

大数据基础平台实施及运维

一、大数据介绍

学习目标

- ☐ 能够了解为什么使用大数据技术
- ☐ 能够了解大数据指的是什么

1. 为什么使用大数据技术？

- 数据量越来越大
- 数据分析的实时性越来越强
- 数据结果的应用越来越广泛

结论：我们需要使用大数据技术

2. 大数据的定义

大数据是收集、整理、处理大容量数据集，并从中获得结果的技术总称。

二、大数据应用领域

学习目标

- ☐ 能够了解大数据应用在哪些领域

1. 广告

- 广告投放
- 广告策略

2. 电信

- 深度包检测
- 流失分析
- 网络质量

3. 金融

- 风险识别
- 预测

4. 能源生物

- 基因组分析
- 地质分析

5. 安全

- 入侵检测
- 图像识别
- 6. 社交游戏
 - 流失分析
 - 社交推荐
 - 使用分析
- 7. 电商零售
 - 推荐系统
 - 交易分析

三、大数据技术处理框架有哪些？

学习目标

- ☐ 能够了解大数据处理框架有哪些
- ☐ 能够了解每种大数据处理框架所对应的产品

1)什么是大数据处理框架？

处理框架和**处理引擎**负责对数据系统中的数据进行计算。

虽然“引擎”和“框架”之间的区别没有什么权威的定义，但大部分时候可以将前者定义为实际负责处理数据操作的组件，后者则可定义为承担类似作用的一系列组件。

2)大数据处理框架有哪些？

- 仅批处理框架

用于批量处理大数据集的处理框架，可对整个数据集进行操作。

例如：

Apache Hadoop，一种以**MapReduce**作为默认处理引擎批处理框架。

- 仅流处理框架

用于对随时进入系统的数据进行计算，是一种“无数据边界”的操作方式。

例如：

Apache Storm

Apache Samza

- 混合处理框架

一些大数据处理框架可同时**处理批处理**和**流处理**工作负载。

例如：

Apache Spark

Apache Flink

总结：

- 处理框架 一组组件
- 处理引擎 具体对数据操作的工具
- 框架分类
 - 仅批处理框架 apache hadoop MapReduce
 - 仅流处理框架 apache storm apache samza
 - 混合处理框架 apache spark apache flink

四、hadoop生态圈

学习目标

- ☐ 能够了解Hadoop历史
- ☐ 能够了解Hadoop项目定义
- ☐ 能够了解Hadoop核心项目
- ☐ 能够了解Hadoop相关项目

1) Hadoop历史

雏形开始于2002年的Apache的Nutch，Nutch是一个开源Java 实现的搜索引擎。它提供了我们运行自己的搜索引擎所需的全部工具。包括全文搜索和Web爬虫。

随后在2003年Google发表了一篇技术学术论文谷歌文件系统（GFS）。GFS也就是google File System，google公司为了存储海量搜索数据而设计的专用文件系统。

2004年Nutch创始人Doug Cutting基于Google的GFS论文实现了分布式文件存储系统名为NDFS。

2004年Google又发表了一篇技术学术论文MapReduce。MapReduce是一种编程模型，用于大规模数据集（大于1TB）的并行分析运算。

2005年Doug Cutting又基于MapReduce，在Nutch搜索引擎实现了该功能。

2006年，Yahoo雇用了Doug Cutting，Doug Cutting将NDFS和MapReduce升级命名为Hadoop，Yahoo开建了一个独立的团队给Goug Cutting专门研究发展Hadoop。

Google和Yahoo对Hadoop的贡献功不可没。

总结：

- NDFS--->HDFS
- MapReduce

2)项目定义

- Apache™Hadoop®项目用于可靠，可扩展的分布式计算的开源软件。
- Apache Hadoop是一个大数据处理框架，允许使用简单的编程模型跨计算机集群分布式处理大型数据集。
- Apache Hadoop可以从单个服务器扩展到数千台计算机
- Apache Hadoop集群中每台计算机都提供本地计算和存储。
- Apache Hadoop集群不是依靠硬件来提供高可用性，而是设计了用于检测和处理应用程序层的故障，从而在计算机集群之上提供高可用性服务。

总结：

- 开源软件
- 大数据处理架构
- 单台服务可以，数千台服务器
- 每台服务器都存自己的数据及运算自己的数据
- 把硬件故障认为常态，通过软件把控故障

3)核心项目

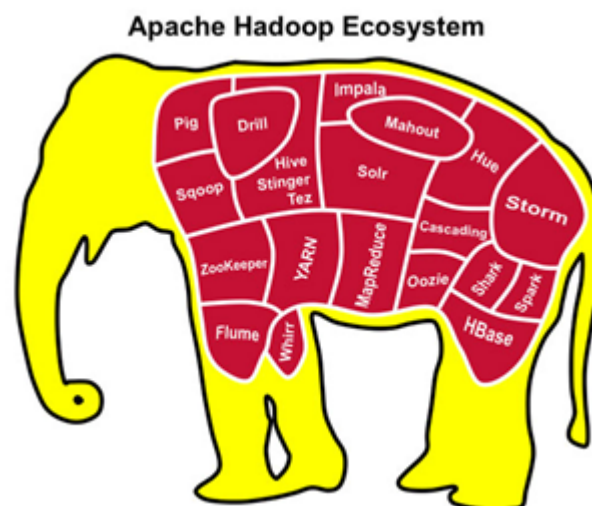
- Hadoop分布式文件系统（HDFS™）：一种分布式文件系统，可提供对应用程序数据的高吞吐量访问。
- Hadoop YARN：作业调度和集群资源管理的框架。
- Hadoop MapReduce：基于YARN的系统，用于并行处理大型数据集。
- Hadoop Common：支持其他Hadoop模块的常用实用程序。
- Hadoop Ozone：Hadoop集群所提供的对象存储。

4) 相关项目

Apache的其他Hadoop相关项目包括：

- Ambari™
 - 基于Web的工具，用于配置，管理和监控Apache Hadoop集群，包括对Hadoop HDFS，Hadoop MapReduce，Hive，HCatalog，HBase，ZooKeeper，Oozie，Pig和Sqoop的支持。
 - Ambari还提供了用于查看集群运行状况的仪表盘，例如热图，以及可视化查看MapReduce，Pig和Hive应用程序的功能，以及以用户友好的方式诊断其性能特征的功能。

