HDFS高可用（HA）

若愚先生L 关注
0.24 2019.04.25 11:09:01 字数 1,757 阅读 515

概述

基于Hadoop 2.x。

NameNode保存着整个HDFS系统的元数据信息，NameNode挂了的话会导致整个HDFS系统不可用。HDFS重启时需要从磁盘上的EditLog生成元数据信息。

HDFS的HA要保证：
1、NameNode节点挂了，仍然有其他NameNode节点可用。（主备）
2、就算有一块磁盘坏了无法修复，HDFS的EditLog也不能丢。（多副本）

若愚先生L 关注
总资产5 (约0.48元)

Hive权限控制：集成Sentry 和 Hue
阅读 393

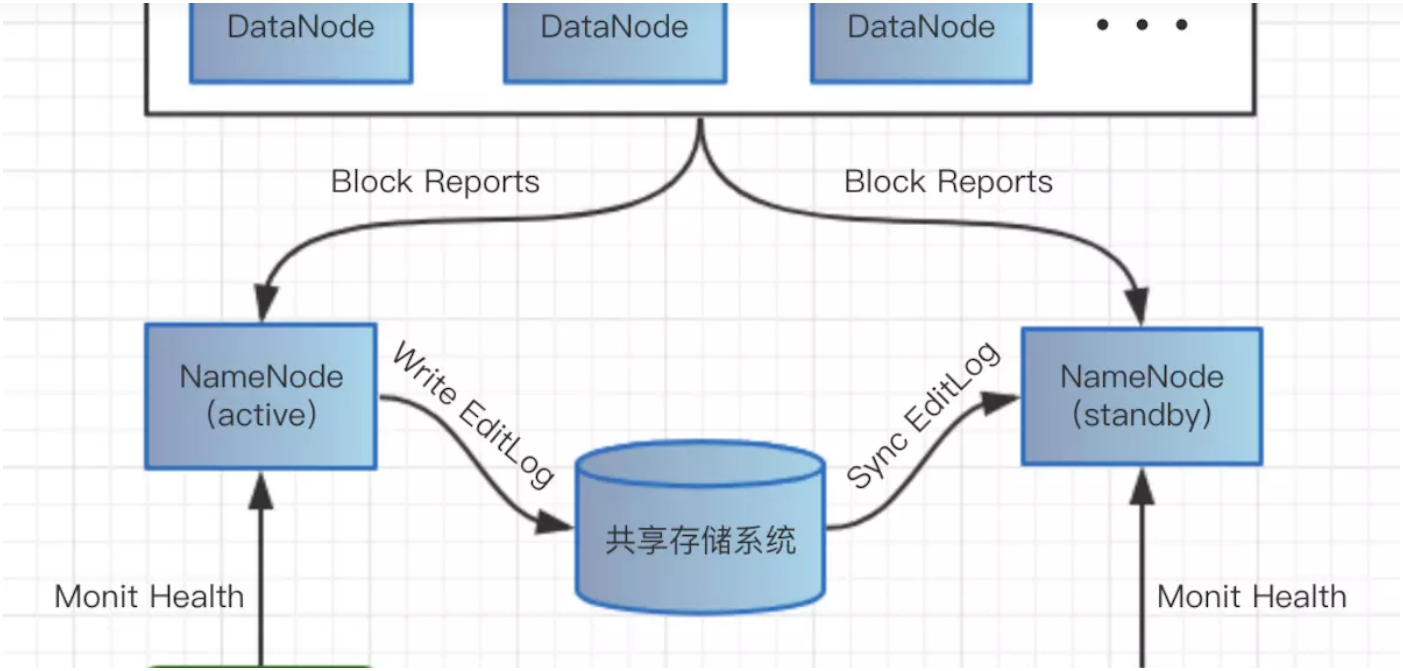
Spark（二）：对RDD的一些理解
阅读 363

推荐阅读

“我刚刚翻到了两年前的备忘录。”
阅读 18,283

外卖员与顾客聊天记录曝光：我必须让你吃到老子的餐
阅读 17,001

“我，14岁开始整容，花掉400万”：看脸的时代，我劝你学着体面
阅读 32,641



HDFS 高可用架构

1、有两个NameNode节点，一个为active主节点，一个为standby备用节点。DataNode同时向这两个NameNode汇报自身存储的Block状况。

