第45卷 第6期 西 安 交 通 大 学 学 报 Vol. 45 No. 6
2011年6月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Jun. 2011

一种提高云存储中小文件存储效率的方案

余思[1,2]，桂小林[1,2]，黄汝维[1]，庄威[1]
(1. 西安交通大学电子与信息工程学院，710049，西安；2. 西安交通大学陕西省计算机网络重点实验室，710049，西安)

摘要：针对基于HDFS(Hadoop distribated file system)的云存储系统中小文件存储效率不高的问题，采用序列文件技术设计了一个云存储系统中小文件的处理方案. 该方案利用多维属性决策理论，综合读文件时间、合并文件时间及节省内存空间大小等指标，得出合并小文件的最优方式，能够在消耗的时间和节省的内存空间之间取得平衡；设计基于层次分析法的系统负载预测算法对系统负载进行预测，从而实现负载均衡的目的；利用序列文件技术对小文件进行合并. 实验结果表明，在不影响存储系统运行状况的基础上，该方案提高了小文件的存储效率.

关键词：云存储；小文件；存储效率；负载预测
中图分类号：TK124 **文献标志码**：A **文章编号**：0253-987X(2011)06-0059-05

Improving the Storage Efficiency of Small Files in Cloud Storage

YU Si[1,2], GUI Xiaolin[1,2], HUANG Ruwei[1], ZHUANG Wei[1]
(1. School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China;
2. Shaanxi Province Key Laboratory of Computer Network, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: An approach based on SequenceFile is proposed to improve storage efficiency of small files in the cloud storage systems that are on the basis of Hadoop distributed file system(HDFS). The approach uses the multi-attribute decision theory and the indices such as reading time, combining time, and saved memory size to obtain an optimal file merging scheme, so that the balance between computing time and memory space is achieved. A system load forecast algorithm is designed based on the analytic hierarchy process to predict the load of the system. SequenceFile is used to combine small files. Experimental results show that, without degrading the performance of storage system, the storage efficiency of small files is improved.

Keywords: cloud storage; small file; storage efficiency; load forecasting

HDFS(Hadoop distributed file system)[1]是一种具有高度容错性质的分布式文件系统模型，可以部署在支持JAVA运行环境的普通机器或虚拟机上，能够提供高吞吐量的数据访问，非常适合部署云存储平台.

HDFS采用主从式架构设计模式(master/slave architecture)，一个名称节点(NameNode)和若干数据节点(DataNode)构成HDFS集群. HDFS的这种单名称节点的设计极大地简化了文件系统的结构，然而也因此引发了HDFS的小文件存储效率低的问题. 因为HDFS中的每个目录和文件的元数据信息都存放在名称节点的内存中，如果系统中存在大量的小文件(指那些比HDFS数据块(默认为64 MB)小得多的文件)，则无疑会降低整个存储系统的存储效率和存储能力.

在各种存储系统中，存在大量这样的小文件. 美国西北太平洋国家实验室2007年的一份研究报告表明，他们系统中有1 200万个文件，其中94%的文件小于64 MB，58%的小于64 kB[2]. 在一些具体的科研计算环境中，也存在大量的小文件，例如，在某

收稿日期：2010-12-10. 作者简介：余思(1987—)，男，博士生；桂小林(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60873071)；IBM共享大学研究(SUR)资助项目(SUR201001X).
网络出版时间：2011-03-18 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20110318.1523.000.html

些生物学计算中可能会产生 3 000 万个文件，而其平均大小只有 190 kB[2].

解决基于 HDFS 的存储系统中小文件存储效率问题的主流思想是将小文件合并或组合为大文件，目前主要的方法分为 2 种，一种是利用 Hadoop 归档(Hadoop archive，HAR)等技术实现小文件合并的方法，另一种则是针对具体的应用而提出的文件组合方法.

