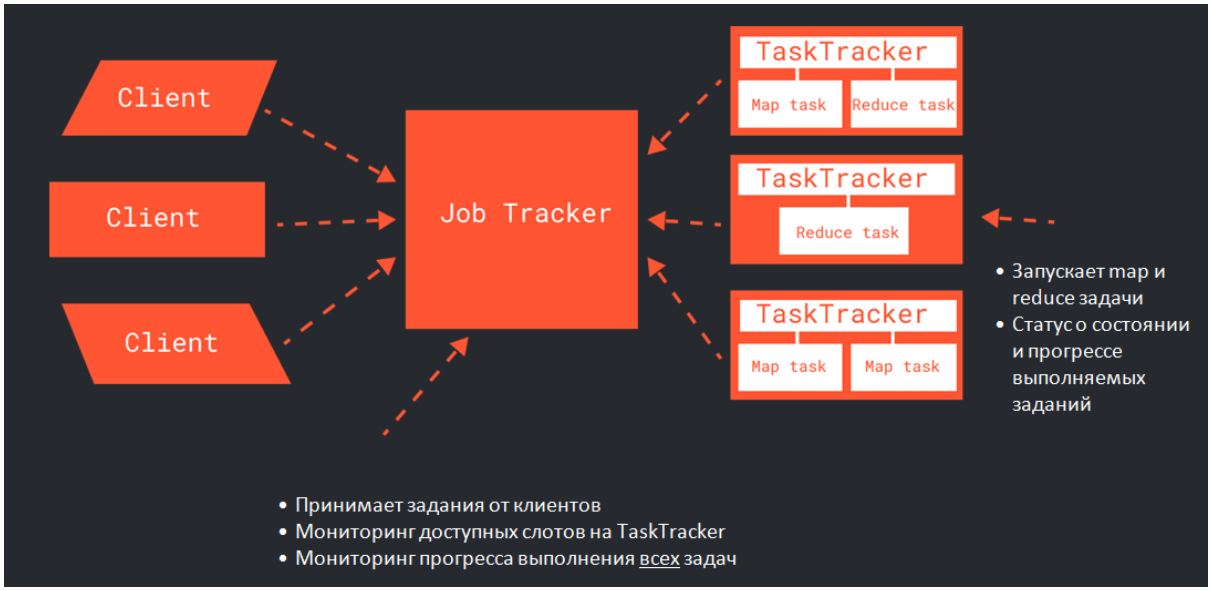
KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 2 урок > Основы HADOOP. YARN, MapReduce

> Оглавление

> Оглавление
> Конфигурация выполнения задач в Hadoop v.1
> Причины поиска альтернативы для Job Tracker
> YARN
> YARN в экосистеме
> YARN Scheduler
    > YARN FIFO Scheduler
    > YARN Capacity Scheduler
    > YARN Fair Scheduler
> MapReduce
> Как происходит обработка данных MapReduce
> Реализация MapReduce в Hadoop
> Составляющие MapReduce каркаса Hadoop
> Глоссарий

> Конфигурация выполнения задач в Hadoop v.1



Client – даёт задание в виде собранного jar-файла, которое попадает на Job Tracker.

Job Tracker – единый сервис-планировщик заданий, который является общим координатором внутри кластера:

- принимает задания от клиентов,
- мониторинг доступных слотов на Task Tracker,
- мониторинг процесса выполнения всех задач.

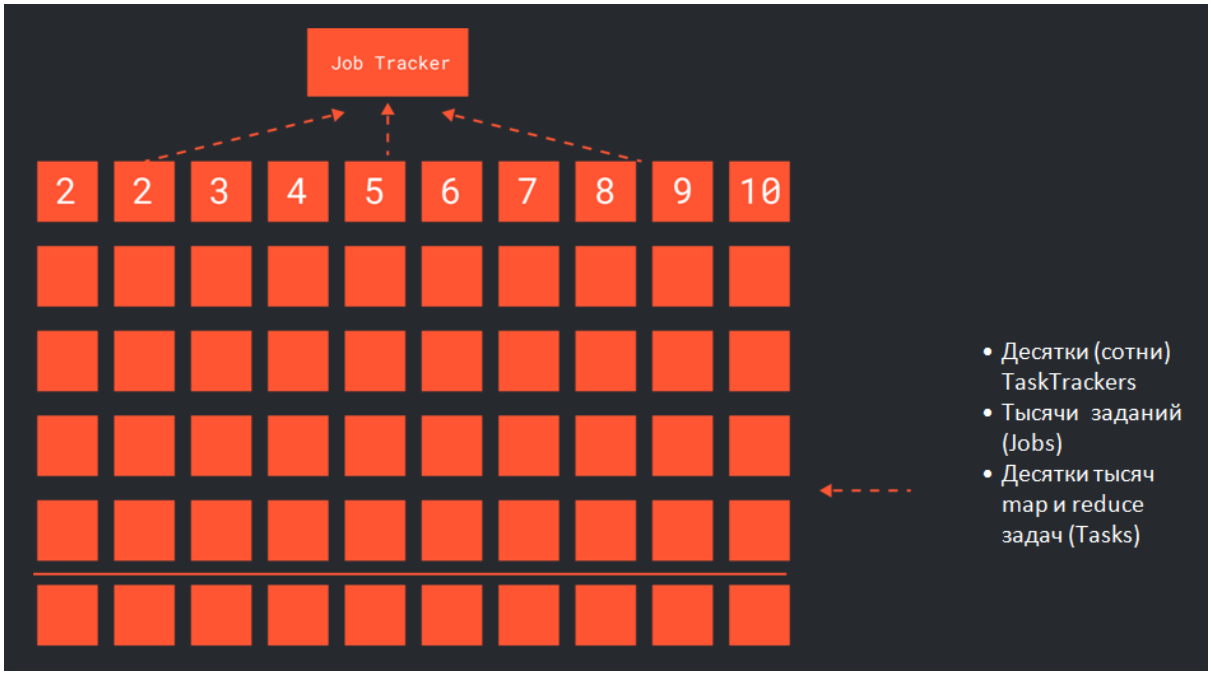
Job Tracker распределяет задания между Task Tracker на основе поступающей от них информации.

