCount



Resource Scheduling and Allocation

Hadoop 1.0 only has HDFS and MapReduce. Resources are scheduled using MapReduce 1.0, which has the following disadvantages:

The master node has SPOFs. Fault recovery depends on the periodic checkpoint, which does not ensure the reliability. When a fault occurs, the system notifies users who have to manually decide whether to perform recalculation.

Resource Scheduling and Allocation

Job and resource scheduling are not distinguished. When MapReduce is running, many concurrent jobs exist in the environment. To solve this, diversified and efficient scheduling policies are needed.

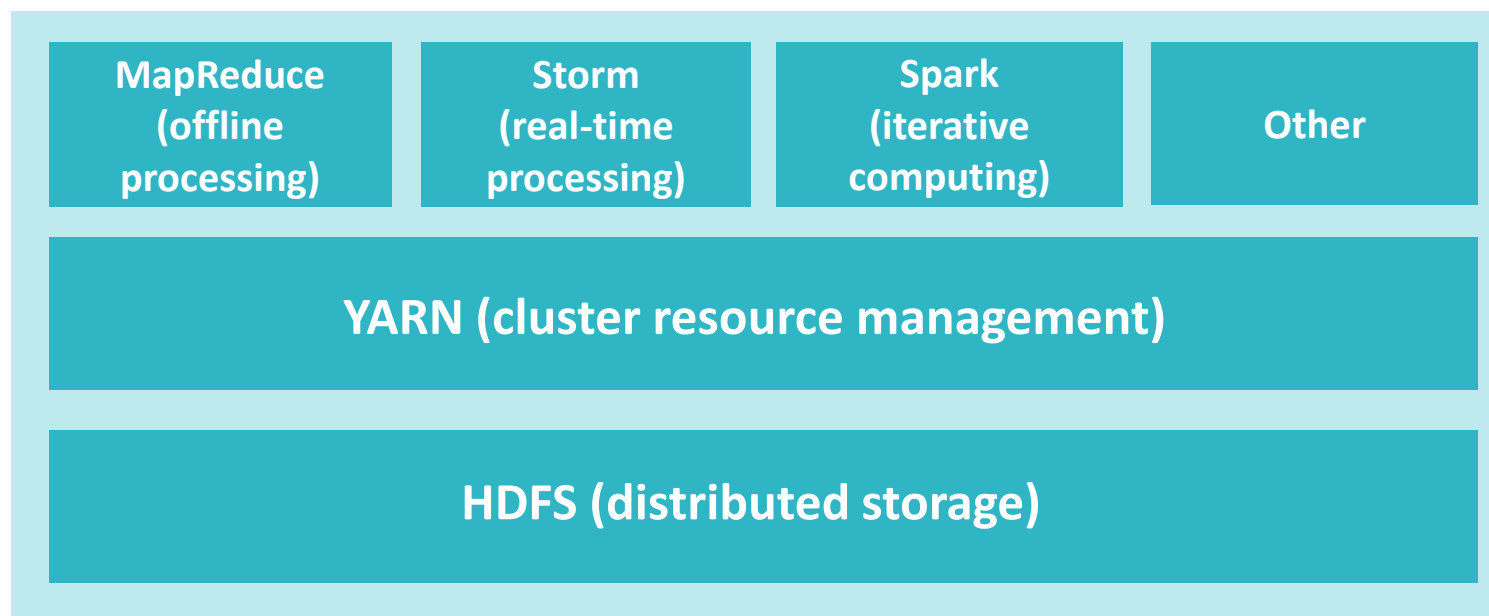
When many jobs are concurrently executed without considering resource isolation or security, it is difficult to ensure that a single job does not occupy too many resources and that user programs are secure.

