基于柔性放置的负载均衡策略研究

刘 群[1] 冯 丹[2] 李 坚[2]

(华中科技大学网络与计算中心 武汉 430074)[1] (华中科技大学计算机学院 武汉 430074)[2]

摘 要 在基于可扩展对象的海量存储系统(Based on Scalable Object Mass Storage System,BSO-MSS)中,负载均衡一直是研究的重点,如何选择存储对象(Storage Object,SO)及数目是关键。因此提出柔性负载均衡策略,它不仅考虑网络对BSO-MSS的影响,而且更关注SO本身,针对SO中不同的存储能力,自适应选择SO数目,采用不同大小的分条进行存储。当SO数目未达到最佳值时,增加SO数目,会减少系统响应时间,提高整个系统的吞吐量。

关键词 基于可扩展对象的海量存储系统,负载均衡,柔性放置

中图法分类号 TP334.5 **文献标识码** A

Research on Workload Balancing Strategy Based on Flexible Layout

LIU Qun[1] FENG Dan[2] LI Jian[2]

(Network and Computer Center, Huazhong University of Science and Technology, Wuhan 430074, China)[1]

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)[2]

Abstract The workload balancing is always the key of study in the Based on Scalable Object Mass Storage System (BSO-MSS). How to choice Storage Objects(SOs) and the number of them are. It presented a strategy of workload balancing in this paper. Not only do we thought over network effect on BSO-MSS, but also we noticed SOs themselves. Aiming at different storing ability of SOs, the system adapts to choice the number of SOs by itself and adopts different size of stripping for data storing. When the number of SOs don't add up to the best value, we are adding the more number, the system has the least response time. So it enhances the whole system performance.

Keywords Based on scalable object mass storage system, Workload balancing, Flexible layout

1 引言

在20世纪80年代,随着网络技术的兴起和普及,各种海量数据(如高性能计算、流媒体、数字博物馆、地理信息系统、气象服务等)的应用,促使存储需求呈指数增长。因此,计算机系统的设计重点从传统以计算处理为中心转移到以数据应用为中心,并将网络服务和存储服务相结合,构成了网络存储系统。正是这些海量数据的应用需求推动了海量存储系统不断的发展和其性能不断的改善,推出新的存储体系结构,从传统的直接存储系统(Direct Access Storage,DAS)、存储区域网(Storage Area Network,SAN)和附网存储(Network Attached Storage,NAS),发展到新一代的基于对象存储(Object Based Storage,OBS)[1,2]技术。

在传统的存储结构中,元数据服务器包括文件管理部分和存储管理部分。文件管理部分提供全局的逻辑视图,包括树型目录结构和文件名列表等,向用户提供用户调用接口;存储管理部分是存储和管理文件的物理视图,把文件的逻辑块映射到底层物理介质。OBS将存储数据的逻辑视图和物理视图分开,元数据服务器仅管理约占负载10%的文件管理部分,而剩下约90%的存储管理部分下移到基于对象的存储设备(Object-Based Storage Device,OSD)[3,4]中,OSD负责其中所有对象的空间分配(对象映射的逻辑块),维护与对象有关的元数据(类似于UNIX/LINUX的inode结构),对象的存储建立在一个"平坦"一维空间中,接口也从基于块的接口变为基于对象的接口,这样与存储管理相关的元数据分布到整个系统各个OSD中。因此OBS克服了NAS和SAN中的不足,既有"块"接口的快速,又有"文件"接口的便于共享,并在扩展性、安全性、高性能和跨平台数据共享等方面更胜一筹。

2 基于可扩展对象的海量存储系统

在基于对象技术的基础上,扩充了对象含义,改进了元数据管理与存储,采用了一种基于可扩展对象的海量存储系统(Based on Scalable Object Mass Storage System,BSO-MSS),它由元数据服务器(MetaData Server,MDS)、存储对象(Storage Object,SO)以及客户对象(Client Object,CO)组成[5,6](如图1所示)。

