# 湖中建仓 湖仓一体

# Hadoop

Hadooop 核心内容：HDFS+Yarn+Mapreduce

## 安装

Hadoop 的安装分为单机方式、伪分布式方式和完全分布式方式。（MDR采用的是完全分布式安装）

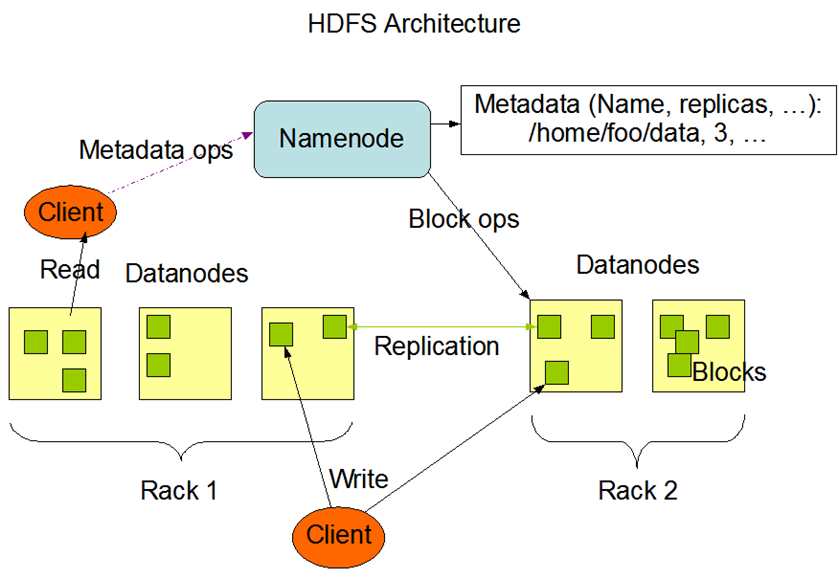
单机模式：主要用于开发测试

伪分布式：伪分布模式 Hadoop 守护进程运行在本地机器上，模拟一个小规模的

的集群。可以使用 HDFS 和 MapReduce

完全分布式：Hadoop 守护进程运行在一个集群上

## HDFS架构图

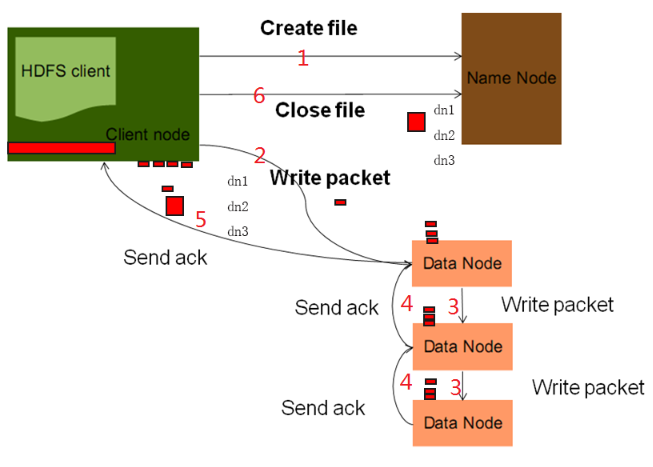


HDFS为了保证数据存储的可靠性和读取性能，对数据进行切块后进行复制并存储在集群的多个节点中。

NameNode: 存储元数据信息 元数据保存在内存/磁盘中 保存文件、block、datanode之间的映射关系

DataNode: 存储block内容 存储在磁盘中 维护了block id到的映射关系文件

## HDFS写流程



## HDFS读流程

计算机生成了可选文字:
HDFS client 
Client nod 
Open file 
Get block 
location 
Close file 
k 
ad block 
ame N Od 
Data Node 
Data Node 
Data Node 

## HDFS删除流程

先在NameNode上执行节点名字的删除。当NameNode执行delete方法时，它只标记操作涉及的需要被删除的数据块，而不会主动联系这些数据块所在的DataNode节点。当保存着这些数据块的DataNode节点向NameNode节点发送心跳时，在心跳应答里，NameNode节点会向DataNode发出指令，从而把数据删除掉。所以在执行完delete方法后的一段时间内，数据块才能被真正的删除掉。

