Hadoop 学习理解

----adoop 基础知识

笔者：电杆

2018 年 10 月 10 日

目录

一、什么是大数据。

大数据的 4V 特征：

1. 数据量大，TB->PB
2. 数据类型繁多，结构化、非结构化文本、日志、视频、图片、地理位置等；
3. 商业价值高，但是这种价值需要在海量数据之上，通过数据分析与机器学习更快速的挖掘出来；
4. 处理时效性高，海量数据的处理需求不再局限在离线计算当中。

现如今，正式为了应对大数据的这几个特点，开源的大数据框架越来越多，越来越强，先列举一些常见的：

- **文件存储**：Hadoop HDFS、Tachyon、KFS
- **离线计算**：Hadoop MapReduce、Spark
- **流式、实时计算**：Storm、Spark Streaming、S4、HeronK-V、NOSQL
- **数据库**：HBase、Redis、MongoDB
- **资源管理**：YARN、Mesos
- **日志收集**：Flume、Scribe、Logstash、Kibana
- **消息系统**：Kafka、StormMQ、ZeroMQ、RabbitMQ
- **查询分析**：Hive、Impala、Pig、Presto、Phoenix、SparkSQL、Drill、Flink、Kylin、Druid
- **分布式协调服务**：Zookeeper
- **集群管理与监控**：Ambari、Ganglia、Nagios、Cloudera Manager
- **数据挖掘、机器学习**：Mahout、Spark MLLib
- **数据同步**：Sqoop
- **任务调度**：Oozie

……..

大数据"的处理方法是：采用多机器、多节点的处理大量数据方法，而采用这种新的处理方法，就需要有新的大数据系统来保证，系统需要处理多节点间的通讯协调、数据分隔等一系列问题。