MDS如同一盏导航灯,提供全局的命名空间,负责整个系统的目录结构、权限控制和文件命名。它建立目录、文件的访问管理,构建一个树状目录结构,包括目录对象和文件对象的创建、删除和访问控制、限额控制等;同时为客户对象提供

到稿日期:2010-06-23 返修日期:2010-10-10 本文受国家自然科学基金项目(60873028),湖北省教学研究项目(2009053),华中科技大学实验技术研究项目资助。

刘 群(1969—),女,博士,高级工程师,主要研究方向为计算机网络与高性能网络存储,E-mail:liliuqun@mail.hust.edu.cn;冯 丹(1970—),女,博士,教授,主要研究方向为计算机系统结构、网络存储等;李 坚(1965—),硕士,助理研究员,主要研究方向为计算机网络及应用。

统一的文件逻辑视图，构造、管理并描述每个文件的分布视图，允许客户对象直接与存储对象联系访问该文件数据。

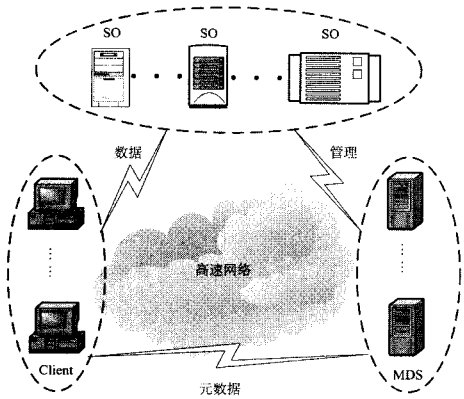


图 1　BSO-MSS 体系结构

MDS 为所有客户端提供了一个全局的、有效的、单一的目录树，目录树包含从不同 MDS 移植到这个命名空间来的目录和文件，并提供 POSIX 标准文件访问接口。同时针对元数据访问的特点，设置缓冲区，将最近访问过的元数据缓存到该缓冲区中，减少磁盘 I/O。由此可见，MDS 的元数据服务在 BSO-MSS 中起着举足轻重的作用。

SO 则是一个智能设备，它包括处理器（CPU）、内存（Memory）、网络接口（NIC）和存储设备（Storage Device），其中存储设备以多种形式存在，可以是 Disk，RAID，NAS 和磁带设备等。依照 SO 的接口与状态，实现数据组织的存储。SO 与传统的存储设备区别不是介质，而是接口，并且在 OBS 基础上进一步丰富和扩展了对象接口的内涵，使之成为一个通用存储访问的接口模式。Client 的应用程序采用 POSIX 方式进行数据访问。

在 BSO-MSS 中，对象存储设备 SO 成百上千，甚至更多，但它并不是一次性建成的，而是依据需求不断地增加新的 SO，淘汰旧的 SO，逐步发展壮大起来。因此，如何合理有效地实现负载均衡，对系统的整体性能以及资源的充分利用都至关重要。

3　常用负载放置方法

比较常见的对象布局有哈希（Hashing）算法和分条（Stripping）技术两种。哈希算法使用哈希函数将数据对象近似随机地分布在不同 SO 上，实现负载平衡和较高的数据分布率。但这种一对一的映射方式不能够充分地利用多 SO 的并行性，同时当增加新的对象存储设备时，需要迁移数据对象，或者当某一 SO 负载过重时，需要将其中一部分数据对象迁移到其他对象存储设备中，这也会导致系统负载的不均衡。分条技术[7]源起于多个磁盘的并行操作，将应用文件分成一个个固定的小块，分条到不同的 SO 中，这个特性提高了 SO 的并行度，能够获得高的集合带宽。