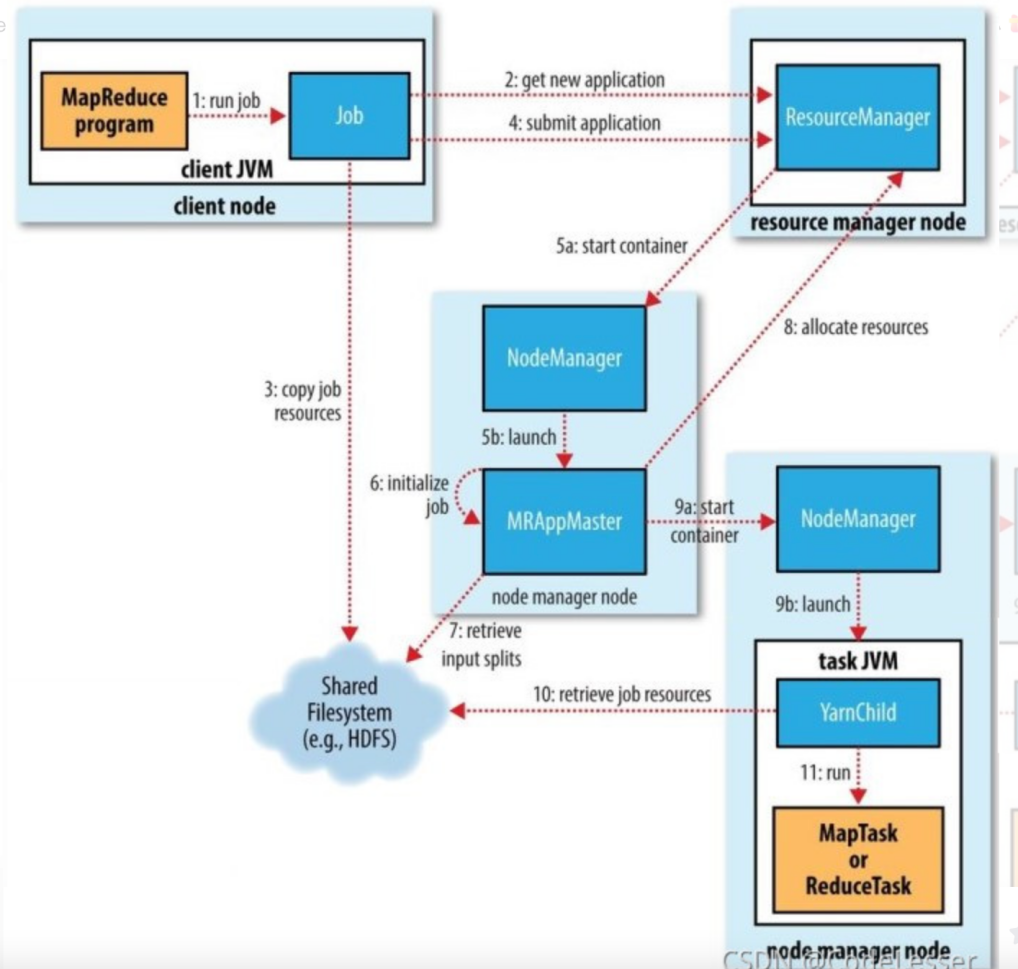
## Map Reduce

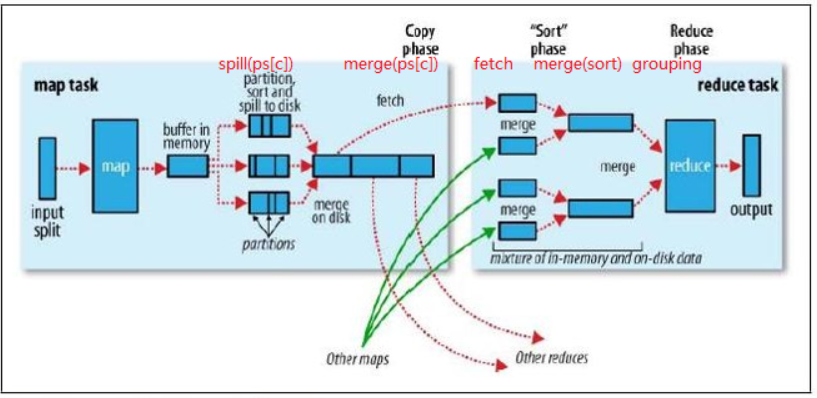
MapReduce是一个分布式计算框架（编程模型），最初由由谷歌的工程师开发，基于GFS的分布式计算框架, 主要用于搜索领域，解决海量数据的计算问题。后来Cutting根据《Google Mapreduce》,设计了基于HDFS的MapReduce分布式计算框架。