Contents

1. MapReduce Overview
2. Functions and Architectures of MapReduce
- 3. YARN overview**
4. Functions and Architecture of YARN
5. Resource Management and Task Scheduling in YARN
6. Enhanced Features of YARN

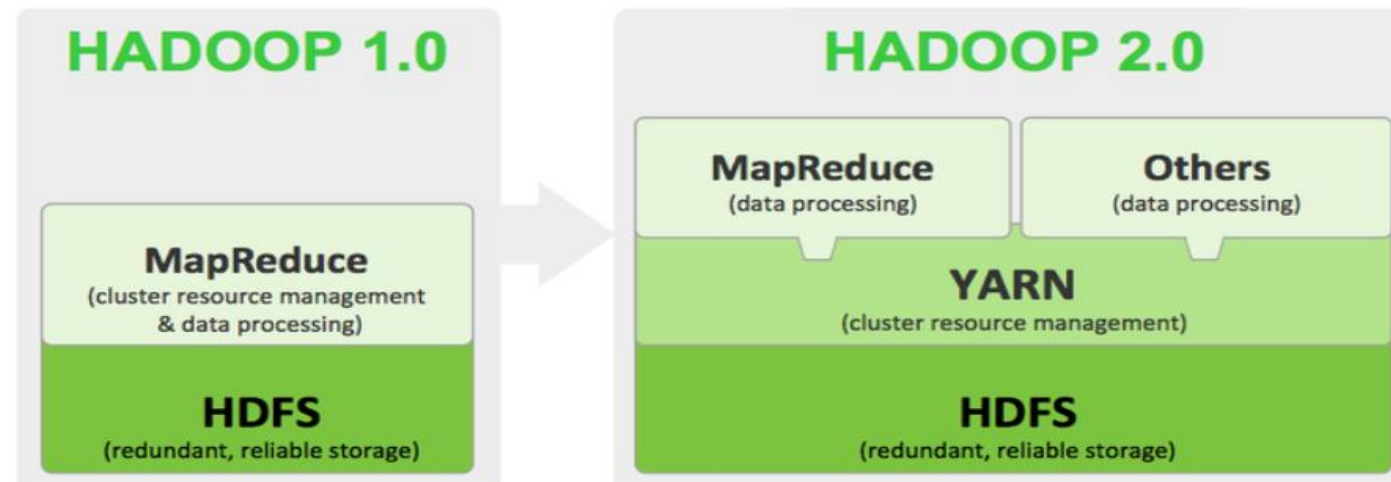
YARN Overview

- Apache Hadoop Yet Another Resource Negotiator (YARN) is a resource management system. It provides centralized resource management and scheduling for upper-layer applications, remarkably improving cluster resource utilization and data sharing.



