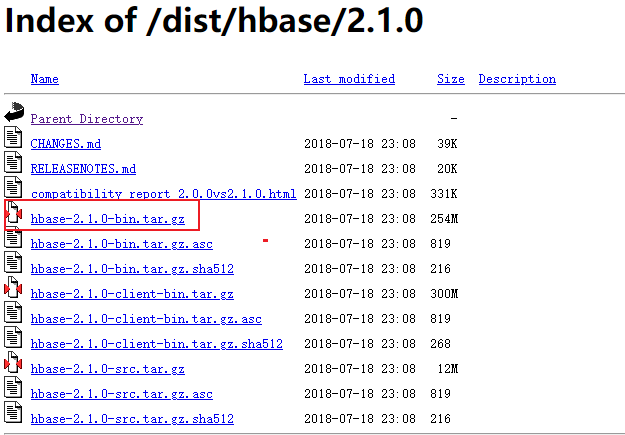
## HBase安装操作

### 上传解压HBase安装包

下载地址:

<https://archive.apache.org/dist/hbase/2.1.0/>



上传并解压:

|  |
| --- |
| cd /export/software  rz上传  解压操作:  cd /export/software  tar -xvzf hbase-2.1.0.tar.gz -C ../server/  创建软连接  cd /export/server/  ln -s hbase-2.1.0/ hbase |

### 修改HBase配置文件

#### hbase-env.sh

|  |
| --- |
| cd /export/server/hbase/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.itcast.cn: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.itcast.cn,node2.itcast.cn,node3.itcast.cn</value>  </property>  <!-- ZooKeeper快照的存储位置 -->  <property>  <name>hbase.zookeeper.property.dataDir</name>  <value>/export/server/zookeeper/zkdatas/</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  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/conf  vim regionservers  node1.itcast.cn  node2.itcast.cn  node3.itcast.cn |

#### 分发安装包与配置文件

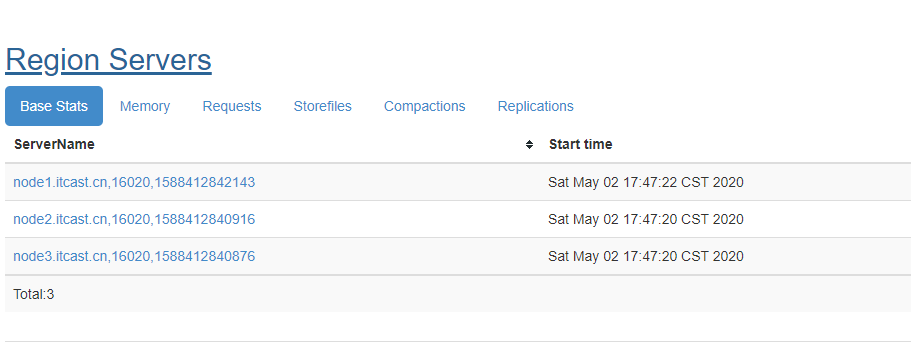
|  |
| --- |
| cd /export/server  scp -r hbase-2.1.0/ node2.itcast.cn:$PWD  scp -r hbase-2.1.0/ node3.itcast.cn:$PWD  在node2和node3创建软链接  cd /export/server  ln -s hbase-2.1.0/ hbase  在node2.itcast.cn和node3.itcast.cn配置加载环境变量  source /etc/profile |

#### 启动HBase

|  |
| --- |
| cd /export/server  # 启动ZK (只是告诉大家需要启动zookeeper, 原来该怎么启动 还怎么启动)  ./start-zk.sh  # 启动hadoop  start-all.sh  # 启动hbase  start-hbase.sh |

#### 验证Hbase是否启动成功

<http://node1.itcast.cn:16010/master-status>

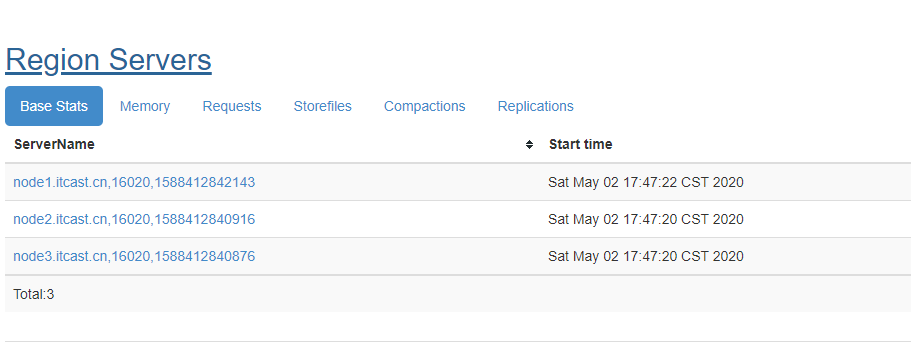


或者:

|  |
| --- |
| # 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.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