二、hadoop 家族

Hadoop 也是由诸多的子项目构成的，下面是组成 Hadoop 的核心项目：

1. HDFS: Hadoop 分布式文件系统(Distributed File System) - HDFS (Hadoop Distributed File System)

2. MapReduce：并行计算框架，0.20 前使用 org.apache.hadoop.mapred 旧接口，0.20 版本开始引入 org.apache.hadoop.mapreduce 的新 API

3. HBase: 类似 Google BigTable 的分布式 NoSQL 列数据库。(HBase 和 Avro 已经于 2010 年 5 月成为顶级 Apache 项目)

4. Hive：数据仓库工具，由 Facebook 贡献。

5. Zookeeper：分布式锁设施，提供类似 Google Chubby 的功能，由 Facebook 贡献。

6. Avro：新的数据序列化格式与传输工具，将逐步取代 Hadoop 原有的 IPC 机制。

7. Pig: 大数据分析平台，为用户提供多种接口。

8. Ambari：Hadoop 管理工具，可以快捷的监控、部署、管理集群。

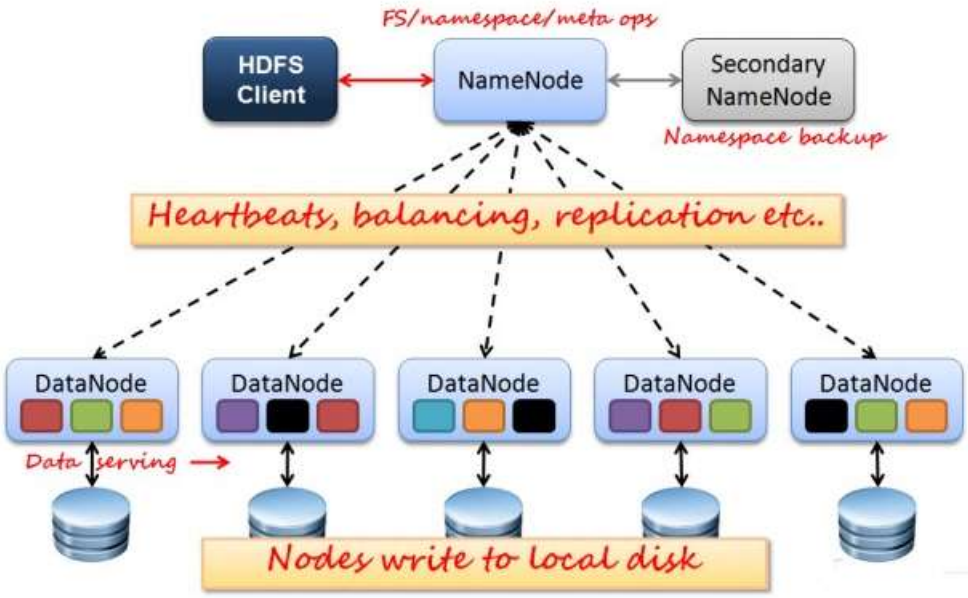
9. Sqoop：于在 HADOOP 与传统的数据库间进行数据的传递。

三、hadoop 详解

3.1 HDFS 详解

3.1.1 HDFS 是什么?

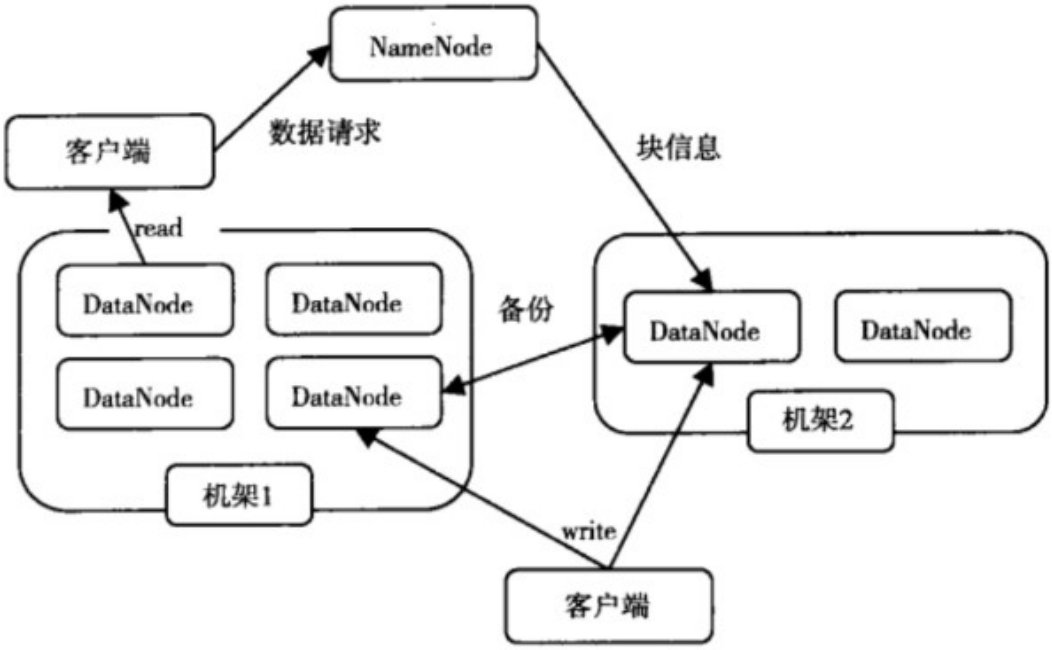
HDFS 分布式文件系统，是 Hadoop 体系中数据存储管理的基础。它是一个高度容错的系统，能检测和应对硬件故障，用于在低成本的通用硬件上运行。HDFS 简化了文件的一致性模型，通过流式数据访问（1 一次写入，多次读取。文件一旦写入不能修改，只能追加。2，它能保证数据的一致性。），提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。 ----分而存之



DFS 有两种 Node，一种是 NameNode，负责记录具体数据的元数据信息，另一种是 DataNode，真正的数据节点。这里的 NameNode 有两个，另一个主要作用是分担主 NameNode 的一部分工作负载。每一个文件的副本

存储在不同的节点上，可以通过配置，让 HDFS 感知机架，这样副本会存储在不通的机架上，即使整个机架坏掉，数据也是可以恢复的。不同副本之间的数据复制由 HDFS 负责。在 NameNode 和 DataNode 之间维持着心跳