- Avro™
数据序列化系统。
- Cassandra™
可扩展的多主数据库，没有单点故障。
- Chukwa™
用于管理大型分布式系统的数据收集系统。
- HBase™
可扩展的分布式数据库，支持大型表的结构化数据存储。
- Hive™
一种数据仓库基础架构，提供数据汇总和即时查询。
- Mahout™
可扩展的机器学习和数据挖掘库。
- Pig™
用于并行计算的高级数据流语言和执行框架。
- Spark™：
用于Hadoop数据的快速通用计算引擎。Spark提供了一种简单而富有表现力的编程模型，支持广泛的应用程序，包括ETL，机器学习，流处理和图形计算。
- Tez™
基于Hadoop YARN构建的通用数据流编程框架，它提供了一个功能强大且灵活的引擎，可以执行任意DAG任务来处理批量和交互式用例的数据。Tez正在被Hadoop生态系统中的Hive™，Pig™和其他框架以及其他商业软件（例如ETL工具）采用，以取代Hadoop™MapReduce作为底层执行引擎。
- ZooKeeper™
用于分布式应用程序的高性能协调服务。



总结：

- 核心项目 hdfs mapreduce yarn
- 相关项目 ambari hbase hive spark zookeeper

五、Hadoop核心项目Hadoop分布式文件系统(HDFS)介绍

学习目标

- ☐ 能够掌握HDFS功能
- ☐ 能够掌握HDFS特点
- ☐ 能够了解HDFS中Block大小
- ☐ 能够掌握HDFS中NameNode作用
- ☐ 能够掌握HDFS中DataNode作用
- ☐ 能够绘制HDFS写数据流程图
- ☐ 能够绘制HDFS读数据流程图

- Hadoop的基础核心就是HDFS和MapReduce，
- Hadoop旗下有很多经典子项目，比如HBase、Hive等，这些都是基于HDFS和MapReduce发展出来的。要想了解Hadoop，就必须知道HDFS和MapReduce是什么。

1)hadoop文件系统定义

- HDFS (Hadoop Distributed File System , Hadoop分布式文件系统)
- 它是一个高度容错性的系统
- 它适合部署在廉价的机器上
- 它能提供高吞吐量的数据访问
- 它适合那些有着超大数据集 (large data set) 的应用程序

超大数据集指的是：海量数据分析、机器学习等

2)hadoop文件系统特点

- 支持大数据文件
非常适合上TB级别的大文件或者一堆大数据文件的存储，如果文件只有几个G甚至更小就没啥意思了。
- 支持文件分块存储

HDFS会将一个完整的大文件平均分块存储到不同**计算节点**上，它的意义在于读取文件时可以同时从多个计算节点上读取不同区块的文件，多主机读取比单主机读取效率要高得多。

- 支持一次写入，多次读取，顺序读取（流式数据访问）

这种模式跟传统文件不同，它不支持动态改变文件内容，而是要求让文件一次写入就不做变化，要变化也只能在文件末添加内容。

- 支持廉价硬件

HDFS可以部署在普通PC机上，这种机制能够让给一些公司用几十台廉价的计算机就可以撑起一个大数据集群。

- 支持硬件故障

HDFS认为所有计算机都可能会出问题，为了防止某个主机失效读取不到该主机的块文件，它将同一个文件块副本分配到其它某几个主机上，如果其中一台主机失效，可以迅速找另一块副本取文件。

总结：

- 支持大文件
- 分块
- 廉价设备
- 支持硬件故障

3)hadoop分布式文件系统关键词

- Block

最基本的存储单位；将文件进行分块处理，通常是128M/块，例如：256M文件会被分为2个Block。

Hadoop 1.x版本，Block默认大小为64M；Hadoop 2.x，Block默认大小为128M。

- Hadoop集群架构(主从架构)

- NameNode(主节点)

用于保存整个文件系统的目录信息、文件信息及分块信息，这是由唯一一台主机专门保存，当然这台主机如果出错，NameNode就失效了。

- 接收用户的操作请求
- 维护文件系统的目录结构
- 管理文件和Block之间的映射管理
- 管理 block 和 DataNode 之间的映射

在Hadoop2.*开始支持activity-standby模式,如果主NameNode失效,启动备用主机运行NameNode。

- DataNode(从节点)

