第38卷 第21期
Vol.38 No.21

计 算 机 工 程
Computer Engineering

2012年11月
November 2012

·软件技术与数据库·

文章编号：1000—3428(2012)21—0040—05 文献标识码：A 中图分类号：TP311.1

Namenode单点故障解决方案研究

邓 鹏，李枚毅，何 诚

(湘潭大学信息工程学院，湖南 湘潭 411105)

摘 要：针对Hadoop分布式文件系统中的Namenode单点故障问题，在研究Secondary Namenode机制、Backup Node机制和Facebook Avatar机制的基础上，提出一种Avatar改进方案。主节点向备用节点转发客户端请求，使用Zookeeper实现故障切换，从而解决Namenode的单点故障问题。利用Petri网模型在理论上证明了该方案的正确性，采用基于有限源的存储网络故障修复模型对该方案的可用性进行定量分析。实验结果表明，该方案具有不丢失数据、快速切换和故障自动恢复的特点。

关键词：云计算；单点故障；Hadoop分布式文件系统；高可用性；Petri网；故障恢复

Research on Namenode Single Point of Fault Solution

DENG Peng, LI Mei-yi, HE Cheng

(School of Information Engineering, Xiangtan University, Xiangtan 411105, China)

【Abstract】Based on the analysis of the Secondary Namenode mechanism, Backup Node mechanism and Facebook Avatar mechanism, an improved scheme of Avatar is proposed to solve the Single Point of Fault(SPOF)of Namenode existing in Hadoop Distributed File System(HDFS) which is a distributed file system of Hadoop. Client request is transmitted from the master node to standby node, and the Zookeeper is used to take over failover. And then the SPOF of Namenode is solved. The correctness of this program is verified by Petri net modal in theory and the quantitative analysis of its availability is conducted by storage network fault repairing model based on finite element method. Experimental result demonstrates the advantages of this program which are nonvolatile, fast failover and automatic failover.

【Key words】cloud computing; Single Point of Fault(SPOF); Hadoop Distributed File System(HDFS); High Availability(HA); Petri net; fault recovery

DOI: 10.3969/j.issn.1000-3428.2012.21.011

1 概述

随着云计算概念的普及，开源云计算平台Hadoop成为了研究的热点。Hadoop分布式文件系统(Hadoop Distributed File System, HDFS)的元数据服务器Namenode一直缺乏可靠的备份机制，导致HDFS面临可用性问题。

单点故障是指引起系统整体失效的部件，当该部件失效时，会使整个系统无法工作。在Hadoop集群中Namenode容易引发单点故障。Hadoop社区发布的Hadoop版本一直未能很好地解决Namenode的单点故障问题。一些使用Hadoop的厂商，如Facebook、淘宝等，针对Namenode的单点问题，提出了各自的解决方案。单点故障会导致停机，研究表明，在云存储系统中，每停机一小时，企业将损失15万美元到645万美元[1]。为了保证Hadoop集群达到24 h×365天的应用需求，需要构建具有高可用性的HDFS。因此，研究Namenode的单点故障问题，具有重要的理论价值和商业价值。本文分析当前HDFS的主要高可用性(High Availability, HA)方案，并在Facebook Avatar机制的基础上设计一种改进方案。

2 相关研究

Namenode是HDFS的元数据服务器，管理并协调数据节点Datanode的工作[2]，其内存中保存整个分布式文件系统的2类元数据：

(1)文件系统的名字空间，即系统目录树。

(2)数据块副本与Datanode的映射，即副本的位置。

第(1)类元数据在Namenode上定期持久化，保存为镜像文件。数据块副本的位置信息没有实现持久化，而是在Namenode启动时，通过接收Datanode的Blockreport获得。

Namenode的磁盘上保存操作日志。客户端对数据或元数据的读写操作被分解为一条或多条日志存入操作日志中。Namenode启动时，会将镜像和操作日志读取到内存中，把日志应用到名字空间中，使得Namenode中的名字空间恢复到宕机之前的状态。在Hadoop集群中一旦Namenode宕机，则整个集群不能再对外提供服务。Namenode的故障恢复时间主要耗费在镜像文件的加载以