MR框架对于程序员的最大意义在于，不需要掌握分布式计算编程，不需要考虑分布式编程里可能存在的种种难题，比如任务调度和分配、文件逻辑切块、位置追溯等工作。这样，软件工程师能够把大部分精力放在核心业务层面上，大大简化了分布式程序的开发和调试周期。

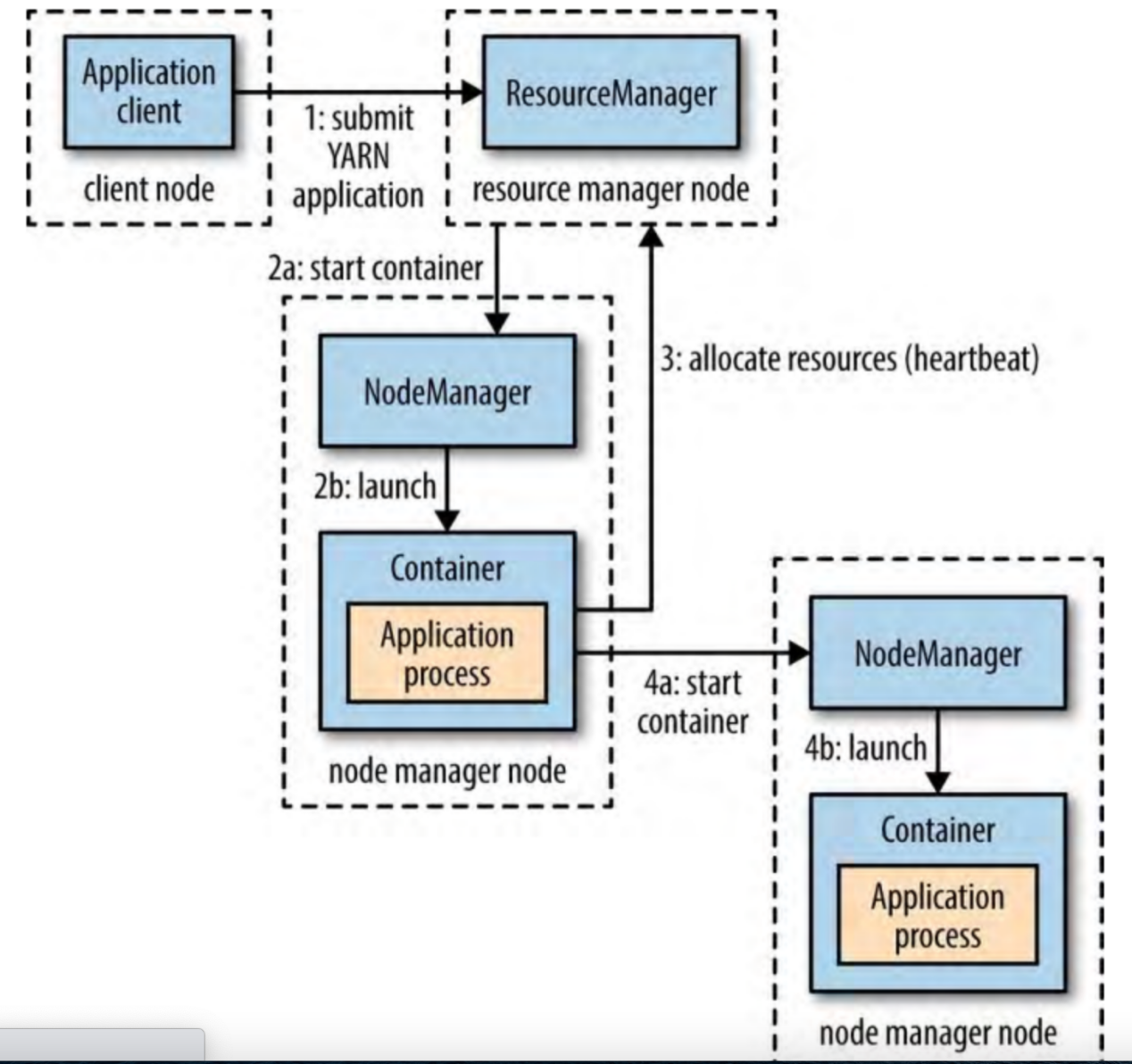
MR由两个阶段组成：Mapper和Reducer，用户只需要实现map()和reduce()两个函数，即可实现分布式计算，非常简单。这两个函数的形参是key、value对，表示函数的输入信息。