YARN Overview

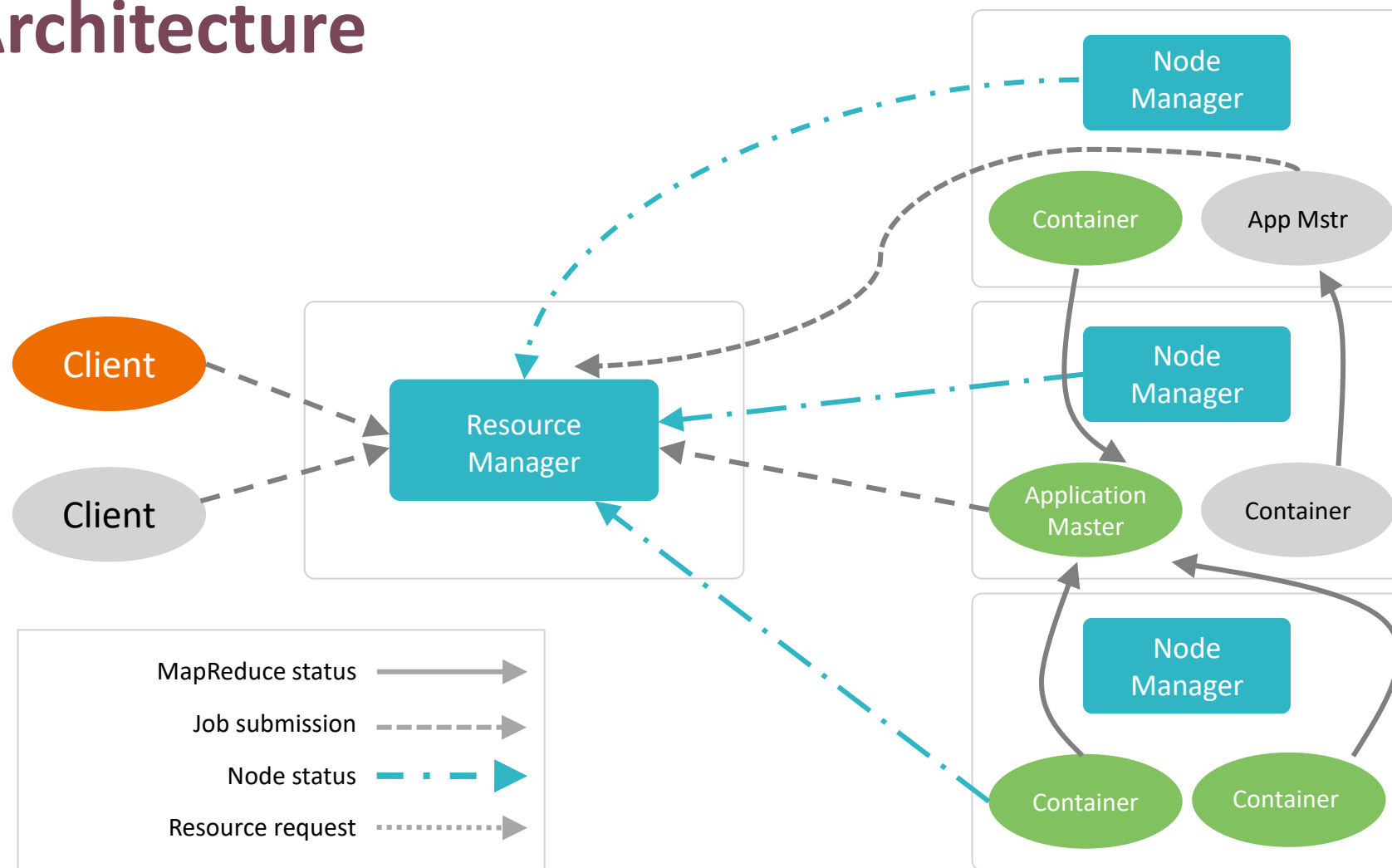
- Hadoop 2.0 introduces the YARN framework to better schedule and allocate cluster resources to overcome the shortcomings of Hadoop 1.0 and support various programming requirements.
- Hadoop 2 has better scalability than Hadoop 1 and is scalable up to 10000 nodes per cluster.



Contents

1. MapReduce Overview
2. Functions and Architectures of MapReduce
3. YARN overview
- 4. Functions and Architecture of YARN**
5. Resource Management and Task Scheduling in YARN
6. Enhanced Features of YARN

YARN Architecture



YARN Architecture

ResourceManager:

Centrally manages and allocates all resources in the cluster.

ApplicationMaster

Applies for and releases resources from the scheduler, requests NodeManager to run tasks, traces application status, and monitors their processes.

YARN Architecture

NodeManager:

Interacts with ApplicationMaster and ResourceManager.

Receives and executes container-related commands, and reports the container health status.

YARN Architecture

Container:

