# Hadoop核心子项目：

1. Core：一套分布式文件系统以及支持Map-Reduce的计算框架
2. HDFS：分布式文件系统，解决大数据存储问题
3. MapReduce是处理大量半结构化数据集合的编程模型（map:任务的分解；reduce：结果的总结）
4. Hive数据仓储工具
5. Hbase分布式数据库

虚拟系统软件Docker

# hadoop1.2

## hdfs

<http://node1:50070>

### 配置

|  |
| --- |
| 配置NameNode：conf/core-site.xml:  <configuration>  <property>  <name>fs.default.name</name>  <value>hdfs://localhost:9000</value>  </property>  <property>  <name>hadoop.tmp.dir</name>  <value>/opt/hadoop-1.2</value>  </property>  </configuration> |
| 配置副本数：conf/hdfs-site.xml:  <configuration>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  </configuration>  配置副本机器：slaves  node2  node3 |
| 配置SecondNode：conf/masters  node2 |
| 取消权限  <property> <name>dfs.permissions</name> <value>false</value> <description> If "true", enable permission checking in HDFS. If "false", permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories. </description> </property> |

### 常用命令

|  |
| --- |
| 格式化dfs  ./hadoop namenode -format |
| 启动dfs  ./start-dfs.sh |
| 停止dfs  ./stop-dfs.sh |
| 配置JAVA\_HOME  vi hadoop-env.sh |

关闭防火墙

|  |
| --- |
| systemctl stop firewalld |

访问测试：

|  |
| --- |
| 创建文件夹  hdfs dfs -mkdir -p /usr/input/hot  删除文件夹  hdfs dfs -rm -r /usr/output |
| 上传文件  hdfs dfs -put ~/data /usr/input/hot  下载文件  hdfs dfs -get /usr/output |
| 执行job  hadoop jar ~/hot.jar com.bjsxt.mr.RunJob |
| 查看job操作  ./hadoop job -help |

### 常见问题

|  |
| --- |
| namenode没启动  可能是hadoop.tmp.dir对应目录没创建，创建目录后再自行格式化 |

## MapReduce

### 配置MapReduce1

|  |
| --- |
| conf/mapred-site.xml  <configuration>  <property>  <name>mapred.job.tracker</name>  <value>node3:9001</value>  </property>  </configuration> |
| 启动  配置的node3，就要在node3上执行  ./start-mapred.sh  访问：<http://node3:50030> |
| 停止  ./stop-mapred.sh |

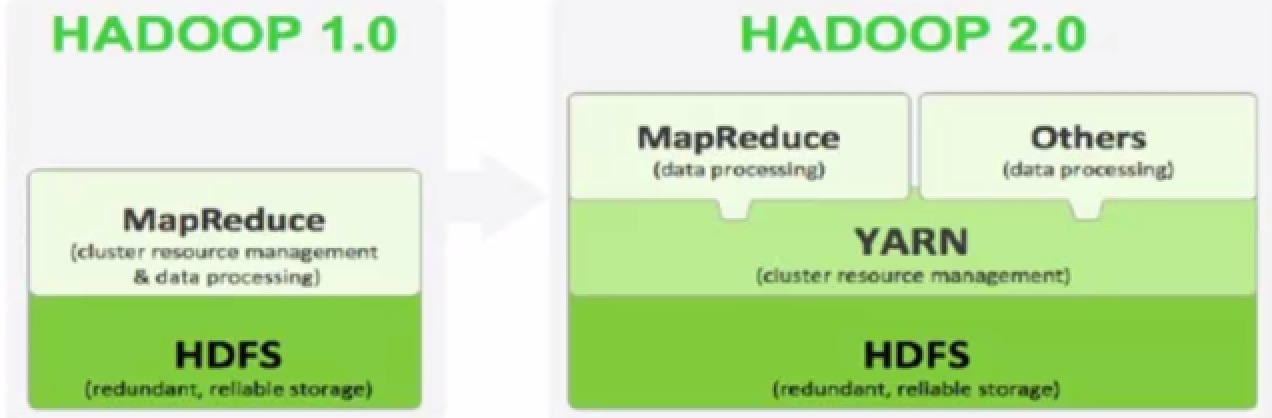
### 配置MapReduce2

|  |
| --- |
| mapred-site.xml  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property> |
| yarn-site.xml  <property>  <name>yarn.resourcemanager.hostname</name>  <value>node1</value>  </property>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>  <value>org.apache.hadoop.mapred.ShuffleHandler</value>  </property> |
| 启动命令：start-yarn.sh  访问：<http://node1:8088>  或<https://node1:8090> |

### 算法

|  |
| --- |
| 关注度权重公式：  W = TF \* Log (N/DF)  TF：当前关键字在该条微博中出现的次数  DF：当前关键字在所有微博中出现的微博条数，一条微博记为1条，不管出现多少次。  N：微博的总条数 |

# hadoop2.x



HDFS：NNFederation、 HA；

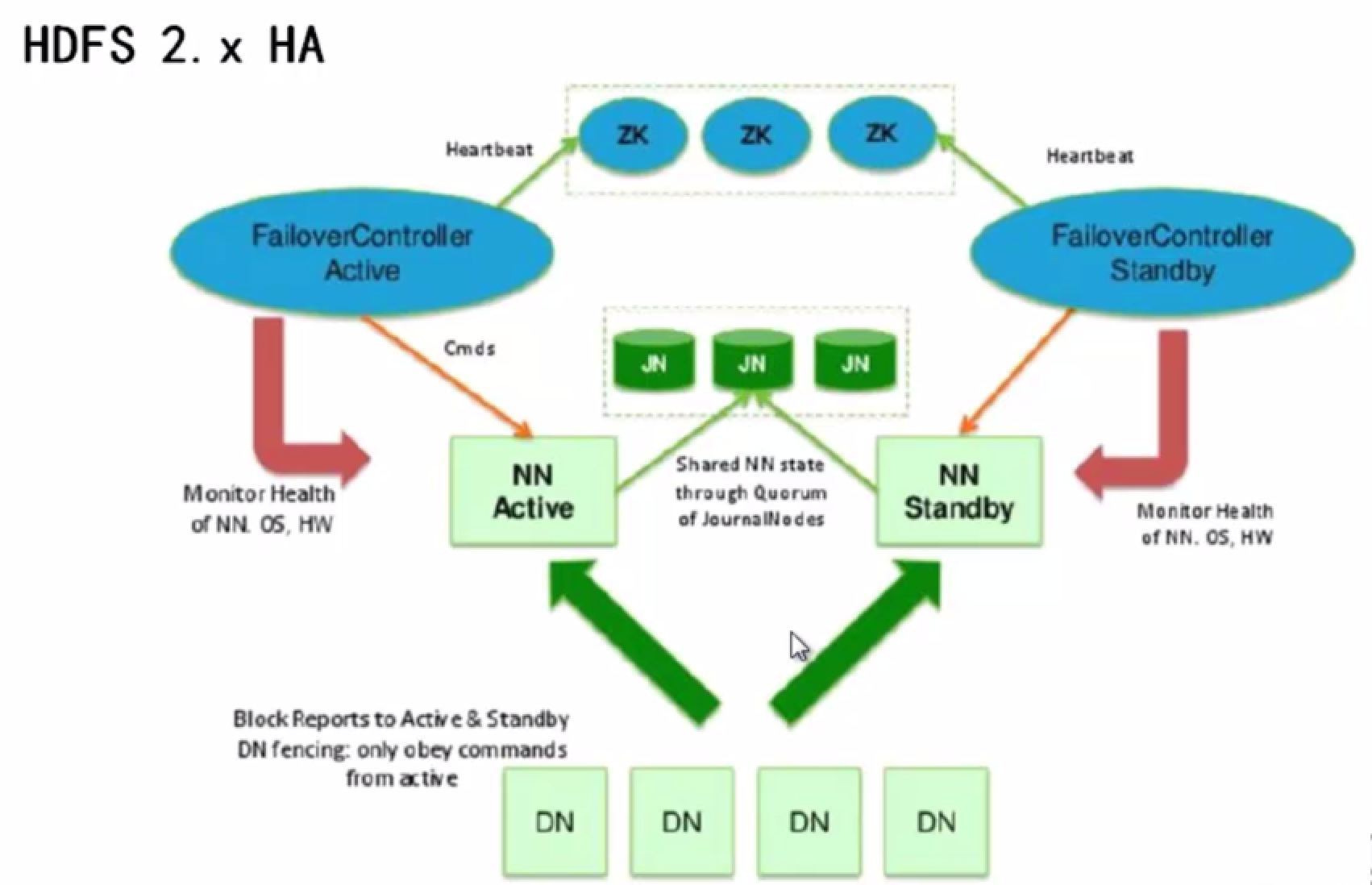
YARN：资源管理系统，从MRv1演化而来，支持多种运算框架。

核心思想：将MRv1中JobTracker的资源管理和任务调度两个功能分开，分别由ResourcesManager和ApplicationManager进程实现

ResourcesManager：负责整个集群的资源管理和调度，一个系统中只有一个。

ApplicationManager：负责应用程序相关事务，比如任务调度，任务监控和容错，每个应用程序应用对应一个ApplicationManager

解决hadoop1的单点故障：



1、HDFS HA ：通过主备NameNode解决，如果主NameNode故障，则切换到备NameNode上，解决内存受限问题：

JN Node是存放NameNode源数据的集群，每个NameNode分管一部分目录

Zookeeper FailoveController对NameNode进行心跳检查，控制NameNode切换的服务，同时也对ZooKeeper进行心跳检查

客户端访问zookeeper，由zookeeper返回正在工作的NameNode给客户端，客户端再访问NameNode。

HDFS 2.X Federation：解决nameNode过大问题，一般通过加内存解决。

MapReduce On YARN:MRv2

将MapReduce作业直接运行在YARN上，而不是由JobTracker和TaskTracker构建的MRv1系统中

基本功能模块：

YARA：负责资源管理和调度

MRAppMaster：负责任务切分、任务调度、任务监控和容错等；

MapTask/ReduceTask：任务驱动引擎，与MRv1一致。

每个MapReduce作业对应一个MRAppMaster

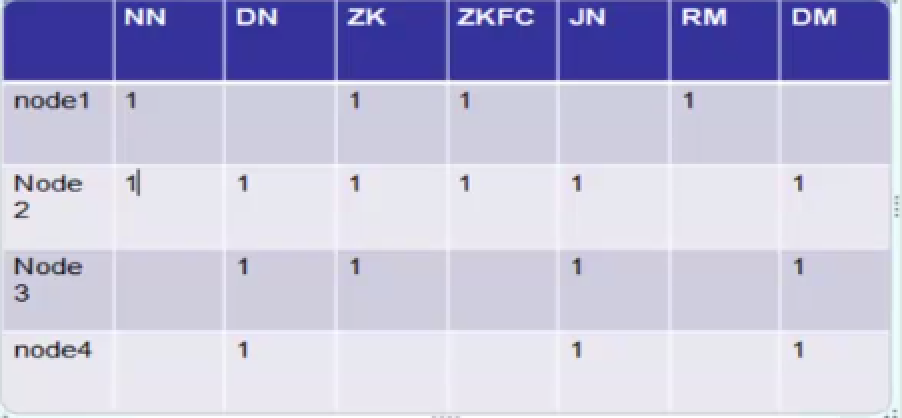
MRAppMaster任务调度

YARN将资源分配给MRAppMaster

MRAppMaster进一步将资源分配给内部的任务

MRAppMaster容器失败后，由YARN重新启动，MRAppMaster重新申请资源

## 安装



|  |
| --- |
| 修改hadoop-env.sh  添加JAVA\_HOME配置 |
| hdfs-site.xml  <property>  <name>dfs.nameservices</name>  <value>bjsxt</value>  </property>  <property>  <name>dfs.ha.namenodes.bjsxt</name>  <value>nn1,nn2</value>  </property>  <property>  <name>dfs.namenode.rpc-address.bjsxt.nn1</name>  <value>node1:8020</value>  </property>  <property>  <name>dfs.namenode.rpc-address.bjsxt.nn2</name>  <value>node2:8020</value>  </property>      <property>  <name>dfs.namenode.http-address.bjsxt.nn1</name>  <value>node1:50070</value>  </property>  <property>  <name>dfs.namenode.http-address.bjsxt.nn2</name>  <value>node2:50070</value>  </property>  <property>  <name>dfs.namenode.shared.edits.dir</name>  <value>qjournal://node2:8485;node3:8485;node4:8485/bjsxt</value>  </property>  <property>  <name>dfs.client.failover.proxy.provider.bjsxt</name>  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>  </property>      <property>  <name>dfs.ha.fencing.methods</name>  <value>sshfence</value>  </property>    <property>  <name>dfs.ha.fencing.ssh.private-key-files</name>  <value>/root/.ssh/id\_dsa</value>  </property>    <property>  <name>dfs.journalnode.edits.dir</name>  <value>/opt/jn/data</value>  </property>    <property>  <name>dfs.ha.automatic-failover.enabled</name>  <value>true</value>  </property> |
| core-site.xml  <property>  <name>fs.defaultFS</name>  <value>hdfs://bjsxt</value>  </property>    <property>  <name>ha.zookeeper.quorum</name>  <value>node1:2181,node2:2181,node3:2181</value>  </property>    <property>  <name>hadoop.tmp.dir</name>  <value>/opt/hadoop2</value>  </property> |
| vi slaves  node2  node3  node4 |

## 常用命令

|  |
| --- |
| 1.  在node1上格式化namenode：bin/hadoop namenode –format  在node1上格式化ZKFC：./hdfs zkfc -formatZK |
| 2. 启动datanode1：  /home/hadoop-2.6/sbin/hadoop-daemon.sh start namenode  或  ./hbase-deaemon.sh start regionserver |
| 3. 在未格式化的namenode自行命令拷贝初始化文件  ./hdfs namenode –bootstrapStandby |
| 4. 停掉所有（不包含ZK）  ./stop-dfs.sh |

# 安装zookeeper

最好是3、5、7奇数个，偶数个是不行的

|  |
| --- |
| 进入conf下，  cp zoo\_sample.cfg zoo.cfg  vi zoo.cfg  修改dataDir=/opt/zookeeper  添加以下配置  server.1=node1:2888:3888  server.2=node2:2888:3888  server.3=node3:2888:3888 |
| 创建文件/opt/zookeeper/myid  node1机器值为1  node2机器值为2  node3机器值为3 |
| 添加zookeeper的bin目录到path环境变量 |
| 启动zookeeper  zkServer.sh start |

# journalnode

|  |
| --- |
| 启动  /home/hadoop-2.6/sbin/hadoop-daemon.sh start journalnode |

# hbase

<http://node1:60010>

|  |
| --- |
| hbase-site.xml  <configuration>  <property>  <name>hbase.rootdir</name>  <value>file:///opt/hbase</value>  <!-- <value>file://bjsxt/hbase</value> -->  </property>  <property>  <name>hbase.zookeeper.property.dataDir</name>  <value>/opt/zookeeper</value>  </property>  <property>  <name>hbase.cluster.distributed</name>  <value>true</value>  </property>  </configuration> |
| hbase-env.sh  export HBASE\_MANAGES\_ZK=false |

## hbase迭代

|  |
| --- |
| ResultScanner results = HbaseClient.*getInstance*().scan("CDC\_VEHICLE\_SCORE");  **for** (Result result : results) {  String rowKey = **new** String(result.getRow(),"utf-8");  System.***out***.println("获得到rowkey:" + rowKey);  **for** (KeyValue keyValue : result.~~raw~~()) {  System.***out***.println("列：" + **new** String(keyValue.~~getFamily~~()) + "====值:" + **new** String(keyValue.~~getValue~~()));  }  // logger.info(result);  } |

|  |  |  |
| --- | --- | --- |
| 操作 | 命令表达式 | 注意 |
| 创建表 | create 'table\_name, 'family1','family2','familyN' |  |
| 添加记录 | put 'table\_name', 'rowkey', 'family:column', 'value' |  |
| 查看记录 | get 'table\_name', 'rowkey' | 查询单条记录，也是hbase 最常用的命令 |
| 查看表中的记录总数 | count  'table\_name' | 这个命令并不快，且目前没有找到更快的方式统计行数 |
| 删除记录 | delete 'table\_name' ,'rowkey','family\_name:column' deleteall 'table\_name','rowkey' | 第一种方式删除一条记录单列的数据 第二种方式删除整条记录 |
|  |
| 删除一张表 | 1、disable 'table\_name' |  |  |
| 2、drop 'table\_name' |  |
| 查看所有记录 | scan "table\_name" ,{LIMIT=>10} | LIMIT=>10 只返回10条记录，否则将全部展示 |  |

# 启动

## 第一次启动顺序

分别在三个机器上启动zookeeper：./zkServer.sh start

分别在三个机器上启动JournalNode: /home/hadoop-2.6/sbin/hadoop-daemon.sh start journalnode

在其中一个namenode上格式化: ./hadoop namenode -format，并启动他/home/hadoop-2.6/sbin/hadoop-daemon.sh start namenode，并在另一个namenode上拷贝./hdfs namenode –bootstrapStandby，启动另一个namenode: /home/hadoop-2.6/sbin/hadoop-daemon.sh start namenode

在其中一个namenode上格式化zkfc ./hdfs zkfc -formatZK

停止上面节点：stop-dfs.sh

全面启动：start-dfs.sh

## 非第一次启动顺序

分别在三个机器上启动zookeeper：zkServer.sh start 对应进程QuorumPeerMain

分别在两个机器上启动zkfc /home/hadoop-2.6/sbin/hadoop-daemon.sh start zkfc 对应进程DFSZKFailoverController

在任意一台机器上启动HDFS：/home/hadoop-2.6/sbin/start-dfs.sh 对应进程NameNode/JournalNode

在需要的机器上启动hbase：/home/hbase/bin/start-hbase.sh 对应进程HMaster/HRegionServer

常见端口

|  |  |  |
| --- | --- | --- |
| **60020** | regionserver | hbase |
| 2181 |  |  |
| 8485 |  |  |
| 8020 |  |  |

## 启动cdc-parent:cloud\_start.sh

~/shell/zookeeper\_start.sh

~/shell/rabbitMQ\_start.sh

~/software/hadoop-2.6.0/sbin/start-dfs.sh

~/software/hadoop-2.6.0/sbin/start-yarn.sh

~/software/hbase-1.1.2/bin/start-hbase.sh

# hbase count

$HBASE\_HOME/bin/hbase org.apache.hadoop.hbase.mapreduce.RowCounter ‘tablename’