基金项目：国家自然科学基金资助项目(61105052)

作者简介：邓 鹏(1988－)，男，硕士研究生、CCF会员，主研方向：云计算；李枚毅，教授、博士；何 诚，硕士研究生

收稿日期：2011-12-31 **修回日期：**2012-03-14 **E-mail：**cloudcomputing2006@163.com

及等待 Datanode 的 Blockreport。以 Facebook 的统计为例，规模为 1.5 亿个文件+1.5 亿个块+2 000 节点的集群，加载名字空间元数据和等待块位置映射信息上报，每个阶段所耗费的时间大概都在 20 min 左右。

3 Namenode 的 HA 方案分析

云存储网络中的 HA 方案主要包括硬件冗余和软件冗余。本文讨论的 HA 方案不包括复杂硬件冗余技术，而只关注基于廉价机器冗余的软件容错。

Hadoop 社区对 Namenode 采用的备份机制存在各种缺陷，很难投入实际应用。Hadoop0.20.x 中 Secondary Namenode 机制和 Hadoop0.21.0 中的 Backup Node 机制，切换时间都过长且有数据丢失的风险，是冷备机制。社交服务网站 FaceBook 针对 Namenode 的单点故障问题，提出 Avatar 机制，实现热备。以下对这 3 种方案做简要介绍和对比。

3.1 Secondary Namenode 机制

在 Hadoop0.20.x 中存在一个运行 Secondary Namenode 进程的节点。Secondary Namenode 定期从 Namenode 上下载镜像和日志进行合并，称为一次 checkpoint，将得到的新的镜像文件上传到 Namenode 替换原来的镜像文件，使 Namenode 上的镜像文件保持最新。Checkpoint 操作极耗内存，Secondary Nomenode 的 checkpoint 操作减轻了 Namenode 的负荷。

3.2 Backup Node 机制

在 Hadoop0.21.0 中，Secondary Namenode 被一个叫做 Checkpointnode 的角色所取代，并且新增了一个 Backup Node 的角色。

Backup Node 和 Namenode 之间有一个流通道，Namenode 通过这个通道直接将日志写入到 Backup Node，实现名字空间的同步。

3.3 Facebook Avatar 机制

Facebook Avatar 是应用在 Hadoop0.20.2 上的补丁程序。Avatar 机制主要包括一个 Primary Namenode、一个 Standby Namenode 和一个 NFS 服务器，Primary Namenode 代替原来的 Namenode，负责对外提供服务。Standby Namenode 是一个处于 Safemode(安全模式)的 Namenode 节点(不接受客户端请求)，这 2 个 Namenode 都可以接收 Blockreport，并通过 NFS 服务器来同步镜像和日志，保持名字空间一致。在 Primary Namenode 宕机时，Standby Namenode 切换为 Primary Namenode 耗时很短。

表 1 对上述 3 种方案的备份方式、运行机制、切换流程和优劣势进行比较(下文中 Secondary Namenode 简写为 SNN，Backup Node 简写为 BN，Primary Namenode 简写为 PN，Standby Namenode 简写为 SN)。