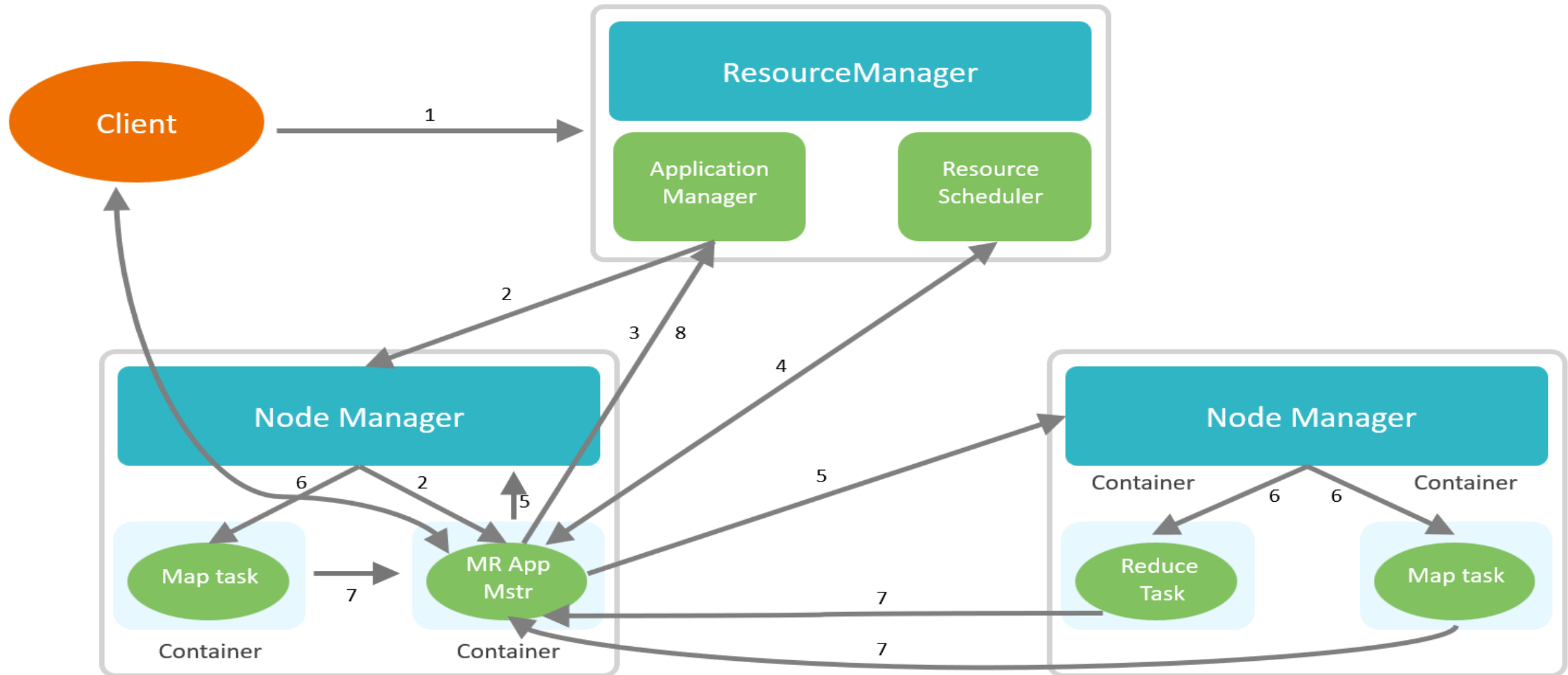
Encapsulates a certain number of resources on a node.

The clients submit jobs. ResourceManager receives the jobs and starts and monitors the first container

App Mstr notifies NodeManager to manage resources and start other containers.

The jobs are running in containers.

Job Scheduling Process of MapReduce on YARN



Job Scheduling Process of MapReduce on YARN

Step 1: A client submits applications to YARN, including ApplicationMaster, commands to start ApplicationMaster, and user programs.

Step 2: ResourceManager allocates the first container to the applications and asks NodeManager to start ApplicationMaster in the container.

Job Scheduling Process of MapReduce on YARN

Step 3: ApplicationMaster registers with ResourceManager so users can directly view the operating status of the applications. Then, ResourceManager applies for resources for each task and monitors the tasks until they finish (that is, repeat steps 4–7).

Step 4: ApplicationMaster applies for and obtains resources from ResourceManager in polling mode.

Job Scheduling Process of MapReduce on YARN

Step 5: Once the resources are obtained, ApplicationMaster asks NodeManager to start the tasks.

Step 6: After setting an operating environment, NodeManager writes task start commands into a script and runs this script to start the tasks.