NameNode 能够感知 DataNode 的状态，DataNode 变得不可用，会启用副本复制



HDFS 的体系结构图

3.1.2 HDFS 功能及特性

HDFS 功能节点

- NameNode（NN）是主节点，存储文件的元数据如文件名，文件目录结构，文件属性（生成时间,副本数,文件权限），以及每个文件的块列表和块所在 DataNode 等。
- DataNode（DN）在本地文件系统存储文件块数据，以及块数据的校验和。
- Secondary NameNode（SNN） 用来监控 HDFS 状态的辅助后台程序，每隔一段时间获 取 HDFS 元数据的快照。

HDFS 特性

- 低成本： 一般来说，HDFS 部署在商用硬件上，所以，在项目的拥有成本方面是非常经济的。
- 数据的种类和数量： 可以将任何类型的数据存储到 HDFS 中，无论是结构化的，非结构化的还是半结构化的。
- 可靠性和容错性： 当将数据存储在 HDFS 上时，它会将给定的数据内部分割为数据块，并以分布的方式将其存储在 Hadoop 集群中。关于哪个数据块位于哪个数据节点上的信息被记录在元数据中。NameNode 管理元数据， DataNode 负责存储数据。名称节点也复制数据，即维护数据的多个副本。数据的这种复制使得 HDFS 非常可靠和

容错。名称节点定期更新元数据并保持复制因子一致。

- 数据完整性： HDFS 不断检查存储的数据的完整性与其校验和。如果发现任何错误，它会向名称节点报告。然后，名称节点创建额外的新副本，因此删除损坏的副本。
- 高吞吐量：吞吐量是单位时间内完成的工作量。通过并行处理数据，大大减少了处理时间，从而实现了高吞吐量。
- 数据局部性： 数据局部性讨论的是将处理单元移动到数据而不是数据到处理单元。在 HDFS 中，我们将计算部分带到数据所在的数据节点。

3.1.3 HDFS 操作命令 FS Shell

HDFS 允许以文件和目录的形式组织管理用户数据，它提供了一个称为 FS shell 的命令行接口让用户与 HDFS 中的数据进行交互。该命令集的语法类似于用户已经熟悉的其它 shell(例如 bach、csh)。下面是一些例子：

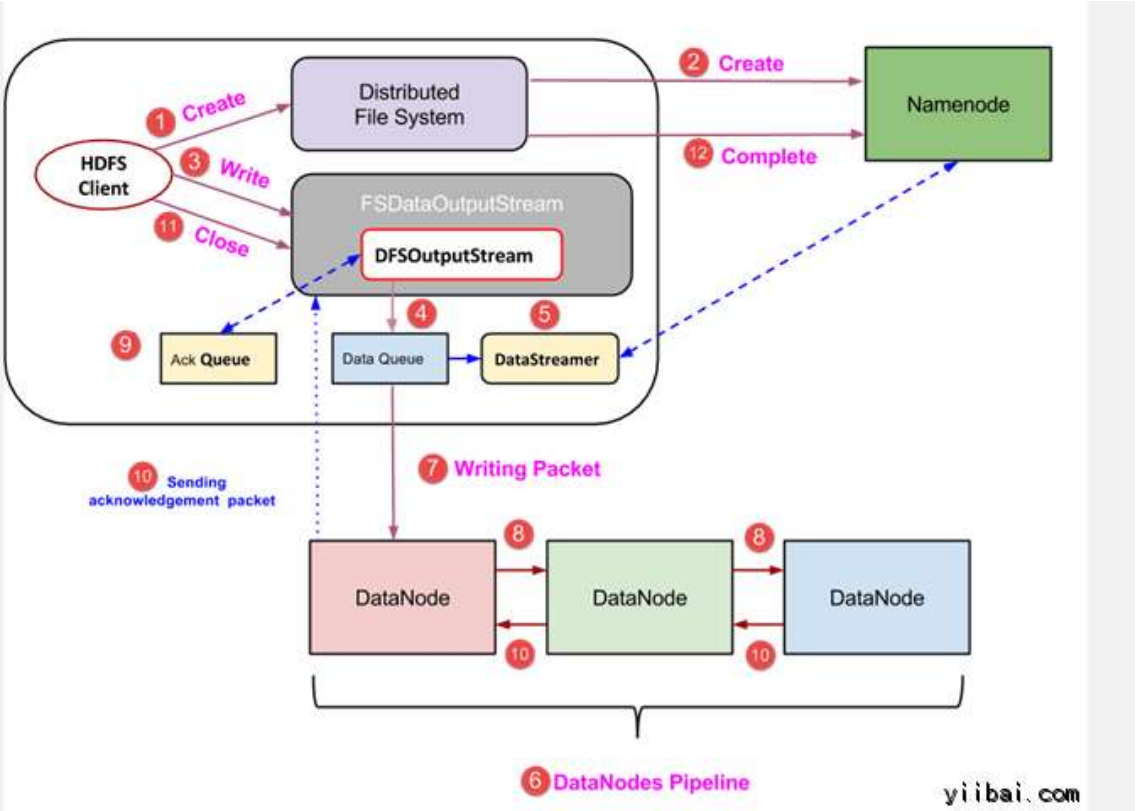
在根目录/下面创建一个 foodir 的子目录 bin/hadoop fs -mkdir /foodir