MR执行流程





## Yarn 运行机制



## Hadoop HA 架构

计算机生成了可选文字:
Heartbeat 
FailoverController 
Active 
Monitor Heal 
ofNN. OS, HW 
NN 
Active 
red NN state wi 
single writer 
(fenced) 
ZK 
Heartbe 
FailoverController 
Standbv 
onitor Health 
OfNN. OS, HW 
NN 
Standby 
lock Reports to Active & Standby 
DN fencing: Update cmds 什 om one 

## Hadoop2和Hadoop3有什么区别

License

Hadoop 2.x - Apache 2.0，开源

Hadoop 3.x - Apache 2.0，开源

支持的最低Java版本

Hadoop 2.x - java的最低支持版本是java 7

Hadoop 3.x - java的最低支持版本是java 8

容错

Hadoop 2.x - 可以通过复制（浪费空间）来处理容错。

Hadoop 3.x - 可以通过Erasure编码处理容错。

数据平衡

Hadoop 2.x - 对于数据平衡使用HDFS平衡器。

Hadoop 3.x - 对于数据平衡使用Intra-data节点平衡器，该平衡器通过HDFS磁盘平衡器CLI调用。

存储Scheme

Hadoop 2.x - 使用3X副本Scheme

Hadoop 3.x - 支持HDFS中的擦除编码。

存储开销

Hadoop 2.x - HDFS在存储空间中有200％的开销。

Hadoop 3.x - 存储开销仅为50％。

存储开销示例

Hadoop 2.x - 如果有6个块，那么由于副本方案（Scheme），将有18个块占用空间。

Hadoop 3.x - 如果有6个块，那么空间9个块，中6块空间，3块用于奇偶校验。

YARN时间线服务

Hadoop 2.x - 使用具有可伸缩性问题的旧时间轴服务。

Hadoop 3.x - 改进时间线服务v2并提高时间线服务的可扩展性和可靠性。

默认端口范围

Hadoop 2.x - 在Hadoop 2.0中，一些默认端口是Linux临时端口范围。所以在启动时，他们将无法绑定。

Hadoop 3.x - 但是在Hadoop 3.0中，这些端口已经移出了短暂的范围。

工具

Hadoop 2.x - 使用Hive，pig，Tez，Hama，Giraph和其他Hadoop工具。

Hadoop 3.x - 可以使用Hive，pig，Tez，Hama，Giraph和其他Hadoop工具。

兼容的文件系统

Hadoop 2.x - HDFS（默认FS），FTP文件系统：它将所有数据存储在可远程访问的FTP服务器上。 Amazon S3（简单存储服务）文件系统Windows Azure存储Blob（WASB）文件系统。

Hadoop 3.x - 它支持所有前面以及Microsoft Azure Data Lake文件系统。

Datanode资源

Hadoop 2.x - Datanode资源不专用于MapReduce，我们可以将它用于其他应用程序。

Hadoop 3.x - 此处数据节点资源也可用于其他应用程序。

MR API兼容性

Hadoop 2.x - 与Hadoop 1.x程序兼容的MR API，可在Hadoop 2.X上执行

Hadoop 3.x - 此处，MR API与运行Hadoop 1.x程序兼容，以便在Hadoop 3.X上执行

支持Microsoft Windows

Hadoop 2.x - 它可以部署在Windows上。

Hadoop 3.x - 它也支持Windows。

插槽/容器

Hadoop 2.x - Hadoop 1适用于插槽的概念，但Hadoop 2.X适用于容器的概念。通过容器，我们可以运行通用任务。

Hadoop 3.x - 它也适用于容器的概念。

单点故障

Hadoop 2.x - 具有SPOF的功能，因此只要Namenode失败，它就会自动恢复。

Hadoop 3.x - 具有SPOF的功能，因此只要Namenode失败，它就会自动恢复，无需人工干预就可以克服它。

HDFS联盟

Hadoop 2.x - 在Hadoop 1.0中，只有一个NameNode来管理所有Namespace，但在Hadoop 2.0中，多个NameNode用于多个Namespace。

