**Hadoop笔记**

对于Google而言，BigTable Hbase为数据库，MapReduce为计算查找，GFS为文件系统。

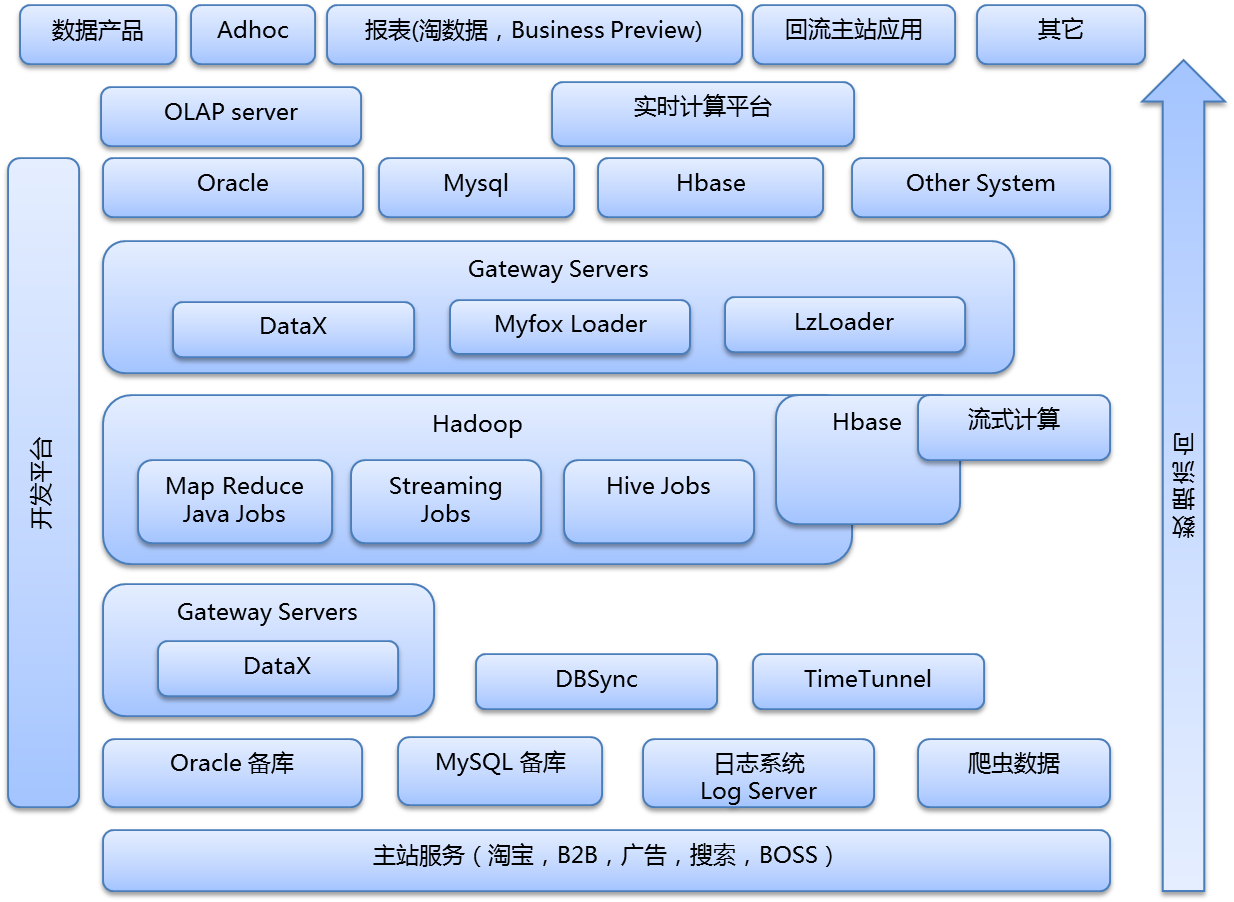
Lucene为hadoop前身，Nutch为Lucene的微缩版，Lucene是一个Java高性能全文索引引擎工具包，它可以方便地嵌入到各种实际应用中实现全文索引搜索功能，而Nutch是一个应用程序，它是一个以Lucene为基础实现的搜索引擎应用。