删除根目录/下面的 foodir 子目录： bin/hadoop fs -rm -R /foodir

查看文件/foodir/myfile.txt 的内容： bin/hadoop fs -cat /foodir/myfile.txt

HDFS 命令详解：https://www.cnblogs.com/zhuxiaojie/p/6391518.html

3.1.4 HDFS 写入数据流程



- 客户端通过调用 DistributedFileSystem 对象的 create() 方法创建一个新的文件，并开始写操作 - 在上面的图中的步骤 1

- DistributedFileSystem 对象使用 RPC 调用连接到 NameNode，并启动新的文件创建。但是，此文件创建操作不与文件任何块相关联。NameNode 的责任是验证文件(其正被创建的)不存在，并且客户端具有正确权限来创建新文件。如果文件已经存在，或者客户端不具有足够的权限来创建一个新的文件，则抛出 IOException 到客户端。否则操作成功，并且该文件新的记录是由 NameNode 创建。
- 一旦 NameNode 创建一条新的记录，返回 FSDataOutputStream 类型的一个对象到客户端。客户端使用它来写入数据到 HDFS。数据写入方法被调用(图中的步骤 3)。
- FSDataOutputStream 包含 DFSOutputStream 对象，它使用 DataNodes 和 NameNode 通信后查找。当客户机继续写入数据，DFSOutputStream 继续创建这个数据包。这些数据包连接排队到一个队列被称为 DataQueue
- 还有一个名为 DataStreamer 组件，用于消耗 DataQueue。DataStreamer 也要求 NameNode 分配新的块，拣选 DataNodes 用于复制。
- 现在，复制过程始于使用 DataNodes 创建一个管道。 在我们的例子中，选择了复制水平 3，因此有 3 个 DataNodes 管道。
- 所述 DataStreamer 注入包分成到第一个 DataNode 的管道中。
- 在每个 DataNode 的管道中存储数据包接收并同样转发在第二个 DataNode 的管道中。
- 另一个队列，“Ack Queue"是由 DFSOutputStream 保持存储，它们是 DataNodes 等待确认的数据包。
- 一旦确认在队列中的分组从所有 DataNodes 已接收在管道，它从 'Ack Queue' 删除。在任何 DataNode 发生故障时，从队列中的包重新用于操作。
- 在客户端的数据写入完成后，它会调用 close()方法(第 9 步图中)，调用 close()结果进入到清理缓存剩余数据包到管道之后等待确认。
- 一旦收到最终确认，NameNode 连接告诉它该文件的写操作完成。

3.2 MapReduce 详解

3.2.1 MapReduce 是什么?

MapReduce 是一种分而治之的思想，谷歌在论文中提出的这种思想成为了后来分布式任务处理的标准。Map 是映射，Reduce 则是归约，对于输入的数据来说，先需要分片，然后通过 Map 对数据进行处理，处理的结果是 k/v

这个 k/v 是针对每个 map 接收到的分片进行的操作，每一个 map 操作输出至少一个

key/value 对，根据 map 的 key 对 map 的输出先进行合并，每一个 key 对应一个 reduce，各个 reduce 做各自的处理 ----分而治之