Hadoop 3.x - Hadoop 3.x还有多个名称空间用于多个名称空间。

可扩展性

Hadoop 2.x - 我们可以扩展到每个群集10,000个节点。

Hadoop 3.x - 更好的可扩展性。 我们可以为每个群集扩展超过10,000个节点。

更快地访问数据

Hadoop 2.x - 由于数据节点缓存，我们可以快速访问数据。

Hadoop 3.x - 这里也通过Datanode缓存我们可以快速访问数据。

HDFS快照

Hadoop 2.x - Hadoop 2增加了对快照的支持。 它为用户错误提供灾难恢复和保护。

Hadoop 3.x - Hadoop 2也支持快照功能。

平台

Hadoop 2.x - 可以作为各种数据分析的平台，可以运行事件处理，流媒体和实时操作。

Hadoop 3.x - 这里也可以在YARN的顶部运行事件处理，流媒体和实时操作。

群集资源管理

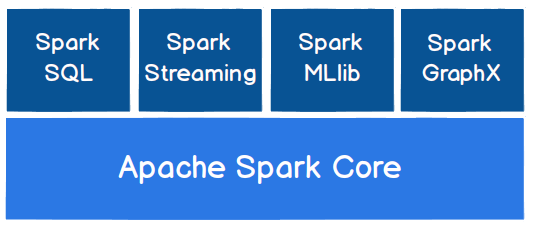
Hadoop 2.x - 对于群集资源管理，它使用YARN。 它提高了可扩展性，高可用性，多租户。

Hadoop 3.x - 对于集群，资源管理使用具有所有功能的YARN。

# Spark

Spark是一种基于内存的快速、通用、可扩展的大数据分析计算引擎。

## Spark 核心模块



Spark Core

Spark Core中提供了Spark最基础与最核心的功能，Spark其他的功能如：Spark SQL，Spark Streaming，GraphX, MLlib都是在Spark Core的基础上进行扩展的

Spark SQL

Spark SQL是Spark用来操作结构化数据的组件。通过Spark SQL，用户可以使用SQL或者Apache Hive版本的SQL方言（HQL）来查询数据。

Spark Streaming

Spark Streaming是Spark平台上针对实时数据进行流式计算的组件，提供了丰富的处理数据流的API。

Spark MLlib

MLlib是Spark提供的一个机器学习算法库。MLlib不仅提供了模型评估、数据导入等额外的功能，还提供了一些更底层的机器学习原语。

Spark GraphX

GraphX是Spark面向图计算提供的框架与算法库。

## Spark 运行模式

Local模式

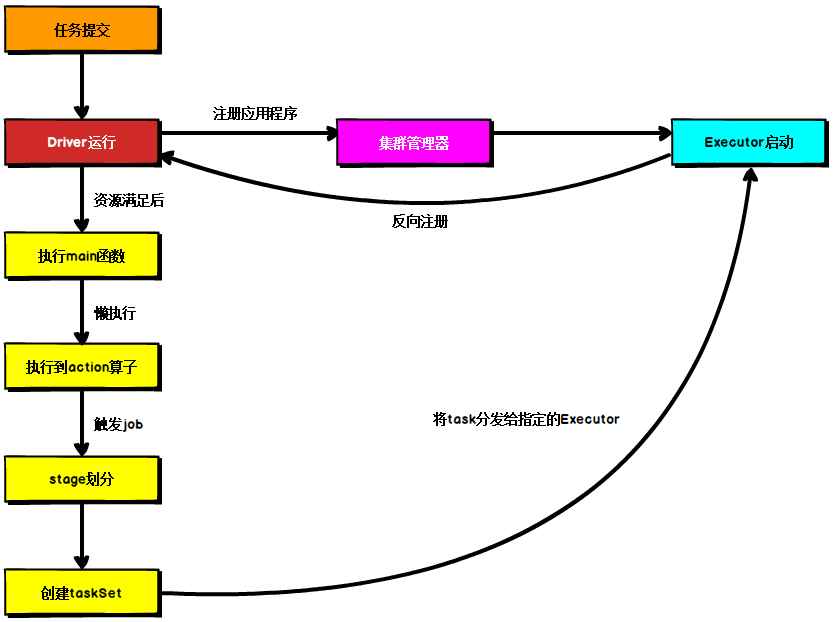
Standalone模式

Yarn 模式

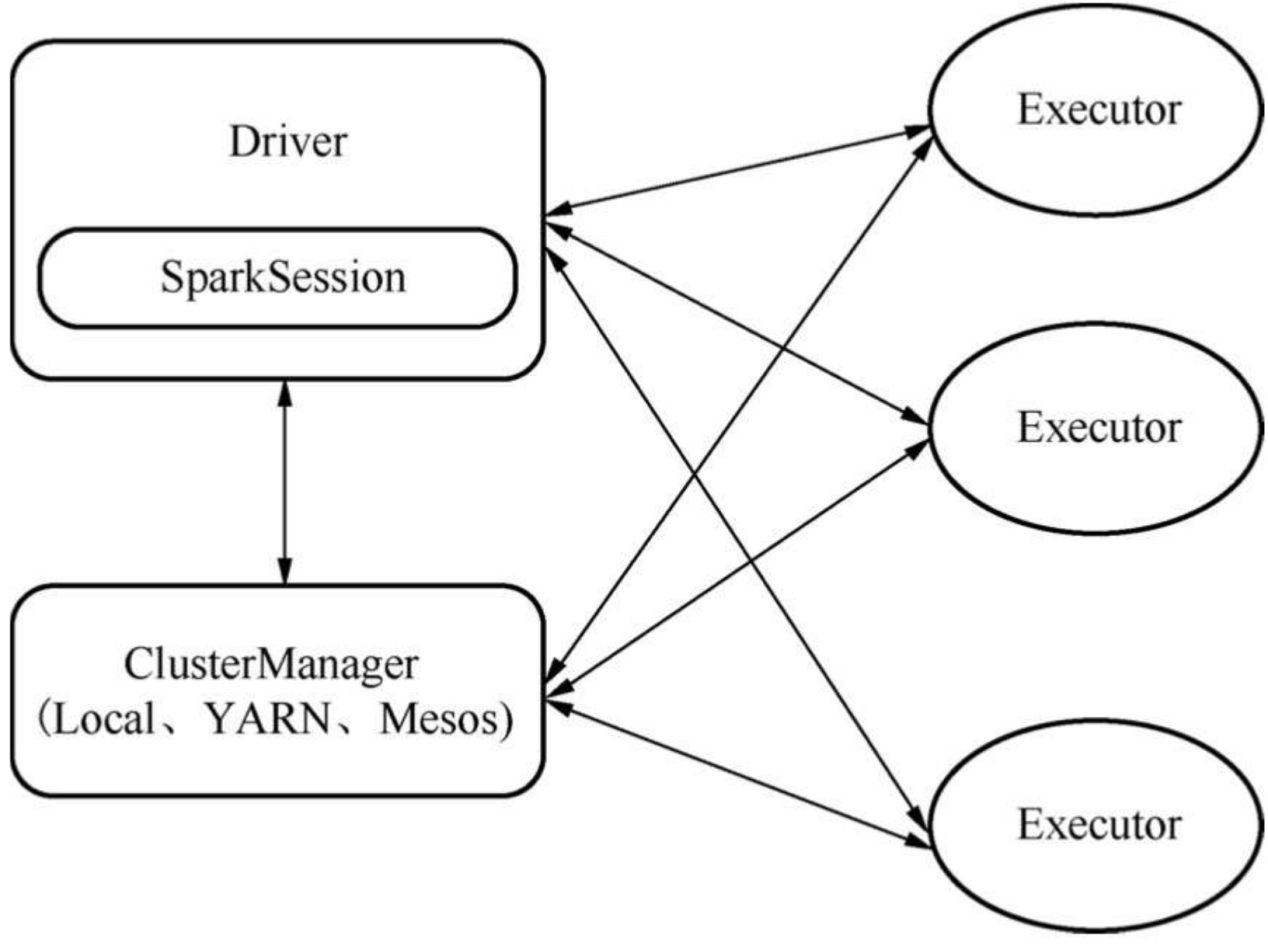
独立部署（Standalone）模式由Spark自身提供计算资源，无需其他框架提供资源。这种方式降低了和其他第三方资源框架的耦合性，独立性非常强。但是你也要记住，Spark主要是计算框架，而不是资源调度框架，所以本身提供的资源调度并不是它的强项，所以还是和其他专业的资源调度框架集成会更靠谱一些。