## 高可用搭建

Hbase的高可用指的就是Master的高可用, 其regionServer本身就是高可用

### 创建backup-masters文件

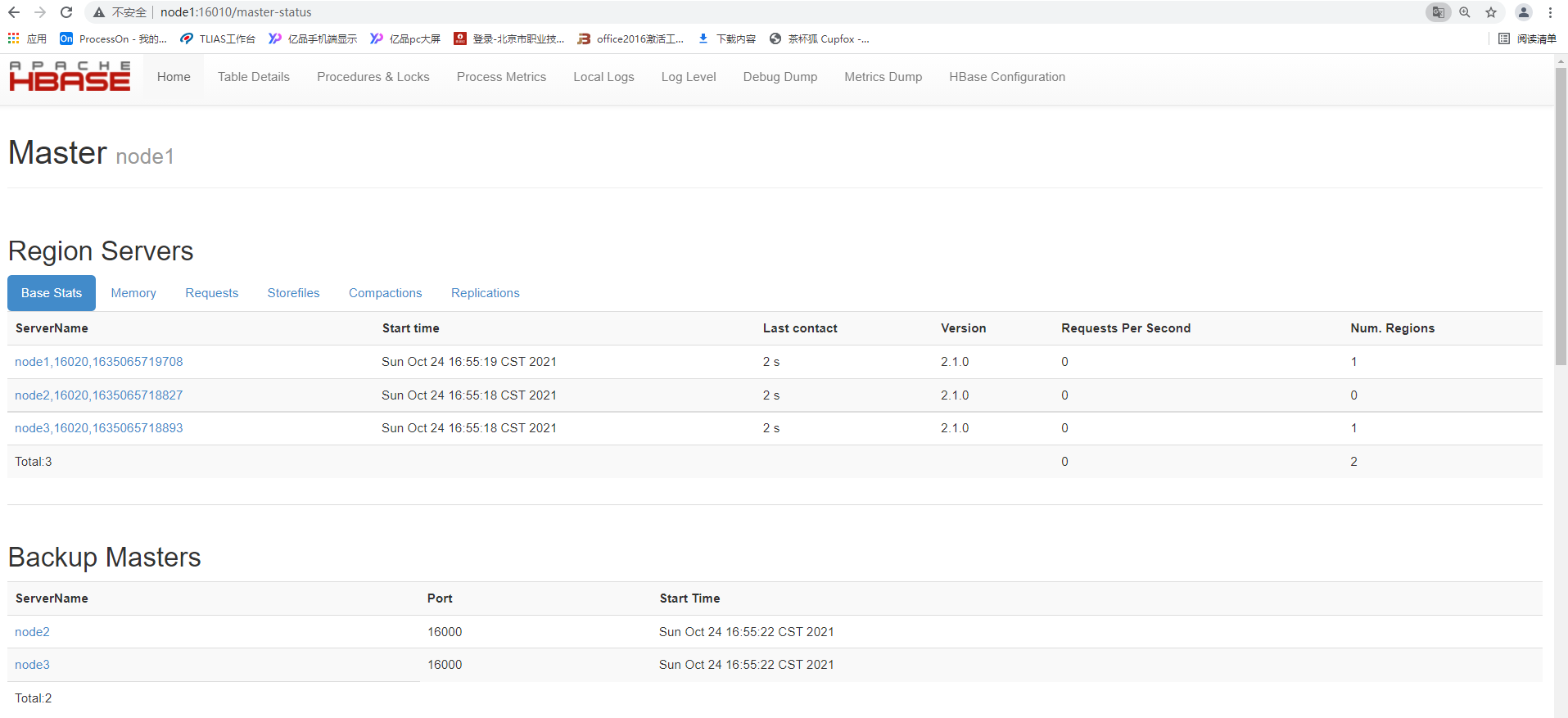
此文件用于标识将那些节点作为hbase的master的备份节点, 仅需要在node1配置即可

|  |
| --- |
| cd /export/server/hbase/conf  vim backup-masters  添加以下内容:  node2.itcast.cn  node3.itcast.cn |

