[**HDFS HA与QJM架构**](http://blog.csdn.net/stark_summer/article/details/44219095)

# 问题导读

**1.HDFS HA与QJM解决了什么问题？  
2.HDFS HA与QJM区别是什么？  
3.在HA（两个namenode）架构下，如何访问hdfs文件？**  
  
  
【使用QJM构建HDFS HA架构(2.2+)】  
    本文主要介绍HDFS HA特性，以及如何使用QJM(Quorum Journal Manager)特性实现HDFS HA。

# 一、背景

HDFS集群中只有一个Namenode，这就会引入单点问题；即如果Namenode故障，那么这个集群将不可用，直到Namenode重启或者其他Namenode接入。  
    有两种方式会影响集群的整体可用性：  
    1、意外的突发事件，比如物理机器crash，集群将不可用，直到管理员重启Namenode。  
    2、系统维护，比如软件升级等，需要关闭Namenode，也会导致集群暂时性的失效。  
  
    HDFS HA特性即解决这个问题，它通过在集群中同时运行2个(redundant)Namenodes，并让active和passive之间热备(hot standby)。当Active Namenode故障失效后，即可快速故障转移到新的Namenode上(passive Namenode)；也可以在计划维护期间，基于管理员发起(administrator-inited)的友好的failover。

# 二、Architecture

在典型的HA架构中，有两个独立的机器作为Namenode，任何时刻，只有一个Namenode处于Active状态，另一个处于standby状态(passive,备份)；Active Namenode用于接收Client端请求，Standy节点作为slave保持集群的状态数据以备快速failover。  
  
    为了让Standby Node与Active Node保持同步，这两个Node都与一组称为JNS的互相独立的进程保持通信(Journal Nodes)。当Active Node上更新了namespace，它将记录修改日志发送给JNS的多数派。Standby noes将会从JNS中读取这些edits，并持续关注它们对日志的变更。Standby Node将日志变更应用在自己的namespace中，当failover发生时，Standby将会在提升自己为Active之前，确保能够从JNS中读取所有的edits；即在failover发生之前，Standy持有的namespace应该与Active保持完全同步。  
  
    为了支持快速failover，Standby node持有集群中blocks的最新位置是非常必要的。为了达到这一目的，Datanodes上需要同时配置这两个Namenode的地址，同时和它们都建立心跳链接，并把block位置发送给它们。  
  
    任何时刻，只有一个Active Namenode是非常重要的，否则将会导致集群操作的混乱，那么两个Namenode将会分别有两种不同的数据状态，可能会导致数据丢失，或者状态异常，这种情况通常称为“split-brain”(脑裂，三节点通讯阻断，即集群中不同的Datanodes却看到了两个Active Namenodes)。对于JNS(Journal Nodes)而言，任何时候只允许一个Namenode作为writer；在failover期间，原来的Standby Node将会接管Active的所有职能，并负责向JNS写入日志记录，这就阻止了其他Namenode基于处于Active状态的问题。

# 三、硬件资源

为了构建HA集群架构，你需要准备如下资源：  
    1、Namenode机器：两台配置对等的物理机器，它们分别运行Active和Standby Node。  
    2、JouralNode机器：运行JouralNodes的机器。JouralNode守护进程相当的轻量级，它们可以和hadoop的其他进程部署在一起，比如Namenodes、jobTracker、ResourceManager等。不过为了形成多数派(majority)，至少需要3个JouralNodes，因为edits操作必须在多数派上写入成功。当然JNS的个数可以 > 3，且通常为奇数(3,5,7)，这样可以更好的容错和形成多数派。如果你运行了N个JNS，那么它可以允许(N-1)/2个JNS进程失效并且不影响工作。  
  
    此外，在HA集群中，standby namenode还会对namespace进行checkpoint操作(继承Backup Namenode的特性)，因此，就不需要在HA集群中运行SecondaryNamenode、CheckpointNode或者BackupNode。事实上，HA架构中运行上述节点，将会出错(不允许)。

# 四、部署

## 一)、配置

和HDFS Federation类似，HA配置向后兼容，运行只有一个Namenode运行而无需做任何修改。新的配置中，要求集群中所有的Nodes都有相同的配置文件，而不是根据不同的Node设定不同的配置文件。  
  
    和HDFS Federation一样，HA集群重用了“nameservice ID”来标识一个HDFS 实例(事实上它可能包含多个HA Namenods)；此外，“NameNode ID”概念被添加到HA中，集群中每个Namenode都有一个不同的ID；为了能够让一个配置文件支持所有的Namenodes(适用与Federation环境)，那么相关的配置参数都以“nameservice ID”或“Namenode ID”作为后缀。  
  