Task Tracker – сервис, который разворачивается на каждом сервере внутри кластера:

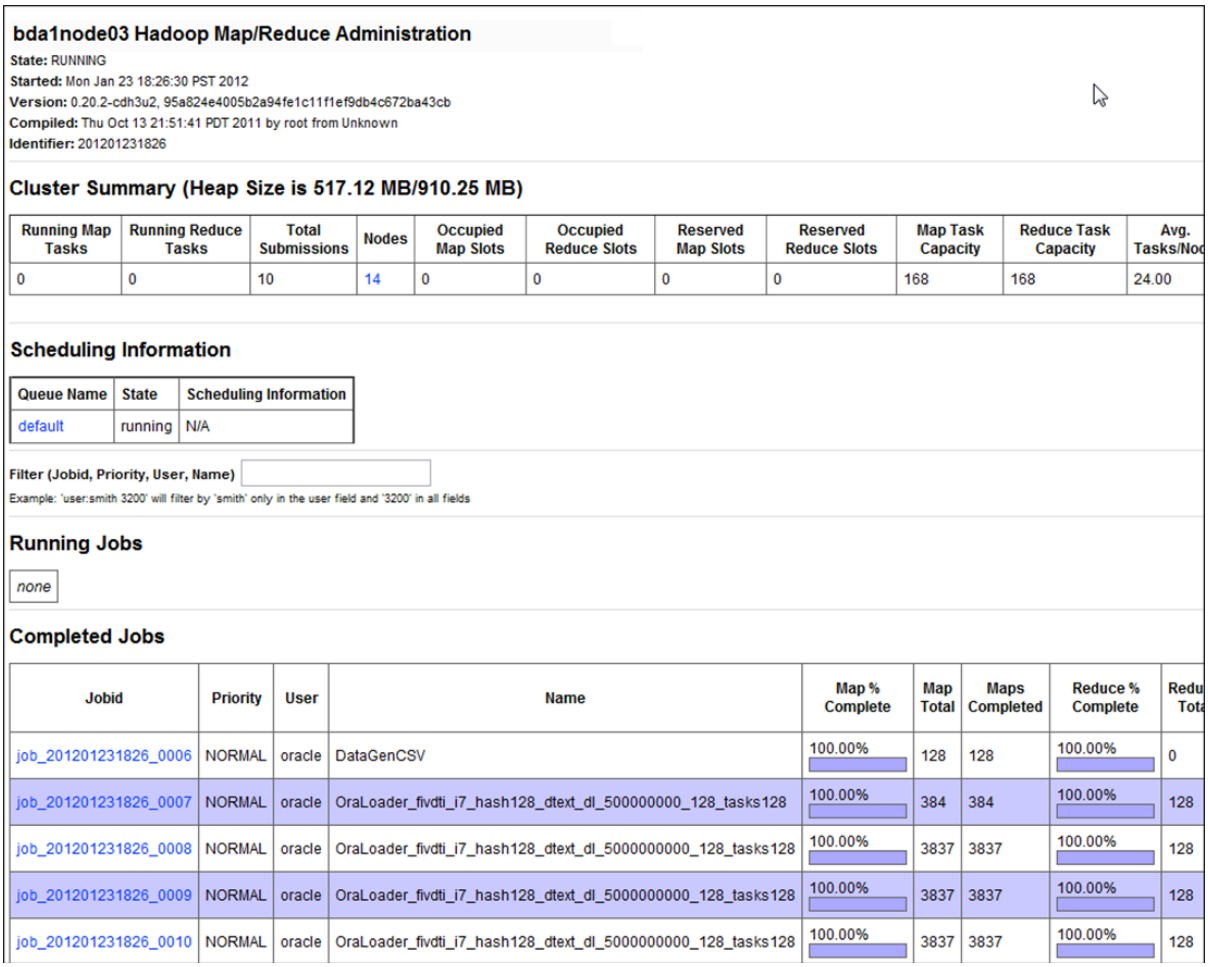
- запускает Map и Reduce задачи и жёстко распределяет вычислительные ресурсы между ними,
- контролирует статус о состоянии и прогрессе выполняемых заданий.

Каждый Task Tracker передаёт информацию о выполняемых задачах и используемых ресурсах в Job Tracker.

> Причины поиска альтернативы для Job Tracker



Могло быть большое количество серверов в кластере, на каждом сервере свой Task Tracker и каждый из них отправляет на Job Tracker информацию о задачах. При большом количестве заданий и большом объёме кластера Job Tracker не справлялся с потоком информации и начинал выходить из строя. Требовалась перезагрузка, возникал простой, могла потеряться информация о текущих задачах, приходилось их перезапускать. Таким образом, требовалось изменить подход к конфигурации.



bda1node03 Hadoop Map/Reduce Administration

State: RUNNING
Started: Mon Jan 23 18:26:30 PST 2012
Version: 0.20.2-cdh3u2, 95a824e4005b2a94fe1c11f1ef9db4c672ba43cb
Compiled: Thu Oct 13 21:51:41 PDT 2011 by root from Unknown
Identifier: 201201231826

Cluster Summary (Heap Size is 517.12 MB/910.25 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Noc
0	0	10	14	0	0	0	0	168	168	24.00

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

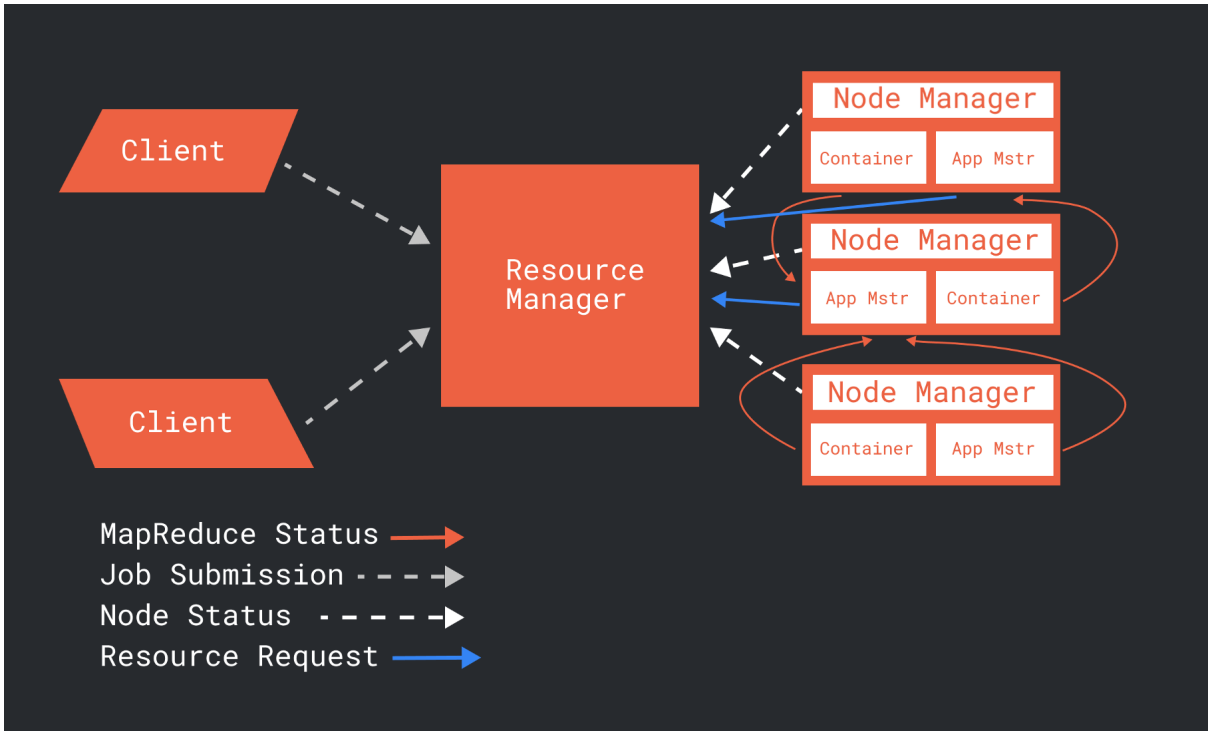
Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Redu Tota
job_201201231826_0006	NORMAL	oracle	DataGenCSV	100.00%	128	128	100.00%	0
job_201201231826_0007	NORMAL	oracle	OraLoader_fivdti_i7_hash128_dtext_dl_500000000_128_tasks128	100.00%	384	384	100.00%	128
job_201201231826_0008	NORMAL	oracle	OraLoader_fivdti_i7_hash128_dtext_dl_5000000000_128_tasks128	100.00%	3837	3837	100.00%	128
job_201201231826_0009	NORMAL	oracle	OraLoader_fivdti_i7_hash128_dtext_dl_5000000000_128_tasks128	100.00%	3837	3837	100.00%	128
job_201201231826_0010	NORMAL	oracle	OraLoader_fivdti_i7_hash128_dtext_dl_5000000000_128_tasks128	100.00%	3837	3837	100.00%	128