表 1 方案性能对比

方案名	备份方式	运行机制	切换流程	优势	劣势
Secondary Namenode 机制	冷备& 手动切换	(1)SNN 通过 RPC 调用 Namenode 上的 rollEditLog()方法，建立临时日志文件 edits.new； (2)SNN 从 Namenode 上下载镜像和日志文件合并； (3)将合并后的镜像上传到 Name node 上，覆盖旧的镜像	(1)修改 IP，在 SNN 上手动启动 Namenode，进入安全模式； (2)在 SNN 上手动执行命令读取 Checkpoint 文件，接收 Datanode 的 Blockreport 信息； (3)SNN 切换为 Namenode，并对外提供服务	使用简单方便，无需开发，简单配置即可	有大量数据丢失的风险；内存中无元数据同步，切换需很长时间，SNN 不适合作 Name node 的备用节点
Backup Node 机制	冷备& 手动切换	(1)握手，BN 通过 RPC 调用获取 NamespaceInfo，并进行版本验证； (2)初始化，设置标志 ready 值为 false，不接受请求； (3)注册，BN 作为存储路径注册到 Namenode 上； (4)创建检查点加载镜像文件	(1)修改 IP，手动启动 BN 为 Name node； (2)BN 等待 Datanode 上报块信息； (3)BN 成功切换为 Namenode，并对外提供服务	Checkpoint 的效率更高；名字空间元数据同步更新，恢复时可以保证与最新的元数据一致	块位置映射信息未在内存同步；切换后需要等待 Blockreport，切换时间较长
Avatar 机制	热备& 手动切换	(1)PN 在运行过程中将日志和镜像同步到 NFS 服务器上； (2)SN 的 Standby 线程定期做 Check Point，Ingest 线程则周期性地读取 PN 同步到 NFS 上的日志以保持元数据同步； (3)SN 在 PN 运行期间处于安全模式，接收 Blockreport，不响应请求	(1)手动执行命名切换 SN 为 PN 的； (2)SN 最后一次读取 NFS 上的日志；退出保护模式，接收客户端的请求； (3)全局变量 currentAvatar 被设置为 Avatar.ACTIVE，即 Primary，切换过程结束	实现热备；SN 切换为 PN 能在 10 ms~1 min 内完成，迅速地切换；引入虚拟 IP 切换后无需手动更改 IP 地址	NFS 的数据同步需要一定时间，会有几秒的数据丢失；引入的虚拟 IP 很不稳定；NFS 服务器是个新单点。

4 Avatar 改进方案的设计与实现

依据表 1 可知，现有方案的缺陷主要体现在数据丢失、故障恢复时间过长和需要人工干预等方面，这使得这些方案并不满足大规模应用的要求。针对以上方案的不足，本文在 Avatar 机制基础上提出一种不丢失数据、快速并且能自动切换的热备方案。

4.1 设计思想

本文方案在 Facebook Avatar 基础上改进，主要基于如下 3 个方面进行设计：

(1)不丢失数据

分析表 1 中 3 种方案可知：因为网络的延迟和传输的时间间隔，元数据依赖网络传输来实现同步会丢失数据。

因此，本文方案采用同步请求的方式来同步元数据产生过程，而不是把已经生成的元数据通过网络传送。每一个请求作为一个原子操作，要么在 2 台 Namenode 上全部执行成功，要么全部失败，元数据的产生过程一致，元数据保持最终一致性。

(2)自动切换

表 1 中的方案在切换 Namenode 时都需要手动键入命令或者修改 IP，不利于集群的管理。本文方案采用 Zookeeper 实现自动切换。Zookeeper 是一个高效、可靠的协同工作系统，实现了类 Paxos 算法，支持 quorum 机制，提供分布式锁服务，Zookeeper 不依赖于 HDFS，是多服务器的系统，没有单点故障的问题。本文方案在 Zookeeper 上创建一个锁，抢占到锁的节点即为 Active Node(当前对外提供服务的 Namenode)，同时 Zookeeper 保存当前 Active Node 的 IP，客户端和 Datanode 通过读取 Zookeeper 中的数据来识别当前的 Active Node。

(3)快速切换

保持 Avatar 原有的主备节点同时接收 Datanode 心跳信息的功能，block 映射信息保持一致，备用节点切换成主节点后无需其他处理便可接管工作。

4.2 方案实现过程

改进的 Avatar 架构如图 1 所示，在 Avatar 的基础上，去掉 NFS 服务器，checkpoint 操作由负载较小的 SN 执行。PN 把接收到的用户请求转发给 SN，只有当收到 SN 发送来的操作成功消息，PN 才认为本次请求成功，否则回滚并重新执行；SN 不再一直处于安全模式，可以接收来自 PN 转发的用户请求，并对元数据做相应的修改，但不对 Datanode 反馈结果也不发送命令。对外服务由 PN 来执行。

