2009年4月 第4期
Vol. 30 No. 4 2009

# 集群多媒体存储系统的两级元数据管理

万继光，詹 玲

（华中科技大学 计算机科学与技术学院 武汉光电国家实验室，湖北 武汉 430074）
E-mail：jgwan@mail.hust.edu.cn

**摘 要**：随着网络上多媒体数据的爆炸性增长，对海量可扩展的存储系统的需求也快速增长. CMSS(Cluster Multimedia Storage System)项目采用分布式存储系统结构：一种自治的高性能的基于PC的存储集群系统. CMSS采用两级的元数据服务器结构，通过分离存储数据的逻辑视图与物理视图，全局逻辑视图由专用的全局元数据服务器来管理，局部逻辑视图和物理视图由各个存储服务器上的本地元数据服务器来管理. 在详细介绍了CMSS系统两级元数据管理方案的同时，进行了相应的试验测试和性能分析.

**关 键 词**：多媒体；集群；存储系统；元数据服务器
**中图分类号**：TP333 **文献标识码**：A **文 章 编 号**：1000-1220(2009)04-0752-05

## Two-level Metadata Management of a Cluster Multimedia Storage System

WAN Ji-guang, ZHAN Ling

(*College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan National Laborartory for Optoelectronics, Wuhan* 430074, *China*)

**Abstract**: As a result of the multimedia rapid growth, demand for large-scale multimedia storage system is growing rapidly. The CMSS (Cluster Multimedia Storage System) project adopts decentralized storage systems-a self-managing high-performance PC-based cluster storage system. This paper describes the design and implementation of the Two-level metadata server for CMSS. The CMSS divides the logical view of the stored data from the physical view. The logical view is managed by the metadata server which is called GMS(Global Metadata Server), and the physical view is managed by a component of storage servers which is called LMS(Local Metadata Server). The paper introduced the design of the CMSS in detail, and perform a intensively performance evaluation study on the CMSS.

**Key words**: multimedia; cluster; storage system; metadata server

## 1 前 言

随着网络上多媒体数据的爆炸性增长，大量的多媒体数据在世界各地产生并共享，从而对海量可扩展的存储系统的需求也快速增长. 现有存储系统技术受到空前的挑战：信息多媒体化对容量产生巨大的需求；365天×24小时不间断的运行要求系统具有极高的可用性；信息量持续不断的增加要求系统具有良好的可扩展性. 网络应用的发展对上述大容量、高可靠性、高可用性、高性能、动态可扩展性、易维护性和开放性等众多方面提出更高的需求，现有技术已受到相当大的压力.

多媒体数据存储有它自己独特的特点：系统存储大量的大文件，典型的文件有几百MB，甚至一些文件达到几GB，因此要重点考虑大文件的优化；主要的负载是大的顺序读，通常一个请求能够读几百KB的数据，小的随机读也存在，但是相对比较少；写操作主要是大的顺序写，且一旦文件写完，很少对文件进行修改；系统的带宽和响应时间都很重要，大量的大的读请求需要很高的带宽，而VOD等应用对响应时间也有很高的要求.

针对多媒体存储的特点，采用PC集群技术，我们设计并实现了一种基于PC集群的高性能多媒体存储系统CMSS (Cluster Multimedia Storage System). CMSS采用两级元数据服务器结构，通过分离存储数据的逻辑视图与物理视图，全局逻辑视图由专用的全局元数据服务器GMS(Global Metadata Server)来管理，局部逻辑视图和物理视图由各个存储服务器上的本地元数据服务器LMS(Local Metadata Server)来管理. 采用两级元数据管理技术，即实现了单一命名空间，又避免了单点失效；即缩短了请求的处理路径，也避免了传统集中元数据服务器的性能瓶颈.

## 2 相关研究工作

我们的研究工作借鉴了国内外关于分布式并行文件系统和存储集群系统的研究工作. 蓝鲸分布式文件系统(BWFS)[1,2]是国家高性能计算机工程技术研究中心自主设计实现的分布式文件系统，它着重于大容量、高I/O吞吐率和高扩展能力等方面特性. Sorrento[3,4]项目主要目的是构建

收稿日期：2007-11-20 基金项目：国家自然科学基金项目(60603075)资助；国家"九七三"重点基础研究发展计划项目(2004CB318203)资助. 作者简介：万继光，男，1972年生，讲师，博士，研究方向为网络存储系统；詹 玲，女，1973年生，讲师，博士研究生，研究方向为光存储系统、网络存储系统.