Старый интерфейс Job Tracker

> YARN

YARN (Yet Another Resource Negotiator) – модуль, появившийся с версией Hadoop 2.0, отвечающий за управление ресурсами кластеров и планирование заданий.



Главная идея YARN — предоставить две основные функции трекера заданий — управление ресурсами и запуск/мониторинг задач, двум отдельным компонентам: глобальному менеджеру ресурсов (Resource Manager) и менеджеру узлов (Node Manager).

Resource Manager (менеджер ресурсов) – планировщик ресурсов, абстрагирующий все вычислительные ресурсы кластера и управляющий их предоставлением. Для выполнения заданий оперирует понятием Container (контейнер).

Resource Manager:

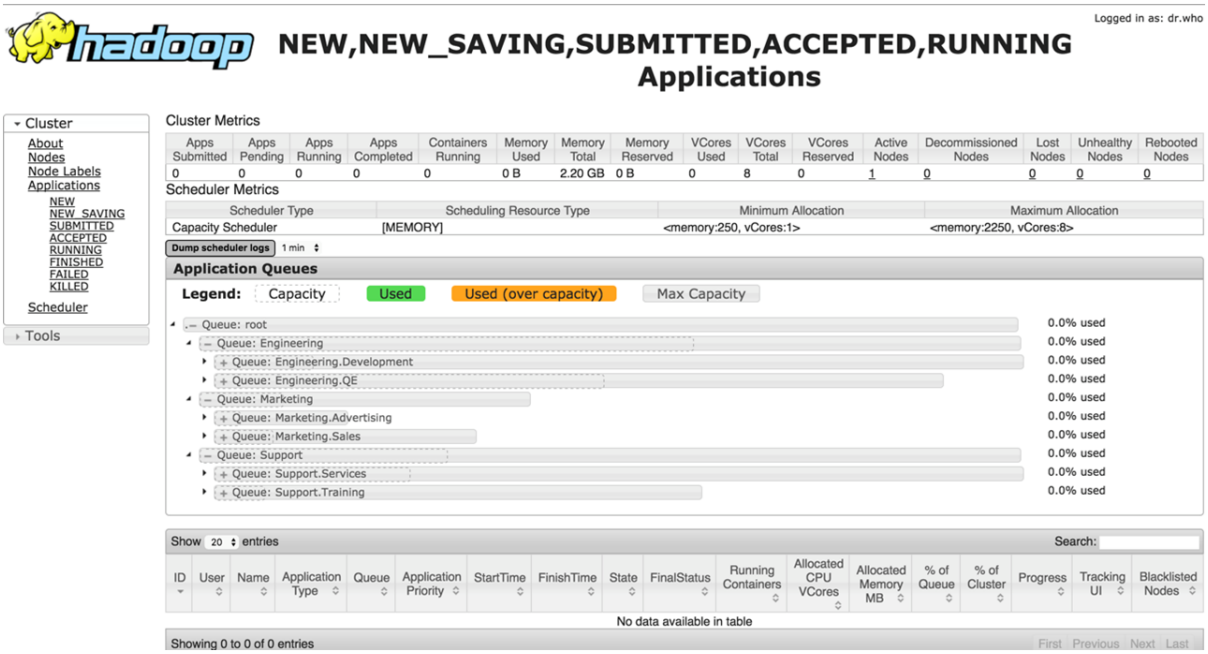
- принимает задание от Client,
- смотрит на доступные ресурсы и выбирает сервер, на котором есть доступные ресурсы,
- запускает на сервере Application Master (главный координатор того, как выполняется задание, следит, чтобы оно выполнялось корректно) и передаёт ему задание,
- выделяет по запросу Application Master контейнеры с ресурсами.

Node Manager (менеджер узлов) — это подчинённый процесс, работающий на каждом узле и отвечающий за запуск контейнеров приложений (Container),

мониторинг использования ресурсов контейнера (Application Master): процессор, память, диск, сеть, и передачу этих данных Resource Manager.

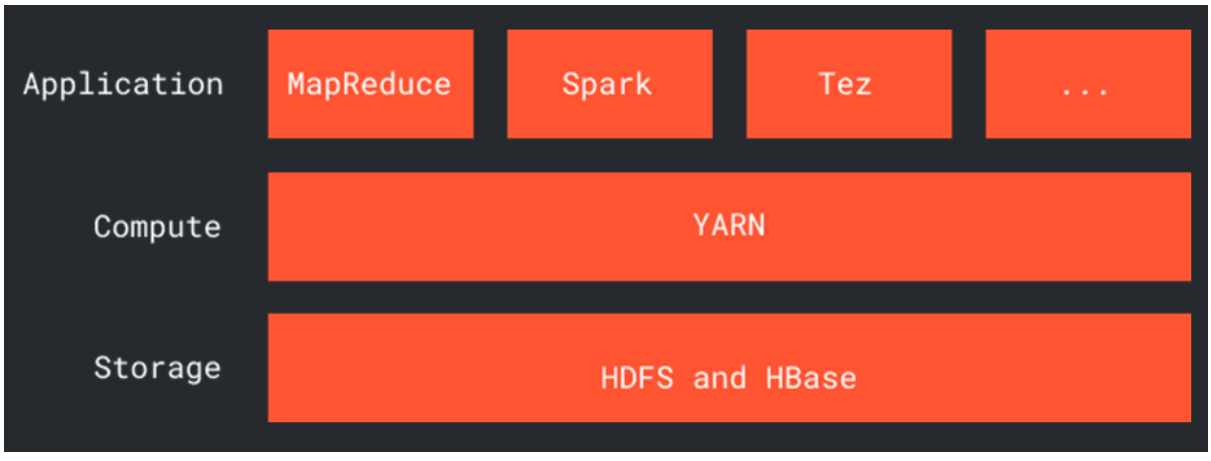
Application Master:

- запрашивает у Resource Manager ресурсы для выполнения заданий,
- взаимодействует с контейнером.