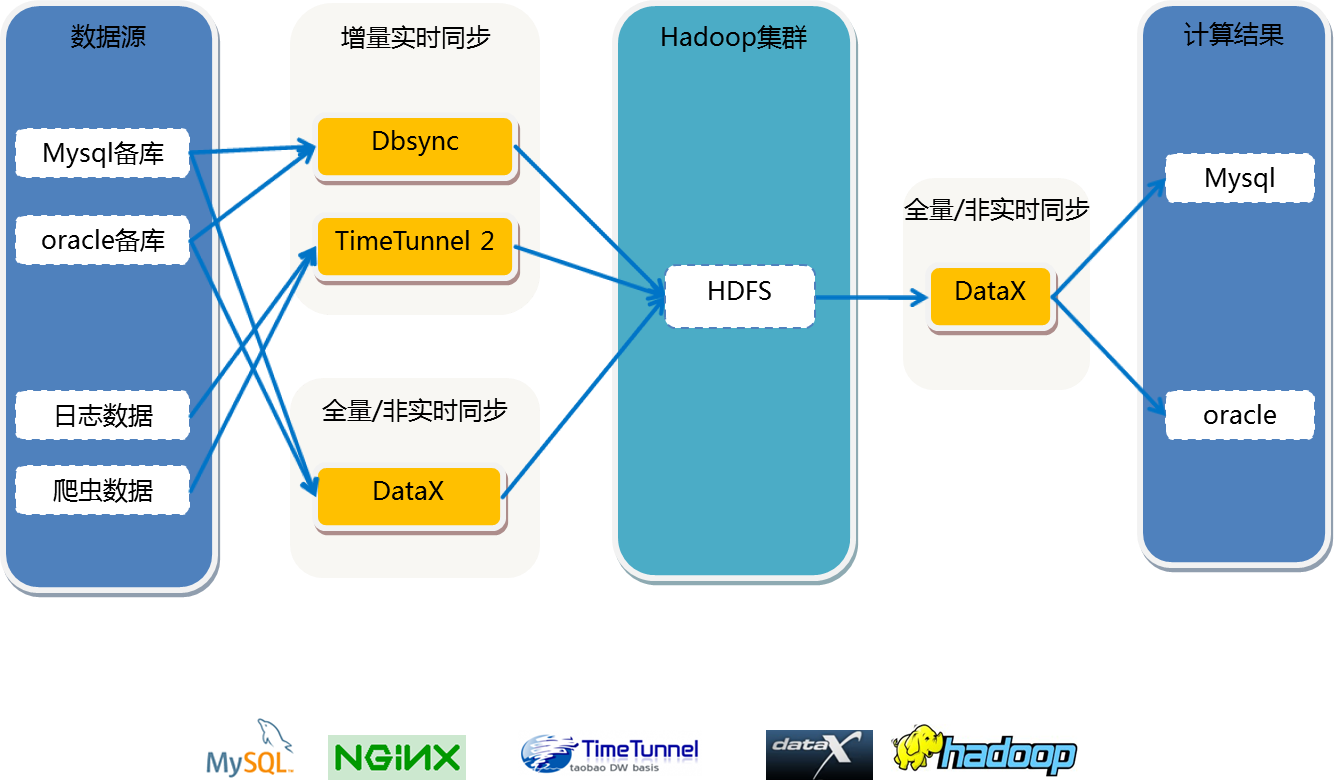
IaaS

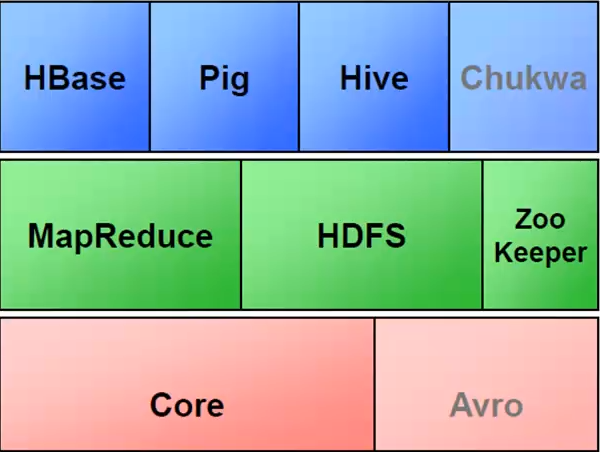
Nutch

Lucene：



web框架





Zoo Keeper：Google Chubby的开源实现，节点通信工具。

Pig：Hadoop客户端，一个轻度的命令语言Pig Latin(可以完成排序、过滤、求和、聚组、关联)，可以将命令转换为MapReduce，减少用户编写java苦恼（Yahoo贡献）。

Hive：数据仓库工具，将SQL转换为MapReduce（只支持部分SQL语言），提供shell、JDBC/ODBC、Thiift、web等接口。

HBase：Google Bigtable的开源实现，列存储非关系型数据库。NoSQL的典型代表。可以使用shell（shell中使用HQL语言进行查询）、web、api等多种方式访问。适合高速写的操作。

Chukwa：架构在Hadoop之上的数据采集与分析框架，主要进行日志采集和分析，通过安装在收集节点的代理采集最原始的日志数据。

Sqoop：利用JDBC接口完成和关系型数据库之间交换数据，hadoop和SQL之间数据交换。

Avro：数据序列化工具，类似于google的protobuf和facebook的thrift模块。

Cassandra：NoSQL分布式Key-Value型数据库，由Facebook贡献，与Hbase类型，借鉴Google Bigtable思想体系。只有顺序写，没有随机写的设计满足高负荷性能，Hbase抢了其风头。

