静静
一种美德

HDFS HA切换后missing block问题分析

页 2015年05月31日 21:03:31

我们Hadoop平台也从Hadoop1.2.1升级到了Hadoop2.4.0版本，当然HDFS HA 也配置到集群中。具体的配置方案（https://issues.apache.org/jira/browse/HDFS-1623）。感恩cloudera 这样伟大的公司，为我们提供了一oop1.2.1迁移到Hadoop2.4.0，同时验证HDFS HA 切换是否对HBase功能产生影响。问题出现了，当我们持续menode)，读出刚才写入的数据，悲剧了，数据读不来，HDFS web页面出现了missing block。下来我们我们分析HBase 相关日志，发现在HA 切换时写入 hfile 文件出现问题。ok，我们现在先验证HDFS H个不难啊，我们用HDFS client代码持续写入大量小文件（10个字节），同时进行HA切换。悲剧再次发生了，H

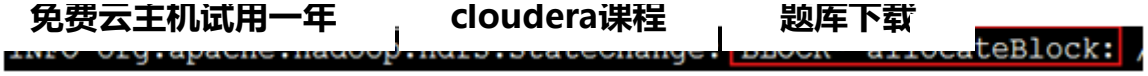
现在问题复现了，先兴奋一把，我们先解决 HDFS HA 切换造成块丢失的问题。解决了这个问题再继续验证HBas

- namenode2是active Namenode，我们切换的时候直接kill掉该节点；
- namenode1是standby Namenode ，我们切换后直接变为active Namenode；

一、问题分析

1）Namenode 分析

我们选择第一个block:blk_1075342606进行分析，切换后namenode1日志如下：



该日志说明已经给相应的block分配了相关的Datanode节点。也就是说在namenode2 inode（Fsimage）和 blo
ode 的inode（Fsimage）和blockmap 忘记了。忘记的同学可以一下看看ggjucheng 同学对namenode源码简要分
2013/02/04/2889386.html。

需要我们记住的是：NameNode作为HDFS中文件目录和文件分配的管理者，它保存的最重要信息，就是下面两

- 文件名=>数据块：持久化为fsimage，持久化到内存中、文件系统的inode相似；
- 数据块=>DataNode列表，BLockmap保存的结构。

Datanode日志

查询Datanode日志，查看blk是否写入到Datanode。方法如下

r i in $(cat ~/software/hadoop/etc/hadoop/slaves); do echo $i; ssh $i "grep blk_1075342606 ~/hadoop/logs/ha

以验证到如下：

这说明blk 已经成功写入到了三个Datanode节点。 数据写入正常，但是namenode 1里面关于该block的元数据不正

3) FileJournalManager同步日志

需要查看edits是否记录该block的edit log文件。在10:53:42开始大量的日志editlog同步。需要确定blk_1075342

4) 综合分析

我们可以做如下分析：对于namenode1日志,blockmap 里面为什么没有该block信息？熟悉HDFS的同学知道，个文件对应的blockid保存入BlocksMap，此时BlocksMap中每个block对应的Datanodes列表暂时为空。创建文件时

在大神（微博名：@王志强-Austin）指导下，结合 Namenode HA方案，standby namenode同步editlog 通过Q
ock给 namenode1的时候，edit log还没同步过来。怎么验证这种猜想了？重启一下这个block对于的Datanode，

我们重启该block report对于的datanode，oh my God！奇迹发生了，该block从页面的missing block队列里面去
数据。

我们回头过来分析namenode1 block report 和edit log 的时间，发现在HDFS HA 切换期间，当Datanode 的incr
时，后台standby没有完全变成activenamenode之前，会出现包含 invalidate block 的后台日志。

1) Datanode blockreport

tanode会进行增量或者全量block report 给namenode。增量的block report间隔时间为100*（heartbeat间隔时间
个小时。下面是代码入口：
reportReceivedDeletedBlocks： 从上次lastDeletedReport后的接收到的block汇报给namenode；所有新加入的
列中；当本次汇报完了以后会清空pendingIncrementalBRperStorage队列；如果汇报失败会将合并到下一次的p
blockReport ：从上次lastBlockReport计算是否过了dnConf.blockReportInterval（默认为1个小时）时间，进行

```
pNamenode.blockReport(bpRegistration, bpos.getBlockPoolId(), reports);
```

Namenode blockreport process

面查看namenode如何处理IncrementalBlockReport，处理入口是：

```
for (ReceivedDeletedBlockInfo rdbi : srdb.getBlocks()) {
  switch (rdbi.getStatus()) {
  case DELETED_BLOCK:
    removeStoredBlock(rdbi.getBlock(), node);
    deleted++;
    break;
  case RECEIVED_BLOCK:
    addBlock(node, srdb.getStorage().getStorageID(),
        rdbi.getBlock(), rdbi.getDelHints());
    received++;
    break;
  case RECEIVING_BLOCK:
    receiving++;
    processAndHandleReportedBlock(node, srdb.getStorage().getStorageID(),
        rdbi.getBlock(), ReplicaState.RBW, null);
    break;
```


根据不同的状态对block处理不同，需要关注RECEIVED_BLOCK、RECEIVING_BLOCK的处理
决：

```
BlockInfo storedBlock = blocksMap.getStoredBlock(block);
    if(storedBlock == null) {
      // If blocksMap does not contain reported block id,
      // the replica should be removed from the data-node.
      toInvalidate.add(new Block(block));
      return null;
    }
```

如果在blocksmap里面找不到该block，就判断该block为invalidate block；并且添加到toInvalidate队列里面。全量
lockReport相似。