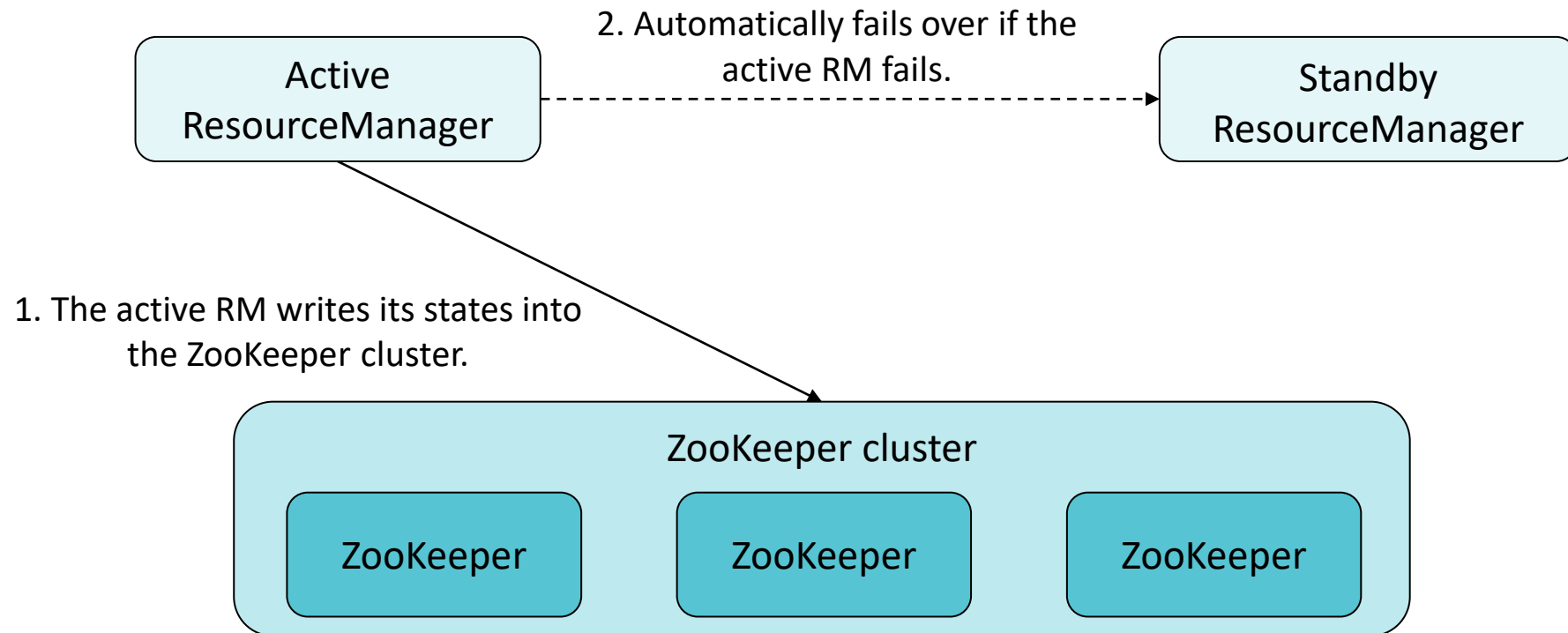
Job Scheduling Process of MapReduce on YARN

Step 7: Each task uses RPC to report status and progress to ApplicationMaster so that ApplicationMaster can restart a task if the task fails. Users can access applications at any time while they are running.