YARN Resource Manager UI (консоль менеджера ресурсов)

> YARN в экосистеме



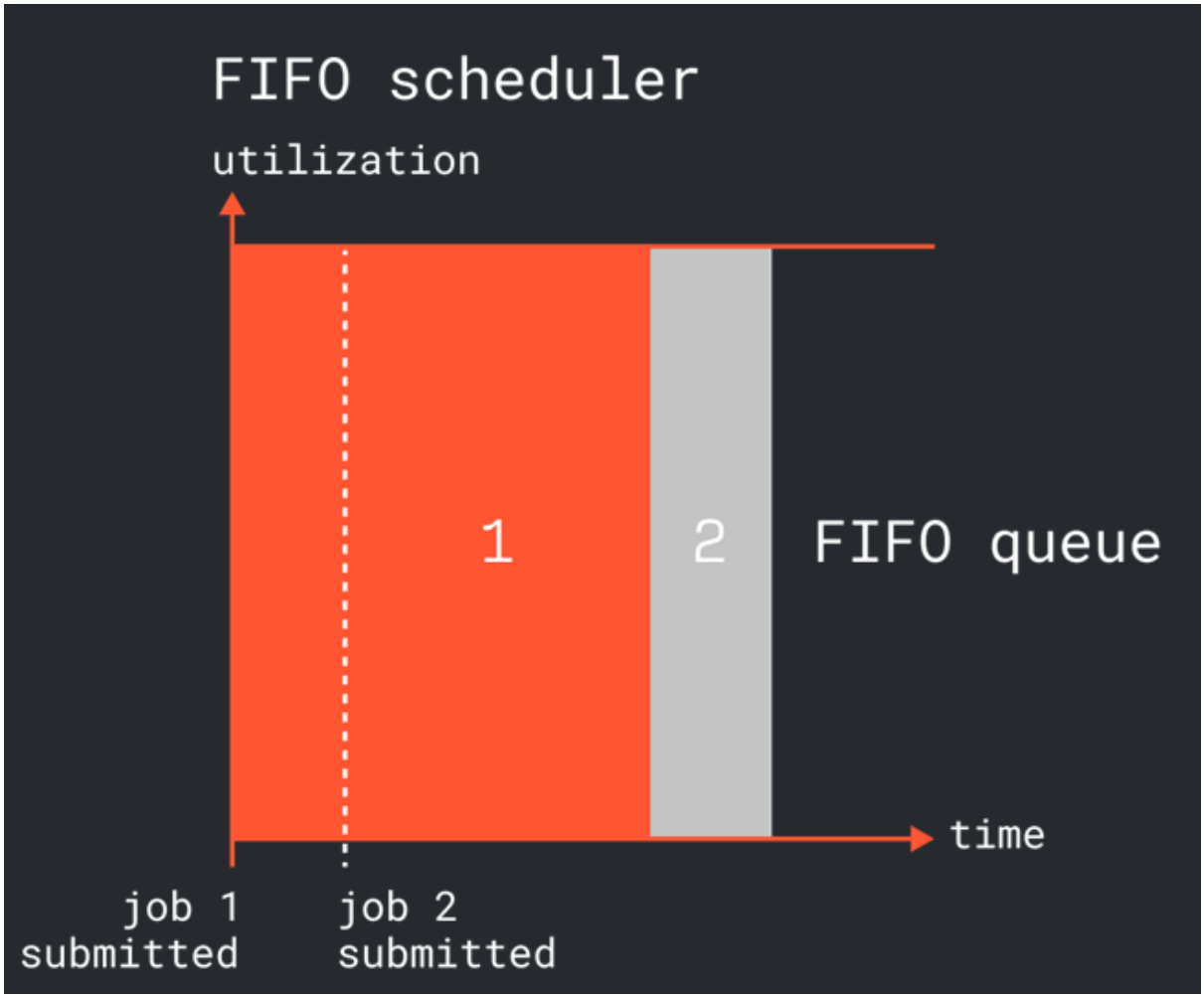
3х уровневая система:

1. Application (уровень приложений). Внутри YARN как внутри вычислительного движка можем запускать задания по какой-либо технологии: MapReduce, Spark, Tez.
2. Compute (уровень вычислений). Задание запускается в кластере, YARN выделяет ресурсы, следит за выполнением задания.
3. Storage (уровень хранения данных). Данные, которые мы обрабатываем берутся из HDFS, HBase или других источников.

> YARN Scheduler

YARN Scheduler – компонент, который занимается планированием выполнения задач, сколько ресурсов выделить задаче.

