Journal of Computer Applications
计算机应用,2012,32(8):2150-2153,2158
ISSN 1001-9081
CODEN JYIIDU
2012-08-01
http://www.joca.cn

文章编号:1001-9081(2012)08-2150-04
doi:10.3724/SP.J.1087.2012.02150

基于纠删码和动态副本策略的 HDFS 改进系统

李晓恺[1,2*],代　翔[1,2],李文杰[1,2],崔　喆[1]
(1.中国科学院 成都计算机应用研究所,成都 610041;　2.中国科学院 研究生院,北京 100049)
(*通信作者电子邮箱 gucas1988@gmail.com)

摘　要:为了让 Hadoop 分布式文件系统(HDFS)达到更高的存储效率以及更加优化的负载均衡能力,针对 HDFS 的多副本存储技术提出了改进方案——Noah。Noah 引入了编码和译码模块,对 HDFS 中的 block 进行编码分解,生成更多数量的数据分片(section),并随机地分散保存到集群当中,替代原有系统的多副本容灾策略;在集群出现节点失效的情况下,通过收集与失效 block 相关的任意 70% 左右的 section 进行原始数据的恢复;同时根据分布式集群运行情况以及对副本数目需求的不同采用动态副本策略。通过相关的集群实验,表明 Noah 在容灾效率、负载均衡、存储成本以及安全性上对 HDFS 作了相应的优化。

关键词: Hadoop 分布式文件系统;分布式存储;数据容灾;负载均衡;动态副本
中图分类号: TP393.027.2;TP311.133.1　　**文献标志码:**A

Improved HDFS scheme based on erasure code and dynamical-replication system

LI Xiao-kai[1,2*], DAI Xiang[1,2], LI Wen-jie[1,2], CUI Zhe[1]
(1. *Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu Sichuan* 610041, *China*;
2. *Graduate University of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: In order to improve the storage efficiency of Hadoop Distributed File System (HDFS) and its load balance ability, this paper presented an improved solution named Noah to replace the original multiple-replication strategy. Noah introduced a coding module to HDFS. Instead of adopting the multiple-replication strategy by the original system, the module encoded every data block of HDFS into a greater number of data sections (pieces), and saved them dispersedly into the clusters of the storage system in distributed fashion. In the case of cluster failure, the original data would be recovered via decoding by collecting any 70% of the sections, while the dynamic replication strategy also worked synchronously, in which the amount of copies would dynamically change with the demand. The experimental results in analogous clusters of storage system show the feasibility and advantages of new measures in proposed solution.

Key words: Hadoop Distributed File System (HDFS); distributed storage; data disaster recovery; load-balance; dynamic replication

0　引言

互联网时代的来临带来了数据的疯狂增长,也使得传统的数据存储模型遭遇了新的挑战:磁盘容量快速增加的同时,其访问速度并没有显著的提高,带来的直接后果是传统的数据存储方式无法胜任互联网时代的海量数据处理任务,Google 率先设计并实现了一个新的分布式文件系统——Google File System (GFS)[1]来解决了上述问题,继而 Apache 发布了 GFS 的开源版本——Hadoop[2],它的核心由 Hadoop 分布式文件系统(Hadoop Distributed File System, HDFS)和 Map-Reduce[3]两部分组成,前者负责存储,后者负责分析处理。

Hadoop 的应用非常广泛,但是它还是存在一些比较突出的问题:存储效率相对较低、集群负载均衡能力比较差、NameNode 单点故障、JobTracer 负载过重,以及小文件问题等。一些高校和科研机构针对这些缺陷做了相关的研究和优化[4-6],文献[7]分析了 5 个影响 Hadoop 性能的因素,分别为计算模型、I/O 模型、数据解析、索引和调度,同时针对这 5 个因素提出了相应的提高性能的方法,最后实验证明,通过这些方法可以将 Hadoop 性能提高 2.5 到 3.5 倍。文献[8-9]从系统层面解决 HDFS 小文件问题,但更多的是集中在 Map-Reduce 部分,对 HDFS 优化的相关研究工作开展的相对较少。而 HDFS 是整个 Hadoop 的基础,它的效率和性能直接影响着 Map-Reduce 的效率,因此对 HDFS 进行相关的研究和改进是非常必要的。针对 HDFS 存储效率不高以及负载均衡能力不足的关键问题,本文提出了一种基于数据编码策略和动态副本策略相结合的改进方案 Noah。实验测试表明,Noah 在保证了集群数据安全性的同时,提高了 HDFS 的数据恢复的速度,优化了 HDFS 的负载均衡能力,降低了 Hadoop 整体的存储成本。这对于提升和改进 Hadoop 及其相关的云计算架构体系的实际运行效率具有明显的意义。

1　HDFS 框架知识

1.1　HDFS 架构[10]

HDFS 采用 master/slave 架构,如图 1 所示。一个 HDFS 集群是由一个 NameNode[11]节点和一定数量的 DataNode[12]节点组成,DataNode 部署在若干机架(Rack)上。NameNode

收稿日期:2012-02-22;**修回日期:**2012-03-26。　**基金项目:**国家 863 计划项目(2008AA01Z402)。
作者简介:李晓恺(1988-),男,安徽池州人,硕士研究生,主要研究方向:分布式存储、分布式计算;　代翔(1983-),男,河南信阳人,博士研究生,主要研究方向:计算机网络、信息安全;　李文杰(1987-),男,湖南益阳人,硕士研究生,主要研究方向:信息编码、分布式存储;　崔喆(1970-),男,四川巴中人,研究员,主要研究方向:软件工程、计算机网络、信息安全。

负责管理文件系统的名字空间(NameSpace)以及客户端对文件的访问。集群中的 DataNode 一般是一个节点一个,负责管理它所在节点上的存储。HDFS 暴露了文件系统的名字空间,用户能够以文件的形式在上面存储数据。从内部看,一个文件被分成一个或多个固定大小的数据块 block(HDFS 中的基本存储单位),这些块存储在一组 DataNode 上。NameNode 执行文件系统的名字空间操作,比如打开、关闭、重命名文件或目录;也负责确定数据块到具体 DataNode 节点的映射。DataNode 负责处理文件系统客户端的读写请求,在 NameNode 的统一调度下进行数据块的创建、删除和复制。集群中单一 NameNode 的结构大大简化了系统的架构,NameNode 是所有 HDFS 元数据的仲裁者和管理者,用户数据永远不会流过 NameNode。

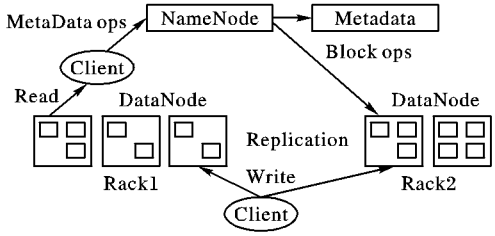

图 1　HDFS 架构

1.2　多副本容灾策略

HDFS 被设计成一个可靠的、容错的分布式文件系统,它允许节点失效的情况出现,所以 HDFS 需要一套自己的容灾机制。HDFS 是通过多副本技术来实现容灾的,默认情况下,HDFS 会将一个 block 的 3 份副本分别保存在集群的不同 DataNode 上。NameNode 全权管理数据块的复制,它周期性地从集群中的每个 DataNode 接收心跳信号和块状态报告(blockreport)。接收到心跳信号意味着该 DataNode 节点工作正常。当 DataNode 出现失效的时候,NameNode 就会将该失效节点上的 block 重新恢复到其他节点上以维持副本的数目。

1.3　多副本策略存在的问题

多副本在 HDFS 中的作用主要有:

1)容错,保证系统的可靠性。

2)实现负载均衡。Hadoop 的核心思想是“将计算移动到数据上”,HDFS 的负载均衡是通过将数据平均分配到各个 DataNode 上实现的;一个节点拥有数据的多少直接决定了它计算负载的大小;在运行过程中,通过移动副本来转移负载。

多副本策略的缺陷主要体现在:

1)HDFS 整体存储成本较高。以常用的 3 副本策略为例,实际所需存储空间为存储数据容量的 3 倍,直接增加了存储硬件成本和文件索引的建立时间。此外,每一份 block 的位置信息均需保存在 NameNode 的内存中,block 数量过多将对 NameNode 的内存造成非常大的压力,降低了系统的可扩展性。

2)负载均衡能力不足。多副本策略对系统中的数据均维持相同数目的副本数量而不去加以区别对待,系统无法根据外界需求的变化动态地去改变副本的数目,这直接导致了系统有限的负载均衡能力。

2　基于 GE 码和动态副本策略的方案 Noah

2.1　Noah 的设计思想

通过上述分析,固定数目的副本策略是造成 HDFS 的存储成本较高以及负载均衡能力不强的主要原因。本文试图在保证数据安全性的同时,降低系统的存储成本并且提供更加灵活的数据负载均衡能力的解决方案。

Noah 的设计思想基于:

1)基于编码的容灾机制取代多副本容灾机制;

2)在新的容灾机制的基础上,采用灵活的副本数目动态变化的策略,取代原有固定数目副本策略。

2.2　基于 GE 码的容灾机制

2.2.1　GE 码

纠删码是一种线性数据编码方式,因为它可以通过引入较少的数据冗余对数据传输和存储提供较高的可靠性保证,纠删码编码的方法被引进到存储系统中。(n,m) 纠删码将 m 个输入的源数据块编码产生 $n(n>m)$ 个数据块,其中任意 m 个数据块都能通过解码重构出原始数据。应用这种编码方法,系统可以允许的数据损坏上限是 $(n-m)$ 个数据块,编译码过程如图 2 所示。

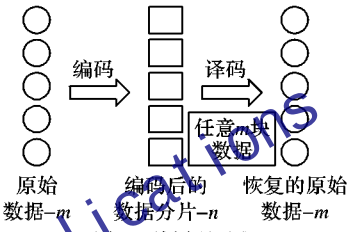

图 2　编译码过程

文献[13]中描述了一种新的纠删码——GE 码,这是一类参数为 $[n,k,(n-k)/2+1]$ 的垂直阵列码。码长为 n,信息位为 k,汉明距离为 $(n-k)/2+1$,所以 GE 码可以纠正 $(n-k)/2$ 个删除错。GE 码用一个矩阵 $\boldsymbol{A}=[a_{i,j}](0\leqslant i<n,0\leqslant j<n)$ 来表示。其中 $[a_{i,j}](0\leqslant i<n-t,0\leqslant j<n)$ 用于存储有效信息,称为信息元;$[a_{i,j}](n-t\leqslant i<n,0\leqslant j<n)$ 用于存储校验元。

GE 码的编码过程即根据信息元生成校验元,步骤如下:

1) 根据矩阵 $\boldsymbol{A}$,构造一个 $(n-t)\times(n-1)$ 的矩阵 $\boldsymbol{A}'$。

2) 生成一个等重码 $C(n-t,n-1,t,(n-t)t/(n-1))$。

3) 将这些等重码上的元素随机地划分为 t 个集合。

4) E 码的校验元就可以由式(1) 生成。

$$a_{i,j}=\bigoplus_{s=0}^{n-k-1} d_{i-(n-k),s};\ i=n-k,\cdots,n-1,j=0,\cdots,n-1 \tag{1}$$

GE 码的译码过程是根据可用的校验元生成信息元,步骤如下:

1)将阵列中所有的校验元的状态标记为“可用”。

2)随机选择一个状态为“可用”的校验元,检查它所校验的信息元是否被删除。如果其中没有信息元被删除,那么将此校验元标记为“无用”;如果其中有且仅有一个信息元被删除,同样将此校验元标记为“无用”,并且此被删除的信息元则可由式(2)恢复。

$$d_{i',j'}=a_{i,j}\oplus\left(\bigoplus_{s=0}^{n-k-1} d_{i-(n-k),s}\right);\ i-(n-k)\neq i',s\neq j' \tag{2}$$

3)重复 2),直到阵列中不再有状态为“可用”的校验元,或者所有状态为“可用”的校验元都校验了至少两个被删除的信息元。

选取 GE 码作为 Noah 的编码方案的主要原因在于:编译

码速度快(线性复杂度算法);数据容灾能力强,文献[13]中给出了该编码方案容灾能力的证明,依据设置的编码初始参数不同,容灾能力在区间[0.30,0.35]上,即只需 70% 的分片数据即可完成对原始数据的恢复。

2.2.2 Noah 的容灾机制

Noah 采用 GE 码设计了一套新的容灾机制,抛弃 HDFS 的"多份镜像复制"策略,使用编码分片方案,对 HDFS 中的 block 数据进行编码得到数据分片(section),使得整个系统集群对于绝大多数数据只保存与该 block 相对应的 section 数据。Noah 的容灾机制核心流程如图 3 所示。

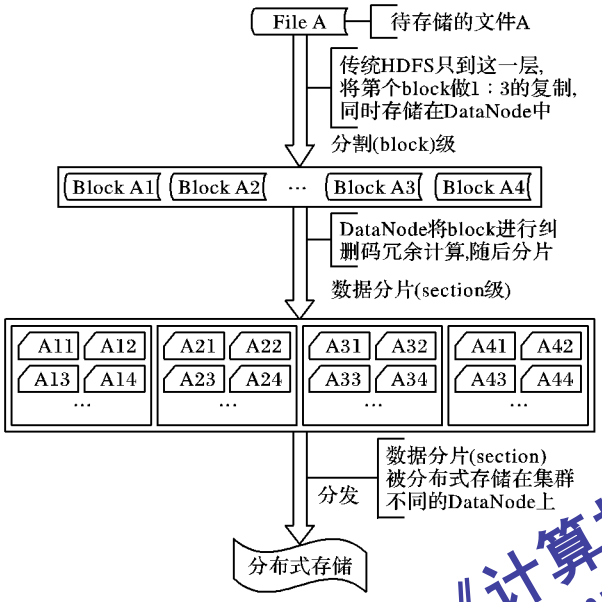

图 3 Noah 的容灾机制核心流程

2.2.3 数据分片(section)的生成

图 4 所显示的 block 的冗余信息是通过 2.2.1 节中描述的 GE 码编码过程生成的校验元信息,编码后的 block 会被分割成一定数量的 section,section 由数据部分和冗余编码部分组成,数据部分可以直接使用,冗余编码部分在恢复数据时使用。方案采用的具体参数为 GE[31,21,6]码,即码长是 31,信息位是 21,汉明距离是 6 的 GE 码。

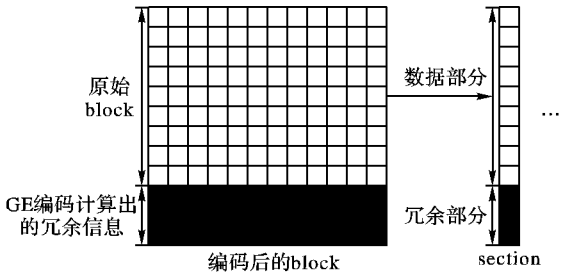

图 4 数据分片(section)的生成过程

2.2.4 Noah 的数据存储策略

Noah 在系统中不再保存 block 的多份副本,它将 block 数据通过 GE 码的编码过程分解成一定数量的 section,之后将这些 section 下发到集群当中。具体流程如下:DataNode 对节点上的 block 进行编码分片处理,得到一定数量(由编码原则及现实需要指定参数的设置)的 section,之后将这些 section 采用随机策略分散保存到集群不同的 DataNode 上,并将存储位置通知 NameNode,NameNode 在元数据中记录下 block 和 section 之间的映射关系以及 section 的存储位置。至此完成对数据的存储工作。

2.2.5 Noah 的数据恢复策略

Noah 的数据恢复策略不同于 HDFS 的副本复制策略,它分两个阶段完成数据恢复:1)section 收集;2)对 section 进行译码操作恢复损坏的 block。在 section 收集阶段,NameNode 会首先查询损失 block 所对应的 section 在集群中的分布,选择其中任意 70% 的 section 集中到一个 DataNode 上进行译码恢复操作。

当 block 数据发生丢失或损坏,NameNode 将查询该丢失的 block 所对应的 section 所在位置,让这些 section 所在的 DataNode 分别将其拥有的 section 发送给指定的 DataNode;该 DataNode 在接收完指定数量的 section 之后进行译码解析操作,从而恢复得到丢失的 block 信息。

当 section 丢失或损坏后,NameNode 将记录下该 section 信息,但并不马上进行恢复操作;当某一 block 的 section 丢失或损坏数量超过一个阈值(一般为与该 block 相关的 section 数量的 30%)时,拥有此 block 的 DataNode 将会把此 block 重新进行编码分片,并将重新分片的 section 下发到不同的 DataNode 上,从而保证 section 的数量维持在一个安全的阈值。

2.3 动态副本策略

动态副本策略保证在系统运行状态下,根据来自客户对数据的不同需求,系统在 Noah 的容灾机制的支持下,针对热点数据通过 section 动态地生成和删除副本,动态地部署副本,使系统维持在一个良好的负载均衡状态。

动态副本策略包括动态副本生成策略和动态副本部署策略。

2.3.1 动态副本生成策略

动态副本生成策略通过区别对待集群中的数据,动态改变副本数目来达到系统整体的相对负载均衡。系统不再为每一个 block 保留固定数量的副本,它会根据系统运行时对副本的需求情况动态地改变副本数量。

NameNode 动态维护一张表,该表记录着一段时间内上层应用对每一个 block 的访问等待队列数目。

1)当 block 副本数量满足应用请求的情况时,对该 block 的应用请求不需要在等待队列中进行排队,可以直接访问。

2)当 block 副本数量不能满足应用请求时,新的应用请求将会在该 block 的等待队列中进行排队等候,系统会依据排队等候的应用请求数量的不同,通过 Noah 的容灾机制生成不同数量的新副本来满足需求。

3)当 block 副本数量远远超过应用请求时,系统会删除多余的副本。

2.3.2 动态副本部署策略

动态副本部署策略通过将 block 在不同 DataNode 之间转移来实现系统负载的转移。DataNode 借助心跳信号向 NameNode 报告本节点的资源负载情况,NameNode 依据这些信息构建全网资源负载图,动态地将负载高的 DataNode 上需求大的 block 向其他负载低的 DataNode 上转移,并且修改内存中相应 block 的位置信息。

2.4 section 的管理

引入 section 之后,如何在 NameNode 上对其进行有效管理成了一个不能回避的问题。若采取和 block 相同的管理策略,在 NameNode 内存中为每一个 section 都生成一个对应的管理对象是不现实的,因为 section 的数量比 block 的数量多了一个数量级,势必会对 NameNode 内存造成非常大的压力,

从而形成瓶颈。以下给出相应的解决方案。

2.4.1　NameNode 内部关键数据结构的变化

为了适应对 section 管理的需求,重新设计了 NameNode 上一个重要的数据结构——BlocksMap[14]。新的数据结构如图 5 所示。

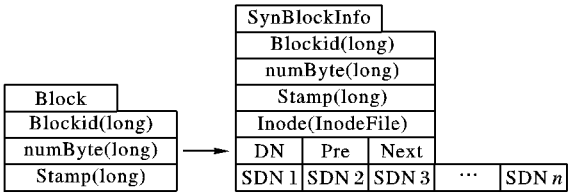

图 5　BlocksMap 数据结构

BlocksMap 实际上是一个由〈Block, BlockInfo〉构成的 Map 表,其中 Block 对象中只记录了 blockid,block 大小以及时间戳信息。而 BlockInfo 是从 Block 对象继承而来,因此除了 Block 对象中保存的信息外,还包括代表该 block 所属的 HDFS 文件的 INodeFile 对象引用和该 block 所属 DataNode 列表信息以及该 block 所对应的所有 section 所属 DataNode 列表信息,在初始化 BlockInfo 对象的时候,DataNode 列表信息是空的,需要通过 DataNode 上传的信息改写相应字段的内容。

NameNode 的本地持久化文件 fsimage[15] 中只存储了名字空间以及文件和 block 之间的映射关系,并没有记录每一个 block 和 section 对应到哪几个 DataNode 的信息。这些 fsimage 中没有记录的信息是在所有 DataNode 启动时,每个 DataNode 对本地磁盘进行扫描,将其上保存的 block 和 section 的信息汇报给 NameNode,NameNode 在接收到每个 DataNode 的信息汇报后,根据这些信息改写内存中 BlockInfo 里的 DN 和 SDN 字段。这样,每一个 block 和 section 所在的 DataNode 位置就成功保存在 NameNode 的内存当中了。

2.4.2　section 的读写操作

对 section 进行读写操作,首先都需要对 section 进行定位。对 section 的定位比较简单,因为在 NameNode 内存中已经记录了它的位置信息,直接查找就可得到它所属的 Datanode,然后通过 DataNode 的本地文件系统即可找到 section 的具体存储位置。

section 的读写操作类似于 block 的读写,都是属于“一次性写入,多次读取”模式,不支持对其部分数据进行更新操作。写 section 实质上是对 block 进行编码分片然后分发到集群当中,收到 section 的 DataNode 节点在本地作记录下该 section 的 ID 以及其所属的 block 的 ID,并通知 NameNode 以使其在内存中记录下 section 的位置信息。

3　方案可行性分析

3.1　实验环境

为了验证 Noah 的可行性,搭建了一个由 1 个 NameNode 节点和 10 个 DataNode 节点的组成的小型集群环境。主要参数配置如下,CPU 为 Intel Core i3(2.1 GHz),内存为 2 GB,硬盘空间为 1 TB,网络带宽为 1 000 Mbps,操作系统为 UBUNTU 10.04,Hadoop 版本为 Hadoop-0.20.203.0。

3.2　GE 码容灾效率

作为阵列纠删码的一种,GE 码的编译码过程中都只用到了异或运算,而异或运算的时间复杂度远远低于有限域运算,所以 GE 码的数据恢复效率非常高。

图 6 是在相同集群环境下,分别使用基于 GE[31,21,6] 码的 Noah 的容灾机制和 3 副本备份容灾机制恢复时间的比较。可以看出,Noah 的容灾机制比备份容灾机制能更快地恢复数据,这是因为采用 Noah 的容灾机制时,可以从多个节点同时传输 section,最终在目标 DataNode 上进行译码恢复数据,这个过程中,译码时间成本相对于网络传输时间是非常小的,所以编码方案能够更快速地恢复损坏的文件,具备更好的容灾效率。

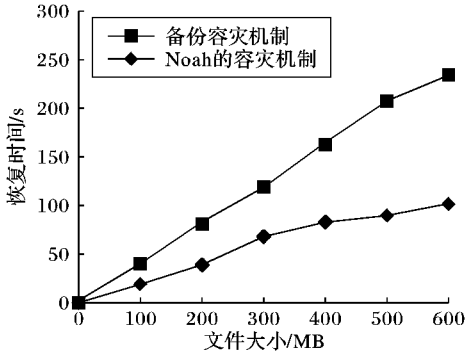

图 6　备份容灾机制和 Noah 的容灾机制数据恢复实验对比

3.3　负载均衡能力

动态副本策略可以显著改善系统的负载均衡能力。为了验证这一点,在实验环境下作了原 HDFS 和 Noah 的负载均衡能力对比实验。假设集群故障率为 10%,图 7 是对比实验的结果。其中,横轴表示集群中的 10 个 DataNode 节点,纵轴表示节点资源利用率的标准差。从中可以看出:原有 HDFS 负载被动较大,而 Noah 采取了动态副本策略,将单个节点的负载平均分配到了整个集群当中,各个节点资源利用率变化很小,非常稳定,处于相对均衡状态。

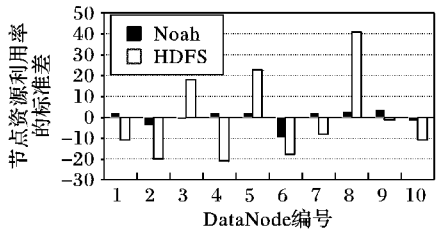

图 7　节点资源负载均衡对比

3.4　存储成本

动态副本策略直接降低了系统整体保有的副本数量,存储成本也随之下降,下降幅度由具体的集群运行参数决定,表 1 是空间占用情况实验的对比数据,存储的实验文件大小是 1 TB,HDFS 采用的是默认的 3 副本。从数据中可以看出,Noah 的总体空间节省比率为 30% ~35%。

表 1　空间占用情况对比

每个 block 的分片数量	空间占用/TB		空间节省比率
	Noah	原 HDFS	
5	1.95	3.00	0.35
11	1.98	3.00	0.34
17	2.02	3.00	0.33
31	2.11	3.00	0.30

3.5　安全性

以 3 副本的 HDFS 为例进行对比,数据安全取决于拥有这 3 个副本所在的 DataNode,若 3 个 DataNode 同时无法提供服务,那么该数据将失效;采用编码方案,该数据被分解更多数量的section(一般在100)分发到集群中,恢复数据时,只需

(下转第 2158 页)

4 结语

CoreScan 算法将网络划分为以次中心节点为核心的 k_{max} 个社区，并将其他节点唯一分配到该社区，从而减小了初始合并社区的数量，使得算法的效率接近线性。实验表明在实际社会网络中，该算法取得了良好的效果，但不适合随机网络。该实验同时说明在实际的社会生活中，一些节点并不与社区外部连接，它们构成的次中心节点存在并影响了社区的结构。如何更好地利用这些次中心节点，以提高算法的正确性是我们下一步的工作。

参考文献：

[1] DANON L, DíAZ-GUILERA A, DUCH J, *et al*. Comparing community structure identification [J]. Journal of Statistical Mechanics: Theory and Experiment, 2005, 2005(9): 09008 – 09018.

[2] FORTUNATO S. Community detection in graphs [J]. Physics Reports, 2010, 486(3): 75 – 174.

[3] FORTUNATO S, BARTHELEMY M. Resolution limit in community detection [J]. Proceedings of the National Academy of Sciences, 2007, 104(1): 36 – 41.

[4] GIRVAN M, NEWMAN M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences, 2002, 99(2): 7821 – 7826.

[5] NEWMAN M E J. Fast algorithm for detection community structure in networks [J]. Physical Review E, 2004, 69(6): 066133.

[6] DUCH J, ARENAS A. Community detection in complex networks using extremal optimization [J]. Physical Review E, 2005, 72(2): 027104.

[7] CLAUSET A. Finding local community structure in networks[J]. Physical Review E, 2005, 72(2): 026132.

[8] WU F, HUBERMAN B A. Finding communities in linear time: a physics approach [J]. The European Physical Journal B: Condensed Matter Physics, 2004, 38(2): 331 – 338.

[9] FORTUNATO S, LATORA V, MARCHIORI M. Method to find community structures based on information centrality [J]. Physical Review E, 2004, 70(5): 056104.

[10] 赵凤霞，谢福鼎. 基于 K-means 聚类算法的复杂网络社团发现新方法[J]. 计算机应用研究, 2009, 26(6): 2041 – 2049.

[11] RADICCHI F, CASTELLANO C, CECCONI F, *et al*. Defining and identifying communities in networks [J]. Proceedings of the National Academy of Sciences, 2004, 101(9): 2658 – 2663.

[12] NEWMAN M E J. Modularity and community structure in networks [J]. Proceedings of the National Academy of Sciences, 2006, 103(23): 8577 – 8582.

[13] NEWMAN M E J. Finding community structure in networks using the eigenvectors of matrices [J]. Physical Review E, 2006, 74(3): 036104.

[14] BERRY J W, HENDRICKSON B, LAVIOLETTE R A, *et al*. Tolerating the community detection resolution limit with edge weighting [J]. Physical Review E, 2011, 83(5): 056119.

[15] GOOD B H, de MONTJOYE Y-A, CLAUSET A. The performance of modularity maximization in practical contexts [J]. Physical Review E, 2011, 81(4): 046106.

[16] LI ZHENPING, ZHANG SHIHUA, WANG RUI-SHENG, *et al*. Quantitative function for community detection [J]. Physical Review E, 2008, 77(3): 036109.

[17] BLONDEL V D, GUILLAUME J-L, LAMBIOTTE R, *et al*. Fast unfolding of communities in large networks [J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10): 10008 – 10014.

（上接第 2153 页）

要其中的任意 70% 的节点，即可容忍集群中不超过 30% 的节点同时失效，提高了数据的安全性。

4 结语

基于 GE 码和动态副本策略的方案主要是针对 HDFS 存储效率不高以及负载均衡能力不足而提出来的。数据实验测试结果表明，Noah 在保证了集群数据安全性的同时，提高了数据恢复的速度，降低了整体的存储成本，优化了 HDFS 的负载均衡能力。

目前该方案遇到的困难主要集中在 section 的管理上，随着集群规模的扩大，section 数量也迅速增加，文件映射表的建立和维护成本也随之增加，从而导致集群的扩展性受到了制约，在今后的工作当中，将进一步研究如何降低所引入的系统复杂性以及如何设计更加优化的 section 分发策略，逐步完善改进的系统方案。

参考文献：

[1] GHEMAWAT S, GOBIOFF H, LEUNG S-T. The Google file system [C]// SOSP 2003: Proceedings of 19th ACM Symposium on Operating Systems Principles. New York: ACM, 2003: 58 – 66.

[2] Hadoop[EB/OL]. [2011 – 12 – 01]. http://hadoop.apache.org/common.

[3] Map-Reduce[EB/OL]. [2011 – 12 – 01]. http://wiki.apache.org/mapreduce/.

[4] BORTHAKUR D. Hadoop and its usage at Facebook [R/OL]. [2011 – 11 – 13]. http://borthakur.com/ftp/newflixalone.pdf.

[5] MACKEY G, SEHRISH S, WANG JUN. Improving metadata management for small files in HDFS [C]// CLUSTER '09: IEEE International Conference on Cluster Computing and Workshops. Piscataway: IEEE, 2009: 1 – 4.

[6] THUSOO A, SARMA J S, JAIN N, *et al*. Hive: A petabyte scale data warehouse using Hadoop[C]// ICDE 2010: Proceedings of 26th IEEE International Conference on Data Engineering. Piscataway: IEEE, 2010: 996 – 1005.

[7] JIANG D, OOI B C, SHI L, *et al*. The performance of MapReduce: an in-depth study[J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 472 – 483.

[8] DONG BO, QIU JIE, ZHENG QINGHUA, *et al*. A novel approach to improving the efficiency of storing and accessing small files on Hadoop: a case study by PowerPoint files[C]// SCC '10: Proceedings of the 2010 IEEE International Conference on Services Computing. Washington, DC: IEEE Computer Society, 2010: 65 – 72.

[9] LIU XUHUI, HAN JIZHONG, ZHONG YUNQIN, *et al*. Implementing WebGIS on Hadoop: a case study of improving small file I/O performance on HDFS[C]// CLUSTER '09: IEEE International Conference on Cluster Computing and Workshops. Piscataway: IEEE, 2009: 45 – 48.

[10] HDFS [EB/OL]. [2011 – 12 – 05]. http://hadoop.apache.org/common/.

[11] NameNode[EB/OL]. [2011 – 12 – 11]. http://wiki.apache.org/namenode/.

[12] DataNode[EB/OL]. [2011 – 12 – 21]. http://wiki.apache.org/datanode/.

[13] 陈铮. 一类新的阵列纠删码理论及应用[D]. 北京：中国科学院研究生院, 2009.

[14] BlocksMap[EB/OL]. [2011 – 02 – 08]. http://www.tbdata.org/archives/1120/.

[15] WHITE T. Hadoop: The definitive guide[M]. Sebastopol, California: O'Reilly Media, 2011: 295.