hadoop hdfs hbase 安装

一. Hadoop 概述

Hadoop HA 即 Hadoop 的高可用，不同于普通的 namenode+second namenode 的模式，second namenode 只能作为 namenode 的冷备份，当 namenode 挂掉后，second namenode 不能自动充当 namenode 的角色。所以为了保证高可用性，就有了 Hadoop HA 的机制。

hdfs HA 中即有至少有两个 namenode，这儿以两个 namenode 为例，一个可以对外提供服务的 namenode 成为 active namenode，待命状态的 namenode 成为 standby namenode。

1.如何做到两个 namenode 的元数据的一致性呢？

将 edits 文件托管给稳定的第三方存放（当然，为了防止第三方挂掉，namenode 本地的 edits 文件也是必要的），standby namenode 就可以定时去第三方取 edits 文件放在内存中，做元素据的修改。这样就可以保证 actice 和 standby 的数据一致性。

2.第三方用什么才能保证高可用？

第三方用的是 qjournal 集群，集群就可以将 edits 文件存放在每个机器中。并且这个集群使用 zookeeper 作为集群的协调，保证数据写入一半以上的机器，才为写入成功。

3.怎么做到 standby 在 active 节点挂掉自动充当 active 的角色呢？

这儿在每个 namenode 对应着一个监控进程 ZKFC。在 active 上 ZKFC 使用 RPC 调

用以判断 active 是否挂掉，如果挂掉，那么在 zookeeper 上将相应的节点删除（分布式锁），在 Standby 节点上的 ZKFC 进程就可以感知到节点删除的信息。Standby 节点上的 ZKFC 进程感知到 active 节点挂掉的信息之后会向 active 节点发送一个杀死 hdfs namenode 的命令，确保 active 死亡，然后再向 standby 发送切换的指令，并向 zookeeper 创建节点。这样就可以让 standby 模式转换为 active 模式。

对于 resourceManager 的 HA,因为不存在数据的一致性,所以只需要向 zookeeper 创建节点（获得分布式锁）即可。

二．环境准备

前置需求

zookeeper 集群，Java 环境，关闭防火墙 ,hadoop 用户

1.节点角色搭配（这儿是 5 台节点）

Hadoop10：NameNode ZKFC

Hadoop11：NameNode ZKFC

Hadoop12：DataNode JournalNode

Hadoop13：DataNode JournalNode

Hadoop14：DataNode JournalNode

2.节点之间的免密登陆，以及 hosts 文件的相应映射配置好

3. 解压安装

```
 tar -zxvf -C hadoop-2.5.0-cdh5.3.0.tar.gz  -C /home/hadoop/

 cd /home/hadoop/hadoop-2.5.0-cdh5.3.0

 ls -al

 drwxr-xr-x. 17 hadoop tass  4096 Jul 26  2017 .

drwxr-xr-x. 14 hadoop tass  4096 Jul 17 14:43 ..

drwxr-xr-x.  2 hadoop tass  4096 Jul 11 19:05 bin

drwxr-xr-x.  3 hadoop tass  4096 Dec 17  2014 cloudera

drwxr-xr-x.  4 hadoop tass  4096 Jul  4  2017 dfs

drwxr-xr-x.  6 hadoop tass  4096 Dec 17  2014 etc

drwxr-xr-x.  5 hadoop tass  4096 Dec 17  2014 examples

drwxr-xr-x.  3 hadoop tass  4096 Dec 17  2014 examples-mapreduce1

-rw-r--r--.  1 hadoop tass 22626 Jul 26  2017 hadoop_etc.tar.gz

drwxr-xr-x.  2 hadoop tass  4096 Dec 17  2014 include

drwxr-xr-x.  3 hadoop tass  4096 Dec 17  2014 lib

drwxr-xr-x.  2 hadoop tass  4096 Dec 17  2014 libexec

drwxr-xr-x.  3 hadoop tass  4096 Jul 20 16:57 logs

drwxr-xr-x.  3 hadoop tass  4096 Dec 17  2014 sbin

drwxr-xr-x.  4 hadoop tass  4096 Dec 17  2014 share

drwxr-xr-x. 17 hadoop tass  4096 Dec 17  2014 src
```