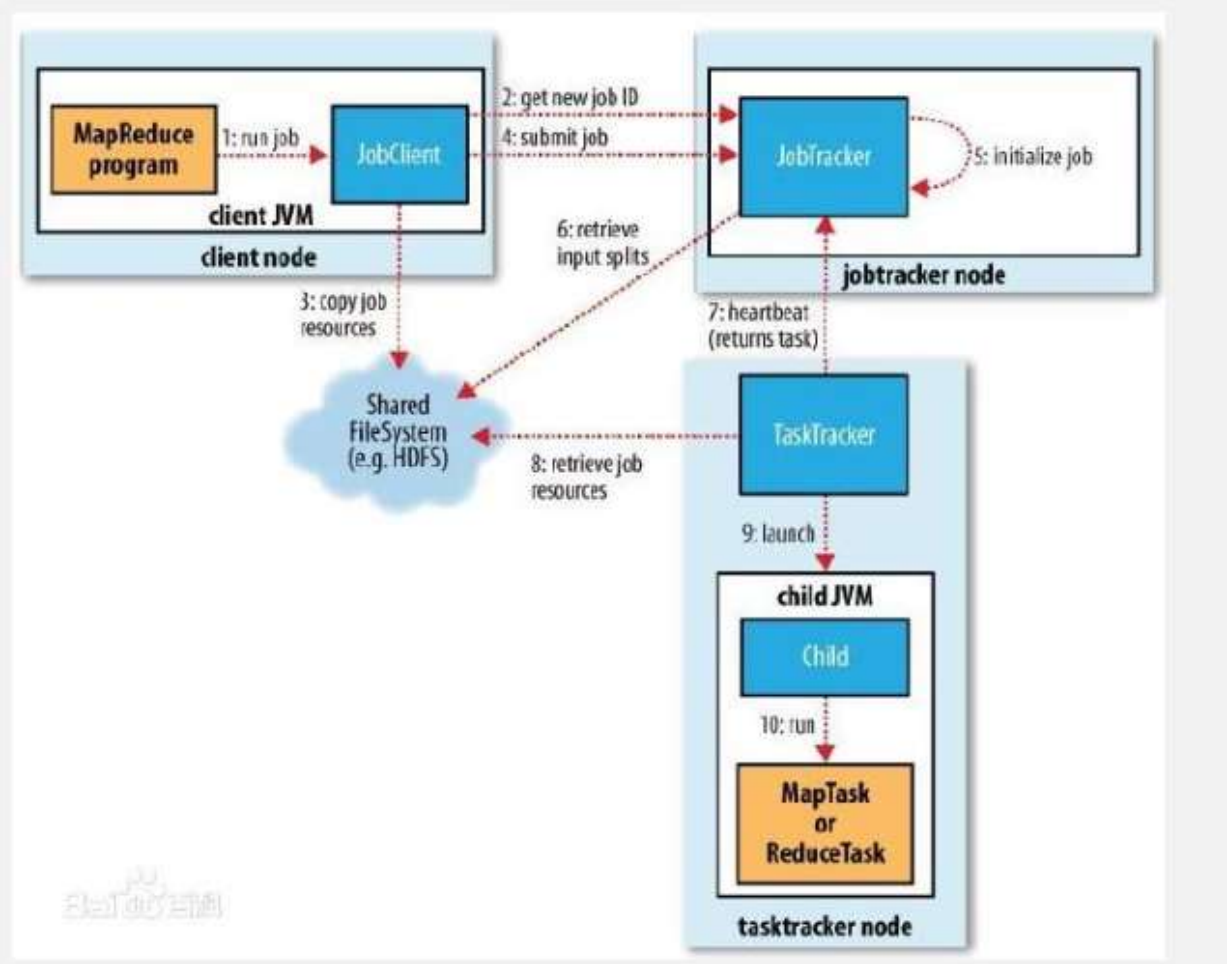
3.2.2 MapReduce 执行流程

MapReduce 程序执行流程：

(1) 开发人员编写好 MapReduce program，将程序打包运行。

(2) JobClient 向 JobTracker 申请可用 Job，JobTracker 返回 JobClient 一个可用 Job ID。

(3) JobClient 得到 Job ID 后，将运行 Job 所需要的资源拷贝到共享文件系统 HDFS 中。



(4) 资源准备完备后，JobClient 向 JobTracker 提交 Job。

(5) JobTracker 收到提交的 Job 后，初始化 Job。

(6) 初始化完成后，JobTracker 从 HDFS 中获取输入 splits(作业可以该启动多少 Mapper 任务)。

(7) 与此同时，TaskTracker 不断地向 JobTracker 汇报心跳信息，并且返回要执行的任务。

(8) TaskTracker 得到 JobTracker 分配(尽量满足数据本地化)的任务后，向 HDFS 获取 Job 资源(若数据是本地的，不需拷贝数据)。