### 重启hbase即可

|  |
| --- |
| stop-hbase.sh  start-hbase.sh |

查看webui，检查Backup Masters中是否有node2.itcast.cn、node3.itcast.cn



## Hbase和hive的集成

### 拷贝hive提供的base通信包

* 1) 将hive提供的一个和hbase集成的通信包放置到hbase的lib目录下

|  |
| --- |
| cd /export/server/hive/lib  cp hive-hbase-handler-3.1.2.jar /export/server/hbase/lib/ |

* 2) 分发给node2和node3

|  |
| --- |
| cd /export/server/hbase/lib  scp -r hive-hbase-handler-3.1.2.jar node2:$PWD  scp -r hive-hbase-handler-3.1.2.jar node3:$PWD |

### 修改 hive-site.xml

|  |
| --- |
| cd /export/server/hive/conf  vim hive-site.xml  输入 i 进入 插入模式  插入以下内容:  <property>  <name>hive.zookeeper.quorum</name>  <value>node1.itcast.cn,node2.itcast.cn,node3.itcast.cn</value>  </property>  <property>  <name>hbase.zookeeper.quorum</name>  <value>node1.itcast.cn,node2.itcast.cn,node3.itcast.cn</value>  </property>  <property>  <name>hive.server2.enable.doAs</name>  <value>false</value>  </property> |

发送给node2和node3

|  |
| --- |
| cd /export/server/hive/conf  scp -r hive-site.xml node2:$PWD  scp -r hive-site.xml node3:$PWD |

### 修改 hive-env.sh

|  |
| --- |
| cd /export/server/hive/conf  vim hive-env.sh  输入i 进入插入模式  添加以下内容:  export HBASE\_HOME=/export/server/hbase/ |

发送给node2和node3

|  |
| --- |
| cd /export/server/hive/conf  scp -r hive-env.sh node2:$PWD  scp -r hive-env.sh node3:$PWD |

### 重启hbase和hive

|  |
| --- |
| 这里省略前置启动zookeeper和hadoop的操作  启动hive:  cd /export/server/hive/bin  nohup ./hive --service metastore &  nohup ./hive --service hiveserver2 &  启动hbase:  start-hbase.sh |

### 测试

* 1) 在hbase中创建一张表, 并添加相关的数据

|  |
| --- |
| create 'hive\_hbase\_score','C1'  put 'hive\_hbase\_score','rk001','C1:name','张三'  put 'hive\_hbase\_score','rk001','C1:age','20'  put 'hive\_hbase\_score','rk001','C1:score','80.5'  put 'hive\_hbase\_score','rk002','C1:name','李四'  put 'hive\_hbase\_score','rk002','C1:age','15'  put 'hive\_hbase\_score','rk002','C1:score','95.2'  put 'hive\_hbase\_score','rk003','C1:name','王五'  put 'hive\_hbase\_score','rk003','C1:age','18'  put 'hive\_hbase\_score','rk003','C1:score','89'  put 'hive\_hbase\_score','rk004','C1:name','赵六'  put 'hive\_hbase\_score','rk004','C1:age','20'  put 'hive\_hbase\_score','rk004','C1:score','50' |

* 2) 在hive创建外部表与hbase进行集成操作

|  |
| --- |
| 格式:  create external table 表名 (  字段1 类型,  字段2 类型,  字段3 类型  ......  ) stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties ("hbase.columns.mapping"=":key,列族:列名,列族:列名....") tblproperties("hbase.table.name"="hbase表名");  注意:  1) 在构建hive表的时候, 理论上 hive的表名和字段名是可以任意的, 但是建议与要映射hbase表保持一致  2) hbase.columns.mapping 设置 hbase中列与 hive中列进行一一映射匹配, 第一个匹配第一个, 第二个匹配第二个,以此类推  3) hbase.table.name 设置当前hive表映射hbase的那个表    示例:  create database day03\_hivetohbase;  use day03\_hivetohbase;  create external table hive\_hbase\_score (  id string,  name string,  age string,  score string  ) stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties("hbase.columns.mapping"=":key,C1:name,C1:age,C1:score") tblproperties("hbase.table.name"="hive\_hbase\_score");  select \* from hive\_hbase\_score; |

