第十章管理Hadoop

## 10.1HDFS

### 10.1.1 永久性的数据结构

**1.namenode的目录结构**

Namenode被格式化之后，将产生如下所示目录结构：

${dfs.name.dir}/

------------current/

---------------------VERSION

---------------------edits

----------------------fsimage

-----------------------fstime

**VERSION文件是java属性文件，**其中包含HDFS的版本信息。该文件一般包含以下内容：

NamespaceID=\*\*\* // 文件系统的唯一标识符

cTime=\*\*\*\* // namenode 储存系统的创建时间

StrogaeType=\*\*\*\*\* // 说明存储目录包含的是namenode数据结构

layoutVersion=\*\*\*\* // 是一个负整数，描述HDFS持久性数据结构（布局）的版本，和Hadoop发布包的版本号无关，只要布局更改，版本号就会递减。

**2.文件系统映像和编辑日志**

文件系统客户端在执行写操作时，这些操作首先是被记录到编辑日志中。

Fsimage文件是文件系统元数据的一个永久性检查点。并非每个写操作都会更新这个文件。因为fsimage是大型文件，如果频繁的执行写操作会是系统运行缓慢。如果namenode出现故障，**可以先将fsimage文件载入内存重构新近的元数据，在执行编辑日志中的各项操作**。事实上namenode在启动阶段就是这样做的。

Fsimage文件包含文件系统中所有目录和文件inode的序列化信息。数据块存储在datanode中，但fsimage并不描述datanode，取而代之的是，namenode将这种块映射关系放在内存中。

如前所述，edits文件会越来越大，尽管这对namenode的运行没有影响，由于恢复编辑日志中的各项操作，namenode的重启操作会比较慢。

解决方案是运行辅助namenode为主namenode内存中的文件系统元数据创建检查点。

1. 辅助namenode请求主namenode**停止使用edits文件**，将操作记录写到一个新的文件中。
2. 辅助namenode**从主namenode获取fsimage 和edits文件**（HTTP GET）
3. 辅助namenode将fsimage文件加载到内存中，逐一执行edits文件中的操作，创建**新的fsimage文件**。
4. 辅助namenode**将新的fsimage文件传**送到主namenode（HTTP POST）；
5. 主namenode用**新的fsimage文件替代之前的fsimage文件，用步骤一的新edits文件代替旧的edits文**件。**同时更新fstime文件记录检查点执行时间。**

最终主namenode拥有最新的fsimage文件和一个更小的edits文件。

在namenode处于安全模式下，**管理员可调用hadoop -dfsadmin -saveNamespce 命令创建检查点。**

创建检查点的触发条件受两个配置参数控制，通常情况下，**辅助namenode每隔一个小时创建检查点**；此外**编辑日志的大小达到64MB时，及时未到一个小时也会创建检查点**。系统每隔五分钟检查一次编辑日志的大小。

**3.辅助namenode的目录结构**

**${fs**.checkpoint.dir**}/**

Current/

VERSION

Edits

Fsimage

Fstime

Previous.checkpoint/ //辅助namenode的检查点数据

VERSION

Edits

Fsimage

fstime

**4. datanode 的目录结构**

和namenode不同，datanode的存储目录是在初始化时创建的

${dfs.data.dir}/current/VERSION

/blk\_<id-1>

/blk\_<id\_1>.meta

………

/blk\_<id\_64>

/blk\_<id\_64>.meta

/subdir0

……………

/subdir63

Datanode 的VERSION文件和namenode 的VERSION文件非常相似。

Datanode的current目录中的其他文件都有blk\_前缀。包括两种文件类型：HDFS块文件和块的元数据文件（meta后缀）。

当目录的数据块的数量增加到一定规模，datanode会创建一个子目录来存放新的数据块及其元数据信息。

同一个datanode的每个磁盘的块都不会重复，不同的datanode之间块会重复。

### 10.1.2 安全模式

**Namenode启动时**，首先将映像文件fsimage载入内存，执行编辑日志edits中各项操作。一旦在内存中成功建立文件系统元数据的映像，则创建一**个新的fsimage文件（不需要借助辅助namenode）和一个空的编辑日志edit**s。此时namenode开始监听RPC和HTTP请求。但是此时，**namenode运行在安全模式**，即**namenode的文件系统对于客户端是只读的。**

系统中数据块的位置不是由namenode维护的，而是以块列表的方式存储在datanode中。在系统正常操作期间，namenode会在内存中保存所有块位置的映射信息。在安全模式下，各个datanode发送最新的块列表信息。 如果namenode认为发送更新信息的datanode节点过少，它会启动块复制进程，将数据块复制到新的datanode节点。在安全模式下，namenode不会向datanode发送任何块复制或块删除的指令。

在启动一个刚刚格式化的HDFS集群，因为系统中没有任何数据块，所以namenode不会进入安全模式。

查看namenode是否处于安全模式： $ hadoop dfsadmin -safemode get

在执行某个命令之前namenode先退出安全模式： $ hadoop dfsadmin -safemode wait

进入安全模式 $hadoop dfsadmin -safemode enter

离开安全模式 $hadoop dfsadmin -safemode leave

### 10.1.3 日志审计

HDFS的日志能够记录所有文件系统访问请求。对日志的审计是在log4j在INFO级别实现的。为了不与namenode的其他日志混在一起，最好配置log4j，将审计日志写到单独的文件中。