结合了哈希算法和分条技术的静态负载放置策略[8]，将小文件直接映射成一个数据对象，通过哈希方法映射到一个 SO 中；而对大文件则分割成多个数据对象，分别放置在不同 SO 中。根据 LLNL（Lawrence Livermore National Laboratory）[9]的负载统计分析结果显示，约有 85% 的文件为 512kB 的，约 15% 的文件大小小于 512kB，因而确定大于 512kB 的视为大文件。但这种放置策略主要考虑网络对其的影响，并不关注 SO 本身，没有考虑 SO 之间的处理器能力、存储容量、带宽存在着差异，当 SO 存在差异时，对象布局策略直接影响系统性能，影响着各个 SO 之间的负载平衡和并行性。本文针对 SO 之间存在的差异，考虑磁盘对系统性能的影响，提出柔性放置的负载均衡策略。

4　负载柔性放置策略

在实际对象放置过程中，系统并不知晓当前对象的平均到达率和平均数据请求长度，但对象的历史访问信息记录在对象的属性中，通过访问 SO 的属性，对历史访问记录进行统计，可以得到负载的特征和 SO 的特性，以响应时间为代价，针对 SO 中不同的存储能力，采用不同大小的分条进行存储，自适应调节系统提高系统性能。

从上可知，当客户端从 MDS 获得相应信息和策略后，就直接与 SO 进行通信，MDS 处理客户端请求只是一个微秒级的常数，在此忽略。因此我们仅考虑客户端和与之通信的各个 SO 之间响应时间。假设一个 512kB 的数据对象，在空间分配等中软件处理开销一般是微秒级，在千兆网络中网络开销通常为微秒级，而磁头定位时间和寻道时间都是毫秒级；若采用串行传输，网络传输时间也只需 4ms，因此，系统不仅要考虑网络对系统的影响，还需考虑磁盘设备的机械动作对系统性能的影响。

数据对象的服务时间包括 3 个方面：(1)软件开销；(2)网络花费的时间；(3)磁盘花费的时间。其中网络花费的时间包括网络延迟开销和网络传输时间[10]；磁盘花费的时间包括磁盘寻道时间、旋转延迟和磁盘数据传输。由于网络延迟开销为微秒级，不考虑网络故障，在此忽略，这样数据对象服务时间为：

$$T_S = T_o + T_{net} + T_{disk} \approx T_o + T_{net\text{-}tf} + T_{disk} \tag{1}$$

$$T_{disk} = T_{seek} + T_{rota} + T_{disk\text{-}tf} \tag{2}$$

式中，T_S 为服务时间，T_o 为软件开销，$T_{net\text{-}tf}$ 为网络传输时间，T_{disk} 为磁盘中花费的时间，具体为 T_{seek} 是磁盘寻道时间，T_{rota} 是磁头旋转定位时间，$T_{disk\text{-}tf}$ 是磁盘中传输时间。

假设 BSO-MSS 中对象存储设备 SO 数量为 n，客户端请求为泊松到达，到达率为 λ，每一个请求分配选中的 m 个 SO 中（$m \leqslant n$），且无副本、无热点情况。在这种分配模式中，许多相互独立的请求都会排列在某一个 SO 队列中，每一个请求统一在所有 SO 中分配，则对于每一个 SO 请求到达率 λ_i，且 $\lambda_i = \lambda/m, m \leqslant n$。由于并发中每一个 SO 都是一个独立的实体可以并行执行，因此并发度为 m 的数据对象请求的执行示意图如图 2 所示。

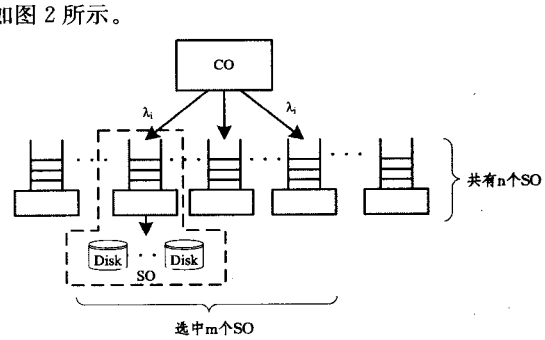


图 2　并发度为 m 的数据对象请求分发情况

根据 M/G/1 排队模型，并发度为 m 的请求在不同的 SO 中服务，SO 的服务时间相当于 m 个独立同分布的变量，因而请求的服务时间取决于执行最慢的 SO。同时对 SO 的请求管理需要一定时间，请求的分解和合并，类似于一种 Fork-