drwxr-xr-x. 3 hadoop tass 4096 Aug 3 2017 tmp

cd etc/Hadoop

ls -al

此为主要 hadoop 的配置文件

-rw-r--r--. 1 hadoop tass 3589 Jul 4 2017 capacity-scheduler.xml

-rw-r--r--. 1 hadoop tass 1335 Jul 4 2017 configuration.xsl

-rw-r--r--. 1 hadoop tass   318 Jul 4 2017 container-executor.cfg

-rw-r--r--. 1 hadoop tass 1954 Jul 4 2017 core-site.xml

-rw-r--r--. 1 hadoop tass   803 Jul 4 2017 fairscheduler.xml

-rw-r--r--. 1 hadoop tass 3589 Jul 4 2017 hadoop-env.cmd

-rw-r--r--. 1 hadoop tass 3473 Jul 4 2017 hadoop-env.sh

-rw-r--r--. 1 hadoop tass 1774 Jul 4 2017 hadoop-metrics2.properties

-rw-r--r--. 1 hadoop tass 2490 Jul 4 2017 hadoop-metrics.properties

-rw-r--r--. 1 hadoop tass 9201 Jul 4 2017 hadoop-policy.xml

-rw-r--r--. 1 hadoop tass 4757 Jul 4 2017 hdfs-site.xml

-rw-r--r--. 1 hadoop tass 1449 Jul 4 2017 httpfs-env.sh

-rw-r--r--. 1 hadoop tass 1657 Jul 4 2017 httpfs-log4j.properties

-rw-r--r--. 1 hadoop tass   21 Jul 4 2017 httpfs-signature.secret

-rw-r--r--. 1 hadoop tass   620 Jul 4 2017 httpfs-site.xml

-rw-r--r--. 1 hadoop tass 3523 Jul 4 2017 kms-acls.xml

-rw-r--r--. 1 hadoop tass 1527 Jul 4 2017 kms-env.sh

-rw-r--r--. 1 hadoop tass 1631 Jul 4 2017 kms-log4j.properties

-rw-r--r--. 1 hadoop tass 5511 Jul 4 2017 kms-site.xml

-rw-r--r--. 1 hadoop tass 11291 Jul 4 2017 log4j.properties

-rw-r--r--. 1 hadoop tass   918 Jul 4 2017 mapred-env.cmd

-rw-r--r--. 1 hadoop tass 1383 Jul 4 2017 mapred-env.sh

-rw-r--r--. 1 hadoop tass 4113 Jul 4 2017 mapred-queues.xml.template

-rw-r--r--. 1 hadoop tass 1179 Jul 4 2017 mapred-site.xml

-rw-r--r--. 1 hadoop tass   758 Jul 4 2017 mapred-site.xml.template

-rw-r--r-- 1 hadoop tass   48 Jul 9 09:23 slaves

-rw-r--r--. 1 hadoop tass 2316 Jul 4 2017 ssl-client.xml.example

-rw-r--r--. 1 hadoop tass 2268 Jul 4 2017 ssl-server.xml.example

-rw-r--r--. 1 hadoop tass 2178 Jul 4 2017 yarn-env.cmd

-rw-r--r--. 1 hadoop tass 4728 Jul 4 2017 yarn-env.sh

-rw-r--r--. 1 hadoop tass 7490 Jul 4 2017 yarn-site.xml

三，配置文件及解析

core-site.xml

<!-- 这儿的 mycluster 表示两个 namenode 组建成的逻辑名字 -->

<property>

<name>fs.defaultFS</name>

```
<value>hdfs://mycluster/</value>
</property>
<!-- 指定 hadoop 临时目录 -->
<property>
<name>hadoop.tmp.dir</name>
<value> /home/hadoop/hadoop-2.5.0-cdh5.3.0/tmp </value>
</property>

<!-- 指定 zookeeper 地址 -->
<property>
<name>ha.zookeeper.quorum</name>
<value> kafkazoo1:2181,kafkazoo2:2181,kafkazoo3:2181</value>
</property>
<property>
  <name>io.file.buffer.size</name>
  <value>131072</value>
</property>

<property>
  <name>hadoop.proxyuser.hadoop.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>hadoop.proxyuser.hadoop.groups</name>
  <value>*</value>
</property>

<property>
  <name>hadoop.native.lib</name>
  <value>true</value>
</property>

<property>
   <name>ha.zookeeper.session-timeout.ms</name>
   <value>600000</value>
</property>

<property>
   <name>ha.failover-controller.cli-check.rpc-timeout.ms</name>
   <value>600000</value>
</property>

<property>
```

```
    <name>ipc.client.connect.timeout</name>
    <value>200000</value>
</property>
```

2.hdfs-site.xml

```
<configuration>
<!--指定 hdfs 的 nameservice 为 bi，需要和 core-site.xml 中的保持一致 -->
<property>
<name>dfs.nameservices</name>
<value>mycluster</value>
</property>
<!-- bi 下面有两个 NameNode，分别是 nn1，nn2 ,这儿的 nn1，nn2 也是逻辑名字，可以自己指定，但是指定后下面的也要随之改变-->
<property>
<name>dfs.ha.namenodes.bi</name>
<value>nn1,nn2</value>
</property>
<!-- nn1 的 RPC 通信地址 -->
<property>
<name>dfs.namenode.rpc-address.mycluster.nn1</name>
<value>hadoop10:9000</value>
</property>