2、ZKFC是一个独立的进程，与NameNode安装在同一服务器上，并监控这个NameNode的健康状况。ZKFC启动的时候会在Zookeeper创建选举节点（临时的ZNode），如果创建成功，ZKFC会告诉这个NameNode它就是active主节点。另外一个ZKFC创建失败，则与它绑定的NameNode就是standby备用节点。ZKFC的具体实现为org.apache.hadoop.hdfs.tools.DFSZKFailoverController。

3、共享存储系统存储EditLog，主备NameNode之间的元数据的同步就是通过这个共享存储系统。当发生主备切换时，新的主NameNode在确认元数据完全同步之后才能继续对外提供服务。HDFS目前默认的共享存储方案是QJM(Quorum Journal Manager)，也就是JournalNode集群，

2、DataNode进程

3、ZKFC进程

4、JournalNode进程

下面来根据源码分析详细的流程。

ZKFC

与本机的NameNode绑定，通过不断监控NameNode的状态来执行相应的操作。

```
// org.apache.hadoop.hdfs.tools.DFSZKFailoverController#main方法

// 创建一个DFSZKFailoverController对象
DFSZKFailoverController zkfc = DFSZKFailoverController.create(
        parser.getConfiguration());
// 调用DFSZKFailoverController对象的run方法
retCode = zkfc.run(parser.getRemainingArgs());
```

ZKFC启动的时候做了啥?

1、根据配置文件，找到与自己在同一个服务器上的NameNode，找不到抛出异常。注意这里只是根据配置文件来判断，这时候本机的NameNode进程可以没有启动。

```
// org.apache.hadoop.hdfs.tools.DFSZKFailoverController#create方法

String nnId = HAUtil.getNameNodeId(localNNConf, nsId);
if (nnId == null) {
  String msg = "Could not get the namenode ID of this node. " +
      "You may run zkfc on the node other than namenode.";
  throw new HadoopIllegalArgumentException(msg);
```

ha/{dfs.nameservices}/ActiveStandbyElectorLock的**临时节点**（{dfs.nameservices}是在配置文件里配置的），哪个ZKFC创建成功，与ZKFC绑定的那个NameNode就是active主节点。创建失败的那个ZKFC就监控这个临时节点的变化。此外第一次启动的时候还会在Zookeeper上创建一个/hadoop-ha/{dfs.nameservices}/ActiveBreadCrumb的**永久节点**，这个永久节点保存着当前Active NameNode的信息，这么做是为了防止脑裂，具体的后面分析。

```
// org.apache.hadoop.ha.ZKFailoverController#doRun方法

// 初始化Zookeeper连接信息，创建ActiveStandbyElector对象
initZK();
```

3、初始化并启动RPC服务，这个服务的协议是org.apache.hadoop.ha.ZKFCProtocol。ZKFC**不仅能主动进行主备切换，还能通过手动执行命令进行主备切换。**

手动主备切换的命令：

①hdfs haadmin -failover --forcefence --forceactive nn2 nn1

②hdfs haadmin -transitionToActive nn2

③hdfs haadmin -transitionToStandby nn1

其他更多命令可以执行hdfs haadmin -help**查看**

注意：②和③这两个命令不会进行fece**（隔离，防止脑裂的措施）操作，要谨慎使用。**

```
// org.apache.hadoop.ha.ZKFailoverController#doRun方法

// 初始化ZKFCRpcServer服务
initRPC();
// 启动ZKFCRpcServer服务
startRPC();
```