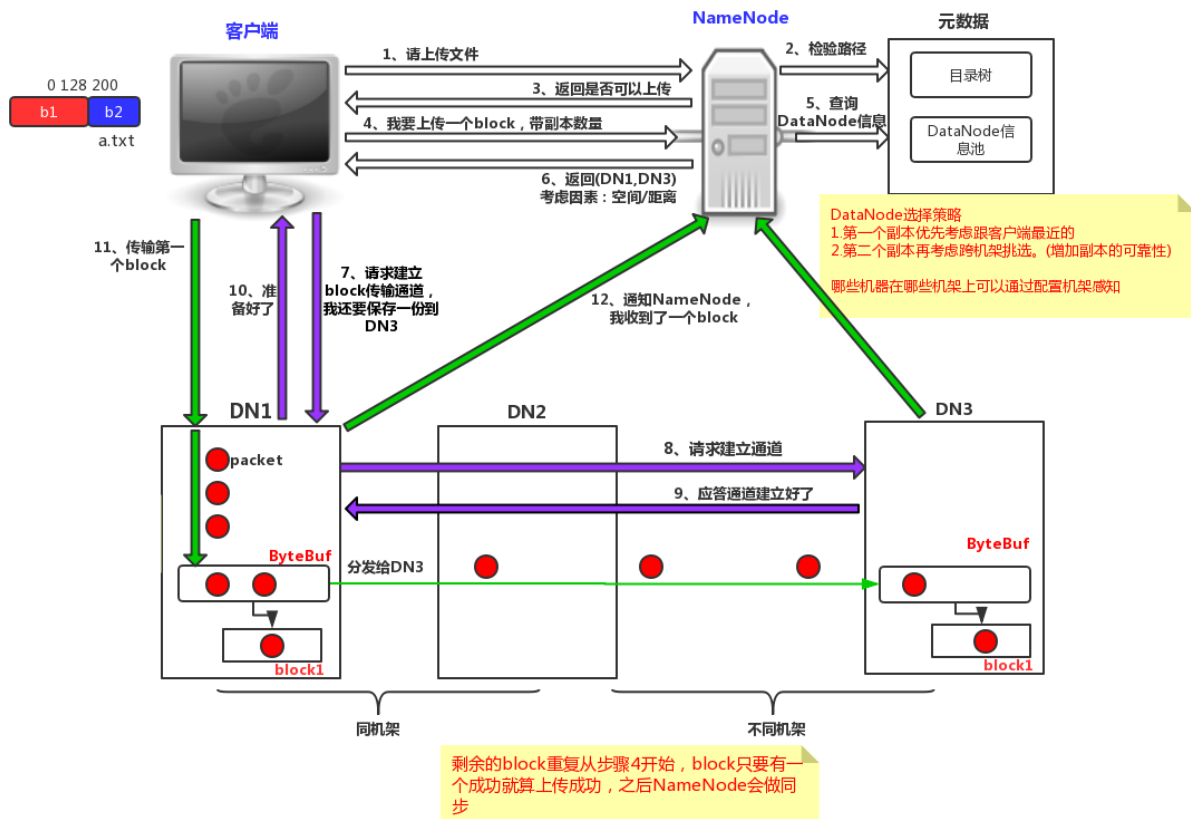
分布在廉价的计算机上,用于存储Block块文件。

- 文件被分成块存储到 DataNode 的磁盘上
- 每个Block(块)可以设置多副本

总结：

- block 文件块 128M/块
- namenode 目录 文件 分块 接收用户访问 文件与block block与datanode
- datanode 存block 副本存储

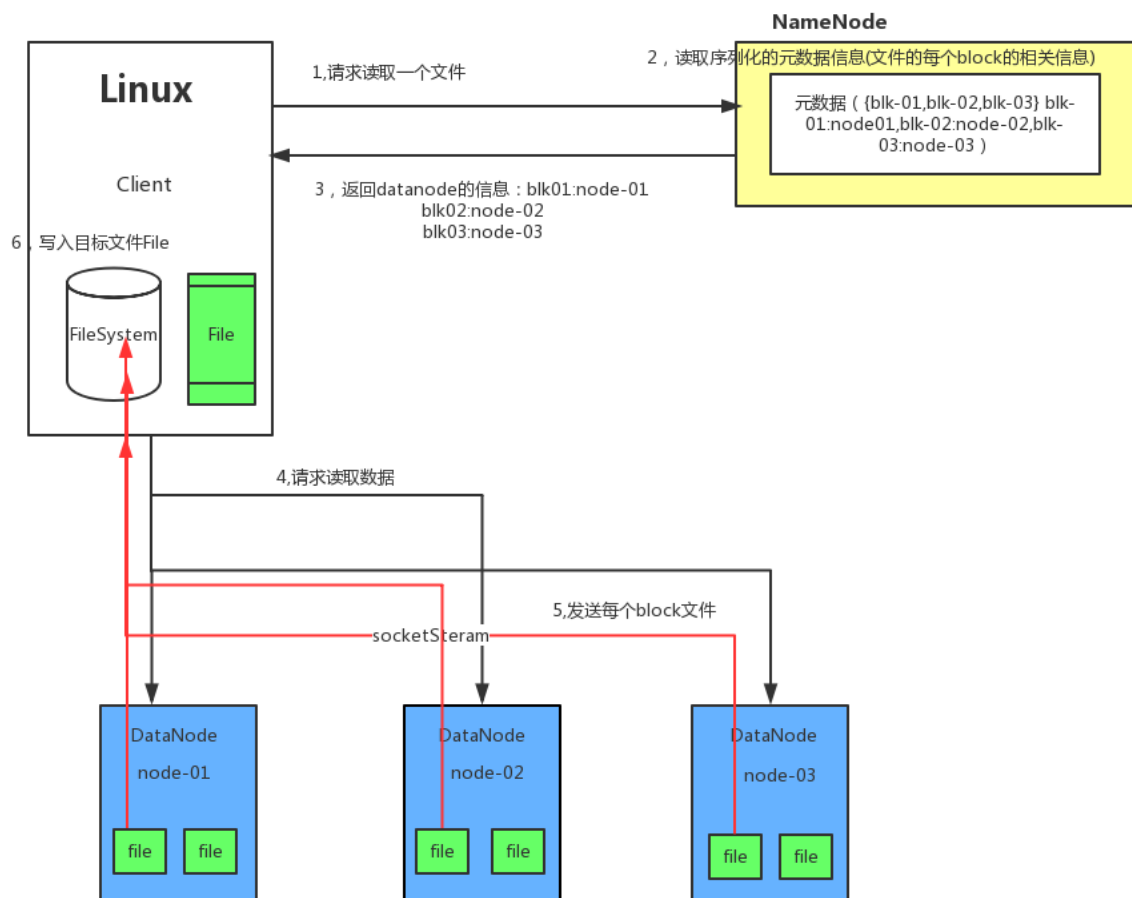
4)hdfs与数据流程



总结：

- 客户端向namenode发起请求
- 客户端向dn发起建立连接请求
- 客户端向dn存储数据

5)hdfs读数据流程



六、Hadoop核心项目编程模型(MapReduce)介绍

学习目标

- ☐ 能够理解MapReduce作用
- ☐ 能够了解MapReduce工作流程

1)MapReduce作用

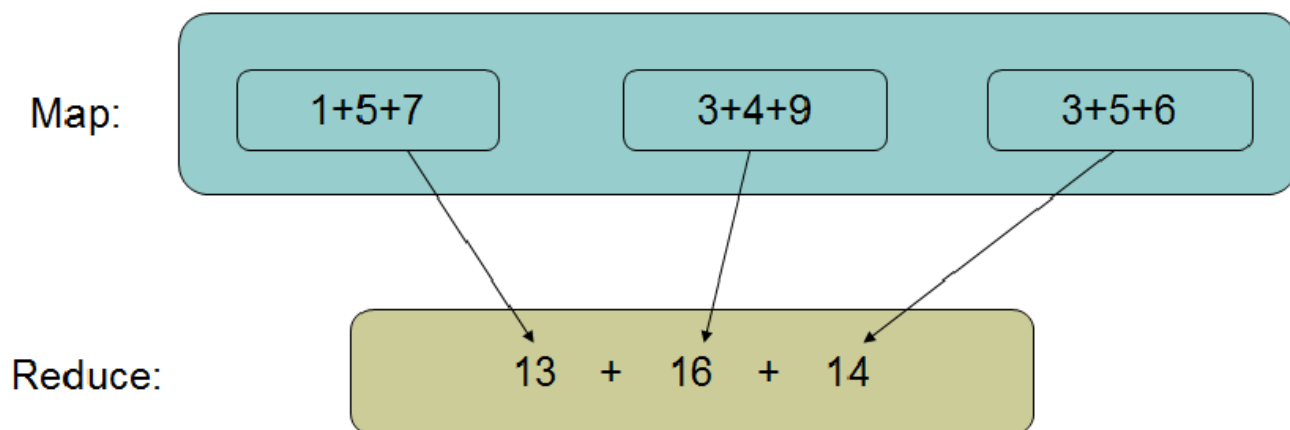
通过对HDFS分布式文件系统的了解，我们已经可以把海量数据存储在集群中DataNode之上了，但这仅是Hadoop工作的第一步，那么如何从海量的数据中找到我们所需要的数据呢，这就是MapReduce要做的事情了。下面通过2个例子给予说明：