Join 过程，请求的管理开销与请求并发度成正比。因此，需先求出这些请求的服务时间和响应时间，再求出最大响应时间。对于每一个请求，服务时间为：

$$T_S \approx m * T_o + T_{net\text{-}tf}/m + T_{seek} + T_{rota} + T_{disk\text{-}tf}/m \quad (3)$$

则 SO 的等待时间主要为等待磁盘工作，假定选定 SO 的数目正好为请求的并发度，每一个 SO 的请求达到率与该磁盘或者磁盘组的请求达到率相同，因此平均等待时间[11]和响应时间[12]分别为：

$$T_W = \frac{\lambda_i E(T_S^2)}{2[1-\lambda_i E(T_S)]} \quad (4)$$

$$T_R = T_W + T_S = \frac{\lambda_i E(T_S^2)}{2[1-\lambda_i E(T_S)]} + T_S \quad (5)$$

由于所选择的 SO 并行工作，因此总 I/O 响应时间为其中一个 SO 的最长响应时间：

$$T_{R\_max} = \alpha * T_R \quad (6)$$

式中，α 为放大因子。由于系统初始时 SO 配置基本相同，因此各个 SO 满足独立同分布条件，对于并发度为 m 的多个 SO 服务时间为其中最慢的 SO 的服务时间所决定，即求取 m 个独立同分布中最长服务时间。下面对 SO 的并行访问最长服务时间进行讨论。

设$\{X_i, i=1,2,\cdots,m\}$为满足同分布的 m 个非负随机变量，X 分布函数为 F_x，用放大因子 $\alpha(m)$表示随机变量 X_i 的最大值，则其分布为：

$$F_{\alpha(m)} = (F_X(y))^m \quad y \geqslant 0 \quad (7)$$

由于随机变量 $\alpha(m)$为非负值，其数学期望表示为：

$$E[\alpha(m)] = \int_0^\infty (1-(F_x(y))^m)\mathrm{d}y \quad (8)$$

由于 $\alpha(m)$为随机变量 X_i 中最大值，它依赖于分布函数 F_x 的概率，当随机变量 X_i 满足不同分布时，$\alpha(m)$表示为[13]：

$$\alpha(m) = \begin{cases} \dfrac{2m}{m+1}, & \text{均匀分布} \\ 0.5772 + \ln(m), & \text{指数分布} \\ 1 + \dfrac{\sigma_X}{\mu_X}\sqrt{2\log(m)}, & \text{正态分布} \end{cases} \quad (9)$$

式中，μ_x，σ_x 分别为函数 F_x 的均值和方差。图 3 表示随机变量 m 的不同放大因子。从图中可以看出，不同的分布放大因子也迥然不同，在此采用均匀分布，即 SO 响应时间为：