Mahout：分布式机器学习和数据挖掘的Lib

HDFS的可靠性：

冗余副本策略

机架策略

心跳机制

安全模式

校验和

回收站

元数据保护

快照机制

hadoop没有当前路径的概念，因此没办法使用cd命令。

map-reduce思想就是分而治之，mapper负责分，reducer负责对map的结果进行汇总，在mapper和reducer之间的一个步骤（可以没有），它可以简化reducer的过程。

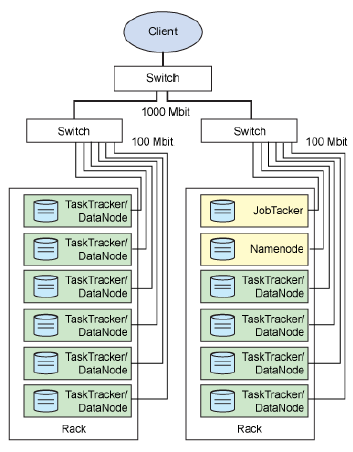
性能调优：

尽量多的reducer

输入：大文件优于小文件

减少网络传输：压缩map的输出

集群中的任何一个节点都可以提交任务。



Hadoop架构

NameNode：HDFS的守护程序，记录文件如何分割成数据块，对内存和I/O进行集中管理，是个单点，发生故障将使集群崩溃，负责管理HDFS的目录树和相关的文件元数据信息，这些信息是以fsimage（HDFS元数据镜像文件）和editlog（HDFS文件改动日志）两个文件存放在本地磁盘。

Secondary NameNode：监控HDFS状态的辅助后台程序，每个集群都有一个，在Namenode故障时人工手动顶上，Secondary Namenode最重要的任务并不是为了NameNode元数据进行热备份，而是定期合并fsimage和edits日志，并传输给NameNode，目的是为了减小NameNode压力，NameNode自己并不会合并fsimage和edits日志。

DataNode：每台服务器运行一个，负责HDFS写到本地文件系统。

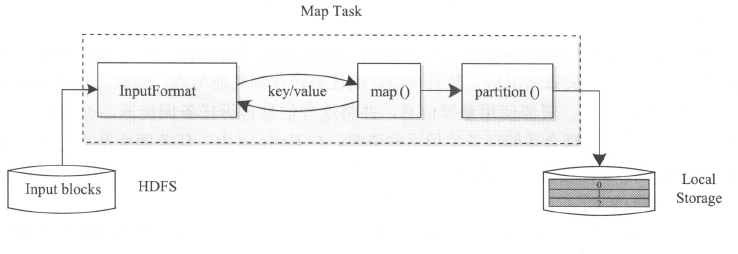
Task Tracker：与datanode结合，管理每个节点上的task，与JobTracker交互。

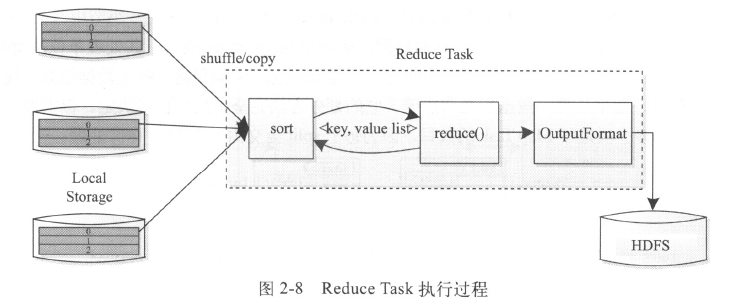
JobTracker：用户处理作业，决定任务调度，监控task，重启失败的task，每个集群只有唯一一个JobTracker。