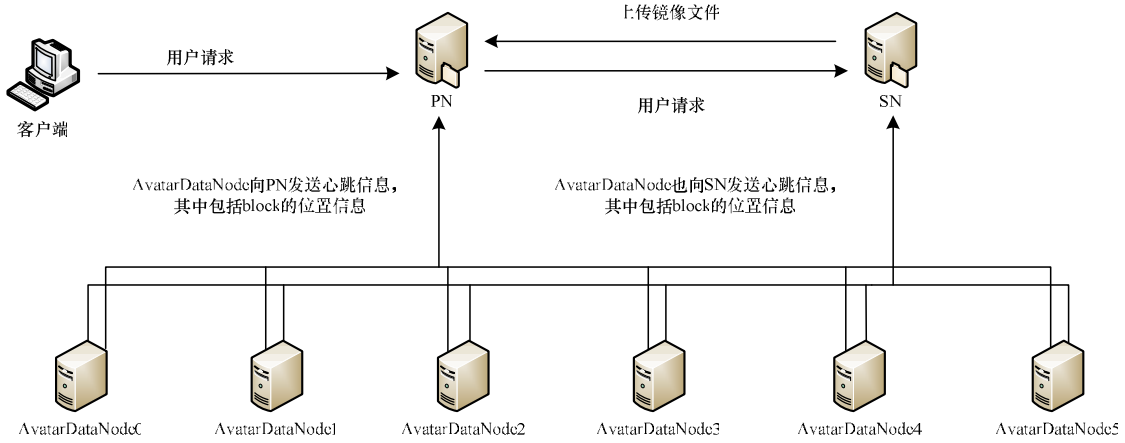

图 1 Avatar 改进方案的架构

实现步骤如下：

Step1 打补丁。在 Hadoop0.20.2 上打上 Avatar 补丁。

Step2 实现转发请求。修改 Namenode 的客户端协议接口实现方法：判断本机是否为当前工作的 Active Node，如果是，则通过 RPC 调用 Standby 中的同名方法，成功完成后发送执行成功消息给 Primary，Primary 收到消息后才正式提交日志，响应请求。如果本机不是 Active Node，则直接执行后续操作。

Step3 实现 Standby 接收请求。修改 Avatar Node，取消 Standby 常驻安全模式；修改 Standby 继承 Namenode，使之拥有接收请求的功能，取消关于 Active Node 的唯一性检测，弃用定期从 NFS 上读取数据的 Injest 线程和 NFS 服务器。

Step4 实现原始元数据同步。SN 开机初始化结束后，启用输入流从 PN 的内存中抓取(PULL)一次元数据到 SN 的内存中。

Step5 SN 实现定期 checkpoint 操作。由于 SN 相对负载比较轻并保存一份与 PN 完全一致的元数据，因此由它定期执行(默认每隔一小时)checkpoint，产生新的磁盘镜像，并将新的磁盘镜像上传到 PN 上，为 PN 分担一部分负载。这份磁盘镜像主要用于重新开机后加载元数据。

Step6 使用 Zookeeper 实现自动切换。Primary 启动初始化过程中在 Zookeeper 上创建一个 EPHEMERAL Znode(临时节点)，保存 Active Node 的 IP 和端口号。当 Active Node 宕机后，Znode 节点自动删除。

Standby 启用线程检测 Znode 是否存在，如果不存在，则创建一个新 Znode 并把自身的 IP 地址和端口写入此节点中，并调用相关脚本切换成为 PN。这一切换过程无需人工干预。

Step7 切换过程 IP 变换对客户端和 Datanode 透明。在客户端和 Datanode 中，启用一个线程实时读取 Znode 中的 IP 和端口地址，作为连接的目标地址。

Step8 重新编译并部署到集群。

4.3 Petri 网建模验证

应用 Petri 网建立改进 HDFS 双机热备的事故模型，将系统的事件作为顶库所，逐步找出导致这一事件的所有可能因素作为中间库所和底库所。依据 Petri 网的关联矩阵求出最小割集，最小割集中的底事件同时发生时，顶事件必然发生，最小割集中任一底事件不发生时，顶事件也不发生[3]，运用这一理论验证 Avatar 改进方案的正确性。图 2 为 HDFS 双机热备的 Petri 网模型，表 2 为对应库所的含义。