Mackey 等[2]利用 HAR 技术实现小文件的合并，从而提高了 HDFS 中元数据的存储效率. Liu 等[3]结合 WebGIS 应用，以 Hadoop 为存储平台开发了 HDWebGIS 原型系统；结合 WebGIS 访问模式的特点，将小文件组合为大文件并为其建立全局索引，从而提高了小文件存储效率. Dong 等[4]针对 BlueSky 系统中 PPT 课件的存储问题，提出了将小文件合并到大文件中并结合预取机制来提高系统存储和访问小文件的效率的方法. 刘立坤等[5]对分布式存储系统中小文件的并发访问进行了优化.

以上的研究工作都是基于文件的合并或组合来解决小文件存储效率不高的问题，然而还存在以下 2 个问题：第一，作为一个完整的系统，在提高小文件存储效率的同时，也应该考虑到系统的负载状况，因为不管是文件合并还是文件组合，对 HDFS 而言都是一个额外的操作；第二，未对小文件合并规模进行研究，即尚未确定多少个小文件合并为一个大文件可以使系统性能达到最优.

基于以上两点，本文提出了一个面向 HDFS 的云存储系统中小文件存储效率的优化方案：采用序列文件技术将小文件合并为大文件，结合多属性决策理论和实验得出合并文件的最优方式，通过基于层次分析法(analytic hierarchy process，AHP)的系统负载预测算法实现系统的负载均衡.

1 小文件存储效率优化方案设计

在构建的云存储系统中，采用多叉树的结构来构建文件索引. 用户将文件上传到云存储系统后，系统会自动根据用户文件的组织形式建立对应的多叉树索引，详细方案见文献[6].

1.1 序列文件合并技术

序列文件(SequenceFile)是 HDFS 提供的一种二进制文件技术，这种二进制文件直接将＜key，value＞对序列化到文件，文件序列化时可实现基于记录或数据块的压缩. 在云存储系统中，对二进制文件采用 SequenceFile 技术将小文件合并为大文件，以小文件的索引号为 key、内容为 value 的形式进行合并，合并的同时实现基于数据块的压缩，这样，在节省名称节点内存空间的同时也节省了数据节点的磁盘空间.

1.2 小文件存储效率优化方案

在基于 HDFS 的云存储系统中，小文件存储效率优化方案如图 1 所示. 为提高对小文件的处理效率，系统为每个用户建立了 3 种队列：第 1 种为序列文件队列(SequenceFile queue，SFQ)，第 2 种为序列文件操作队列(SequenceFile operation queue，SFOQ)，第 3 种为备用队列(Backup queue，BQ). 其中，SFQ 用于小文件的合并，SFOQ 用于对合并后小文件的操作，BQ 用于操作的小文件数超过 SFQ 或 SFOQ 长度的情况. 3 种队列的长度一致，可通过实验得出队列长度的最优值. 下面介绍具体的处理流程.

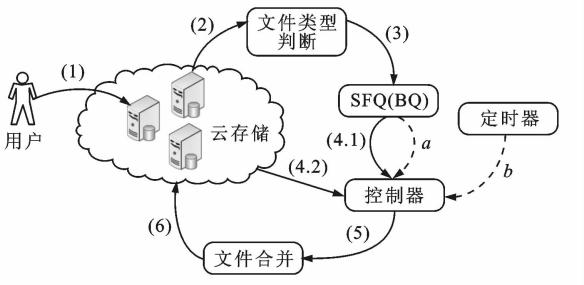

图 1 小文件存储效率优化方案

如图 1 所示，用户将本地的文件上传至云存储服务器(过程 1)，然后服务器开始对该文件的类型进行判断(过程 2)，如果是小文件，将该文件的索引号放入 SFQ 中(过程 3). 当 SFQ 满时，将发送“队列满”信号(QF)给控制器，如图中虚线 a 所示，而当定时器到定时点时，将发送“时间到”信号(TU)给控制器，如虚线 b 所示. 接收到 QF 或者 TU 信号后，控制器开始读取 SFQ 的相关信息(过程 4.1)，对系统负载进行计算(过程 4.2)(具体算法在第 2 节中介绍)，并据此决定是否进行小文件的合并(过程 5). 文件合并后完成小文件与大文件之间的映射(过程 6). 控制器的具体处理逻辑如图 2 所示.