## Spark 运行架构

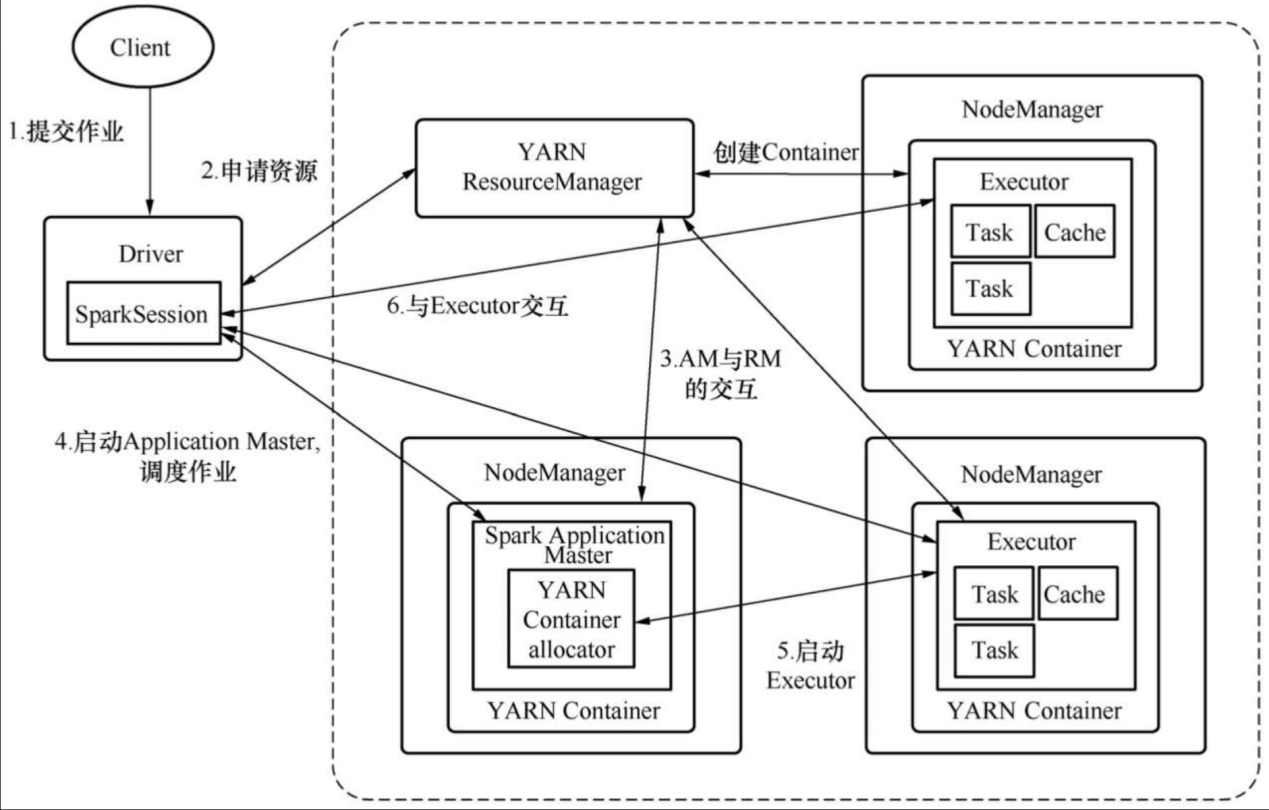
任务提交



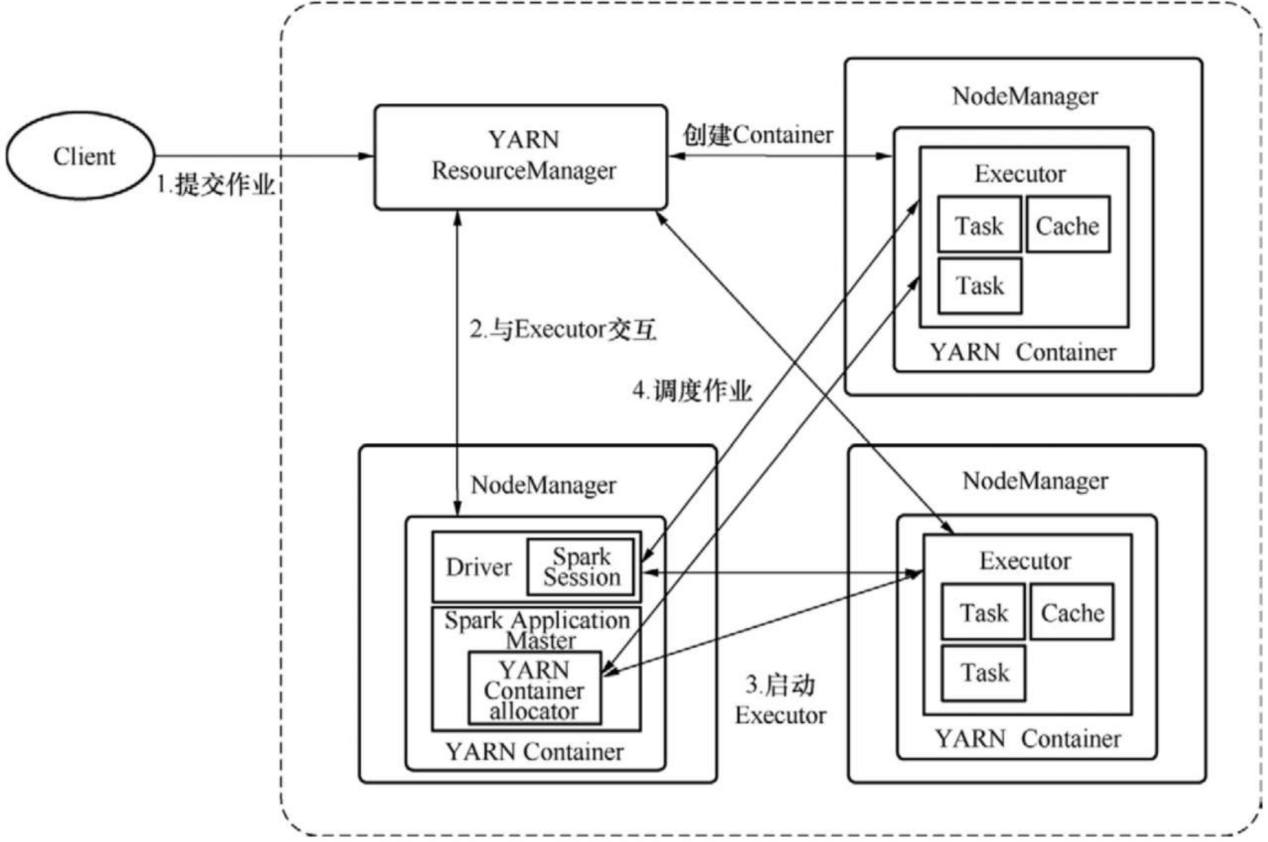
任务执行



Spark on yarn (yarn-client)



Spark on yarn (yarn-cluster)