NameNode的健康状况。**所有的操作触发都是根据NameNode的健康状况由HealthMonitor触发的，可以说HealthMonitor是整个ZKFC服务的引擎。**

```
// org.apache.hadoop.ha.ZKFailoverController#doRun方法

// 初始化并启动HealthMonitor监控
initHM();
```

HealthMonitor内部有一个守护线程MonitorDaemon，它负责执行NameNode的健康检查并触发相应的操作。

```
// org.apache.hadoop.ha.HealthMonitor.MonitorDaemon#run方法

while (shouldRun) {
    try {
      // 循环直到连接上NameNode
      loopUntilConnected();
      // 开始执行NameNode的健康检查
      doHealthChecks();
    } catch (InterruptedException ie) {
      Preconditions.checkState(!shouldRun,
          "Interrupted but still supposed to run");
    }
}
```

```
// org.apache.hadoop.ha.HealthMonitor#doHealthChecks方法

while (shouldRun) {
  // NameNode目前的状态：INITIALIZING，ACTIVE，STANDBY，STOPPING
  HAServiceStatus status = null;
```

```
      proxy.monitorHealth();
      healthy = true;
    } catch (Throwable t) {
      // 如果是HealthCheckFailedException异常则当前NameNode的状态是SERVICE_UNHEALTHY。
      if (isHealthCheckFailedException(t)) {
        LOG.warn("Service health check failed for " + targetToMonitor
            + ": " + t.getMessage());
        enterState(State.SERVICE_UNHEALTHY);
      } else {
        // 由于ZKFC与NameNode部署在同一服务器上，所以两者之间不存在网络异常。这里可以直接认定当前NameNode
        LOG.warn("Transport-level exception trying to monitor health of " +
            targetToMonitor + ": " + t.getCause() + " " + t.getLocalizedMessage());
        RPC.stopProxy(proxy);
        proxy = null;
        enterState(State.SERVICE_NOT_RESPONDING);
        Thread.sleep(sleepAfterDisconnectMillis);
        return;
      }
    }

    if (status != null) {
      // 设置NameNode的状态，并触发相关的回调操作
      setLastServiceStatus(status);
    }
    if (healthy) {
      // 设置HealthMonitor的检查状态：INITIALIZING，SERVICE_NOT_RESPONDING，SERVICE_HEALTHY，SERVICE_
      // HEALTH_MONITOR_FAILED，并触发相关的回调操作。
      enterState(State.SERVICE_HEALTHY);
    }
  }
```

ZKFC启动完成后，选主和主备切换等操作是怎么触发的呢?

在初始化HealthMonitor对象的时候，设置了回调。

```
private void initHM() {
```

```
    healthMonitor.start();
}
```

只有健康检查的状态是SERVICE_HEALTHY的NameNode才具备选举权。

ZKFC是如何防止脑裂的?

ZKFC会在Zookeeper上创建：

/hadoop-ha/{dfs.nameservices}/ActiveStandbyElectorLock — **临时节点，用来进行选主。**

/hadoop-ha/{dfs.nameservices}/ActiveBreadCrumb — **永久节点，用来存放**active NameNode **信息。**

当出现ZKFC所在JVM因为负载高或者Full GC时间长，这时候会导致Zookeeper客户端与Zookeeper服务端之间的心跳不正常，如果超过了session超时时间，Zookeeper服务端就会删除/hadoop-ha/{dfs.nameservices}/ActiveStandbyElectorLock这个临时节点，这个删除操作被立马被standby NameNode绑定的ZKFC感知到，然后创建/hadoop-ha/{dfs.nameservices}/ActiveStandbyElectorLock临时节点成功，最终成为新的active NameNode节点。这种情况下一定的时间内旧的active NameNode任然认为自己是active主节点。这时候整个HDFS系统存在两个active NameNode，产生了脑裂，这对强一致性的HDFS系统是不能容忍的。