案例1：

问题：

求和：1+5+7+3+4+9+3+5+6=？

答案：



案例2

问题

一个银行有上亿储户，银行希望找到存储金额最高的金额是多少？

答案

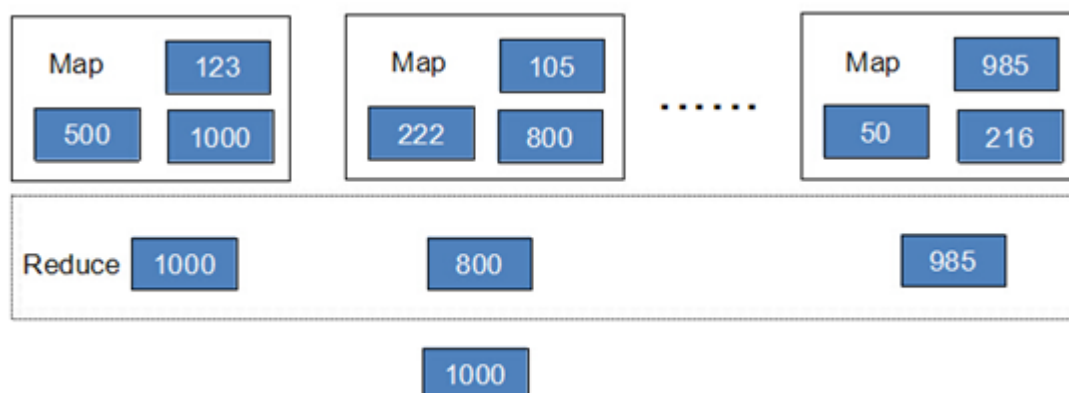
方法1：按照传统的计算方式，我们会这样(java代码)：

```
1 | Long moneys[] ...
2 | Long max = 0L;
3 | for(int i=0;i<moneys.length;i++){
4 |     if(moneys[i]>max){
5 |         max = moneys[i];
6 |     }
7 | }
```

此种方法存在问题：如果数组元素比较少的话，完全可以胜任，但是如果数组中的元素数量是海量的话，那么这个方法就会浪费非常多的时间。

方法2：

首先数字是分布存储在不同块中的，以某几个块为一个Map，计算出Map中最大的值，然后将每个Map中的最大值做Reduce操作，Reduce再取最大值给用户。



结论：

将大的数据分析分成小块逐个分析，最后再将提取出来的数据汇总分析，最终获得我们想要的内容。

通俗说MapReduce是一套从海量源数据提取、分析元素，最后返回结果集的方法。

当然怎么分块分析，怎么做Reduce操作非常复杂，Hadoop已经提供了数据分析的实现，我们只需要编写简单的需求命令即可达成我们想要的结果。

总结：

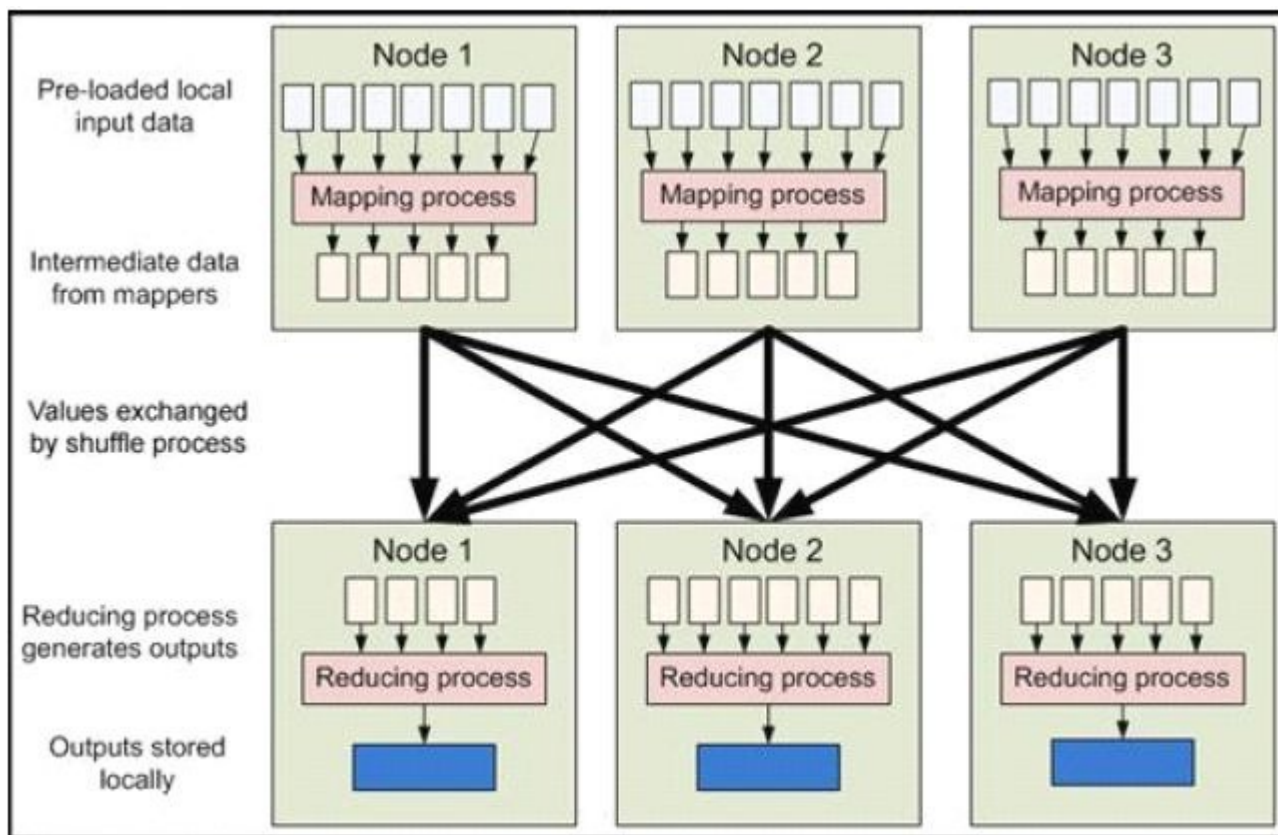
- map 把大数据分成小数据，进行计算 通过洗牌的方式给reduce
- reduce 对map的结果进行汇总