   修改hdfs-site.xml，增加如下几个配置参数，其参数的顺序无关。

1、dfs.nameservices：nameservice的逻辑名称。可以为任意可读字符串；如果在Federation中使用，那么还应该包含其他的nameservices，以","分割。

1. <property>
2. <name>dfs.nameservices</name>
3. <value>hadoop-ha,hadoop-federation</value>
4. </property>

2、dfs.ha.namenodes.[nameservice ID]：

1. <property>
2. <name>dfs.ha.namenodes.hadoop-ha</name>
3. <value>nn1,nn2</value>
4. </property>

 3、dfs.namenode.rpc-address.[nameservice ID].[namenode ID]

1. <property>
2. <name>dfs.namenode.rpc-address.hadoop-ha.nn1</name>
3. <value>machine1.example.com:8020</value>
4. </property>
5. <property>
6. <name>dfs.namenode.rpc-address.hadoop-ha.nn2</name>
7. <value>machine2.example.com:8020</value>
8. </property>

其中nameservice ID需要和1)匹配，namenode ID需要和2) 匹配。配置项的值为相应namenode的hostname以及通讯端口号(Client与namenode RPC通讯端口)，它和non-ha模式下“dfs.namenode.rpc-address”作用一样。每个namenode ID都需要单独配置。  
  
    你可以根据需要，配置“dfs.namenode.servicerpc-address”，格式和上述一致。(SNN，backup节点与Namenode通讯地址)

4、dfs.namenode.http-address.[nameservice ID].[namenode ID]

1. <property>
2. <name>dfs.namenode.http-address.hadoop-ha.nn1</name>
3. <value>machine1.example.com:50070</value>
4. </property>
5. <property>
6. <name>dfs.namenode.http-address.hadoop-ha.nn2</name>
7. <value>machine2.example.com:50070</value>
8. </property>

各个namenode的HTTP地址。它和non-ha下的"dfs.namenode.http-address"配置作用一样。

5、dfs.namenode.shared.edits.dir：

1. <property>
2. <name>dfs.namenode.shared.edits.dir</name>
3. <value>qjournal://node1.example.com:8485;node2.example.com:8485;node3.example.com:8485/hadoop-ha</value>
4. </property>

配置JNS组的url地址，Namenodes将会从JNS组中读写edits。这是一个共享存储区，Active Namenode写入，Standby Node读取，每个Namenodeservice必须配置足够多的JNS地址(>=3，多数派)，每条的格式为：  
    “qjournal://host1:port1;host2:port2;host3:port3/journalId”  
    其中journalId需要和上述配置中的“nameserviceID”匹配。

1. <property>
2. <name>dfs.journalnode.rpc-address</name>
3. <value>0.0.0.0:8485</value>
4. </property>
5. <property>
6. <name>dfs.journalnode.http-address</name>
7. <value>0.0.0.0:8480</value>
8. </property>

此外，我们还需要在相应的JournalNodes上增加上述配置。

6、dfs.client.failover.proxy.provider.[nameservice ID]：  
    HDFS Client链接Namenode所使用的类，Client可以通过此类来判定哪个Namenode为Alive，并与它保持通信。目前hadoop中唯一的实现类为"ConfiguaredFailoverProxyProvider"。

1. <property>
2. <name>dfs.client.failover.proxy.provider.hadoop-ha</name>
3. <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
4. </property>

7、dfs.ha.fencing.methods：在failover期间用来隔离Active Namenode的脚本或者java 类列表。  
    虽然JNS可以确保集群中只有一个Active Node写入edits，这对保护edits一致性很重要，但是在failover期间，有可能Acitive Node仍然存活，Client可能还与其保持连接提供旧的数据服务，我们可以通过此配置，指定shell脚本或者java程序，SSH到Active NameNode然后Kill Namenode进程。它有两种可选值(具体参见官方文档)：  
    1) sshfence：SSH登录到Active Namenode，并Kill此进程。首先当前机器能够使用SSH登录到远端，前提是已经授权(rsa)。  
    2) shell：运行shell指令隔离Active Namenode。

1. <property>
2. <name>dfs.ha.fencing.methods</name>
3. <value>shell(/path/to/my/script.sh arg1 arg2 ...)</value>
4. </property>

复制代码

“()”之间为shell脚本的路径，以及参数列表。

8、fs.defaultFS(core-site.xml)：  
    在non-ha下，这个参数值为namenode的地址：“hdfs://namenode:8020”；不过在HA架构下，将使用namenservice名称替代[回答了第三个问题]