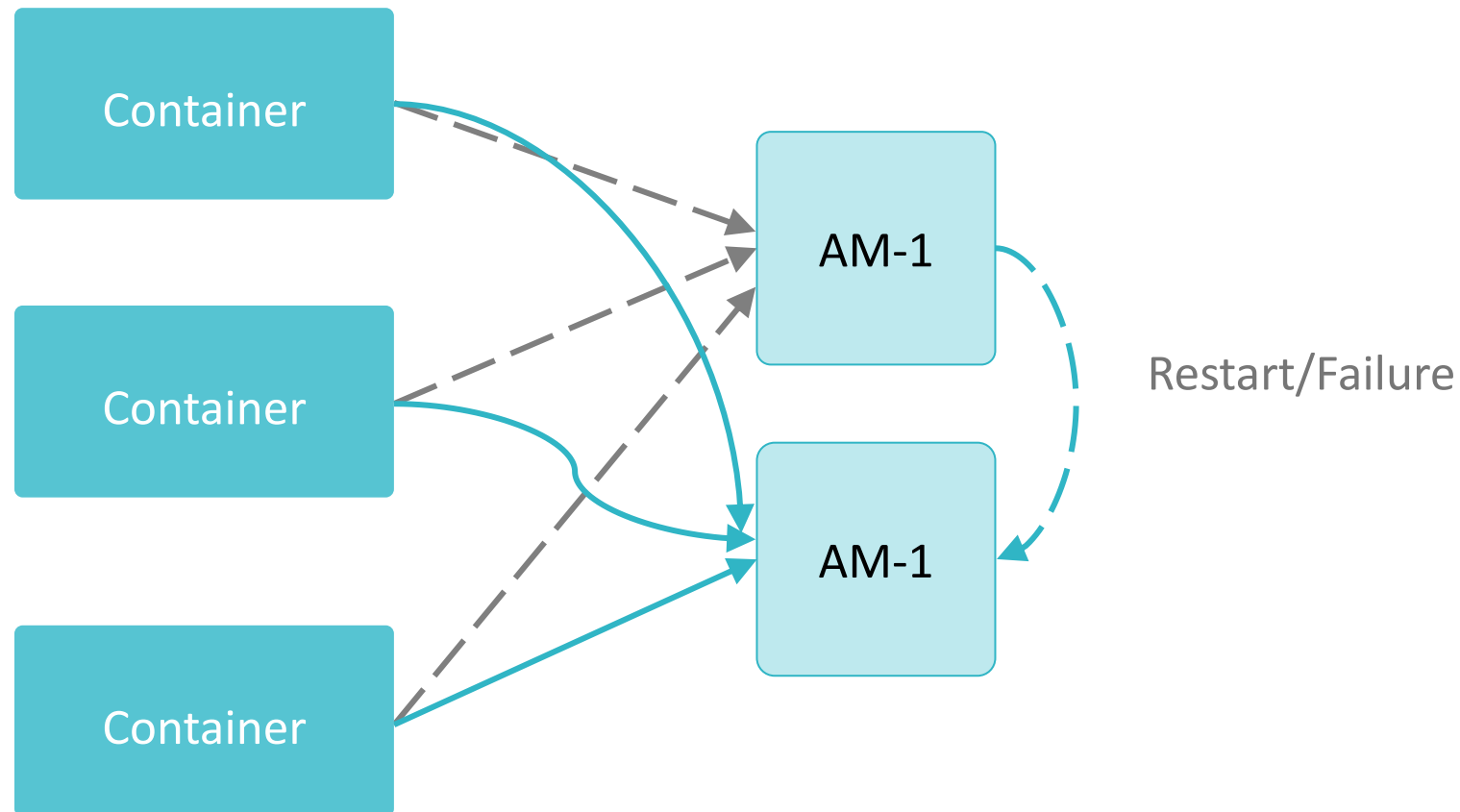
Step 8: After the applications end, ApplicationMaster deregisters itself from ResourceManager and closes

HA Solution of YARN

- ResourceManager in YARN is responsible for resource management and task scheduling of the entire cluster. The YARN HA solution uses redundant ResourceManager nodes to solve SPOFs.



Fault Tolerance Mechanism of YARN ApplicationMaster



Contents

1. MapReduce Overview
2. Functions and Architectures of MapReduce
3. YARN overview
4. Functions and Architecture of YARN
- 5. Resource Management and Task Scheduling in YARN**
6. Enhanced Features of YARN

Resource Management

- Available memory and CPU resources allocated to each NodeManager can be configured through the following parameters (on the YARN service configuration page):
 - **yarn.nodemanager.resource.memory-mb**: size of physical memory that YARN can allocate to a container

Resource Management

- **yarn.nodemanager.vmem-pmem-ratio**: ratio of the virtual memory to the physical memory
- **yarn.nodemanager.resource.cpu-vcore**: number of vCores that can be allocated to a container

Task scheduling in Hadoop YARN

YARN supports different scheduling algorithms, that govern how resources are allocated to different applications and queues. These algorithms help ensure fair sharing of resources.

Each node in the cluster is managed by a NodeManager. NodeManagers are responsible for managing resources on individual nodes and launching containers to execute tasks.