2)MapReduce工作流程

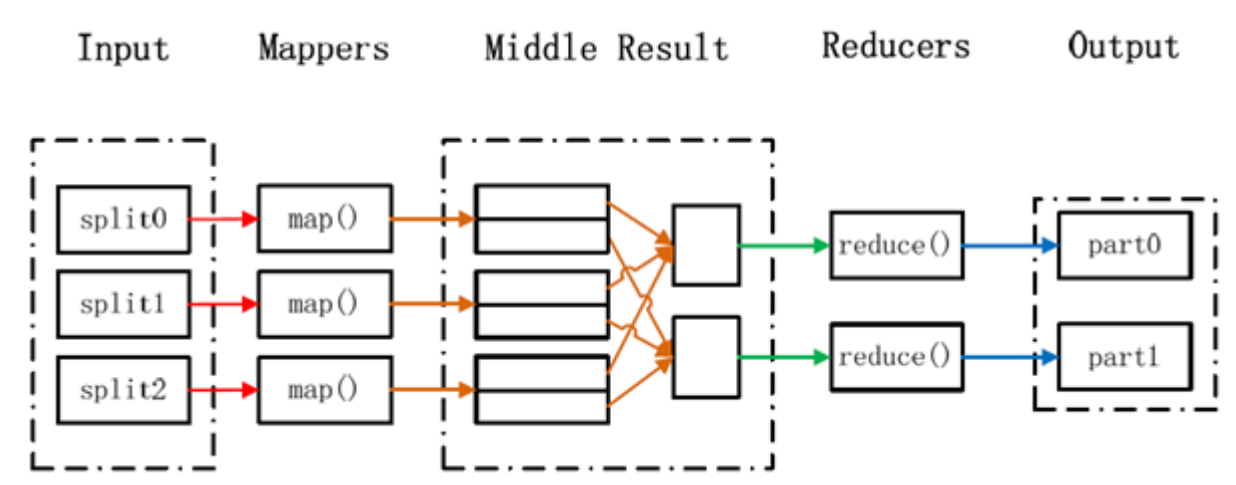
MapReduce 框架的核心步骤主要分两部分：Map 和Reduce。

当向MapReduce 框架提交一个计算作业时，它会首先把计算作业拆分成若干个Map 任务，然后分配到不同的节点(DataNode)上去执行，每一个Map 任务处理输入数据中的一部分，当Map 任务完成后，它会生成一些中间文件，这些中间文件将会作为Reduce 任务的输入数据。Reduce 任务的主要目标就是把前面若干个Map 的输出汇总到一起并输出。

从高层抽象来看，MapReduce的数据流图如下图所示：



MapReduce工作流程



总结：

- map
- reduce

七、Hadoop部署区分依据

学习目标

- ☐ 能够了解Hadoop部署的意义
- ☐ 能够了解不同部署模式区分依据

1)要求

通过部署Hadoop过程了解Hadoop工作方式，进一步了解Hadoop工作原理。

2)本地模式、伪分布式、完全分布式区分依据

主要的区别依据是NameNode、DataNode、ResourceManager、NodeManager等模块运行在几个JVM进程、几个机器。如下表所示：

模式名称	各个模块占用JVM进程数	各个模块运行在几台机器上
单机	1	1
伪分布式	N	1
完全分布式	N	N
HA+完全分布式	N	N

八、单机(本地模式)部署

学习目标

- ☐ 能够了解Hadoop默认部署模式
- ☐ 能够掌握Hadoop部署软件包获取
- ☐ 能够对部署完成的Hadoop进行测试

1.1)单机部署模式介绍

- 单机(本地模式)是Hadoop的默认部署模式。
- 当配置文件为空时，Hadoop完全运行在本地。
- 不需要与其他节点交互，单机(本地模式)就不使用HDFS，也不加载任何Hadoop的守护进程。
- 该模式主要用于开发调试MapReduce程序的应用逻辑。

1.2)部署软件包获取

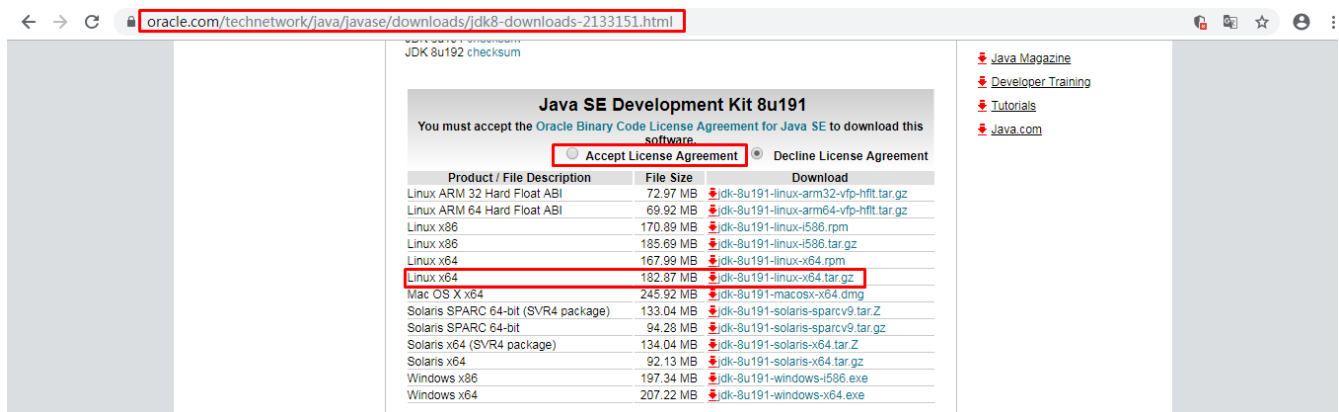
1.2.1) 获取hadoop软件包



Version	Release date	Source download	Binary download	Release notes
2.8.5	2018 Sep 15	source (checksum signature)	binary (checksum signature)	Announcement
3.1.1	2018 Aug 8	source (checksum signature)	binary (checksum signature)	Announcement
2.7.7	2018 May 31	source (checksum signature)	binary (checksum signature)	Announcement
3.0.3	2018 May 31	source (checksum signature)	binary (checksum signature)	Announcement
2.9.1	2018 May 3	source (checksum signature)	binary (checksum signature)	Announcement

```
1 [root@localhost ~]#wget
http://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-
2.8.5.tar.gz
```