3) FileJournalManager editlog

rollEditLog生成入口如下：

```
CheckpointSignature rollEditLog() throws IOException {
```

```
    checkSuperuserPrivilege(); 3      checkOperation(OperationCategory.JOURNAL);
    writeLock();
    try {
      checkOperation(OperationCategory.JOURNAL);
      checkNameNodeSafeMode("Log not rolled");
      if (Server.isRpcInvocation()) {
        LOG.info("Roll Edit Log from " + Server.getRemoteAddress());
      }
      return getFSImage().rollEditLog();
    } finally {
      writeUnlock();
    }
  }
```


namenode正常运行过程中，每个edit log最大为1M。每两分钟生成一个新的editlog。也就是说standby namenode

d edit log的加载入口为loadFSEdits，实现过程如下：

```
 long startTime = now();
      FSImage.LOG.info("Start loading edits file " + edits.getName());
      long numEdits = loadEditRecords(edits, false, expectedStartingTxId,
          startOpt, recovery);
      FSImage.LOG.info("Edits file " + edits.getName()
          + " of size " + edits.length() + " edits # " + numEdits
          + " loaded in " + (now()-startTime)/1000 + " seconds");
```

将目前的editlog加载到内存中，尽最大可能和active namenode保持同步。下面这段代码值得关注，可以查看加载

```
if (FSNamesystem.LOG.isDebugEnabled()) {
        FSNamesystem.LOG.debug(op.opCode + ": " + path +
            " numblocks : " + addCloseOp.blocks.length +
            " clientHolder " + addCloseOp.clientName +
            " clientMachine " + addCloseOp.clientMachine);
      }
```

如果需要查看standby Namenode edit log详细同步加载过程，可以打开该方法对于class文件的debug日志，具体方

```
hadoop daemonlog --setlevel Namenode: 9000  org.apache.hadoop.HDFS.server.Namenode. FSEd
```

3. 问题解决

通过上面分析日志和源码分析我们得出如下结论：
ill active Namenode 时候，HDFS ha发生了切换。如果Datanode 的incremental block report 先于 standby Na
e block；当 standby 完全切换完后变为active Namenode，这些新加入的block依然是invalidate block。需要Da
生产集群出现这样的情况，不可能当HA发生切换的时候去重启invalidate block 对应的Datanode。可以采用如下

- Datanode发现namenode 发生切换的时候做一次全量blockreport。这种方案的优点是只需要
缺点是每次发送Namenode HA 切换所有Datanode都要重新做一次全量blockreport，通信开
时做一次全量blockreport。
- namenode专门起一个线程；当standby namenode变为activenamenode以后发现有invalidat
blockreport命令。这种方案的优点是只让有部分Datanode参与全量blockreport，减少通讯开
端代码，维护专门的线程去查找invalidate block对应的Datanode并发送相关命令。

文为了先解决问题，实现了第一种方案。在后续的文章中我们会继续验证第二种方案的可行性
：

```
commit edf77140caba9761e22fb6660de4f8d11add216b
Author: Ricky Yang <494165115@qq.com>
Date:   Fri Apr 24 08:13:33 EDT 2015
missing block when HA switch
diff   /hadoop-HDFS-project/hadoop-HDFS/src/main/java/org/apache/hadoop/HDFS/server/
>       private DatanodeProtocolClientSideTranslatorPB activeNamenode;
>        DatanodeProtocolClientSideTranslatorPB nowactiveNamenode = bpos.getActiveNN
>        if((nowactiveNamenode != null) && !nowactiveNamenode.equals(activeNamenode)
>         LOG.info("active nn changed and block report agained");
>         lastBlockReport= now()-dnConf.blockReportInterval-1;
 >        activeNamenode = nowactiveNamenode;
>         }
```

1) 在BPServiceActor.java设置全局变量保持当前activeNamenode；

2) 获取当前的activeNamenode 保存为nowactiveNamenode；

3) activeNamenode不为空并且namenode发生了切换时；

4) 调整lastblockreport时间，让BPServiceActor满足blockreport时间；

```
if (startTime - lastBlockReport <= dnConf.blockReportInterval) {
      return null;
    }
```

保存当前active Namenode;
该BPServiceActor 进行block report

处巧妙修改了block report时间判断条件lastBlockReport，提前触发了全量blockreport。

测试总结

编译HDFS源码，替换相应的hadoop-HDFS-2.4.0.jar包，重启HDFS和HBase。进行下面的测试：

est-1	HDFS 可用性
试步骤	1）持续写入HDFS小文件 2）kill 掉active Namenode 3）读出文件对比 4）查看后台日志和web页面
期	1）HDFS 读写正确 2）后台页面也web页面正常
测试结果	符合预期

Test-1	HBase可用性
测试步骤	1）持续写入HBase数据 2）kill 掉active Namenode 3）读出刚才写成的数据对比 4）查看后台日志和web页面
预期	1）HBase读写正确 2）HDFS后台页面也web页面正常
测试结果	符合预期

当kill active Namenode 时候，HDFS HA发生了切换。如果Datanode 的incremental block report 先于 standby 成了invalidate block；当 standby 完全切换完后变为active Namenode，这些新加入的block依然是invalidate block 原因是edit log还没有同步过来，在namespace里面没有创建相应的inode和blockmap信息。当edit log同步完了以后 需要修改的代码较少，可以暂时使用该方案。在生产集群已经运行1个多月，目前没有暴露出问题。

开源软件出现问题是很正常的，学习分析日志并与代码结合不失是一种解决问题的好方法。

文章标签： hadoop HDFS HA 大数据