(9) 获取资源后，TaskTracker 会开启 JVM 子进程运行任务。

注：

(3)中资源具体指什么？主要包含：

- 程序 jar 包、作业配置文件 xml
- 输入划分信息，决定作业该启动多少个 map 任务
- 本地文件，包含依赖的第三方 jar 包(-libjars)、依赖的归档文件(-archives)和普通文件(-files)，如果已经上传，则不需上传

3.3 Yarn 详解

3.3.1 Yarn 是什么?

Apache Hadoop YARN （Yet Another Resource Negotiator，另一种资源协调者）是一种新的 Hadoop 资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度。它将资源管理和处理组件分开，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。

为什么要使用 YARN?

1. 提出了 HDFS Federation，它让多个 NameNode 分管不同的目录进而实现访问隔离和横向扩展。对于运行中 NameNode 的单点故障，通过 NameNode 热备方案（NameNode HA）实现

2. YARN 通过将资源管理和应用程序管理两部分分剥离开，分别由 ResouceManager 和 ApplicationMaster 负责，其中，ResouceManager 专管资源管理和调度，而 ApplicationMaster 则负责与具体应用程序相关的任务切分、任务调度和容错等，每个应用程序对应一个 ApplicationMaster

3. YARN 具有向后兼容性，用户在 MRv1 上运行的作业，无需任何修改即可运行在 YARN 之上。

4. 对于资源的表示以内存为单位 (在目前版本的 Yarn 中，没有考虑 cpu 的占用)，比之前以剩余 slot 数目更合理。

5. 支持多个框架， YARN 不再是一个单纯的计算框架，而是一个框架管理器，用户可以将各种各样的计算框架移植到 YARN 之上，由 YARN 进行统一管理和资源分配。目前可以支持多种计算框架运行在 YARN 上面，比如 MapReduce、Storm、Spark、Flink 等

6. 框架升级更容易， 在 YARN 中，各种计算框架不再是作为一个服务部署到集群的各个节点上（比如 MapReduce 框架，不再需要部署 JobTracler、 TaskTracker 等服务），而是被封装成一个用户程序库（lib）存放在客户端，当需要对计算框架进行升级时，只需升级用户程序库即可。