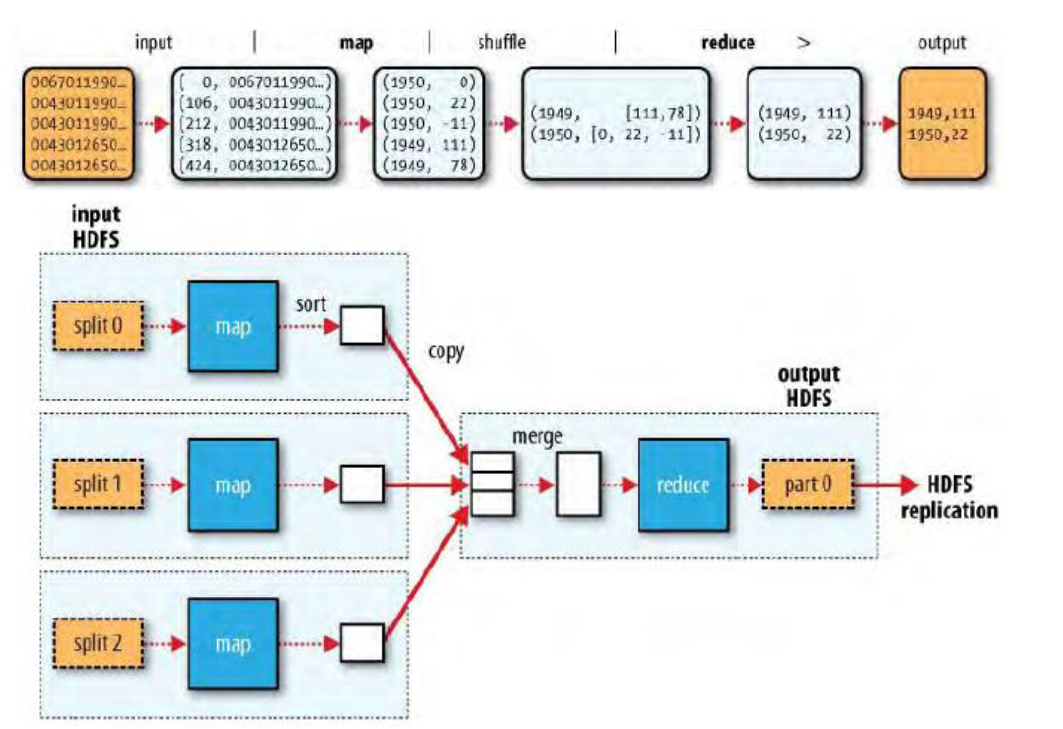
Hadoop使用Master-slave模式：

Master：NameNode，SecondaryNode，JobTracker，浏览器。

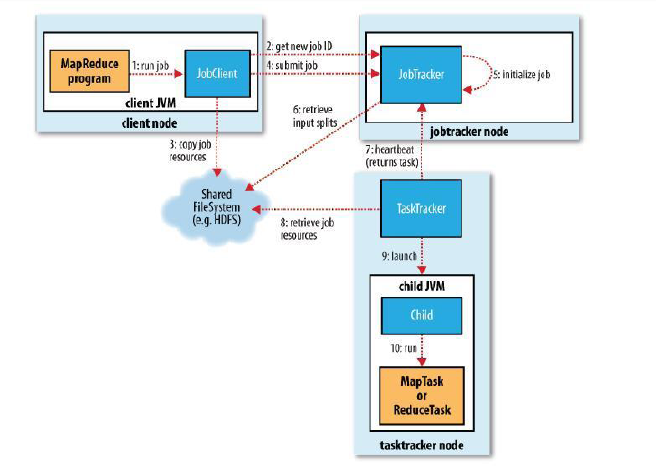
Slave：TaskTracker，DataNode。







MapReduce 典型场景



Hadoop集群监控第三方工具：Ganglia ChuKwa Openstack

1. **安装部署hadoop2.2.0**

自行在家中搭建了一个3节点的hadoop模型，其中macbook在win7中安装了两个虚拟机，分别为alance\_macbook 和alance\_macbook2，在Thinkpad的win7环境中安装了一个虚拟机，为alance\_thinkpad，具体的mac地址，以及各个节点的固定IP地址分配如下表:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hostname | Mac地址 | IP地址 | Hadoop用途 | HBase用途 |
| thinkpad | 00-21-5D-97-25-0C | 192.168.2.100 | / | / |
| macbook | 20-C9-D0-47-2C-95 | 192.168.2.101 | / | / |
| alance\_macbook1 | 00-0C-29-9F-EA-04 | 192.168.2.102 | NameNode/Secondary NameNode  ResouceManager | HMaster |
| alance\_macbook2 | 00-0C-29-37-8F-8A | 192.168.2.103 | DataNode/NodeManager | RegionServer |
| alance\_thinkpad | 00-0C-29-FD-25-7C | 192.168.2.104 | DataNode/NodeManager | RegionServer |

1. 配置hosts文件，使用sudo vim /etc/hosts命令，内容如下：

192.168.2.102 alance\_macbook1

192.168.2.103 alance\_macbook2

192.168.2.104 alance\_thinkpad

此处一定需要注意的地方在于，linux下居然不能够解析下划线，因此导致系统没有起来，需要将下划线该为-，而且需要将slaves文件中的也要改正过来。更改之后的hosts文件如下：