1.2.2) 获取JDK软件包



Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	162.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

```
1 [root@localhost ~]#firefox http://download.oracle.com
```

1.3)部署

1.3.1)jdk部署

```
1 [root@localhost ~]#tar xf jdk-8u191-linux-x64.tar.gz -C /usr/local
2 [root@localhost ~]# cd /usr/local
3 [root@localhost local]# mv jdk1.8.0_191 jdk
```

解压到指定目录后，请修改目录名称

1.3.2) hadoop部署

```
1 [root@localhost ~]# tar xf hadoop-2.8.5.tar.gz -C /opt
2 [root@localhost ~]# cd /opt
3 [root@localhost opt]# mv hadoop-2.8.5 hadoop
```

解压至指定目录后，请修改目录名称

1.3.3) Linux系统环境变量

```
1 [root@localhost ~]#vim /etc/profile
2 export JAVA_HOME=/usr/local/jdk
3 export HADOOP_HOME=/opt/hadoop
4 export PATH=${JAVA_HOME}/bin:${HADOOP_HOME}/bin:$PATH
```

1.3.4) 应用测试

1.3.4.1)加载环境变量

```
1 [root@localhost ~]#source /etc/profile
```

1.3.4.2)测试hadoop可用性

```
1 [root@localhost ~]#mkdir /home/input
2 [root@localhost ~]#cp /opt/hadoop/etc/hadoop/*.xml /home/input
3 [root@localhost ~]#hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
  examples-2.8.5.jar wordcount /home/input/ /home/output/00
4 [root@localhost ~]#cat /home/output/00/*
```

输出目录中有_SUCCESS文件说明JOB运行成功，part-r-00000是输出结果文件。

1.3.4.3)词频统计练习

要求：

1. 制作一个文件，里面包含10-20不同的或部分相同的单词。
2. 使用wordcount方法实现单词出现频率统计。

九、伪分布式部署

学习目标

- ☐ 能够了解伪分布式部署模式
- ☐ 能够正确修改配置文件
- ☐ 能够掌握YARN架构及架构角色功能
- ☐ 能够对已部署的Hadoop集群进行应用测试

1)伪分布式部署模式介绍

- Hadoop守护进程运行在本地机器上，模拟一个小规模的的集群。
- 该模式在单机模式之上增加了代码调试功能，允许你检查内存使用情况，HDFS输入/输出，以及其他的守护进程交互。

2)获取软件包

可参考：第八节 1.2.1与1.2.2小节

3) 修改配置文件

主要涉及的配置文件有：hadoop-env.sh、mapred-env.sh、yarn-env.sh、core-site.xml

3.1) 修改hadoop-env.sh、mapred-env.sh、yarn-env.sh文件中JAVA_HOME参数

```
1 [root@localhost ~]#vim ${HADOOP_HOME}/etc/hadoop/hadoop-env.sh
```

```
1 修改JAVA_HOME参数为：
2 export JAVA_HOME="/usr/local/jdk"
```

3.2)修改core-site.xml

```
1 [root@localhost ~]#vim ${HADOOP_HOME}/etc/hadoop/core-site.xml
```

```
1 (1)配置fs.defaultFS
2 <property>
3 <name>fs.defaultFS</name>
4 <value>hdfs://hd1:8020</value>
5 </property>
6
7 (2)配置hadoop临时目录
8 <property>
9 <name>hadoop.tmp.dir</name>
10 <value>/opt/data/tmp</value>
11 </property>
```

配置临时目录前，请先创建此目录,不创建也可以。

HDFS的NameNode数据默认都存放这个目录下，查看 `*-default.xml` 等默认配置文件，就可以看到很多依赖 `${hadoop.tmp.dir}` 的配置。

默认的 `hadoop.tmp.dir` 是 `/tmp/hadoop-${user.name}`，此时有个问题就是NameNode会将HDFS的元数据存储在这个/tmp目录下，如果操作系统重启了，系统会清空/tmp目录下的东西，导致NameNode元数据丢失，是个非常严重的问题，所有我们应该修改这个路径。

3.3) 配置hdfs-site.xml

```
1 [root@localhost ~]#vim ${HADOOP_HOME}/etc/hadoop/hdfs-site.xml
```

```
1 <property>
2 <name>dfs.replication</name>
3 <value>1</value>
4 </property>
```

dfs.replication配置的是HDFS存储时的备份数量，因为这里是伪分布式环境只有一个节点，所以这里设置为1。

3.4)格式化hdfs

```
1 [root@localhost ~]#hdfs namenode -format
```

格式化是对HDFS这个分布式文件系统DataNode进行分块，统计所有分块后的初始元数据的存储在NameNode中。

格式化后，查看core-site.xml里hadoop.tmp.dir（本例是/opt/data目录）指定的目录下是否有了dfs目录，如果有，说明格式化成功。

3.5)查看hdfs临时目录

```
1 [root@localhost ~]#ls /opt/data/tmp/dfs/name/current
```

fsimage是NameNode元数据在内存满了后，持久化保存到的文件。

fsimage*.md5 是校验文件，用于校验fsimage的完整性。

seen_txid 是hadoop的版本

version文件里保存：

- namespaceID：NameNode的唯一ID。
- clusterID:集群ID，NameNode和DataNode的集群ID应该一致，表明是一个集群。

4)启动角色

请把hadoop安装目录中的sbin目录中的命令添加到/etc/profile环境变量中，不然无法使用hadoop-daemon.sh

4.1) 启动namenode

```
1 [root@localhost ~]#hadoop-daemon.sh start namenode
```

4.2) 启动datanode

```
1 [root@localhost ~]#hadoop-daemon.sh start datanode
```

4.3)验证

JPS命令查看是否已经启动成功，有结果就是启动成功了。

```
1 [root@localhost ~]#jps
```

5) HDFS上测试创建目录、上传、下载文件

5.1)创建目录

```
1 [root@localhost ~]#hdfs dfs -mkdir /test
```

5.2)上传文件

```
1 [root@localhost ~]#hdfs dfs -put  
2 ${HADOOP_HOME}/etc/hadoop/core-site.xml /test
```