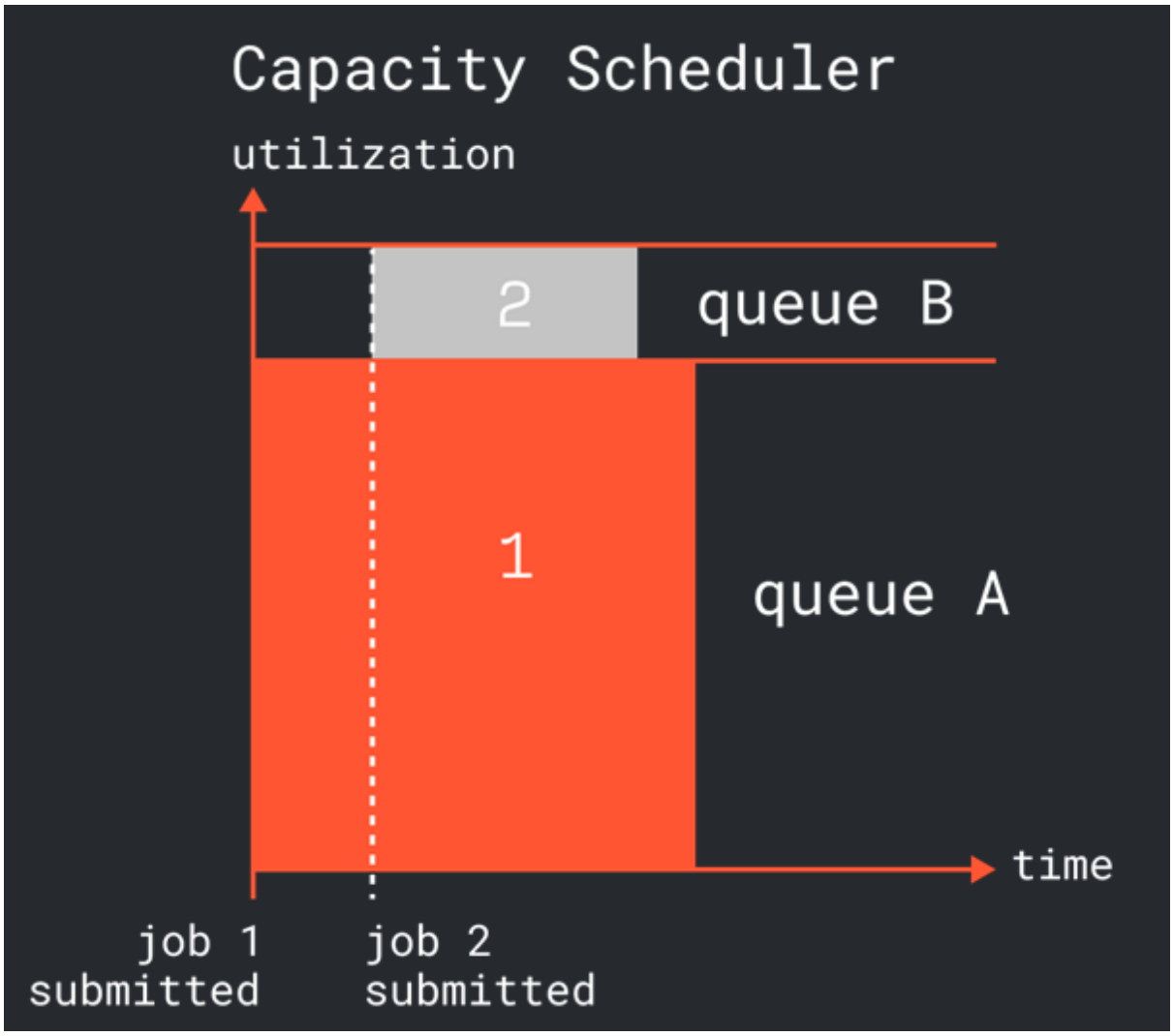
> YARN FIFO Scheduler



Первое задание, которое пришло, выполняется сначала. Второе задание будет выполнено только после выполнения первого задания и т.д. Таким образом формируется FIFO очередь.

YARN FIFO Scheduler крайне редко используется на практике.

> YARN Capacity Scheduler

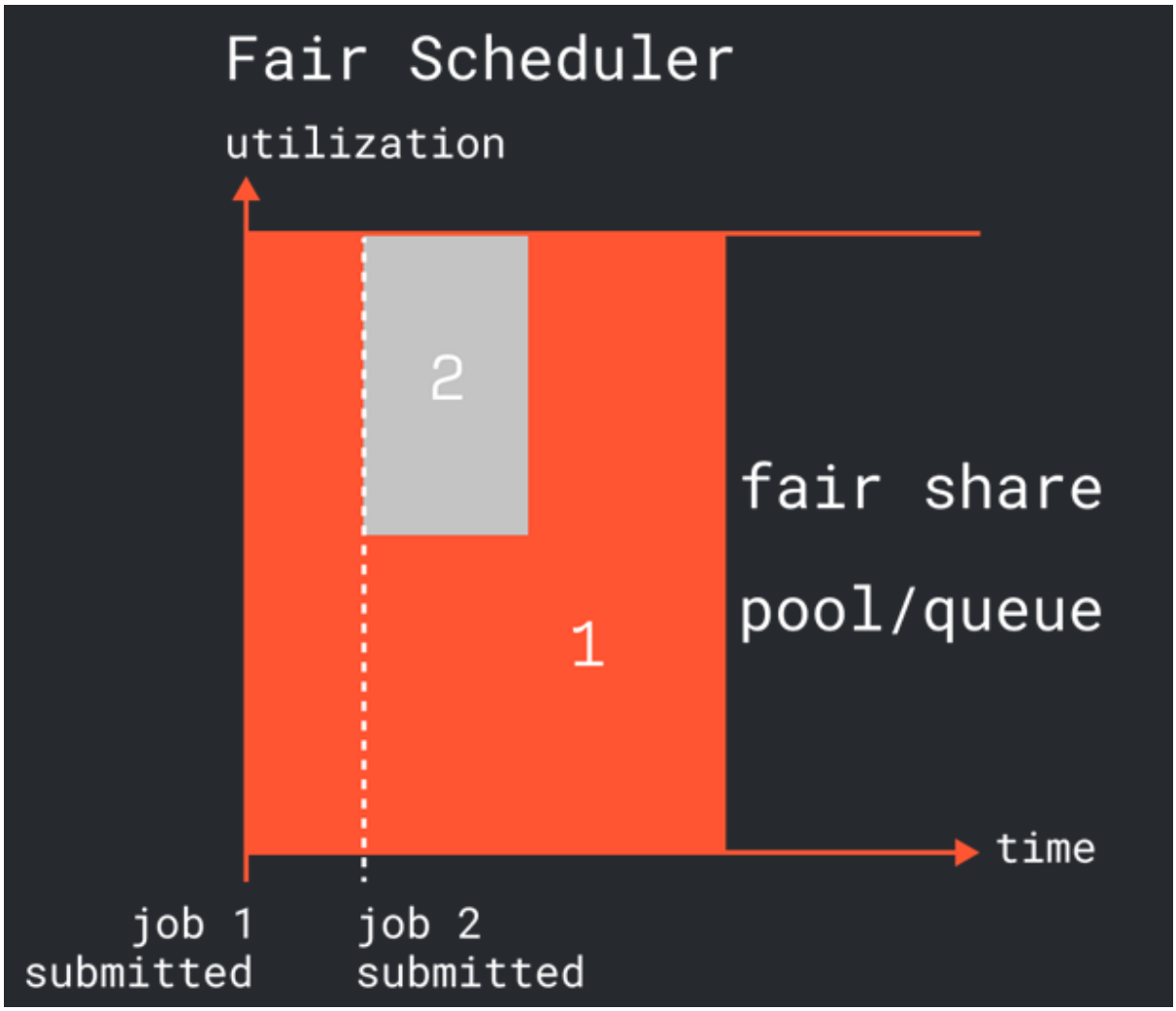


Основан на оценке ёмкости доступных ресурсов, распараллеливает выполнение заданий относительно FIFO Scheduler, но не позволяет правильно организовать утилизацию ресурсов. Если ресурсы заняты большими заданиями, то маленькие задания всё равно будут стоять в очереди и ожидать высвобождения ресурсов.