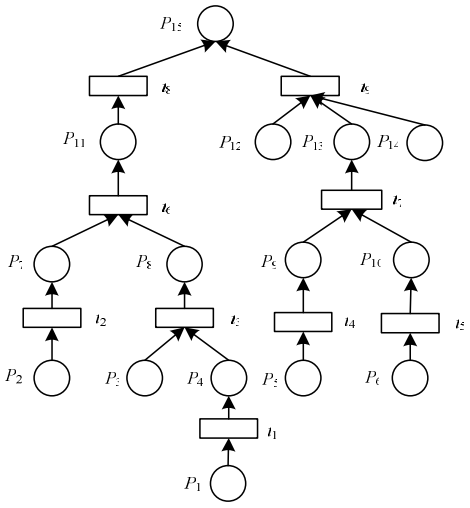

图 2 HDFS 双机热备的 Petri 网模型

表 2 库所及其含义

库所	含义
P_1	接受的请求相同
P_2	同时接受 Blockreport
P_3	原始名字空间相同
P_4	相同的操作序列
P_5	快速 checkpoint
P_6	实时流式传输日志
P_7	数据块副本映射信息同步
P_8	名字空间同步
P_9	日志与旧的镜像文件合并
P_{10}	操作日志同步
P_{11}	内存中元数据同步
P_{12}	数据快速加载到内存
P_{13}	磁盘镜像同步
P_{14}	快速接收 Blockreport
P_{15}	HDFS 双机热备

图 2 的 Petri 网模型对应的关联矩阵 $\boldsymbol{A}^{\mathrm{T}}$ 为：

$$\boldsymbol{A}^{\mathrm{T}}=\begin{bmatrix}-1&0&0&0&0&0&0&0&0\\0&-1&0&0&0&0&0&0&0\\0&0&-1&0&0&0&0&0&0\\1&0&-1&0&0&0&0&0&0\\0&0&0&-1&0&0&0&0&0\\0&0&0&0&-1&0&0&0&0\\0&1&0&0&0&-1&0&0&0\\0&0&1&0&0&-1&0&0&0\\0&0&0&1&0&0&-1&0&0\\0&0&0&0&1&0&-1&0&0\\0&0&0&0&0&1&0&-1&0\\0&0&0&0&0&0&0&0&-1\\0&0&0&0&0&0&1&0&-1\\0&0&0&0&0&0&0&0&-1\\0&0&0&0&0&0&0&1&1\end{bmatrix}$$

依据关联矩阵的最小割集求解算法[4]，求得最小割集为{P_1,P_2,P_3}、{P_5,P_6,P_{12},P_{14}}。依据表 2 可知，{P_1,P_2,P_3}中 P_1 的含义是接收相同的请求，在 Avatar 改进方案中满足该条件；P_2 的含义是同时接收 Blockreport，这是原有的 Avatar 机制中已经实现的功能；P_3 的含义是原始元数据相同，在实现过程中的 Step4 中实现。

综上所述，改进的 Avatar 方案满足最小割集{P_1,P_2,P_3}的所有条件，顶事件 HDFS 双机热备必然发生，即 Avatar 改进方案可以实现预期的目标，解决 Namenode 单点故障。