<!-- nn1 的 http 通信地址 -->
```

```
<property>
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>hadoop10:50070</value>
</property>
<!-- nn2 的 RPC 通信地址 -->
<property>
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>hadoop11:9000</value>
</property>
<!-- nn2 的 http 通信地址 -->
<property>
<name>dfs.namenode.http-address.mycluster.nn2</name>
<value>hadoop11:50070</value>
</property>
<!-- 指定 NameNode 的 edits 元数据在 JournalNode 上的存放位置 -->
<property>
<name>dfs.namenode.shared.edits.dir</name>
<value>
qjournal://hadoop12:8485;hadoop13:8485;hadoop14:8485/mycluster
</value>
</property>
<!-- 开启 NameNode 失败自动切换 -->
```

```
<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>
<!-- 配置失败自动切换实现方式 -->
<property>
<name>dfs.client.failover.proxy.provider.bi</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverPr
oxyProvider</value>
</property>
<!—namenode 元数据存放位置 -->
<property>
                <name>dfs.namenode.name.dir</name>
                <value>file:///home/hadoop/hadoop-2.5.0-
cdh5.3.0/dfs/name</value>
        </property>
<!—edit log 存在位置 -->
        <property>
                <name>dfs.journalnode.edits.dir</name>
                <value>/home/hadoop/hadoop-2.5.0-
cdh5.3.0/dfs/journal</value>
        </property>
```

```
<!—data 数据 所有数据都将存在此目录下,如磁盘不够可继续添加磁盘  -->
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:///home/hadoop/data</value>
        </property>
<!—副本数，默认每份数据都会存 3 份 --->
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
        <property>
                <name>dfs.image.transfer.bandwidthPerSec</name>
                <value>1048576</value>
        </property>
        <property>
                <name>dfs.datanode.du.reserved</name>
                <value>5368709120</value>
        </property>
        <property>
                <name>dfs.blocksize</name>
                <value>268435456</value>
        </property>
```

```
<property>
        <name>dfs.namenode.handler.count</name>
        <value>20</value>
</property>
<property>
        <name>dfs.datanode.handler.count</name>
        <value>50</value>
</property>
<property>
        <name>dfs.datanode.socket.write.timeout</name>
        <value>10800000</value>
</property>
<property>
         <name>dfs.client.socket-timeout</name>
         <value>600000</value>
</property>
<property>
        <name>dfs.datanode.max.transfer.threads</name>
        <value>409600</value>
</property>
<property>
        <name>fs.trash.interval</name>
```

```
        <value>10080</value>
        <value>268435456</value>
</property>
<property>
        <name>dfs.namenode.handler.count</name>
        <value>20</value>
</property>
<property>
        <name>dfs.datanode.handler.count</name>
        <value>50</value>
</property>
<property>
        <name>dfs.datanode.socket.write.timeout</name>
        <value>10800000</value>
</property>
<property>
         <name>dfs.client.socket-timeout</name>
         <value>600000</value>
</property>
<property>
        <name>dfs.datanode.max.transfer.threads</name>
        <value>409600</value>
```

```
        </property>
        <property>
                <name>fs.trash.interval</name>
                <value>10080</value>
        </property>
        <property>
                <name>dfs.permissions.enabled</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.support.append</name>
                <value>true</value>
        </property>
        <property>
                <name>dfs.datanode.max.transfer.threads</name>
                <value>8192</value>
        </property>
</configuration>
```