1. <property>
2. <name>fs.defaultFS</name>
3. <value>hdfs://hadoop-ha</value>
4. </property>

9、dfs.journalnode.edits.dir：  
    指定journalNode存储edits文件的本地路径。  
  
    最终，上述配置信息，需要在server和Client端同时配置才能有效的适应HA与failover特性。

## 二)、部署

上述配置调整完毕后，我们就可以启动journalNodes守护进程，默认的"sbin/start-dfs.sh"脚本会根据"dfs.namenode.shared.edits.dir"配置，在相应的Datanode上启动journalNodes。当然我们可以使用:："bin/hdfs start journalnode"分别在相应的机器上启动。  
    一旦JournalNodes启动成功，它们将会从Namenode上同步metadata。  
    1、如果你的HDFS集群是新建的，那么需要在每个Namenode上执行"hdfs namenode -format"指令。  
    2、如果你的namenodes已经format了，或者是将non-ha转换成ha架构，你应该在将其中一个namenode上的metadata复制到另一台上(dfs.namenode.name.dir目录下的数据)，然后在那个没有format的新加入的namenode上执行"hdfs namenode -bootstrapStandby"。运行这个指令需要确保JournalNodes中持有足够多的edits。  
    3、如果你将一个non-ha的Namenode(比如backup，其已经formated)切换成HA，你需要首先运行"hdfs -initializeSharedEdits"，这个指令将本地Namenode中的edits初始化Journalnodes。  
  
    此后，你就可以启动HA Namenodes。可以通过配置指定的HTTP地址(dfs.namenode.https-address)来查看各个Namenode的状态，Active or Standby。

## 三)、管理员指令

HA集群启动后，我们可以通过一些指令来管理HDFS集群。“bin/hdfs haadmin -DFSHAAdmin”指令，其可选参数：  
    1、-transitionToActive <namenode id>与-transitionToStandbyl <namenode id>：将指定的namenode ID切换为Active或者standby。这个指令并不会触发“fencing method”，所以不常用，我们通常使用"hdfs haadmin -failover"来切换Namenode状态。  
    2、-failover [--forcefence] [--foreactive] <serviceId-fist> <serviceId-second>：在两个Namenode之间failover。这个指令会触发将first节点failover到second节点。如果first处于standby，那么只是简单的将second提升为Active。如果first为Active，那么将会友好的将其切换为standby，如果失败，那么fencing methods将会触发直到成功，此后second将会提升为Active。如果fencing method失败，那么second将不会被提升为Active。  
    例如："hdfs haadmin -DFSHAAdmin -failover nn1 nn2"  
    3、-getServiceState <serviceId>：获取serviceId的状态，Active还是Standby。链接到指定的namenode上，并获取其当前的状态，打印出“standby”或者“active”。我可以在crontab中使用此命令，用来监测各个Namenode的状况。  
    4、-checkHealth <serviceId>：检测指定的namenode的健康状况。

# 五、自动Failover

    上述介绍了如何配置手动failover，在这种模式下，系统不会自动触发failover，即不会将Standby提升为Active，即使Active已经失效。接下来介绍如何实现自动failover。

## 一)、组件

Automatic Failover中，增加了2个新的组件：zookeeper集群，ZKFailoverController进程(简称为ZKFC)。  
    Zookeeper是一个高可用的调度服务，可以保存一系列调度数据，当这些数据变更(notify)时可以通知Client，以及监控(montitor)Clients失效，自动failover的实现将依赖于Zookeeper的几个特性：  
    1、Failure delection：失效检测，每个Namenode将会和zookeeper建立一个持久session，如果Namenode失效，那么次session将会过期失效，此后Zookeeper将会通知另一个Namenode，然后触发Failover。  
    2、Active Namenode election：zookeeper提供了简单的机制来实现Acitve Node选举，如果当前Active失效，Standby将会获取一个特定的排他锁(lock)，那么获取(持有)锁的Node接下来将会成为Active。  
  