4.4 可用性定量分析

采用 RAS(Reliability, Availability, Serviceability)衡量系统的可用性[5-6]，可靠性一般用平均故障间隔(Mean Time to Failure, MTTF)评估，可维护性一般用故障平均修复时间(Mean Time to Repair, MTTR)评估，可用性是指在允许最大故障节点数内，系统按规范成功运行的概率，一般用系统正常运行时间占总时间的比例表示[7]。假设每个节点的故障和恢复概率满足负指数分布，则系统故障率为 $\lambda=1/M_{\mathrm{MTTF}}$，修复率为 $\mu=1/M_{\mathrm{MTTR}}$，单节点可用性表示为：

$$A=\frac{\mu}{\mu+\lambda} \tag{1}$$

假设网络是可靠的，则 Avatar 改进方案可能的 3 种状态如图 3 所示。

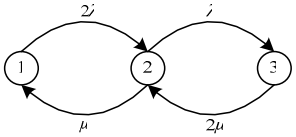

图 3 Avatar 改进方案的状态转换

(1)状态 1，PN 和 SN 都正常运行，系统能正常工作。

(2)状态 2，PN 和 SN 有一个失效，另一个正常，系统能正常工作。

(3)状态 3，PN 和 SN 都失效，系统不能工作。

令状态 1、状态 2 和状态 3 的稳定概率分别为 P_1、P_2 和 P_3，Avatar 改进方案的可用度 $A=1-p_3$。

假设系统可并行修复 2 个故障节点，结合式(1)，依据有限源存储网络的故障修复模型可用度计算公式[8]可得：

$$A=1-p_3=1-\left(\frac{\lambda}{\mu+\lambda}\right)^2 \tag{2}$$

依据如下假设：

(1)PN 节点和 SN 节点平均故障时间间隔为：

$M_{\mathrm{MTTF}}\geqslant 1\ 000$ h

失效率为：

$$\lambda=\frac{1}{M_{\mathrm{MTTF}}}=0.001$$

(2)PN 和 SN 节点平均故障修复时间为 2 h，修复率为：

$\mu=0.5$

基于上述假设，由式(2)可得：

$$A=1-\left(\frac{\lambda}{\mu+\lambda}\right)^2=99.999\ 60\%$$

Avatar 改进方案具有高可用性。

5 Avatar 改进方案的测试与分析

测试环境包括 18 台物理服务器，配置为 Intel Core i3 2.93 GHz CPU，内存 4 GB，千兆以太网卡，一共 18 个节点，1 个 PN 节点，1 个 SN 节点，10 个 Datanode 节点、

6 个 Client 节点。

5.1 时间切换测试

在同一物理集群的不同数据量下，对 Hadoop0.20.2、Hadoop0.21.0，Avatar 和 Avatar 改进方案进行多次切换测试，切换时间对比如图 4 所示。

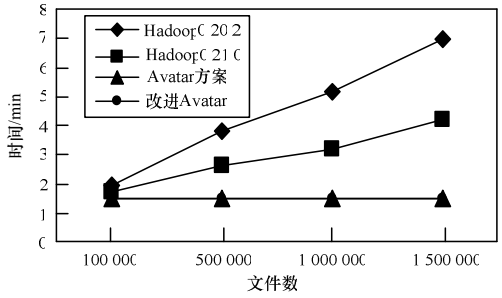

图 4 不同方案的切换时间对比

从图 4 可以看出，采用 Secondary Namenode 机制的 Hadoop0.20.2 随着集群文件数增加，切换时间基本呈线性增加。采用 Backup Node 机制的 Hadoop0.21.0 接近线性增长，因为无需重新加载名字空间，所以总体切换时间约为 Hadoop0.20.2 的一半。Avatar 的切换时间与集群文件数关系不大，稳定在 13 s 左右。Avatar 改进方案和原始的 Avatar 方案切换时间相差不大，也在 13 s 左右。由此可见，改进方案中采用 Zookeeper 能实现快速切换，并避免了使用虚拟 IP 的不稳定性，且切换过程完全自动，无需人工干预，最重要的是整个集群中不存在为了解决单点问题而引入的新单点(类似原始 Avatar 方案中的 NFS 节点)，这些都是改进 Avatar 的优势。

5.2 数据完整性测试

在同一个物理集群上，客户端向不同版本的 Hadoop 写数据，每台客户端写 100 个 128 MB 的文件。文件名从 10001~10100。客户端将文件写入 HDFS 期间，拔掉当前处于 Namenode 状态的机器网线，一段时间后 Namnode 切换到另一节点上，表 3 为写入测试数据的记录结果。

表 3 写入测试数据的记录结果

方案	准备写入的文件	切换前写入成功的文件	切换后写入成功的文件	写入数据结束后集群上可以查看到的文件	丢失的文件
Hadoop0.20.2	10001~10100	10001~10011	10012~10100	10012~10100	11
Hadoop0.21.0	10001~10100	10001~10010	10011~10100	10001~10009、10011~10100	1
Avatar 方案	10001~10100	10001~10010	10011~10100	10001~10008、10011~10100	2
Avatar 改进方案	10001~10100	10001~10009	10010~10100	10001~10100	0

表 3 中不同版本的 Hadoop 在写数据时切换 Namenode 的过程中系统处于无 Namenode 的状态，写入数据的客户端找不到连接服务器而导致数据写入失败，但写入失败的异常可以捕获，采取重写机制(检测到写入失败异常，即反复重试，直到写入成功)避免数据拒绝写入的问题。

表 3 中丢失的文件(第 6 列)是通过切换前写入成功的文件(第 3 列)加切换后写入成功的文件(第 4 列)与写入数据结束后集群上可查看到的文件(第 5 列)对比得出的结果。这些文件是因为其对应的元数据丢失，导致 Hadoop 系统将这些文件自动删除，造成数据永久丢失，数据丢失是没有异常抛出的。

由表 3 记录结果可知，Hadoop0.20.2 默认，1 h 做一次 checkpoint，依此同步元数据，所以，在本次测试中，切换前写入的数据全部丢失。Hadoop0.21.0 切换过程需要加载 block 映射信息，另外还需人工切换，并且因为切换前的最后几条操作日志未能同步，导致少量数据丢失。Avatar 方案因为 NFS 每 3 秒同步一次元数据，所以，会有数据的丢失，本次测试中丢失 2 个文件。Avatar 改进方案自动切换并且因为实现元数据的同步，所以没有数据丢失。

6 结束语

Namenode 单点一直是困扰 Hadoop 使用厂商的问题，也在一定程度上约束了 Hadoop 的广泛应用。本文分析了当前主要的 HA 方案：Secondary Namenode 机制，Backup Node 机制和 Facebook Avatar 机制，总结这些方案的优劣势，并在此基础上提出 Avatar 改进方案，具有不丢失数据、快速故障恢复、自动切换等优势，解决了 Namenode 的单点问题，提高了 HDFS 的可用性，对于 Hadoop 的推广使用有着重要的意义。今后将改进 Avatar 系统，将其应用于大规模云计算中并保证系统的稳定性。

参考文献

[1] Barraza O. Achieving 99.9998% Storage Uptime and Availability. Dot Hill Systems Corp[EB/OL]. (2002-08-19). http://www.dothill.com/products/whitepapers/5-9s_wp.pdf.

[2] The Apache Software Foundation. HDFS Architecture Guide[EB/OL]. [2011-05-04]. http://hadoop.apache.org/common/docs/current/hdfs_design.html.

[3] Bonet P, Llado C M, Puijanger R, et al. PIPE v2.5: A Petri Net Tool for Performance Modeling[EB/OL]. (2010-07-03). http://www.doc.ic.ac.uk/wjk.

[4] 武 滢, 谢里阳, 李进冬. 应用 Petri 网的关联矩阵求最小割集的新方法[J]. 中国机械工程, 2008, 19(9): 1045-1047.

[5] Xu Shiyi. On Dependability of Computing Systems[J]. Journal of Computer Science and Technology, 1999, 14(2): 198-206.

[6] Anderson T, Laprie J C. Dependability: Basic Concepts and Terminology[M]. [S. l.]: Springer-Verlag, 1990.

[7] Rueda A, Pawlak M. Pioneers of the Reliability Theories of the Past 50 Years[C]//Proc. of RAMS'04. New York, USA: ACM Press, 2004: 102-109.

[8] 韩德志, 傅 丰. 高可用存储网络关键技术的研究[M]. 北京: 科学出版社, 2009.

编辑 陆燕菲