当控制器接收到信号时，首先判断信号类型，如果是 QF，则调用基于 AHP 的系统负载预测算法计算系统负载. 如果得到的系统负载低于系统设定的阈值，则开始合并文件(包括 SFQ 和 BQ)，并取消系统中的 TU 信号；如果系统负载大于系统设定的阈值，则进一步判断 BQ 的数量，若 BQ 数量小于某个值(例如 3)，则新建 BQ，将 SFQ 转移到 BQ 中并推

迟合并操作(系统中设定推迟的时间为 30 min)，设定 TU 信号，若 BQ 数量大于 3，则将 BQ 中的小文件进行合并，取消系统中的 TU 信号.

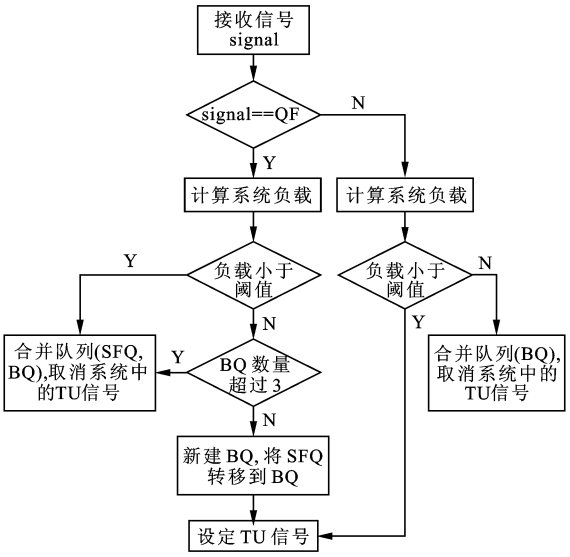

图 2　控制器控制逻辑

如果接收到的是 TU 信号，计算系统负载并判断是否大于系统设定的阈值. 若负载大于阈值，则推迟合并操作并设定 TU 信号；若负载小于阈值，则合并 BQ 中的小文件，取消系统中的其他 TU 信号.

2　基于 AHP 的系统负载预测算法

系统负载预测通常定义为基于 CPU 利用率、内存利用率、带宽利用率和系统平均吞吐量等系统属性对系统运行状态进行的多属性决策.

层次分析法(AHP)[7]是美国运筹学家托马斯萨迪提出的一种层次权重决策分析方法，是对定性问题进行定量分析的一种简便、灵活而又实用的多准则决策方法.

负载计算得到的是一个即时值或历史值，即只能够得到当前或以前时刻的系统负载，然而对小文件的操作是在系统负载计算之后，因此需要根据系统负载的历史信息来推测下一时刻的系统负载. 基于此，本文设计了基于 AHP 的系统负载预测算法. 该算法通过获取系统属性的历史信息，经过 2 次 AHP 分析，最终可得到系统负载的预测值.

算法依据系统属性的重要性，将每个时刻的系统负载属性值经过 AHP 分析融合为单一的决策属性值，然后依据决策属性值的时间重要性，经过第二次 AHP 分析最终得到下一时刻的系统负载值. 具体步骤如下.

步骤 1　决策属性值计算.

(1)构造比较矩阵 $\boldsymbol{A}$. $p_i(i=1,2,\cdots,n)$表示系统属性值. 比较矩阵 $\boldsymbol{A}$ 中元素的定义如表 1 所示，表中属性的大小关系为属性重要性的比较.