# Hudi

插入更新

增量拉取

## 时间轴

在它的核心，Hudi维护一条包含在不同的即时时间所有对数据集操作的时间轴，从而提供，从不同时间点出发得到不同的视图下的数据集。

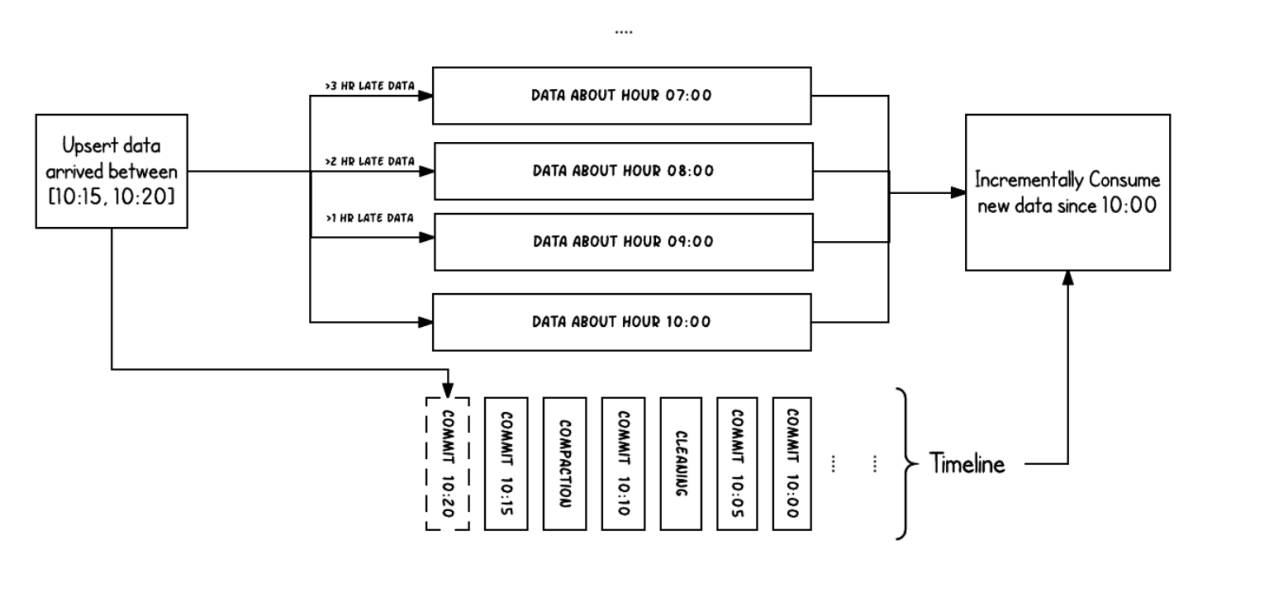
Hudi即时包含以下组件

操作类型 : 对数据集执行的操作类型

即时时间 : 即时时间通常是一个时间戳(例如：20190117010349)，该时间戳按操作开始时间的顺序单调增加。

状态 : 即时的状态

Hudi保证在时间轴上执行的操作的原子性和基于即时时间的时间轴一致性。



## 存储类型

写时复制：

仅使用列文件格式（例如parquet）存储数据。通过在写入过程中执行同步合并以更新版本并重写文件。

读时合并：

使用列式（例如parquet）+ 基于行（例如avro）的文件格式组合来存储数据。 更新记录到增量文件中，然后进行同步或异步压缩以生成列文件的新版本。



## 视图

Hudi支持以下存储数据的视图

**读优化视图** : 在此视图上的查询将查看给定提交或压缩操作中数据集的最新快照。 该视图仅将最新文件切片中的基本/列文件暴露给查询，并保证与非Hudi列式数据集相比，具有相同的列式查询性能。

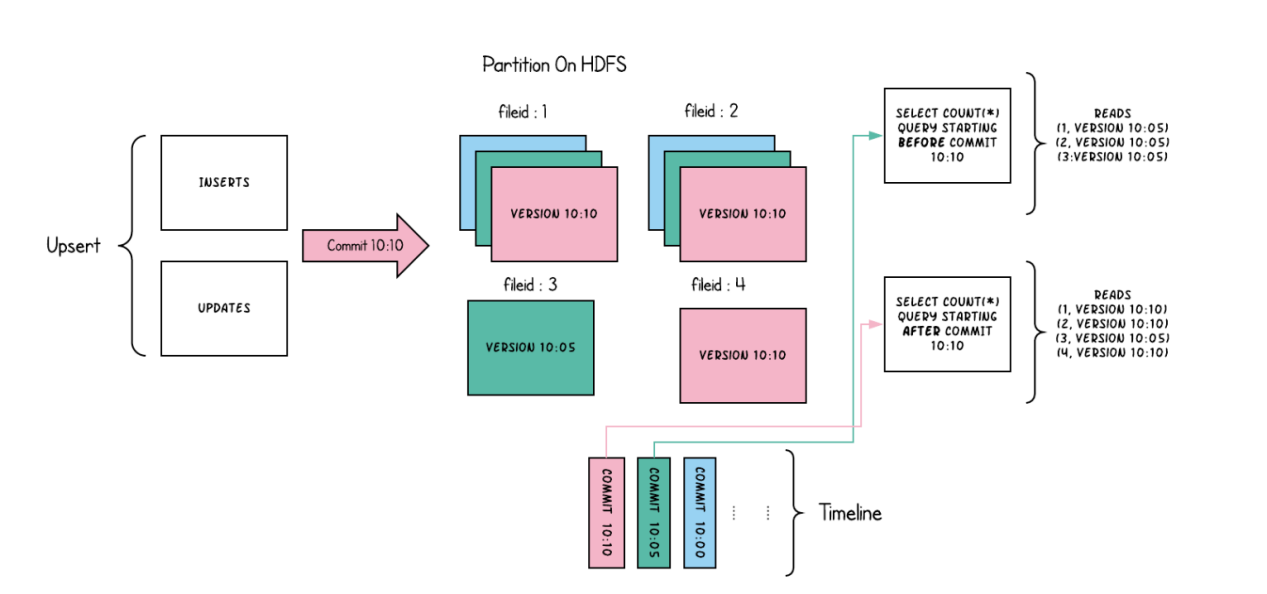
**增量视图** : 对该视图的查询只能看到从某个提交/压缩后写入数据集的新数据。该视图有效地提供了更改流，来支持增量数据管道。

**实时视图** : 在此视图上的查询将查看某个增量提交操作中数据集的最新快照。该视图通过动态合并最新的基本文件(例如parquet)和增量文件(例如avro)来提供近实时数据集（几分钟的延迟）。



## 写时复制存储

**写时复制存储中的文件片仅包含基本/列文件，并且每次提交都会生成新版本的基本文件。 换句话说，我们压缩每个提交，从而所有的数据都是以列数据的形式储存。在这种情况下，写入数据非常昂贵（我们需要重写整个列数据文件，即使只有一个字节的新数据被提交），而读取数据的成本则没有增加。 这种视图有利于读取繁重的分析工作。**



随着数据的写入，对现有文件组的更新将为该文件组生成一个带有提交即时时间标记的新切片，而插入分配一个新文件组并写入该文件组的第一个切片。 这些文件切片及其提交即时时间在上面用颜色编码。针对这样的数据集运行SQL查询（例如：select count(\*)统计该分区中的记录数目），首先检查时间轴上的最新提交并过滤每个文件组中除最新文件片以外的所有文件片。 如您所见，旧查询不会看到以粉红色标记的当前进行中的提交的文件，但是在该提交后的新查询会获取新数据。因此，查询不受任何写入失败/部分写入的影响，仅运行在已提交数据上。

写时复制存储的目的是从根本上改善当前管理数据集的方式，通过以下方法来实现:

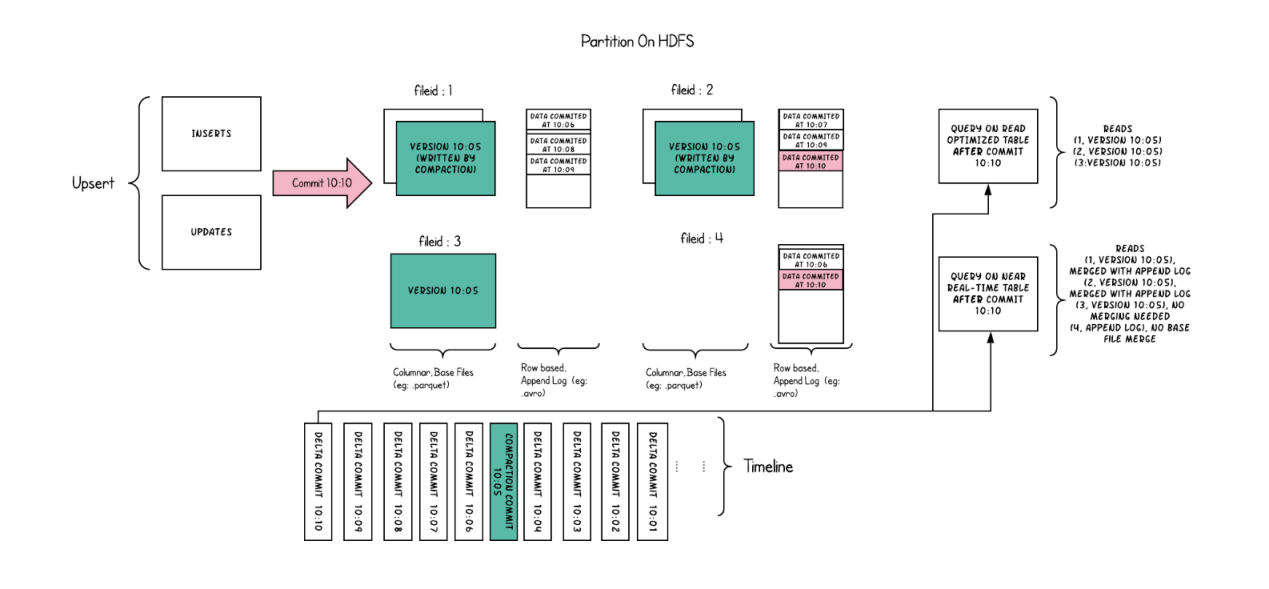
1 优先支持在文件级原子更新数据，而无需重写整个表/分区，

2 能够只读取更新的部分，而不是进行低效的扫描或搜索，

3 严格控制文件大小来保持出色的查询性能（小的文件会严重损害查询性能）。

## 读时合并存储

**读时合并存储是写时复制的升级版，从某种意义上说，它仍然可以通过读优化表提供数据集的读取优化视图（写时复制的功能）。 此外，它将每个文件组的更新插入存储到基于行的增量日志中，通过文件id，将增量日志和最新版本的基本文件进行合并，从而提供近实时的数据查询。因此，此存储类型智能地平衡了读和写的成本，以提供近乎实时的查询。 这里最重要的一点是压缩器，它现在可以仔细挑选需要压缩到其列式基础文件中的增量日志（根据增量日志的文件大小），以保持查询性能（较大的增量日志将会提升近实时的查询时间，并同时需要更长的合并时间）。**



现在，我们每1分钟左右就有一次提交，这是其他存储类型无法做到的。

现在，在每个文件id组中，都有一个增量日志，其中包含对基础列文件中记录的更新。 在示例中，增量日志包含10:05至10:10的所有数据。与以前一样，基本列式文件仍使用提交进行版本控制。因此，如果只看一眼基本文件，那么存储布局看起来就像是写时复制表的副本。

定期压缩过程会从增量日志中合并这些更改，并生成基础文件的新版本，就像示例中10:05发生的情况一样。

有两种查询同一存储的方式：读优化（RO）表和近实时（RT）表，具体取决于我们选择查询性能还是数据新鲜度。

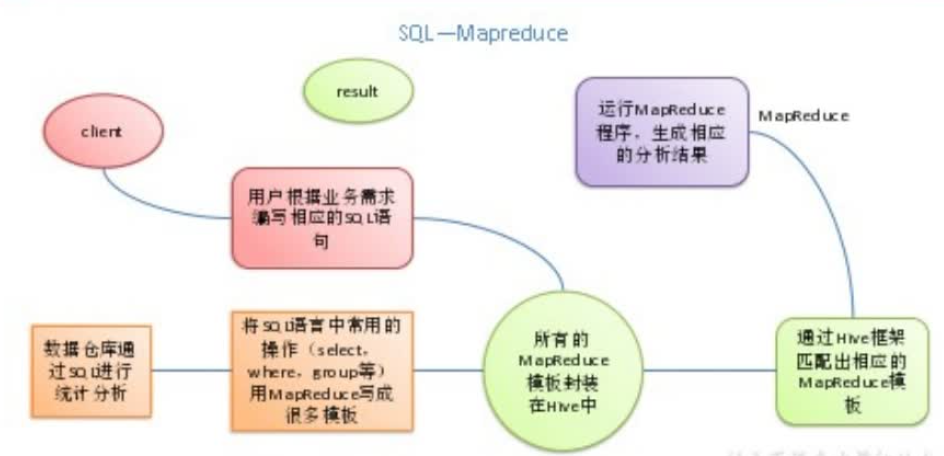
对于RO表来说，提交数据在何时可用于查询将有些许不同。 请注意，以10:10运行的（在RO表上的）此类查询将不会看到10:05之后的数据，而在RT表上的查询总会看到最新的数据。

何时触发压缩以及压缩什么是解决这些难题的关键。 通过实施压缩策略，在该策略中，与较旧的分区相比，我们会积极地压缩最新的分区，从而确保RO表能够以一致的方式看到几分钟内发布的数据。

**读时合并存储上的目的是直接在DFS上启用近实时处理，而不是将数据复制到专用系统，后者可能无法处理大数据量。** 该存储还有一些其他方面的好处，例如通过避免数据的同步合并来减少写放大，即批量数据中每1字节数据需要的写入数据量。

# Hive

Hive：由Facebook开源用于解决海量结构化日志的数据统计。Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张表，并提供类SQL查询功能。本质是：将HQL转化成MapReduce程序。



1 Hive处理的数据存储在HDFS

2 Hive分析数据底层的默认实现是MapReduce

3执行程序运行在Yarn上

优点

操作接口采用类SQL语法，提供快速开发的能力（简单、容易上手）。

避免了去写MapReduce，减少开发人员的学习成本。

Hive的执行延迟比较高，因此Hive常用于数据分析，对实时性要求不高的场合。

Hive优势在于处理大数据，对于处理小数据没有优势，因为Hive的执行延迟比较高。

Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。