5.3)读取内容

```
1 [root@localhost ~]#hdfs dfs -cat /test/core-site.xml
```

5.4)下载文件到本地

```
1 | [root@localhost ~]#hdfs dfs -get /test/core-site.xml
```

6)配置yarn

6.1)Yarn介绍

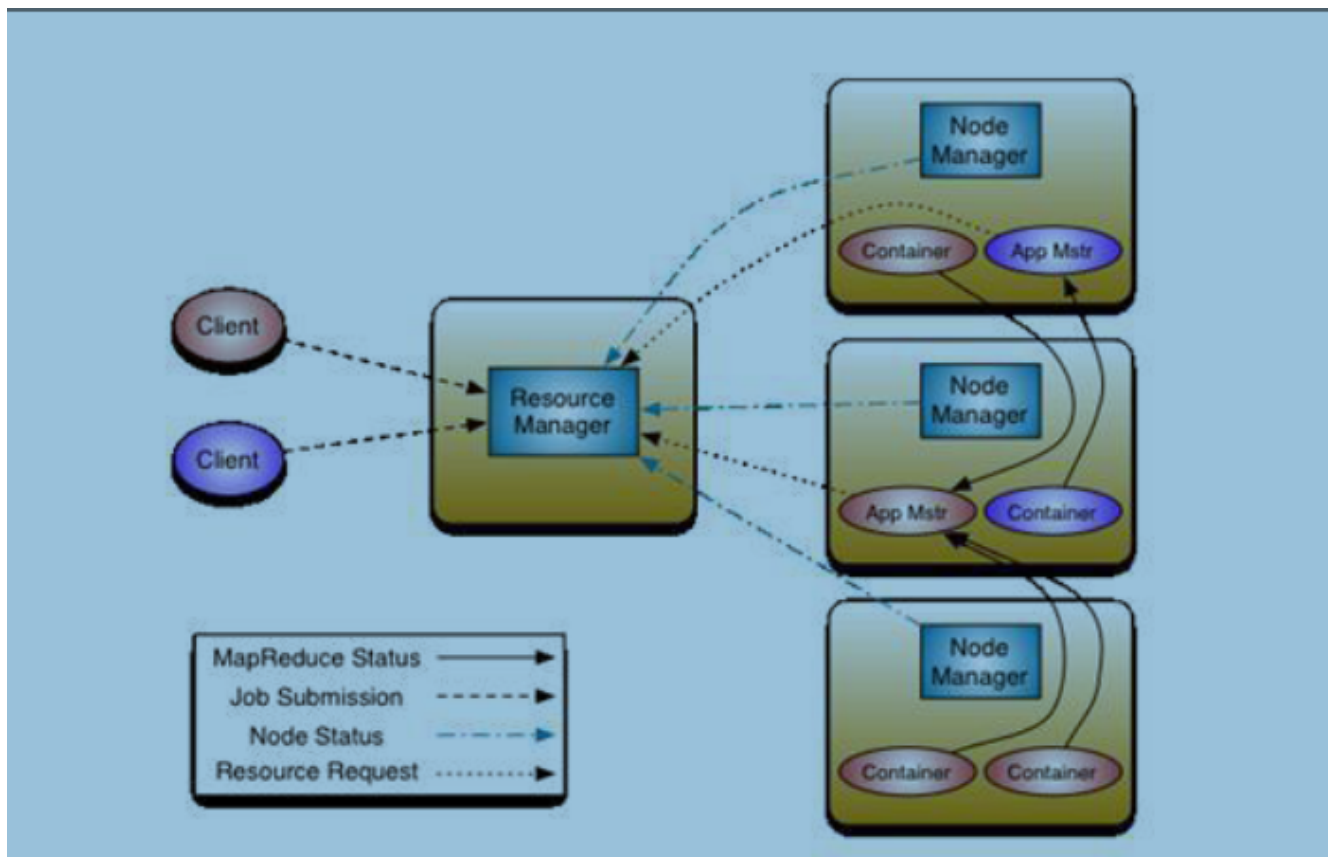
- A framework for job scheduling and cluster resource management.
 - 功能：任务调度 和 集群资源管理
- YARN (Yet An other Resouce Negotiator) 另一种资源协调者
是 Hadoop 2.0新增加的一个子项目，弥补了Hadoop 1.0(MRv1)扩展性差、可靠性资源利用率低以及无法支持其他计算框架等不足。
- Hadoop的下一代计算框架MRv2将资源管理功能抽象成一个通用系统YARN
- MRv1的 jobtracker和tasktrack也不复存在，计算框架 (MR, storm, spark)同时运行在之上，使得hadoop进入了多计算框架的弹性平台时代。

总结：

- yarn是一种资源协调者
- 从mapreduce拆分而来
- 带来的好处：让hadoop平台性能及扩展性得到更好发挥

6.2)使用Yarn好处

- 在某些时间，有些资源计算框架的集群紧张,而另外一些集群资源空闲。 那么这框架共享使用一个则可以大提高利率些集群资源空闲。
- 维护成本低。
- 数据共享。 避免了集群之间移动数据。
- YARN 主从架构
 - ResourceManager 资源管理
 - NodeManager 节点管理
 - ResourceManager
负责对各个NodeManager 上的资源进行统一管理和任务调度。
 - NodeManager
在各个计算节点运行，用于接收RM中ApplicationsManager 的计算任务、启动/停止任务、和RM中Scheduler 汇报并协商资源、监控并汇报本节点的情况。