4.slaves 文件 （从节点 ）存放数据的节点

```
Hadoop12
Hadoop13
```

Hadoop14

拷贝到其他服务器

```
scp -r hadoop*  hadoop11:/home/hadoop/
```

四，启动顺序

1.先启动 zookeeper

```
zkServer.sh start
```

2.启动 hadoop12,hadoop13,hadoop14 的 journalnode

```
sbin/hadoop-daemon.sh start journalnode
```

3.格式化 HDFS(在 hadoop10 或者 hadoop11 上执行格式化）

```
bin/hdfs namenode -format
```

注意：格式化完成之后，为了保证两个 namenode 的数据一致，需要把 hadoop10 的 namenode 的数据目录复制到 hadoop11 的相应的目录下

4.格式化 ZKFC(在 hadoop10 或者 hadoop11 上执行格式化）

```
bin/hdfs zkfc -formatZK
```

5.启动 HDFS(在 hadoop10 或者 hadoop11 上启动)

```
$HADOOP_HOME/sbin/start-dfs.sh
```

五，验证安装

hadoop10 和 hadoop11

jps 含有 namenode 和 zkfc 两个进程

7776 NameNode

1398 Jps

8280 DFSZKFailoverController

Hadoop12-14

jps 含有 JournalNode DataNode 两个进程 和规划一至

10828 JournalNode

1704 DataNode

常用运维

哪个程序死了 就重新启动哪个进程

sbin/hadoop-daemon.sh start datanode

sbin/hadoop-daemon.sh start nameode

sbin/hadoop-daemon.sh start journalnode

也可以 sbin/start-dfs.sh

哪里出错了 就去看$HADOOP_HOME 下的日志

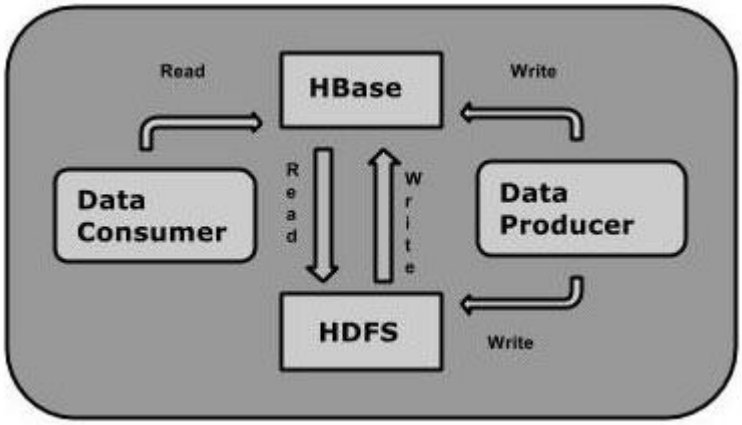
Ps： 对于添加机器，可以直接 scp ，注意不要拷贝 data/ name/ 下的数据 ，