Task scheduling in Hadoop YARN

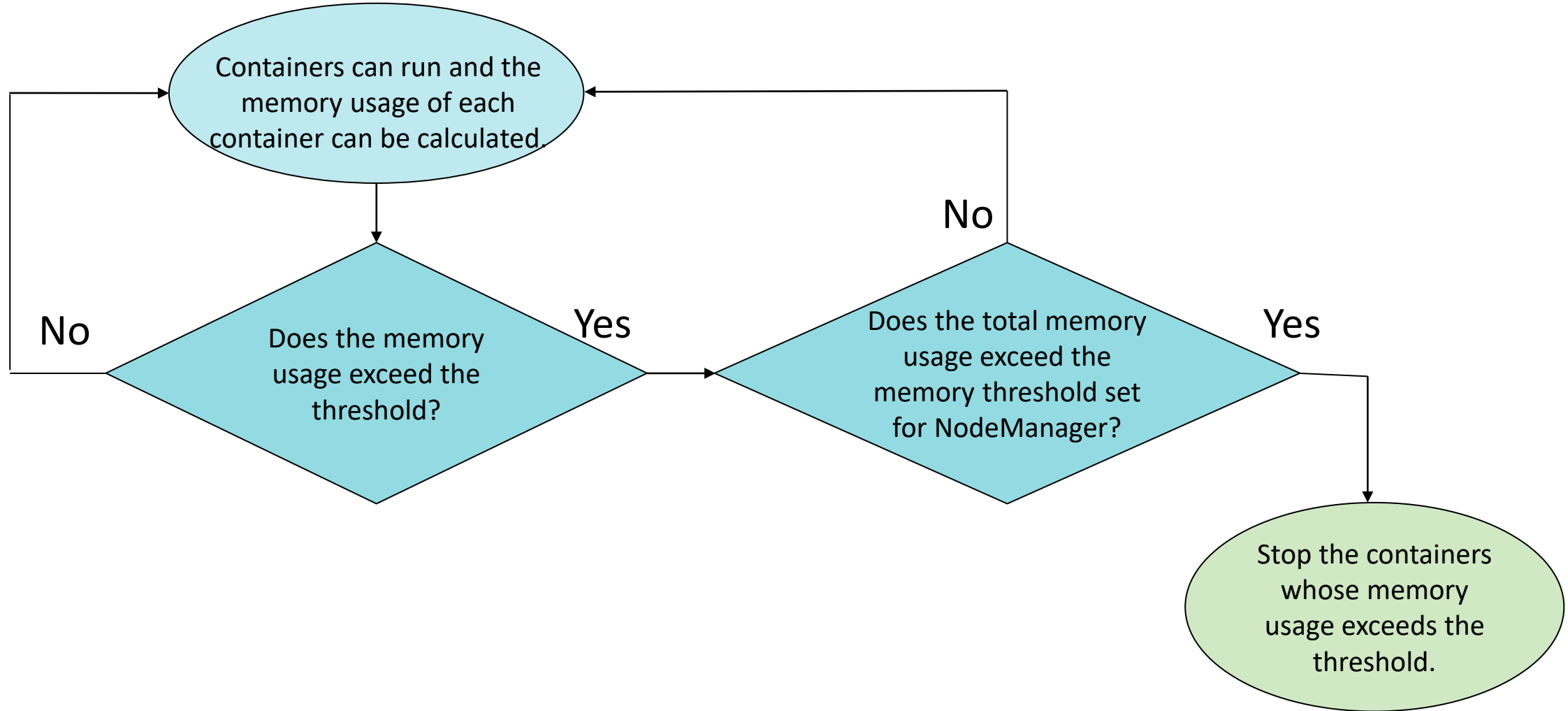
When the ResourceManager receives resource requests from the ApplicationMaster, it considers various factors like queue capacities, application priorities, and available resources on each node. Based on these considerations, it decides how many containers and which nodes to allocate to the application.

Once the ResourceManager makes the allocation decision, it informs the ApplicationMaster about the allocated containers.

Contents

1. MapReduce Overview
2. Functions and Architectures of MapReduce
3. YARN overview
4. Functions and Architecture of YARN
5. Resource Management and Task Scheduling in YARN
- 6. Enhanced Features of YARN**

Enhanced Feature — Dynamic YARN Memory Management



Summary

- This chapter describes the use cases and basic architectures of MapReduce and YARN, principles of YARN resource management and task scheduling, and enhanced features of YARN in Huawei MRS clusters.

Thank You
For Your Attention