$$T_{R\_max} = \frac{2m}{m+1} * T_R \quad (10)$$

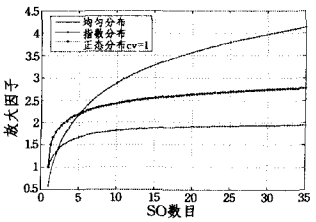


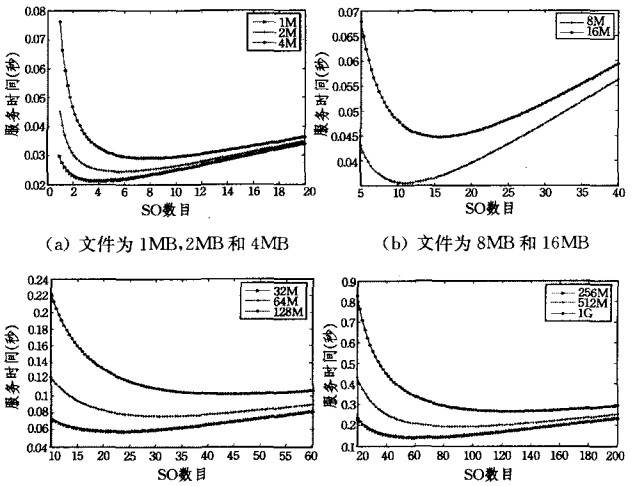
图 3　SO 数目与服务时间放大因子的关系

选择 Maxtor 80G 磁盘，当请求到达磁盘时，旋转延迟在[0，0.0084]之间，寻道时间在[0，0.018]之间，则 T_{seek} = 0.0084/2=0.0042 秒，T_{rota} = 0.018/2=0.009 秒；对于软件开销，由于在 Linux 2.4 中执行 Ext2 文件系统相关功能的代码约 2000 条，自行编写包括网络模块和对象存储服务模块的代码不足 1000 条，均采用 C 语言，对于 1000 条的语句，按 1：10 的比例换算机器代码约在 10000 条，执行时间约为 10ns[14]，则执行时间约 30ns，与毫秒级相比可忽略不计，同时考虑到客户端连续分派子请求到不同的 SO 需要一定的开销，参考文献[15]并作相关测试，软件开销取 0.001 秒。参数如表 1 所列。

表 1　系统采用的参数值

参数名称	参数值
磁盘容量	80GB
最大旋转延迟	0.0083s
寻道时间	0.018s
数据传输率	133MB/s
软件开销	0.001s
网络带宽	1000Mb/s

图 4 显示文件大小为 1MB，2MB，4MB，8MB，16MB，32MB，64MB，128MB，256MB，512MB 以及 1GB 的服务时间与对象存储设备的关系。同样以 512kB 为界，小于 512kB 的文件通过哈希函数映射到某一个对象存储设备中，而大于 512kB 的文件，分条到不同的对象存储设备中。



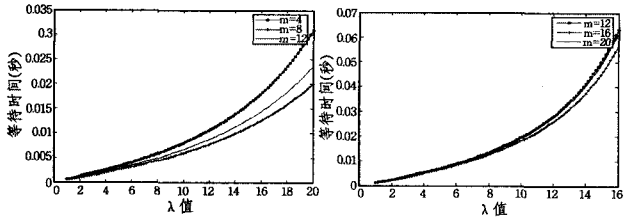
(a) 文件为 1MB，2MB 和 4MB　(b) 文件为 8MB 和 16MB

(c) 文件为 32MB，64MB 和 128MB　(d) 文件为 256MB，512MB 和 1GB

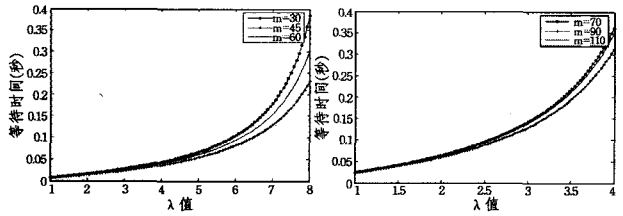
图 4　对象存储设备与平均服务时间的关系

从图 4(a)可以看出，当文件小于或者等于 1MB 时，对象存储设备个数比较合理值为 4；当文件大于 1MB、小于 2MB 时，对象存储设备个数比较合理值为 6；当文件大于 2MB、小于 4MB 时，对象存储设备个数比较合理值为 8。图 4(b)显示了当文件大于 4MB、小于 8MB，对象存储设备个数比较合理值为 12；当文件大于 8MB、小于 16MB 时，对象存储设备个数比较合理值为 16。在图 4(c)中，当文件大于 16MB、小于 32MB 时，对象存储设备个数比较合理值为 20；当文件为大于 32MB、小于 64MB 时，对象存储设备个数比较合理值为 30；当文件为大于 64MB、小于 128MB 时，对象存储设备个数比较合理值为 45。同样，图 4(d)表示文件大于 128MB、小于 256MB 时，对象存储设备个数比较合理值为 65；当文件为大于 256MB、小于 512MB 时，对象存储设备个数比较合理值为 90；当文件为大于 512MB、小于 1GB 时，对象存储设备个数比较合理值为 120。