    ZKFailoverControllor(ZKFC)是一个zookeeper客户端，它主要用来监测和管理Namenodes的状态，每个Namenode机器上都会运行一个ZKFC程序，它的职责为：  
    1、Health monitoring：ZKFC间歇性的使用health-check指令ping本地的Namenode，Namenode也会及时的反馈自己的health status。如果Namenode失效，或者unhealthy，或者无响应，那么ZKFS将会标记其为“unhealthy”。  
    2、Zookeeper session manangement：当本地Nanenode运行良好时，ZKFC将会持有一个zookeeper session，如果本地Namenode为Active，它同时也持有一个“排他锁”(znode)；这个lock在zookeeper中为“ephemeral” znode(临时节点)，如果session过期，那么次lock所对应的znode也将被删除。(参见zookeeper特性)  
    3、Zookeeper-based election：如果本地Namenode运行良好，并且ZKFS没有发现其他的的Namenode持有lock(比如Active失效后，释放了lock)，它将尝试获取锁，如果获取成功，即“赢得了选举”，那么此后将会把本地Namenode标记为Active，然后触发Failover：首先，调用fencing method，然后提升本地Namenode 为Active。  
  
    具体Failover过程和详细内容，请参见[HDFS-2185](https://issues.apache.org/jira/browse/HDFS-2185)。

## 二)、配置

    在Automatic Failover中，需要把一个重要的配置项添加到hdfs-site.xml中。

1. <property>
2. <name>dfs.ha.automatic-failover.enabled</name>
3. <value>true</value>
4. </property>

此外还需要在core-site.xml中，增加如下配置：

1. <property>
2. <name>ha.zookeeper.quorum</name>
3. <value>zk1.example.com:2181,zk2.example.com:2181,zk3.example.com:2181</value>
4. </property>

上述zookeeper集群为即备，尽可能选择相对平稳的zk集群。  
  
    其中"dfs.ha.automatic-failover.enabled"可以为每个nameservice ID分别配置：dfs.ha.automatic-failover.enabled.[nameservice ID]。此外在core-site.xml中还可以配置Zookeeper Client的相关参数，比如sessionTimeout，这些配置项以"ha.zookeeper"开头，其中"dfs.ha."开头的部分配置项可以用来设定fencing method的相关控制。

## 三)、初始化HA状态

上述准备工作结束后，我们还需要在zookeeper中初始化HA的状态，通过执行“hdfs zkfc -formatZK”，此命令将会在zookeeker中创建一个znode，用来保存HA或failover的数据。

## 四)、启动集群

可以使用"start-dfs.sh"这个便捷的指令，它启动了hdfs所需要的所有守护进程，当然包括ZKFC。也可以使用"hadoop-daemon.sh start zkfc"手动启动ZKFC客户端。

## 五)、检验Failover

一旦Automatic Failover集群启动之后，我们需要检测Failover是否符合预期。首先，我们需要通过命令(getServiceState)或者在Namenode的Web UI上查看各个Namenode的状态，确认两个Namenode是否分别处于Active和Standby；此后，你可以手动关闭Active Namenode，比如使用kill -9 <pid num>，在确定Acitve Node失效后，再次检测原来的Standby是否已经提升为Active；不过因为zookeeper session过期判定需要达到sessionTimeout(可配置，ha.zookeeper.session-timeout)，这个failover过程可能需要滞后数秒，默认为5秒。  
  
    如果没有按照预期failover，那么你需要检测配置文件是否正确，zk服务是否正确。此外，我们还可以使用上述DFSHAAadmin指令多次尝试。

# 六、FAQ

1、ZKFC和Namenodes守护进程的启动顺序是否重要？  
    No，对于指定的Namenode，你可以在其之前或者之后启动ZKFC均可以，ZKFC只是调度Namenode的存活状态，如果不启动ZKFC，此Namenode将无法参与自动failover过程。  
    2、是否需要额外的monitoring？  
    你需要在Namenode机器上，添加额外的monitor用来监控ZKFC是否运行。在某些情况下，zookeeper集群的故障可能导致ZKFC意外中断，你需要适时的重启ZKFC。此外，还需要监控Zookeeper集群的运行状况，如果Zookeeper集群失效，那么HA集群将无法failover。  
    3、如果Zookeeper失效，将会怎么样？  
    如果zookeeper集群故障，那么Automatic Failover将不会触发，即使Namenode失效，这也意味着ZKFC无法正常运行。不过，如果Namenodes正常(即使有一个失效)，那么HDFS系统将不会受到影响。因为HDFS Client并没有基于zookeeper做任何事情，当zookeeper集群仍需要尽快的恢复以避免当前Active失效而造成的“split-brain”等问题。  
    4、是否可以在Namenodes之间指定优先级？  
    NO，这是不能支持的。首先启动的Namenode将作为Active，我们只能认为控制Namenode启动的顺序来做到“优先级”。  
    5、在Automatic Failover中，手动Failover怎么做？  
    和普通的Failover一样，我们总是可以通过"hdfs haadmin -DFSHAAdmin -failover"来实现手动Failover。