结构：

ResourceManager：整个集群只有一个，负责集群资源的统一管理和调度处理客户端请求

- 启动/监控 ApplicationMaster
- 监控 NodeManager
- 资源分配与调度

NodeManager：整个集群有多个，负责单节点资源管理和使用

- 单个节点上的资源管理和任务调度
- 处理来自 ResourceManager 的命令
- 处理来自 ApplicationMaster 的命令

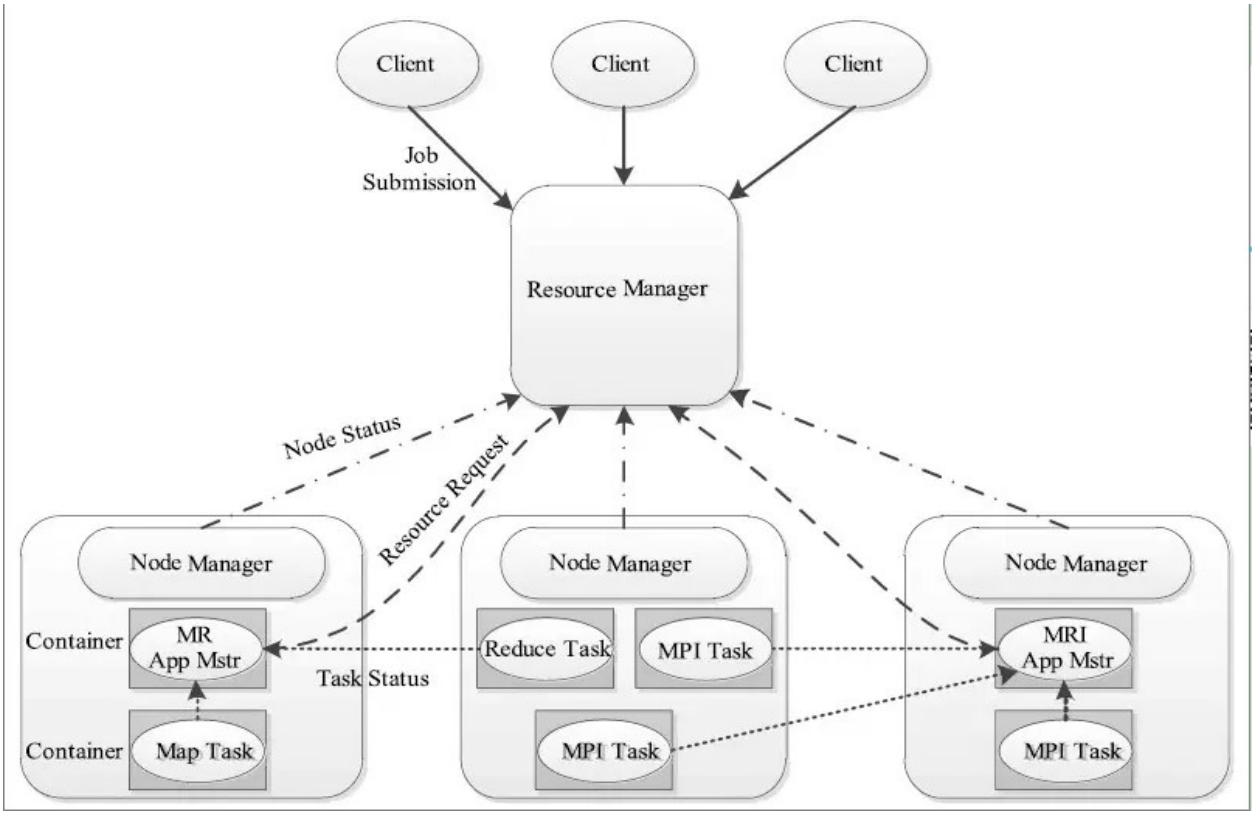
ApplicationMaster：每个应用有一个，负责应用程序的管理

- 数据切分
- 为应用程序申请资源，并进一步分配给内部任务
- 任务监控与容错

Container：1、对任务运行环境的抽象 2、描述一系列信息

- 任务运行资源（节点、内存、CPU）
- 任务启动命令
- 任务运行环境

物理架构：



四、添加新的节点到 hadoop 集群

首先，把新节点的 IP 或主机名 加入主节点（master）的 conf/slaves 文件。

然后登录新的从节点，执行以下命令：

```
$ cd path/to/hadoop
$ bin/hadoop-daemon.sh start datanode
$ bin/hadoop-daemon.sh start tasktracker
```

然后就可以在 namanode 机器上运行 balancer，执行负载均衡

```
$bin/hadoop   balancer
```

五 、正则表达式

分打印字符非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

字符	描述
\cx	匹配由 x 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。注意 Unicode 正则表达式会匹配全角空格符。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

特殊字符

所谓特殊字符，就是一些有特殊含义的字符，如上面说的 runoo*b 中的 *，简单的说就是表示任何字符串的意思。如果要查找字符串中的 * 符号，则需要对 * 进行转义，即在其前加一个 \: runo*ob 匹配 runo*ob。

许多元字符要求在试图匹配它们时特别对待。若要匹配这些特殊字符，必须首先使字符"转义"，即，将反斜杠字符\ 放在它们前面。下表列出了正则表达式中的特殊字符：

特别字符	描述
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。

( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \(和 \)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用 \. 。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
|	指明两项之间的一个选择。要匹配 |，请使用 \\|。

限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 * 或 + 或 ? 或 {n} 或 {n,} 或 {n,m} 共 6 种。

正则表达式的限定符有：

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。

+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 、"does" 中的 "does" 、 "doxy" 中的 "do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配 n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中 n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

参考

本文档参考许多网上博客，先后不分等级，帮助同等要重，致谢！

参考：https://www.cnblogs.com/zhuxiaojie/p/6391518.html

http://www.runoob.com/