图 5 表示文件大小分别为 4MB，16MB，128MB 和 512MB 时，平均等待时间、对象存储设备数目与平均请求到达率的关系。从图中可以看出，在对象存储设备数目未达比较合理值时，随着对象存储设备数目的增加，在相同的平均请求到达率下平均等待时间减少；然而，当超过对象存储设备数目的比较合理值后，在相同的平均请求到达率下平均等待时间反而增大了。



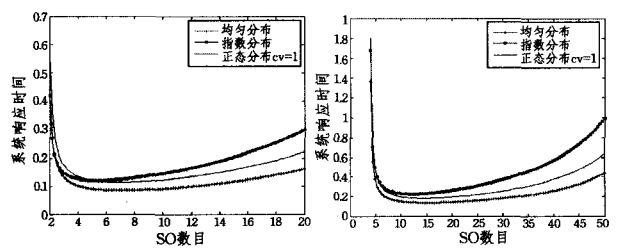
(a) 文件为 4MB 的平均等待时间　　(b) 文件为 16MB 的平均等待时间



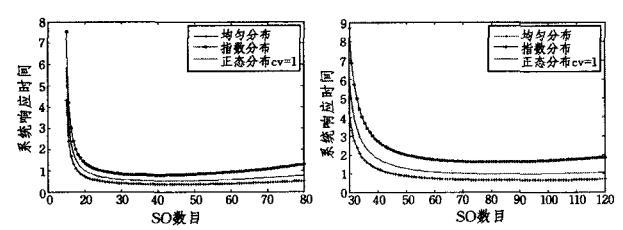
(c) 文件为 128MB 的平均等待时间　　(d) 文件为 512MB 的平均等待时间

图 5　对象存储设备与平均响应时间的关系

依据上述情况，以 $\lambda_i=20$、文件大小为 4MB，$\lambda_i=12$、文件大小为 16MB，$\lambda_i=6$、文件大小为 128MB，$\lambda_i=3$、文件大小为 512MB 为例，在不同的分布时系统响应时间与对象存储设备的关系如图 6 所示。从图中可以看出，对象存储设备所取得最佳值与所取得比较合理值相近，当对象存储设备未到达最佳值时，对象存储设备数目越多，I/O 响应时间越短，一旦超过对象存储设备最佳值，I/O 响应时间又逐步回升。



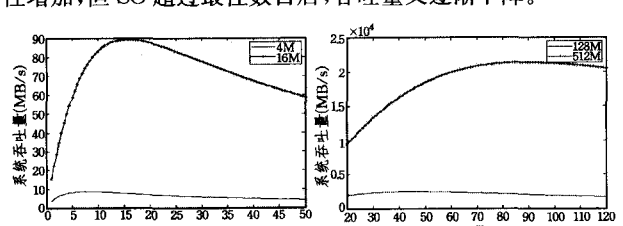
(a) 文件为 4MB 的系统响应时间　　(b) 文件为 16MB 的系统响应时间



(c) 文件为 128MB 的系统响应时间　　(d) 文件为 512MB 的系统响应时间

图 6　系统响应时间与对象存储设备的关系

当 $\lambda_i E(T_s)=1$ 时，则 $\lambda_{max}=m/E(T_s)$，最大吞吐量 $TP_{max}=\lambda_{max}\times B\times E[L]$，其中 B 为数据分条大小，即 $B=L/m$，L 为 I/O 请求大小。图 7 显示在不同 SO 数目中系统最大吞吐量，可以看出，随着 SO 的增加，系统吞吐量几乎成线性增加，但 SO 超过最佳数目后，吞吐量又逐渐下降。