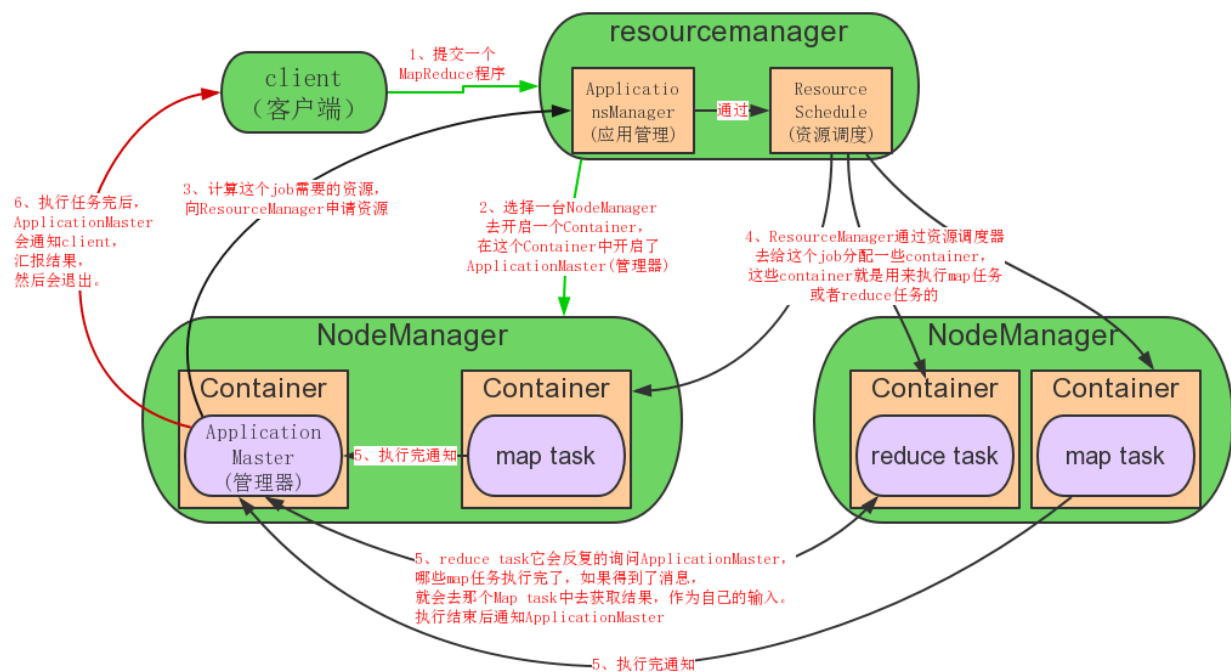
6.3) 配置mapred-site.xml

默认没有mapred-site.xml文件，但是有个mapred-site.xml.template配置模板文件。复制模板生成mapred-site.xml。

```
1 [root@localhost ~]#cp /opt/hadoop/etc/hadoop/mapred-site.xml.template  
  /opt/hadoop/etc/hadoop/mapred-site.xml
```

```
1 <property>  
2 <name>mapreduce.framework.name</name>  
3 <value>yarn</value>  
4 </property>
```

指定mapreduce运行在yarn框架上。



6.4) 配置 yarn-site.xml

```

1 <property>
2   <name>yarn.nodemanager.aux-services</name>
3   <value>mapreduce_shuffle</value>
4 </property>
5 <property>
6   <name>yarn.resourcemanager.hostname</name>
7   <value>hd1</value>
8 </property>

```

yarn.nodemanager.aux-services配置了yarn的默认混洗方式，选择为mapreduce的默认混洗算法。

yarn.resourcemanager.hostname指定了Resourcemanager运行在哪个节点上。

6.5) 启动 yarn

```
1 [root@localhost ~]# yarn-daemon.sh start resourcemanager
```

```
1 [root@localhost ~]# yarn-daemon.sh start nodemanager
```

```
1 [root@localhost ~]# jps
```

6.6)YARN的Web页面

YARN的Web客户端端口号是8088，通过<http://hd1:8088/>可以查看。

6.7)测试

在Hadoop的share目录里，自带了一些jar包，里面带有一些mapreduce实例小例子，位置在share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.5.jar，可以运行这些例子体验刚搭建好的Hadoop平台，我们这里来运行最经典的WordCount实例。

6.7.1) 创建目录

```
1 [root@localhost ~]#hdfs dfs -mkdir -p /test/input
```

创建原始文件:

在本地/opt/data目录创建一个文件wc.input,内容如下:

tom jime

hadoop hive

hbase hadoop tom

6.7.2)上传文件

将wc.input文件上传到HDFS的/test/input目录中:

```
1 [root@localhost ~]#hdfs dfs -put /opt/data/wc.input /test/input
```

6.7.3)运行实例

```
1 [root@localhost ~]#yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-  
2 2.8.5.jar wordcount /test/input /test/output
```

6.7.4)查看输出结果

```
1 [root@localhost ~]#hdfs dfs -ls /test/output
```

output目录中有两个文件，_SUCCESS文件是空文件，有这个文件说明Job执行成功。

part-r-00000文件是结果文件，其中-r-说明这个文件是Reduce阶段产生的结果，mapreduce程序执行时，可以没有reduce阶段，但是肯定会有map阶段，如果没有reduce阶段这个地方有是-m-。

一个reduce会产生一个part-r-开头的文件。

7)停止hadoop

```
1 [root@localhost ~]#hadoop-daemon.sh stop namenode
2 [root@localhost ~]#hadoop-daemon.sh stop datanode
3 [root@localhost ~]#yarn-daemon.sh stop resourcemanager
4 [root@localhost ~]#yarn-daemon.sh stop nodemanager
```

扩展

8)开启历史服务及查看

Hadoop开启历史服务可以在web页面上查看Yarn上执行job情况的详细信息。可以通过历史服务器查看已经运行完的Mapreduce作业记录，比如用了多少个Map、用了多少个Reduce、作业提交时间、作业启动时间、作业完成时间等信息。

```
1 [root@localhost ~]#mr-jobhistory-daemon.sh start historyserver
```

开启后，可以通过Web页面查看历史服务器：

<http://localhost:19888>

9) 开启日志聚集

MapReduce是在各个机器上运行的，在运行过程中产生的日志存在于各个机器上，为了能够统一查看各个机器的运行日志，将日志集中存放在HDFS上，这个过程就是日志聚集。

Hadoop默认是不启用日志聚集的。在yarn-site.xml文件里配置启用日志聚集。

```
1 <property>
2     <name>yarn.log-aggregation-enable</name>
3     <value>true</value>
4 </property>
5 <property>
6     <name>yarn.log-aggregation.retain-seconds</name>
7     <value>106800</value>
8 </property>
```

yarn.log-aggregation-enable:是否启用日志聚集功能。

yarn.log-aggregation.retain-seconds：设置日志保留时间，单位是秒。

以上内容要复制给集群所有主机

9.1)重启yarn

```
1 [root@localhost ~]#stop-yarn.sh
2 [root@localhost ~]#start-yarn.sh
```

9.2)重启历史服务

```
1 [root@localhost ~]#mr-jobhistory-daemon.sh stop historyserver
2 [root@localhost ~]#mr-jobhistory-daemon.sh start historyserver
```