192.168.2.102 alance-macbook1

192.168.2.103 alance-macbook2

192.168.2.104 alance-thinkpad

1. 将hadoop加入到/etc/sudoers文件中，使其拥有sudo的权限。

hadoop ALL=(ALL:ALL) ALL

1. 配置hadoop的bashrc文件，导入java和hadoop（网上好多教程都漏了这一步！）环境变量。

export JAVA\_HOME=/home/hadoop/java/jdk1.7.0\_55

export JRE\_HOME=/home/hadoop/java/jdk1.7.0\_55/jre

export CLASSPATH=.:$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$JAVA\_HOME:$PATH

export HADOOP\_DEV\_HOME=/home/hadoop/hadoop-2.2.0

export PATH=$PATH:$HADOOP\_DEV\_HOME/bin

export PATH=$PATH:$HADOOP\_DEV\_HOME/sbin

export HADOOP\_MAPARED\_HOME=${HADOOP\_DEV\_HOME}

export HADOOP\_COMMON\_HOME=${HADOOP\_DEV\_HOME}

export HADOOP\_HDFS\_HOME=${HADOOP\_DEV\_HOME}

export YARN\_HOME=${HADOOP\_DEV\_HOME}

export HADOOP\_CONF\_DIR=${HADOOP\_DEV\_HOME}/etc/hadoop

最好系统环境变量/etc/profile也加入如下的配置：

export JAVA\_HOME=/home/hadoop/java/jdk1.7.0\_55

export JRE\_HOME=/home/hadoop/java/jdk1.7.0\_55/jre

export CLASSPATH=.:$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$JAVA\_HOME:$PATH

1. 对于hadoop首先是需要配置hadoop-env.sh将java的环境变量加入到其中，添加：

export JAVA\_HOME=/home/hadoop/java/jdk1.7.0\_55

1. 然后需要配置core-site.xml文件，该文件主要用来配置NameNode的端口号以及IP地址。配置如下：

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://192.168.2.102:9000</value>

</property>

</configuration>

//NameNode 端口号

1. 然后配置hdfs-site.xml文件，配置如下：

<configuration>

<property>

<name>dfs.data.dir</name>

<value>/home/hadoop/hadoop-2.2.0/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

</configuration>

由于hadoop2.2.0中并没有masters配置文件，NameNode是在hdfs-site.xml文件中配置完成，即需要在NameNode（alance\_macbook1）中添加额外配置代码（NameNode 和SecondaryNameNode为同一个节点）：

<property>

<name>dfs.http.address</name>

<value>192.168.2.102:50070</value>

<description>Secondary get fsimage and edits via dfs.http.address</description>

</property>

<property>

<name>dfs.secondary.http.address</name>

<value>192.168.2.102:50090</value>

<description>NameNode get the newest fsimage via dfs.secondary.http.address</description>

</property>

//50070可以用来观测NameNode节点的端口号

//50090可以用来观测SecondNameNode节点的端口号

然后在每个在每个slave节点中，配置slaves文件，添加如下内容：

alance-macbook2

alance-thinkpad

1. 最后配置mapred-site.xml文件，默认情况下不存在该文件，需要自己创建：

cp mapred-site.xml.template mapred-site.xml, mapred-site.xml配置内容如下：

<configuration>

<property>

<name>mapred.job.tracker</name>

<value>192.168.2.102:9001</value>

</property>

</configuration>

上述过程完成了真个hadoop的安装过程，下面使用

hadoop namenode -format

命令完成NameNode节点的初始化工作，然后通过如下命令：

start-all.sh

命令启动所有的节点。

hadoop fs -ls /

查看hadoop根目录下的所有文件，主要ls后面要加空格和/

hadoop fs -put test1.txt /

将test1.txt上传至hdfs中，注意要使用空格和/指定根目录。

hadoop fs -get new /input

将hadoop目录中的文件下载到本地。

hadoop fs –rmr /input

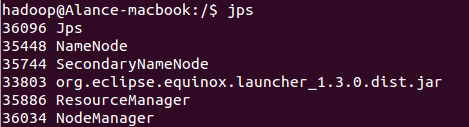
删除hadoop中的文件。

运行hadoop中的wordcount，测试mapreduce的运行状况，具体代码如下：

hadoop jar ~/hadoop-2.2.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount /input /output

hadoop2.2.0中是实际上是没有jobtracker的，所以<http://192.168.2.102:50030/>这个地址是没办法访问的，可以通过访问resource manager的监控地址：<http://192.168.2.102:8088/>