表 1　比较矩阵元素的定义

属性关系	$p_i<p_j$	$p_i=p_j$	$p_i>p_j$
a_{ij}	0	1	2

(2)构造判断矩阵 $\boldsymbol{A}^*$. 对比较矩阵 $\boldsymbol{A}$ 进行如下变换，得到 $\boldsymbol{A}^*$

$$\left.\begin{aligned} a_{ij}^* &= \frac{r_i-r_j}{r_{\max}-r_{\min}}(b_{\mathrm{m}}-1)+1, \quad r_i \geqslant r_j \\ a_{ij}^* &= 1, \quad r_{\max}=r_{\min} \\ a_{ij}^* &= \left[\frac{|r_i-r_j|}{r_{\max}-r_{\min}}(b_{\mathrm{m}}-1)+1\right]^{-1}, \quad r_i<r_j \end{aligned}\right\} \tag{1}$$

式中：$r_i=\sum_{j=1}^{m}a_{ij}$；$r_{\max}=\max(r_i)$；$r_{\min}=\min(r_i)$；$b_{\mathrm{m}}=r_{\max}/r_{\min}$.

(3)计算最优传递矩阵 $\boldsymbol{C}$

$$c_{ij}=\frac{1}{n}\sum_{k=1}^{n}\lg(a_{ik}^*/a_{jk}^*), \quad \forall i,j,k=1,2,\cdots,n$$

最后得权值 $w_j=1/\sum_{i=1}^{n}10^{c_{ij}}$，　$j=1,2,\cdots,n$.

(4)计算决策属性值

$$d_i=\sum_{j=1}^{n}(p_{ij}w_{ij}), \quad i=1,2,\cdots,n$$

步骤 2　系统负载值计算. 根据时间顺序的重要性构造比较矩阵 $\boldsymbol{A}$，按步骤 1 中的(2)、(3)步计算 t_1，$t_2,\cdots,t_n$ 时刻的决策属性值 $d_1,d_2,\cdots,d_n$ 的权重 w_i $(i=1,2,\cdots,n)$，最后得到 t_{n+1}时刻的系统负载

$$d_{n+1}=\sum_{i=1}^{n}(w_i d_i)$$

通过本文提出的这一算法，可以实现对系统负载的预测，从而将对小文件的操作控制在某个能够均衡系统负载的时刻进行.

3　实　验

为提高小文件的操作效率，系统为每个用户建立了 SFQ 和 SFOQ. 在这一节中，将通过实验研究 SFQ 长度对云存储系统的影响，选取读取文件时间、合并文件时间和节省的内存空间作为参考指标，以得到小文件合并的最优方式.

在基于 HDFS 的云存储系统中，对文件的操作主要有上传、下载、读取等. 合并操作对上传没有影响，下载的核心操作也是读取，因此选取读取文件时

间作为参考指标. 提高名称节点内存利用率是本文的主要工作,因此将通过合并文件节省的内存空间作为参考指标之一. 合并文件的效率是影响存储系统性能的一个重要因素,故也将合并文件时间作为参考指标.

3.1 实验方案与实验结果

我们将通过 3 个实验分别获取在 SequenceFile 中读取小文件的平均时间、合并文件的平均时间以及合并所能节省的内存空间等指标值,并通过 AHP 分析数据,得出 SFQ 长度与系统性能的关系.

6 台 Dell 服务器构成云存储环境,服务器的配置均为 CPU 8 Intel Xeon 2.13 GHz,内存 8 GB,硬盘 500 GB,操作系统均为 Ubuntu Server 9.04,Hadoop 版本为 0.20.0.

实验 1 统计合并文件的平均时间(t_1). 按照 SFQ 长度分别为 100、200、300、400、500、600、700、800、900、1 000 合并小文件 50 次,并且在不同的时段重复这样的实验 10 次. 统计这 10 种情况下合并文件所需时间的平均值,最终得到合并文件的时间,结果如图 3 所示.

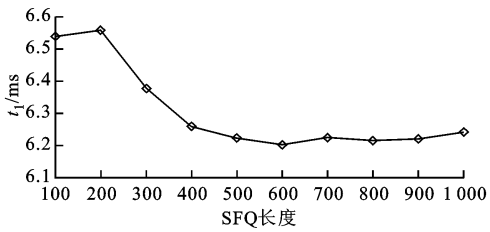

图 3 合并文件的平均时间

实验 2 统计读取小文件的平均时间(t_2). 小文件合并为 SequenceFile 之后,读取小文件的时间主要分为在 SequenceFile 中查找小文件的时间和获取小文件内容的时间 2 部分,因此,小文件合并之后读取文件的时间与该文件在 SequenceFile 中所处的位置有关. HDFS 提供的 API 中采用顺序查找算法进行文件查找,因此读取的文件在 SequenceFile 中位置越靠后所需的时间越长. 在实验 1 中得到的 10 个大文件中以 10 为步长读取小文件,获取其平均时间作为读取该大文件中小文件的平均时间,实验结果如图 4 所示.

实验 3 统计合并 10 000 个小文件节省的内存空间. 将 10 000 个小文件上传到云存储系统,统计其占用名称节点的内存空间,然后分别按照 SFQ 长度为 100、200、300、400、500、600、700、800、900、1 000进行合并,获取合并后占用名称节点的内存空间,两者之差即为合并操作所节省的内存空间,实验结果如图 5 所示.

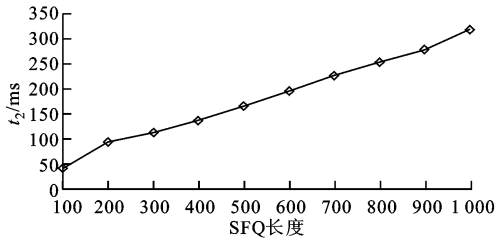

图 4 读取小文件的平均时间

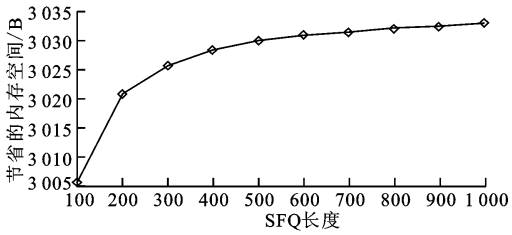

图 5 节省的内存空间

3.2 实验结果分析

3.2.1 *数据标准化* 将实验指标转化为逆指标(越小越好的指标),分别利用 Min-Max 方法和 Z-Score 方法对转化为逆指标的实验数据进行标准化[8-9],结果如图 6、图 7 所示.

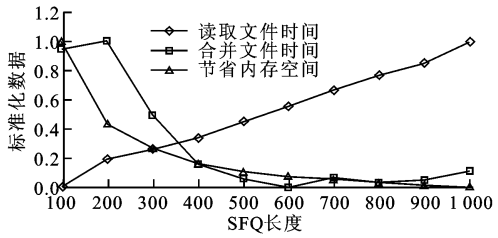

图 6 Min-Max 法的标准化数据

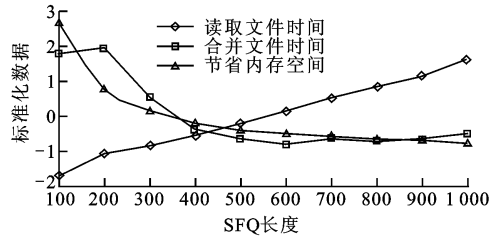

图 7 Z-Score 法的标准化数据

3.2.2 *系统性能决策值计算* 利用 AHP 进行权重计算. 由于读取文件是最频繁的操作,因此认定读取文件时间为 3 个指标中最重要的,节省的内存空间其次. 据此,计算 3 个指标的权重如表 2 所示.

表 2 权重

指标	读取文件时间	节省内存空间	合并文件时间
权重	0.637 0	0.258 3	0.104 7

将标准化的数据与相应的权重相乘之后相加，得到系统性能决策值，如图 8 所示.