### 10.1.4 工具

**1.dfsadmin 工具**

Dfsadmin工具广泛，调用形式如下： hadoop dfsadmin

仅当用户具有超级用户权限，才可以使用这个工具修改HDFS的状态。

**2.fsck工具**

Hadoop 提供fsck工具来检查HDFS中文件的健康状况。

Fsck工具从给定的目录开始循环遍历文件系统的命名空间，并检查它所找到的所有文件，该工具获取文件数据块的元数据并找出问题或检查他们是否一致。

Fsck输出文件的大部分内容都可以理解，下面仅说明部分信息。

过多复制的块：复本数超出最小复本级别的块

仍需要复制的块：

错误复制的块

损坏的块

缺失的复本

损坏的块和缺失的块是最需要考虑的。默认fsck不会对这类块进行任何操作。但可以让fsck执行某一项操作：

**Move** 使用-move选项将受影响的文件移到HDFS的/lost+found目录。这些受影响的文件分裂成连续的块列表，可以帮助用户挽回损失。

**删除** 使用 -delete 删除受影响的文件

**查找**一个文件的数据块

**3. datanode块扫描器**

各个datanode会运行一个块扫扫描器，定期检测本节点上所有的块。可以依靠DataBlockScanner所维护的块列表依次扫描块。查看是否存在校验和错误。

在默认情况下，块扫描器每三周就会检测块。

**4，均衡器**

随着时间的推移，各个datanode上的块分布会越来越不均匀。不均衡的集群会降低MapReduce的本地性，导致部分datanode相对更加繁忙。

均衡器（balancer）程序是一个hadoop守护进程，它将块从忙碌的datanode移到相对空闲的datanode，从而重新分配块。同时坚持复本放置策略，将复本分散到不同机架。 调用下面指令启动均衡器：

$ start-balancer.sh

## 10.2 监控

监控的目标在检测集群在何时未提供所期望的服务。需要为集群预留额外的空间，即使有一小部分的宕机，也不会影响整个系统的运作。

除了以下即将介绍的工具以外，管理员还可以定期运行一些测试脚本。

### 10.2.1 日志 {}

1.设置日志级别

2.获取堆栈跟踪

### 10.2.2 度量 {}

度量从属于特定的上下文，目前，hadoop使用dfs，mapred，rpc和jvm四个上下文。

度量和计数器的差别在哪里？

主要区别在应用范围不同：度量由hadoop守护进程收集，而计数器先针对MapReduce任务收集，在针对整个作业进行汇总。

用户群也不同，度量为管理员服务，计数器主要为MapReduce用户服务。

两者的数据采集和聚集过程也不同。计数器是通过RPC心跳传播的，度量的收集机制独立于接受更新的组件。

### 10.2.3 java管理扩展（JMX）

Java管理扩展是一个标准的java api，可监控和管理应用。Hadoop包含多个托管bean，可以将hadoop度量发布给支持JMX的应用。

JDK 自带一个Jconsole的工具来浏览JVM中Mbean。

可以通过设置java的系统属性来启动远程访问JMX的所有选项，即编辑conf/hadoop-env.sh文件来进行设置。

## 10.3 维护

### 10.3.1 日常管理过程

**1.元数据备份**

方法一： 直接保存这些元数据文件的副本

方法二：整合到namenode上正在使用的文件中

最直接的元数据备份方法：是将辅助namenode上previous.checkpoint子目录存档，放到异地站点。还需要测试复本的一致性，测试方法很简单：启动一个本地的namenode守护进程，查看他是否能够将fsimage和edits文件载入内存。

**2.数据备份**

Hadoop中存储着海量数据，判断哪些数据需要备份在哪里备份就极具挑战性。关键在于给数据划分不同的优先级，无法重新生成的数据优先级最高。

不要误以为HDFS的复本技术足以胜任数据备份任务。

Distcp是一个理想的备份工具，其并行的文件复制功能可以将备份文件存储到其他HDFS集群（最好版本不同，避免hadoop软件纰漏而丢失数据）或其他hadoop文件系统（S3等）。

**3.文件系统检查（fsck）**

**4.文件系统均衡器**

### 10.3.2 委任和解除节点

**1.委任新节点**

预先指定一些经过审核的节点从中挑选新节点。

被允许连接到namenode的所有datenode放在一个文件中，文件名称有dfs.hosts属性指定。该文件放到namenode的本地文件系统中，每行对应一个datanode的网络地址。

Hadoop控制脚本使用slaves文件执行面向整个集群范围的操作。

Hadoop2.x 和hadoop1.x版本添加和解除节点的方式肯定不同，所以这边不记录

**2. 解除旧节点**

HDFS能够容忍namenode故障，但并不意味着允许随意终止datanode。

正确的方法： **用户将退出的datanode告知namenode，Hadoop系统就可在这些datanode停机之前将块复制到其他datanode**

### 10.3.3 升级

升级操作暗藏着数据丢失的风险。

规划过程最好包含在一个小型测试集群上的测试过程。

如果文件系统的布局并未改变，升级集群就非常容易。

成功升级版本之后：

1. 从集群中移除旧的配置文件；
2. 在代码和配置文件针对“被弃用（deprecation）”警告信息进行修复。

HDFS的数据和元数据升级

如果新旧HDFS的文件系统布局不同，则namenode无法正常工作

最可靠的判定文件系统升级是否必要的方法就是在一个测试集群在做实验。

升级HDFS会保存前一版本的元数据和数据复本，但这意味着要两倍的存储开销。因为datanode使用硬链接保存指向同一块的两个应用（当前版本和前一版本）。

用户可以保存前一个版本的文件系统，但无法回滚多个版本，因为新的一个升级任务，要删除前一版本，这个称为（定妥升级）

仅当文件系统健康时才可升级

升级之前最好清除临时文件