在NameNode中运行jps可以得到如下的截图：



1. **安装部署zookeeper-3.3.4**

安装zookeeper，解压缩zookeeper-3.3.4.tar.gz，conf目录下将zoo\_sample.cfg 改为 zoo.cfg修改配置文件conf/zoo.cfg，内容如下所示：

tickTime=2000

dataDir=/home/hadoop/zookeeper/zookeeper-3.4.5

clientPort=2181

initLimit=5

syncLimit=2

dataLogDir=/home/hadoop/zookeeper/zookeeper-3.4.5/logs

server.1=alance-macbook1:2888:3888

server.2=alance-macbook2:2888:3888

server.3=alance-thinkpad:2888:3888

上面已经在一台机器slave-01上配置完成zooKeeper，现在可以将该配置好的安装文件远程拷贝到集群中的各个结点对应的目录下：

在我们配置的dataDir指定的目录下面，创建一个myid文件，里面内容为一个数字，用来标识当前主机，conf/zoo.cfg文件中配置的server.X中X为什么数字，则myid文件中就输入这个数字，例如：

 echo "1" > /home/hadoop/zookeeper/zookeeper-3.4.5/myid

配置hadoop的bashrc文件

export ZOOKEEPER\_HOME=/home/hadoop/zookeeper/zookeeper-3.4.5

export PATH=$PATH:$ZOOKEEPER\_HOME/bin

zookeeper的常用命令如下：

zkServer.sh start //启动zookeeper

zkServer.sh stop //关闭zook

zkServer.sh status //查看zookeeper状态，可以查询谁是leader，谁是follower

1. **安装部署hbase-0.98.1-hadoop2**

安装部署完全分布式Hbase，下载hbase-0.98.1-hadoop2-bin.tar.gz文件（hadoop2代指支持hadoop2.x的版本，需要 格外小心Hbase的版本，之前在Hadoop2.2.0环境中安装hbase-0.96.2，会有莫名其妙的错误，浪费了很多时间也没有得到解决，最后安装了hbase-0.98.1，可以正常运行）。

1. 先设置hbase-env.sh文件，设置内容如下：

export JAVA\_HOME=/home/hadoop/java/jdk1.7.0\_55

export HBASE\_CLASSPATH=/home/hadoop/hadoop-2.2.0/etc/hadoop

export HBASE\_HEAPSIZE=1000

export HBASE\_LOG\_DIR=/home/hadoop/Hbase/hbase-0.96.2-hadoop2/logs

export HBASE\_MANAGES\_ZK=false

1. 配置hadoop的bashrc文件

export HBASE\_HOME=/home/hadoop/Hbase/hbase-0.98.1-hadoop2

export PATH=$PATH:$HBASE\_HOME/bin

1. 配置hbase的conf/hbase-site.xml文件

<configuration>  
<property>  
<name>hbase.rootdir</name>  
<value>hdfs://192.168.2.102:9000/hbase</value>  
<description>HBase数据存储目录</description>  
</property>  
<property>  
<name>hbase.cluster.distributed</name>  
<value>true</value>  
<description>指定HBase运行的模式：false：单机/伪分布；true：完全分布</description>  
</property>  
<property>  
<name>hbase.master</name>  
<value>hdfs://192.168.2.102:60000</value>  
<description>指定Master位置</description>  
</property>

<property>  
<name>hbase.zookeeper.property.dataDir</name>  
<value>/home/hadoop/zookeeper/zookeeper-3.4.5</value>  
</property><property>  
<name>hbase.zookeeper.quorum</name>  
<value>192.168.2.103,192.168.2.104</value>  
<description>指定ZooKeeper集群</description>  
</property>

<property>  
<name>hbase.master.info.bindAddress</name>  
<value>0.0.0.0</value>  
<description>The bind address for the HBase Master web UI  
</description>  
</property>

</configuration>

hbase shell

对于hbase.master.info.bindAddress的配置需要注意，该项默认值是0.0.0.0，若改为某个结点的主机名或IP时，若在另外一个结点上使用start-hbase.sh启动hbase会失败，原因是使用start-hbase.sh启动时，会将当前结点作为master,即在当前结点上启动master服务，但如果hbase.master.info.bindAddress是另外一个结点，那么另外一个主机的地址是肯定无法bind到当前主机上的，所以HMaster服务就起不来了。