Запускаем задание в очереди A, также нам нужно запустить задание в очереди B. Если у нас есть доступные ресурсы, то второе задание будет запущено

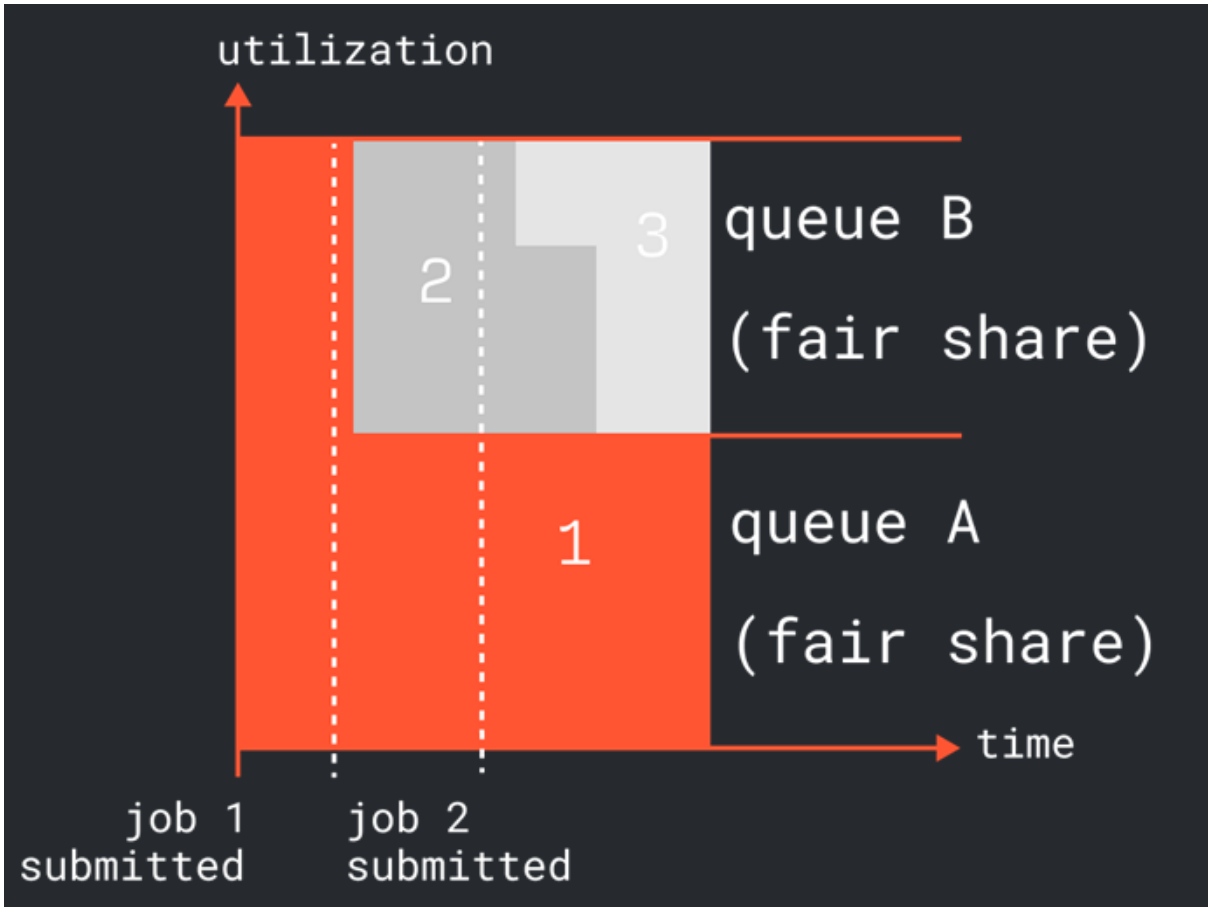
параллельно. Если у нас недостаточно ресурсов, чтобы запустить третье задание, то оно будет ожидать окончания первого или второго задания.

> YARN Fair Scheduler



Если наши ресурсы заняло большое задание и приходит на выполнение небольшое задание, то часть ресурсов задействованных для выполнения первого задания будут выделены для второго задания. Некоторые контейнеры для первого задания будут остановлены. Мы можем остановить ресурсоёмкие контейнеры, которые почти выполнились, и их придётся потом запускать повторно.

Большие задания принято запускать в отдельных очередях.

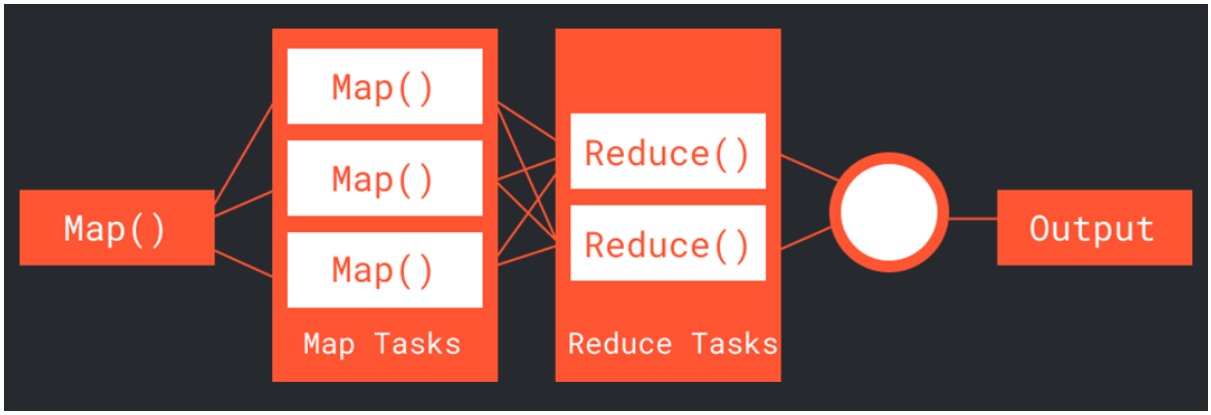


Сначала запустилось большое задание, которое заняло все ресурсы, потом запустилось второе не такое большое задание и заняло некоторое количество ресурсов. Далее запускается третье задание, которое забирает часть ресурсов второго задания. Таким образом, формируется две очереди, которые балансируют для честного использования ресурсов кластера.

Внутри очереди конкретному заданию можно назначить приоритет (целочисленное значение), чем больше приоритет, тем быстрее задание отправится на выполнение. Приоритеты назначаются редко.

> MapReduce

MapReduce – это модель параллельных вычислений, представленная компанией Google, которая используется для работы с большими объёмами данных в Hadoop кластере.