/hadoop-ha/{dfs.nameservices}/ActiveBreadCrumb这个永久节点就是为了解决这个问题的。

```
private boolean becomeActive() {
    assert wantToBeInElection;
    if (state == State.ACTIVE) {
      // already active
      return true;
    }
    try {
```

```
13
14        if (LOG.isDebugEnabled()) {
15          LOG.debug("Becoming active for " + this);
16        }
17
18        // 成为新的Active节点
19        appClient.becomeActive();
20        state = State.ACTIVE;
21        return true;
22      } catch (Exception e) {
23        LOG.warn("Exception handling the winning of election", e);
24        // Caller will handle quitting and rejoining the election.
25        return false;
26      }
27  }
```

在进行 fencing 的时候，会执行以下的操作：

1、首先尝试调用这个旧 Active NameNode 的 HAServiceProtocol服务的 transitionToStandby方法，看能不能把它转换为Standby状态。

2、如果 transitionToStandby 方法调用失败，那么就执行 Hadoop 配置文件之中预定义的隔离措施，Hadoop 目前主要提供两种隔离措施：

sshfence：**通过 SSH 登录到目标机器上，执行命令 fuser 将对应的进程杀死；**

shellfence：**执行一个用户自定义的 shell 脚本来将对应的进程隔离；**

一般使用sshfence方法，配置如下：

```
1   <property>
2     <name>dfs.ha.fencing.methods</name>
3     <value>sshfence(hdfs:22)</value>
4   </property>
5   <property>
6     <name>dfs.ha.fencing.ssh.private-key-files</name>
```

注意：当fence的方式选择SSH的方式时，如果Active NameNode所在的服务器宕机了，这时候会主从切换失败，因为fence操作会失败，HDFS宁愿服务不可用，也要保证数据的一致性。这时候Standby Name对应的ZKFC会一直尝试fence操作，直到fence成功。假如Active NameNode所在的服务器宕机了没有及时重启的话，这段时间里HDFS服务不可用。

JournalNode集群

Active NameNode与Standby NameNode之间的元数据同步是通过JournalNode集群来完成的，当Active NameNode的元数据有任何修改时，会通知所有的JournalNode，当半数以上的JournalNode接收成功时才会此次元数据修改操作成功。Standby NameNode不断从JournalNode监听Edit Log的变化然后读取变更信息，把变化应用于自己的命名空间。Standby NameNode可以确保在主从切换时，命名空间状态已经完全同步了。

NameNode

JournalNode

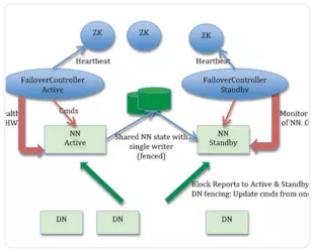
若愚先生L
总资产5 (约0.48元) 共写了1.9W字 获得48个赞 共21个粉丝

推荐阅读

更多精彩内容 >

冰解的破-HDFS

HDFS是Hadoop Distribute File System 的简称，也就是Hadoop的一个分布式文件系...



大佛爱读书 阅读 113 评论 0 赞 0

HDFS HA

翻译： http://hadoop.apache.org/docs/stable/hadoop-project-d...

金刚_30bf 阅读 116 评论 0 赞 1

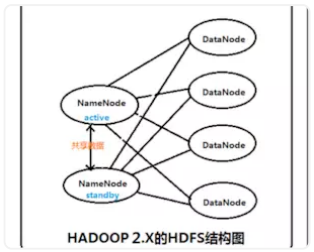
HDFS HA启用

翻译： https://www.cloudera.com/documentation/enterprise/lat...

金刚_30bf 阅读 841 评论 1 赞 1

hdfs体系结构-cdh5.7.1之hdfs各角色含义

(一)分布式文件系统概述 数据量越来越多，在一个操作系统管辖的范围存不下了，那么就分配到更多的操作系统管理的磁盘中...



时待吾 阅读 272 评论 0 赞 0

HA集群搭建