1. hbase的启动命令

hbase-daemon.sh start master //启动master

start-hbase.sh //启动hbase，不过该命令貌似包含了启动master的内容

hbase-daemon.sh stop master //关闭master

stop-hbase.sh //关闭hbase，不过该命令貌似包含了关闭master的内容

hbase shell //shell命令行接口，进入后可使用list显示

同时，可以通过：http://192.168.2.102:60010 查看hbase允许状态，也可通过其判断是否成功启动。

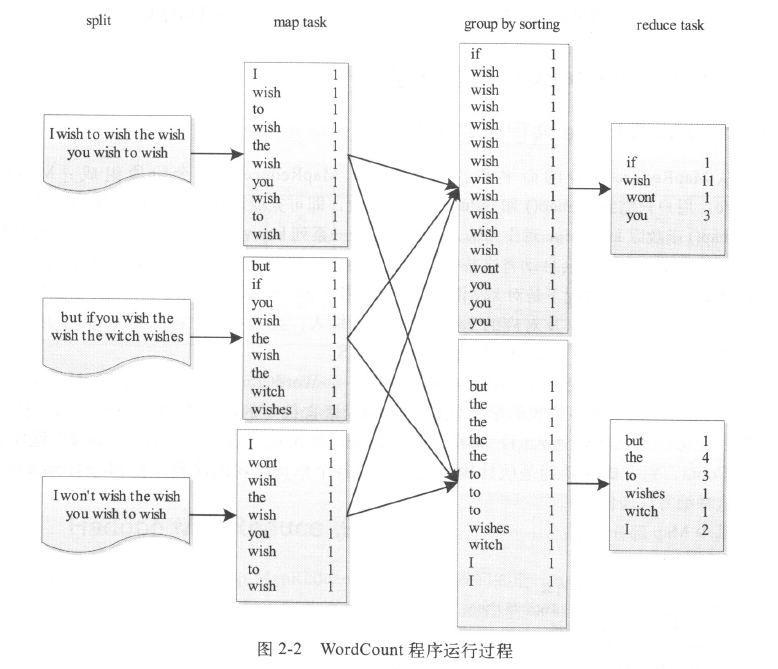
对于zookkeeper和hbase的启动、关闭顺序如下：

启动顺序：hadoop-->zookeeper-->hbase

关闭顺序：hbase-->zookeeper-->hadoop

zookeeper需要在每个节点上手动开启，hbase只需要在HMaster节点上开启即可。

**基于MapReduce 的API编程**



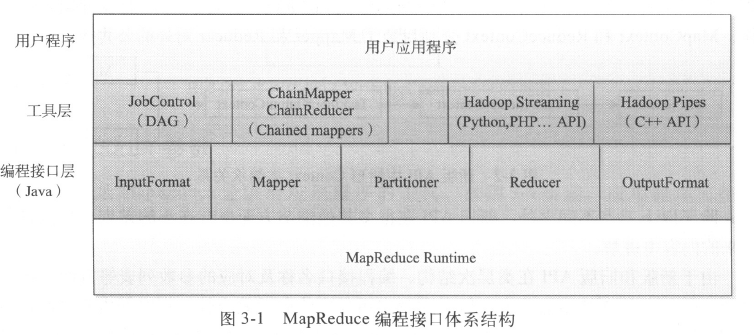
MapReduce对外提供了5个可编程组件，分别是InputFormat，Mapper，Partitioner，Reducer和OutputFormat，大部分情况用户只需要编写Mapper和Reducer即可，第二层是工具层，位于基本的Java API之上，主要是为了方便用户编写复杂的MapReducer程序，该层主要提供了4个编程工具包。

JobControl：方便用户编写有依赖关系的作业，如果一个作业需要有多个依赖性的MapReduce过程完成，则需要使用JobControl来实现。

ChainMapper：方便用户编写链式作业，即在Mapper或者Reducer阶段存在多个Mapper，但是Reducer必须只有一个。

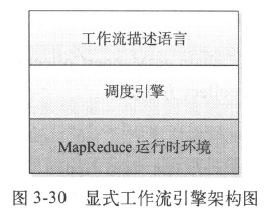
Hadoop Streaming：方便用户采用非Java语言编写作业，允许用户指定可执行文件或者脚本作为Mapper Reducer。

Hadoop Pipes:专门为C/C++用户编写MapReducer程序提供的工具包。



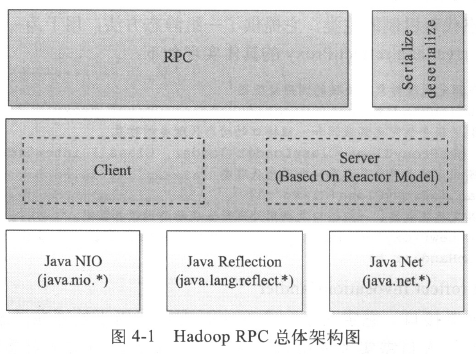
Hadoop还提供了很多的开源工作流引擎，主要可以概括为两类：隐式工作流和显示工作流：