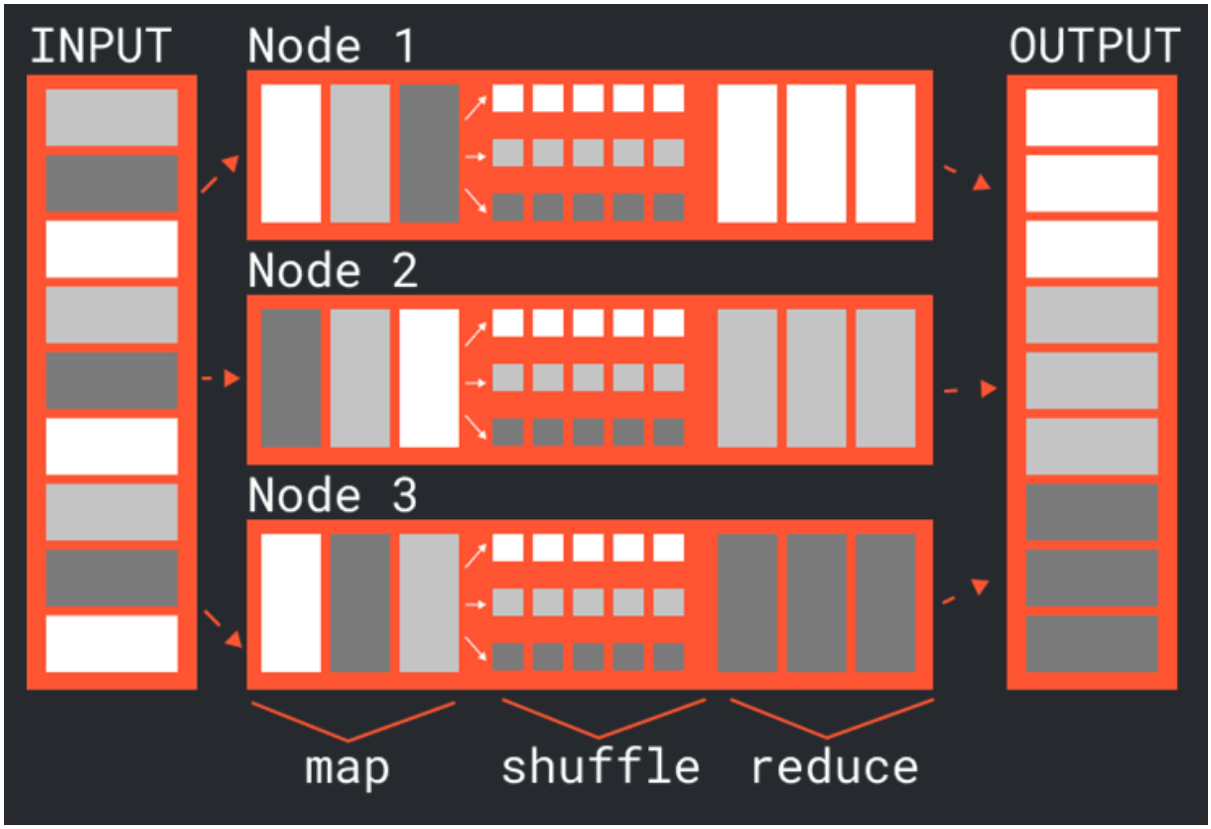


Модель вычислений MapReduce состоит из стадий:

1. Map: выполняемая параллельно фаза отображения, в которой входные данные разделяются на конечное множество блоков для последующей обработки. Как правило, вычисления легковесны, выполняются параллельно и независимо. Например, нам нужно по какому-то условию отфильтровать данные.
2. Shuffle: операция передачи данных от Map к Reduce (перемешивание и рассылка).
3. Reduce: фаза свёртки, в которой вывод фазы отображения агрегируется для получения конечного результата. На каждый Reducer попадут значения с одним ключом (будут сгруппированы). Например, операции группировки, сортировки, нахождения агрегированных значений.

Property	Map	Reduce
На входе	ключ и значение	ключ и массив значений
На выходе	ключ и значение	ключ и значение

> Как происходит обработка данных MapReduce



1. Данные на входе (INPUT) разбиваются на части по определённому правилу, которое зависит от формата самих данных.

2. Каждая часть данных обрабатывается независимо на разных серверах. Внутри каждого сервера все записи с одним ключом собираются и доставляются до Reducer-а с помощью операции Shuffle, которая является ресурсоёмкой и самой сложной, т.к. нужно сгруппировать все данные от Mapper-а, отсортировать на Mapper-е, доставить до Reducer-а и отсортировать внутри Reducer-а.

3. Внутри каждого Mapper-а есть циклический буфер в памяти – условно-бесконечный файл, который получает на вход результат Map оператора и записывает в виде ключ-значение. От каждого ключа считаем hash – целочисленное число, делим целочисленно этот hash на количество Reducer-ов, и в зависимости от остатка от деления определяем к какому Reducer-у нужно отправить данные.

Когда циклического буфера не хватает, то происходит Spill – сброс данных на диск. Программа, выполняющая Map операцию, записывает данные до заполнения буфера, затем содержимое буфера скидывается на локальные диски серверов (для каждого Reducer-а создаётся своя папка).

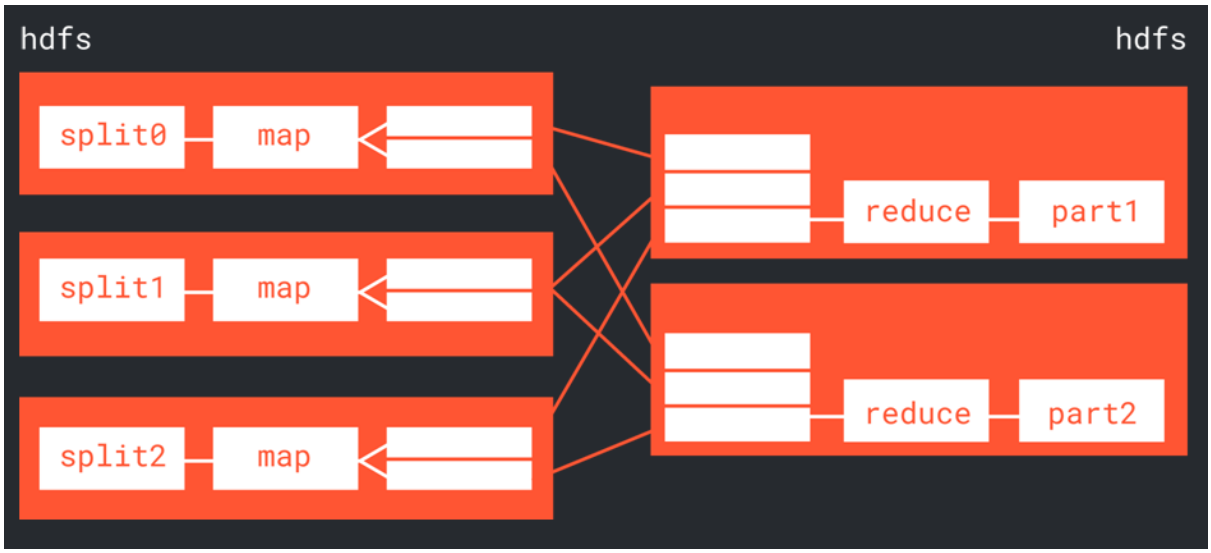
4. Теперь для каждого Reducer-а есть некоторое количество файлов, и, чтобы снизить нагрузку на сеть, выполняются 2 операции:

- сортировка – производится Mapper-ом, используется алгоритм сортировки слиянием, собирается порция данных для каждого Reducer-а,
- пересылка – Reducer сам забирает подготовленные данные с нужным количеством потоков.

5. Reducer забрал сортированные каждым Mapper-ом файлы, но теперь их нужно соединить и произвести глобальную сортировку. Reducer сортирует и объединяет все эти файлы сортировкой слиянием и получает файл с ключами с одним остатком от деления.

6. Reducer выполняет необходимую агрегирующую операцию и формирует OUTPUT выход (ключ и значение).

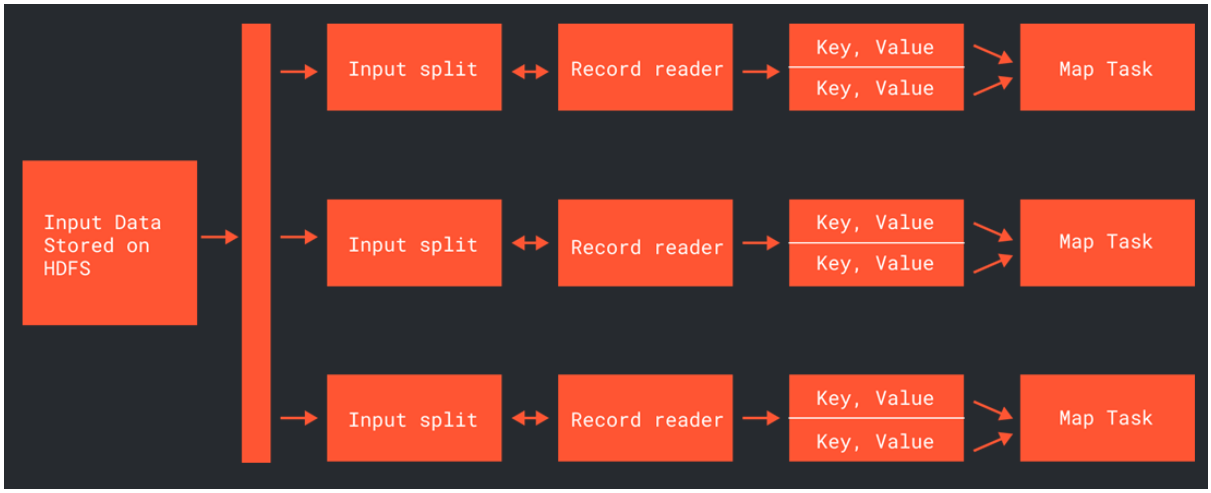
> Реализация MapReduce в Hadoop



Составляющие части процесса:

1. Split – небольшая порция данных, которая идёт на вход каждого Mapper-а. Split-ы формируются в зависимости от формата данных, какое используется сжатие, какой входной формат мы указали в MapReduce задании (Input Format).
2. Mapper производит свои вычисления.
3. Mapper готовит для каждого Reducer-а свою порцию данных.

4. Reducer выкачивает из каждого Mapper-а свою порцию данных.
5. Reducer производит свою операцию и готовит выходные данные (Output Format).



Более подробная схема обработки данных до Mapper

1. Входные данные лежат на HDFS.
2. Данные разбиваются на Split-ы и попадают в Mapper.
3. Mapper в зависимости от Input Format с помощью Record Reader (класс, который занимается преобразованием данных в зависимости от формата данных) правильно преобразует в записи по виду ключ-значение.
4. Данные в виде ключа и значения попадает в Map Task, где производится Map операция.
5. На выходе из Mapper-а получаем данные в виде ключа и значения и передаём их дальше.

> Составляющие MapReduce каркаса Hadoop

- Input File
- Input Format
- Input Split
- Record Reader

- Mapper (key-value)
- Reducer (key-value)
- Output (key-value)

> Глоссарий

JAR-файл – это Java-архив, который представляет собой ZIP-архив содержащий часть программы на языке Java.

Hadoop Streaming (потоковая передача Hadoop) – это утилита, поставляемая с дистрибутивом Hadoop, которая может использоваться для запуска программ для анализа больших данных (могут использоваться различные языки: Java, Scala, Unix, Perl, Python, Bash, Ruby и др.).

Map task – задача отображения.

Reduce task – задача свёртки.

Container (контейнер) – Java-процесс, с выделенным количеством ресурсов (память, процессор) на заданном узле.

MapReduce – фреймворк для вычисления распределенных задач на узлах (node) кластера и модель распределённых вычислений от компании Google, используемая в технологиях Big Data для параллельных вычислений над очень большими (до нескольких петабайт) наборами данных в компьютерных кластерах.

Spark – фреймворк для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop.

Tez – фреймворк, работающий поверх Hadoop YARN и предоставляющий возможность использовать сложный направленный ациклический граф (DAG) задач для обработки данных.

FIFO (first in first out – «первым пришёл – первым ушёл») – способ организации и манипулирования данными относительно времени и приоритетов. Тот, кто приходит первым, тот и обслуживается первым, пришедший следующим ждёт, пока обслуживание первого не будет закончено, и так далее.

Сортировка слиянием (merge sort) – алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например —

потоки) в определённом порядке. Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

Record Reader – класс-итератор, реализующий интерфейс Input Split, определяет количество данных, но не определяет как получить доступ к ним. Класс, реализующий интерфейс Record Reader, загружает данные и преобразует их в пары ключ-значение, подходящие для процесса отображения (Mapper). Экземпляр класса, реализующего интерфейс Record Reader, создаётся на основе входного формата.

Input Format – класс, который проверяет, что входные данные существуют, а также проверяет входную конфигурацию, создаёт входные сплиты (Input Splits) на основании блоков данных, создаёт реализацию Record Reader для получения пар ключ/значение из входных сплитов. Эти пары ключ/значение одна за одной будут отправлены отображению (Mapper).

Output Format – класс, который занимается тем, каким образом правильно нужно сохранить результат вычисления задания.