建立 ssh ， 直接在新增的服务器上 sbin/hadoop-daemon.sh start datanode

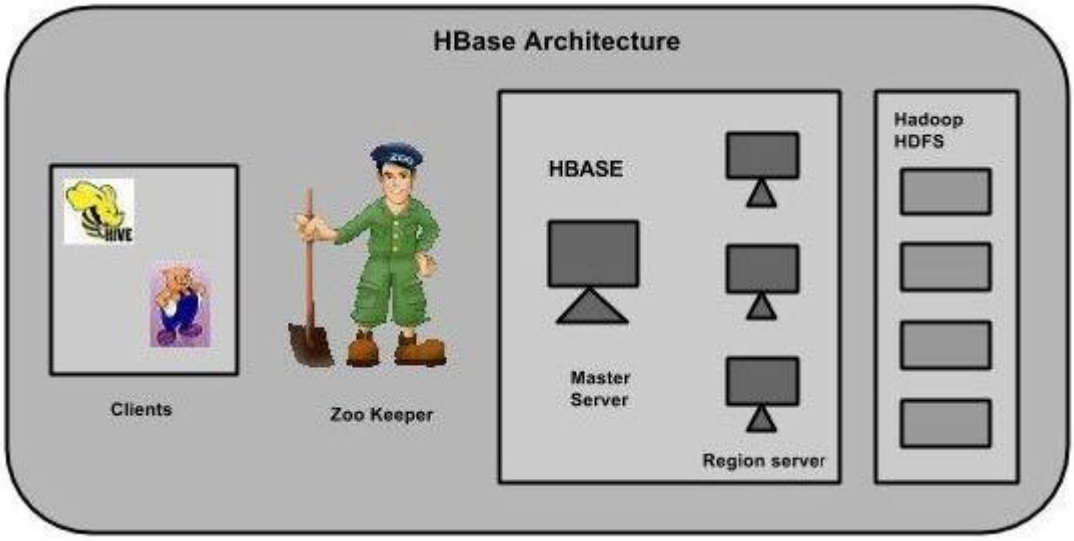
即可。

HBASE 安装

HBase 是什么?
HBase 是建立在 Hadoop 文件系统之上的分布式面向列的数据库。它是一个开源项目，是横向扩展的。
HBase 是一个数据模型，类似于谷歌的大表设计，可以提供快速随机访问海量结构化数据。它利用了 Hadoop 的文件系统（HDFS）提供的容错能力。
它是 Hadoop 的生态系统，提供对数据的随机实时读/写访问，是 Hadoop 文件系统的一部分。
人们可以直接或通过 HBase 的存储 HDFS 数据。使用 HBase 在 HDFS 读取消费/随机访问数据。 HBase 在 Hadoop 的文件系统之上，并提供了读写访问。



在 HBase 中，表被分割成区域，并由区域服务器提供服务。区域被列族垂直分为“Stores”。Stores 被保存在 HDFS 文件。下面显示的是 HBase 的结构。
注意：术语“store”是用于区域来解释存储结构。



HBase 有三个主要组成部分：客户端库，主服务器和区域服务器。区域服务器可以按要求添加或删除。

主服务器 (HMaster)

分配区域给区域服务器并在 Apache ZooKeeper 的帮助下完成这个任务。

处理跨区域的服务器区域的负载均衡。它卸载繁忙的服务器和转移区域较少占用的服务器。

通过判定负载均衡以维护集群的状态。

负责模式变化和其他元数据操作，如创建表和列。

区域 (region)