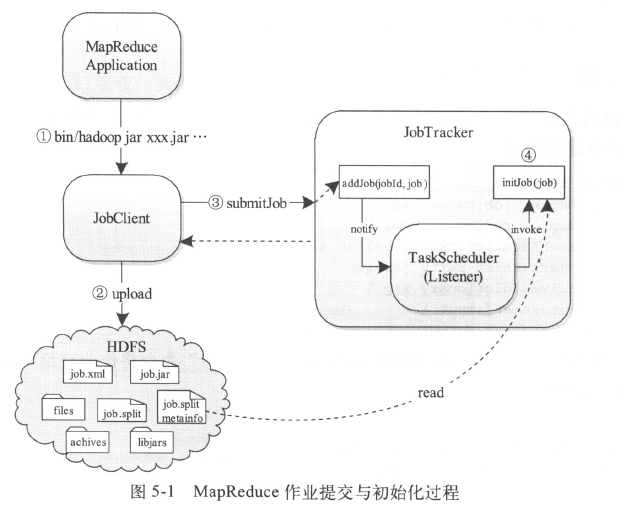




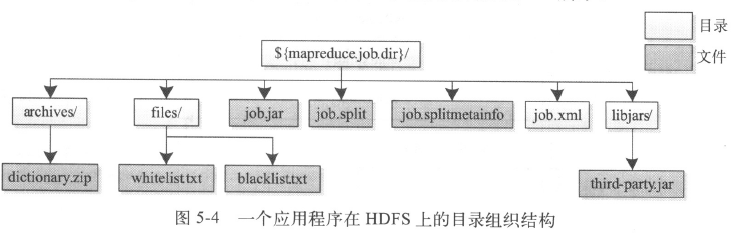
Hadoop中的xml配置文件中主要有3个配置参数，name、value、和description，如果管理员不想让用户程序修改某些属性的属性值，可以将该属性值的final参数设置为true，即<final>true</final>

RPC(Remote Procedure Call):远程过程调用时一种常用的分布式网络通信协议，是分布式系统的基础，Hadoop的RPC主要分为四个部分，分别是序列化层、函数调用层、网络传输层和服务器端处理框架。RPC主要使用了Java中的三个java包（java.lang.reflect, java.net, java.nio）

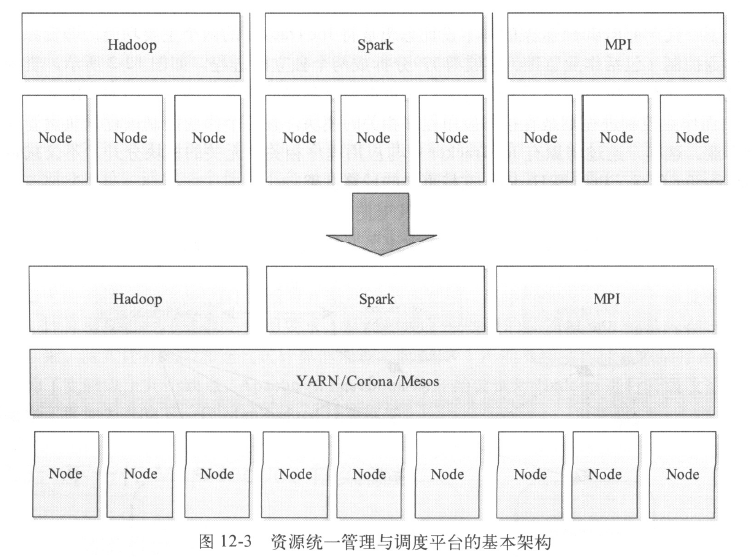




hadoop提交作业的三种方式：Java API、Hadoop Streaming和Hadoop Pipes，后面两种方式底层其实也是调用了Java API



随着互联网的高速发展，基于数据密集型应用的计算机框架不断出现，从支持离线处理的MapReduce到支持在线处理的Storm，从迭代式计算框架Spark到流式处理框架S4，各种框架都解决了某一类的问题，比如在搜索引擎公司中，一种可能的技术方案如下：网页建索引采用MapReduce，自然语言处理采用Spark，对性能要求很好的数据挖掘算法是用MPI。考虑到资源的利用率、运维成本等因素，公司一般希望将所有这些框架部署到一个公共的集群中，让它们资源共享，并对资源进行统一使用，这样便诞生了资源统一管理与调度平台，典型代表是YARN（Yet Another Resource Negotiator），facebook的Corona和Berkeley的Mesos。



YARN是下一代MapReduce框架，它的基本设计思想是将JobTracker拆分成两个独立的服务，一个全局的资源管理器ResourceManager和每个应用程序特有的ApplicationMaster，其中ResourceManager负责整个系统的资源管理和分配，而ApplicationMaster则负责单个应用程序的管理。

hadoop下常用命令

|  |  |
| --- | --- |
| 命令 | 完成任务 |
| hadoop dfs -get | 从hadoop文件系统中取回文件 |
| hadoop dfs -put | 将本地文件上传到hadoop系统中 |
| start-all.sh | 启动hadoop |
| hadoop namenode -format | 格式化namenode |
| hadoop dfsadmin –safemode enter | 进入安全模式 |
| hadoop dfsadmin –safemode leave | 退出安全模式 |
| hadoop-daemon.sh start datanode | 单独启动datanode |
| start-balancer.sh | 进行数据负载均衡 |
| start-dfs.sh | 单独启动dfs |
| start-mapred.sh | 单独启动(2.2.0中改为mapred start-yarn.sh) |
| hadoop-daemon.sh start datanode | 在datanode上运行这条命令，可以启动本地的datanode，不需整个系统的重启。 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |