## HBase安装操作

### 上传解压HBase安装包

|  |
| --- |
| tar -xvzf hbase-2.1.0.tar.gz -C ../server/ |

### 修改HBase配置文件

#### hbase-env.sh

|  |
| --- |
| cd /export/server/hbase-2.1.0/conf  vim hbase-env.sh  # 第28行  export JAVA\_HOME=/export/server/jdk1.8.0\_241/  export HBASE\_MANAGES\_ZK=false |

#### hbase-site.xml

|  |
| --- |
| vim hbase-site.xml  ------------------------------  <configuration>  <!-- HBase数据在HDFS中的存放的路径 -->  <property>  <name>hbase.rootdir</name>  <value>hdfs://node1:8020/hbase</value>  </property>  <!-- Hbase的运行模式。false是单机模式，true是分布式模式。若为false,Hbase和Zookeeper会运行在同一个JVM里面 -->  <property>  <name>hbase.cluster.distributed</name>  <value>true</value>  </property>  <!-- ZooKeeper的地址 -->  <property>  <name>hbase.zookeeper.quorum</name>  <value>node1,node2,node3</value>  </property>  <!-- ZooKeeper快照的存储位置 -->  <property>  <name>hbase.zookeeper.property.dataDir</name>  <value>/export/server/zookeeper-3.4.6/data</value>  </property>  <!-- V2.1版本，在分布式情况下, 设置为false -->  <property>  <name>hbase.unsafe.stream.capability.enforce</name>  <value>false</value>  </property>  </configuration> |

#### 配置环境变量

|  |
| --- |
| # 配置Hbase环境变量  vim /etc/profile  export HBASE\_HOME=/export/server/hbase-2.1.0  export PATH=$PATH:${HBASE\_HOME}/bin:${HBASE\_HOME}/sbin  #加载环境变量  source /etc/profile |

#### 复制jar包到lib

|  |
| --- |
| cp $HBASE\_HOME/lib/client-facing-thirdparty/htrace-core-3.1.0-incubating.jar $HBASE\_HOME/lib/ |

#### 修改regionservers文件

|  |
| --- |
| cd /export/server/hbase-2.1.0/conf  vim regionservers  node1  node2  node3 |

#### 分发安装包与配置文件

|  |
| --- |
| cd /export/server  scp -r hbase-2.1.0/ node2:$PWD  scp -r hbase-2.1.0/ node3:$PWD  在node2和node3配置加载环境变量  source /etc/profile |

#### 启动HBase

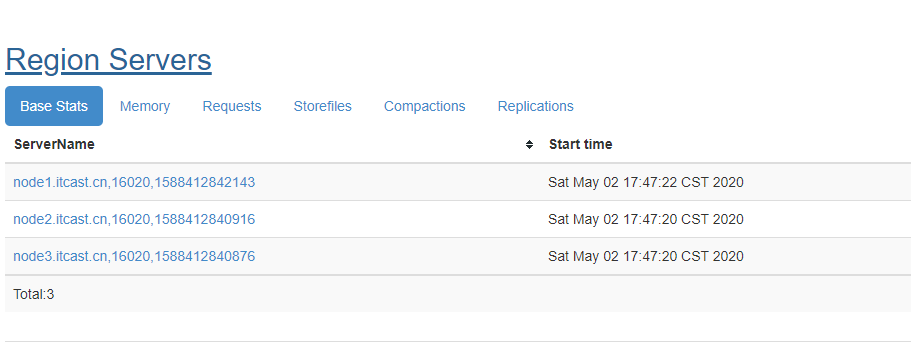
|  |
| --- |
| cd /export/server  # 启动ZK  ./start-zk.sh zkServer.sh 单个启动每个节点的zk; 或者使用自己的一键启动zookeeper脚本  # 启动hadoop  start-dfs.sh / start-all.sh  # 启动hbase  start-hbase.sh |

#### 验证Hbase是否启动成功

|  |
| --- |
| # 启动hbase shell客户端  hbase shell  # 输入status  [root@node1 onekey]# **hbase shell**  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/export/server/hadoop-2.7.5/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/export/server/hbase-2.1.0/lib/client-facing-thirdparty/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  HBase Shell  Use "help" to get list of supported commands.  Use "exit" to quit this interactive shell.  Version 2.1.0, re1673bb0bbfea21d6e5dba73e013b09b8b49b89b, Tue Jul 10 17:26:48 CST 2018  Took 0.0034 seconds  Ignoring executable-hooks-1.6.0 because its extensions are not built. Try: gem pristine executable-hooks --version 1.6.0  Ignoring gem-wrappers-1.4.0 because its extensions are not built. Try: gem pristine gem-wrappers --version 1.4.0  2.4.1 :001 > **status**  1 active master, 0 backup masters, 3 servers, 0 dead, 0.6667 average load  Took 0.4562 seconds  2.4.1 :002 > |

### WebUI

[http://node1:16010/master-status](http://node1.itcast.cn:16010/master-status)



### 安装目录说明

|  |  |
| --- | --- |
| 目录名 | 说明 |
| bin | 所有hbase相关的命令都在该目录存放 |
| conf | 所有的hbase配置文件 |
| hbase-webapps | hbase的web ui程序位置 |
| lib | hbase依赖的java库 |
| logs | hbase的日志文件 |

### 参考硬件配置

针对大概800TB存储空间的集群中每个Java进程的典型内存配置：

|  |  |  |
| --- | --- | --- |
| 进程 | 堆 | 描述 |
| NameNode | 8 GB | 每100TB数据或每100W个文件大约占用NameNode堆1GB的内存 |
| SecondaryNameNode | 8GB | 在内存中重做主NameNode的EditLog，因此配置需要与NameNode一样 |
| DataNode | 1GB | 适度即可 |
| ResourceManager | 4GB | 适度即可（注意此处是MapReduce的推荐配置） |
| NodeManager | 2GB | 适当即可（注意此处是MapReduce的推荐配置） |
| HBase HMaster | 4GB | 轻量级负载，适当即可 |
| HBase RegionServer | 12GB | 大部分可用内存、同时为操作系统缓存、任务进程留下足够的空间 |
| ZooKeeper | 1GB | 适度 |

推荐：

* Master机器要运行NameNode、ResourceManager、以及HBase HMaster，推荐24GB左右
* Slave机器需要运行DataNode、NodeManager和HBase RegionServer，推荐24GB（及以上）
* 根据CPU的核数来选择在某个节点上运行的进程数，例如：两个4核CPU=8核，每个Java进程都可以独立占有一个核（推荐：8核CPU）
* 内存不是越多越好，在使用过程中会产生较多碎片，Java堆内存越大， 会导致整理内存需要耗费的时间越大。例如：给RegionServer的堆内存设置为64GB就不是很好的选择，一旦FullGC就会造成较长时间的等待，而等待较长，Master可能就认为该节点已经挂了，然后移除掉该节点