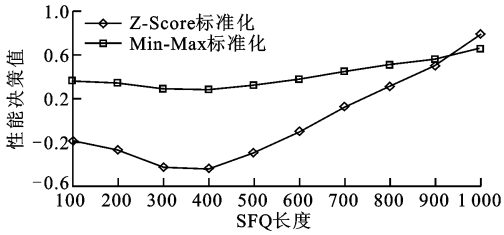

图 8　分析结果

3.2.3　结果分析　从图 8 可以看到，两种数据标准化方法都反映出一个规律，即在本文的实验环境中，性能决策值随着 SFQ 长度的增大呈现一种类似开口向上的抛物线状变化，并且在 SFQ 长度为 400 时取得最小值. 由于我们采用了逆指标进行计算，因此当性能决策值最小时，表示系统性能达到了最优. 由此可以得出结论：在本文的云存储环境中，SFQ 长度取 400 是小文件合并的最优方式；根据基于 AHP 的系统负载预测算法对系统运行状况监控的结果，可以得到小文件合并的最佳时间.

通过实验可知，小文件合并的规模越大，名称节点消耗的内存空间将越少，与此同时，对小文件的操作(读取、删除等)以及合并文件所花费的时间代价也将越大. 在其他基于 HDFS 的存储系统中采用本文的方案进行分析和部署，都可在时间消耗和内存利用率之间实现一种最优平衡，实现在小文件存储效率提高的同时不影响系统性能的目标.

4　结　语

本文针对基于 HDFS 的云存储系统中小文件存储效率不高的问题，提出了一套完整的解决方案. 在该方案中，采用 SequenceFile 技术将小文件以队列的形式合并为大文件，从而实现了节省名称节点所占内存空间的目的，同时也实现了对合并之后的小文件的透明操作. 在确定影响队列长度的指标之后，通过实验获取指标值，采用数据标准化方法和三标度层次分析法确定队列长度的最优值，使得小文件的合并能在合并时间、文件操作时间和节省内存空间之间达到一种平衡. 基于负载均衡的目的，本文设计了基于 AHP 的负载预测算法对系统负载进行预测.

在以后的工作中，我们将从以下两个方面来进行改进：①将小文件的合并以及小文件的读取改进为 Map-Reduce 任务，从而提高操作的效率；②对 SequenceFile 中的小文件查找算法进行改进，提高小文件查找效率.

参考文献：

[1] BORTHAKUR D. The hadoop distributed file system: architecture and design [EB/OL]. [2010-08-25]. http://hadoop.apache.org/core/docs/current/hdfs_design.pdf.

[2] MACKEY G, SEHRI S, WANG Jun. Improving metadata management for small files in HDFS [C/OL]// Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops. [2010-08-10]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5289133.

[3] 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/OL]// Proceedings of 2009 IEEE International Conference on Cluster Computing and Workshops. [2010-08-10]. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5289196.

[4] 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]// Proceedings of the 7th International Conference on Services Computing. Piscataway, NJ, USA: IEEE, 2010: 65-72.

[5] 刘立坤，武永卫，徐鹏志，等. CorsairFS：一种面向校园网的分布式文件系统[J]. 西安交通大学学报，2009，43(8)：43-47.
LIU Likun, WU Yongwei, XU Pengzhi, et al. CorsairFS: a distributed file system for campus network [J]. Journal of Xi′an Jiaotong University, 2009, 43(8): 43-47.

[6] HUANG Ruwei, YU Si, ZHUANG Wei, et al. Design of privacy-preserving cloud storage framework [C]// Proceedings of the 9th International Conference on Grid and Cloud Computing. Piscataway, NJ, USA: IEEE, 2010: 128-132.

[7] SATTY T L. Axiomatic foundation of the analytic hierarchy process [J]. Management Science, 1986, 32(7): 841-855.

[8] HAN Jiawei, KAMBER M. Data mining: concepts and techniques [M]. San Francisco, CA, USA: Morgan Kaufmann, 2006.

[9] 马立平. 统计数据标准化：现代统计分析方法的学与用：(三)[J]. 北京统计，2000(3)：34-35.

(编辑　葛赵青　武红江)