区域只不过是表被拆分，并分布在区域服务器。

区域服务器

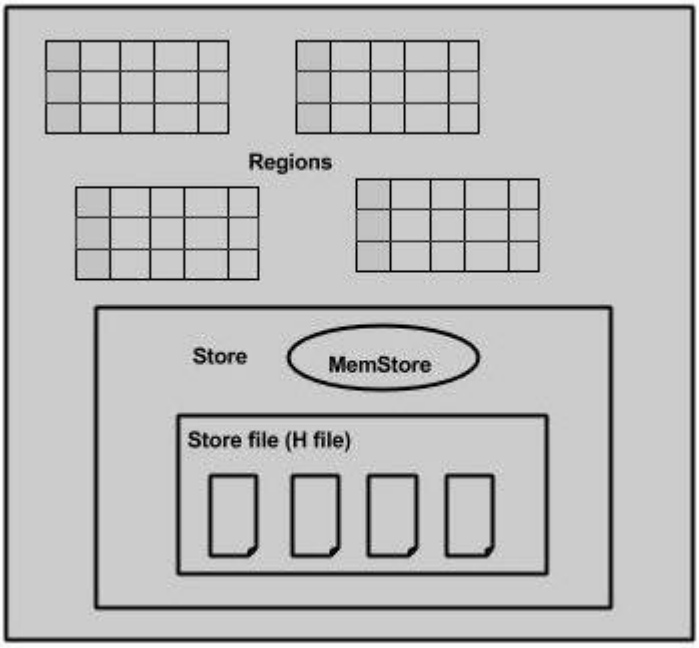
区域服务器拥有区域如下 -

与客户端进行通信并处理数据相关的操作。

句柄读写的所有地区的请求。

由以下的区域大小的阈值决定的区域的大小。

需要深入探讨区域服务器：包含区域和存储，如下图所示：



存储包含内存存储和 HFiles。memstore 就像一个高速缓存。在这里开始进入了 HBase 存储。数据被传送并保存在 Hfiles 作为块并且 memstore 刷新。

Zookeeper

Zookeeper 管理是一个开源项目，提供服务，如维护配置信息，命名，提供分布式同步等

Zookeeper 代表不同区域的服务器短暂节点。主服务器使用这些节点来发现可用的服务器。

除了可用性，该节点也用于追踪服务器故障或网络分区。

客户端通过与 zookeeper 区域服务器进行通信。

在模拟和独立模式，HBase 由 zookeeper 来管理。

一. 功能分配

Hadoop10：HMaster

Hadoop11：HMaster

Hadoop12： regionserver

Hadoop13： regionserver

Hadoop14： regionserver

二. 安装

解压 habse

拷贝 hadoop 下 core-site 和 hdfs-site.xml 至$HBASE_HOME/conf 下

Vi hbase-site.xml

```
<configuration>
<!---hbase 在 hadoop 下的存储根目录 -->
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://mycluster/hbase</value>
</property>
```

```
<property>
    <name>hbase.master.port</name>
    <value>16000</value>
</property>
<!--zk-->
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>iovdc13,iovdc14,iovdc15</value>
</property>
<property>
      <name>hbase.zookeeper.property.clientPort</name>
      <value>2181</value>
</property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
<property>
    <name>hbase.tmp.dir</name>
    <value>/home/hadoop/hbase-0.98.6-cdh5.3.0/tmp</value>
</property>
<property>
```

```
    <name>hbase.regionserver.handler.count</name>
    <value>30</value>
</property>
<!---habse 一个 region 最大的文件大小 5G-->
<property>
    <name>hbase.hregion.max.filesize</name>
    <value>5368709120</value>
</property>
<property>
    <name>hbase.hstore.blockingWaitTime</name>
    <value>90000</value>
</property>
<property>
    <name>hbase.hregion.memstore.mslab.enabled</name>
    <value>true</value>
</property>
<!--- HFile 写入磁盘的大小 -->
<property>
    <name>hbase.hregion.memstore.flush.size</name>
    <value>536870912</value>
</property>
<property>
```

```
    <name>hbase.rpc.timeout</name>
    <value>180000</value>
</property>
<property>
    <name>hbase.client.scanner.timeout.period</name>
    <value>180000</value>
</property>
</configuration>
```

```
vi regionservers
hadoop12
hadoop13
hadoop14
$HBASE_HOME/sbin/start-hbase.sh
Hadoop11:/ $HBASE_HOME/sbin/hbase-demaon.sh start master
```

Hbase 运维

```
hbase-demaon.sh start master
hbase-demaon.sh start regionserver
```

更多运维信息，通过查看$HBASE_HOME/log 定位