(a) 4MB 和 16MB 的系统吞吐量　　(b) 128MB 和 512MB 的系统吞吐量

图 7　不同 SO 数目的系统最大吞吐量

结束语　在海量存储系统中，负载均衡一直是研究的重点，与负载有关的因素较多，如请求队列长度、CPU 处理能力、内存大小、网络带宽和磁盘带宽等。

由于 BSO-MSS 具有分布式存储系统体系结构特点，MDS 能够将用户文件对象分割为若干个数据对象分布到多个 SO 中。而如何选择 SO 及数目是关键，本文提出的柔性负载均衡策略，不仅考虑网络对 BSO-MSS 的影响，而且更关注 SO 本身，针对 SO 中不同的存储能力，自适应选择 SO 数目，采用不同大小的分条进行存储。当 SO 数目未达到最佳值时，SO 数目越多，I/O 响应时间越短，系统能响应的请求越多，且吞吐量越大。因此，采用负载柔性放置能够使 BSO-MSS 具有更高的吞吐量、可扩展性和负载均衡。

参 考 文 献

[1] Intel Corporation. Object-Based Storage: The Next Wave of Storage Technology and Devices[OL]. January 2004. http://www.intel.com/labs/storage/osd

[2] Panasas Inc. Object Storage Architecture-Defining a new generation of storage systems built on distributed, intelligent storage devices[OL]. Rev. 1. 0-10/19/2003. http://www.panasas.com/docs

[3] Mesnier M, Ganger G R, Riedel E. Object-based storage [J]. Communication Magazine, IEEE, 2003, 41(8): 84-90

[4] Hospodor A, Miller E L. Interconnection architectures for petabyte-scale high-performance storage systems[A]// the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies[C]. April 2004: 273-81

[5] Liu Qun, Feng Dan, Qin Ling-jun, et al. A Framework for Accessing General Object Storage[A]// the 2006 International Workshop on Networking, Architecture, and Storages (IWNAS2006[C]. Dalian, China: IEEE, 2006: 145-148

[6] Liu Qun, Feng Dan. An Approximate Analytic Performance Model of Object-Based Storage[A]// the International Conference on Computational Science and Its Applications, LNCS 3980[C]. Berlin: Spring, 2006: 1045-1052

[7] Cabrera L-F, Long D D E. Swift: Using Distributed Disk Striping to Provide High I/O Data Rates[J]. Computing Systems, 1991, 4(4): 405-439

[8] Wang Fang, Zhang Shun-da, Feng Dan, et al. A Hybrid Scheme for Object Allocation in a Distributed Object-Storage System [A]// the International Conference on Computational Science [C]. May 2006: 396-403

[9] Roselli D, Lorch J, Anderson T. A comparison of file system workloads[A]// the 2000 USENIX Annual Technical Conference[C]. June 2000: 41-54

[10] Hennessy J L, Patterson D A. 计算机体系结构量化研究方法(英文版 第 3 版)[M]. 北京: 机械出版社, 2002: 793-802

[11] 罗荣桂. 排队模型及其应用[M]. 武汉: 华中科技大学出版社, 1990

[12] 冯丹. 外存储系统并行性研究[D]. 武汉: 华中科技大学, 1997

[13] Kim M Y, Tantawi A N. Asynchronous Disk Interleaving: Approximating Access Delays[J]. IEEE Transactions on Computers, 1991, 40(7): 801-810

[14] 周可. 外存储系统数据组织与体系结构[D]. 武汉: 华中科技大学, 2003

[15] Simitci H, Reed D A. Adaptive Disk Striping for Parallel Input/Output[A]// the 7th NASA Goddard Conference on Mass Storage Systems and Technologies[C]. March 1999: 88-102