一个基于宽带网络的 PC 或工作站存储集群,它将分布的存储设备虚拟为一个可扩展的卷. Clemson 大学的并行虚拟文件系统(PVFS)[5,6]项目用来为运行 Linux[R] 操作系统的 PC 群集创建一个开放源码的并行文件系统,PVFS 将数据存储到多个群集节点的已有的文件系统中;多个客户端可以同时访问这些数据.但是,PVFS 集中的元数据管理成为整个系统的瓶颈,可扩展性受到一定限制;而且采用静态配置,不具备动态扩展性. Lustre[7,8]是 HP,Intel,Cluster File System 公司等开发的 Linux 集群并行文件系统,是第一个基于对象存储设备的,开源的并行文件系统. Lustre 采用分布式的锁管理机制来实现并发控制,元数据和文件数据的通讯链路分开管理. OceanStore[9-11]通过基于 Internet 的分布式存储系统来解决不断增长的存储需求. OceanStore 提供透明的存储共享,即使设备丢失或损坏,它也能够保护数据不丢失.

## 3　体系结构设计

多媒体数据具有数据量巨大的特征,并且要求有很好的可扩展性,在对其存储和传输等方面均有很高的要求,而且多媒体数据中的音频和视频都是随时间而变化的信息,即是具有很强时间特性,因此,要求存储系统能对其进行实时处理,这就要求存储系统有更高的响应时间.

多媒体应用的特点是读请求远远多于写请求,对存储数据的修改很少,因此 CMSS 采用基于 PC 的存储集群结构,每个节点管理自己的存储设备,共同组织为一个可扩展的高性能存储卷,图 1 给出了 CMSS 的通用结构,CMSS 由三种服务器节点组成:全局元数据服务器(GMS)、存储服务器和多媒体服务器节点.全局元数据服务器负责整体系统的全局元数据管理;存储服务器组织分布式的存储设备,给用户提供一个

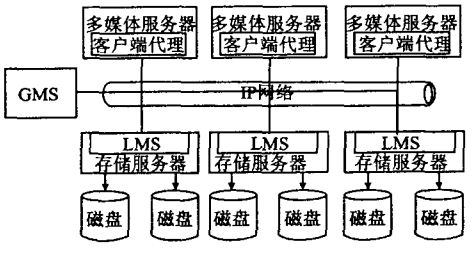


图 1　CMSS 体系结构

Fig. 1　Architecture of CMSS

可扩展的存储卷,存储在该卷上的数据采用层次目录结构组织;多媒体服务器通过 CMSS 客户端代理来访问 CMSS 系统.改变传统的把所有用户请求都要通过全局元数据服务器方式,而是让用户请求分散到各个存储服务器,这样大大增强各个存储服务器的并行度,实现了负载平衡,从而提高了整个集群的性能.

CMSS 系统采用高自治、自维护的分布式策略,各个存储服务器能够独立地维护自己的存储资源和元数据及数据本身,并能够独立提供存储服务,从而提高系统的可扩展性. CMSS 系统采用智能数据放置策略,通过记录数据的访问信息,分析数据的相关性,将有相关性的数据尽可能放置在同一个存储服务器,从而减少存储服务器之间的数据相关性,从而将任何节点的失效对其它节点的影响降低到最小,提高系统的可用性.

对于数据的放置,提供两种策略,一种是分布式放置策略,就是将文件和目录分布到各个存储服务器上,分布的粒度可以是文件或子目录,文件本身不被拆分到不同的存储服务器上.这种策略保证一个文件在一个存储服务器上的完整性,存储服务器具有比较好的自治性,从而保证系统的可扩展性.

另外一种是并行放置策略,就是将文件安装 RAID0 的方式拆分放置到各个存储服务器上,它类似于 PVFS 的数据放置策略,被拆分的文件由 GMS 负责管理.这种策略能够解决热点数据负载平衡的问题,这个系统具有更好的性能,但是,由于这种文件是由 GMS 来管理,会增加 GMS 的负载,降低存储服务器的自治性,也降低整个系统的可扩展性.

两种放置策略各有自己的优缺点,一般建议将常用的文件采用并行放置策略,从而提高访问的性能;将不常用的文件采用分布放置策略,从而保证系统的可扩展性.采用智能的动态放置策略是本项目的下一步研究工作,就是系统根据记录的访问信息,动态的将数据在两种放置策略之间转换.

## 4　两级元数据管理

当前集中式存储系统元数据服务器需要管理系统中的每个文件的每个数据块,其限制了系统的性能和可扩展性.因为元数据服务器是在存储服务器和应用服务器的数据通路上,它成为系统性能和容量扩展性的主要瓶颈.

当前存储系统的元数据服务器主要功能包括两个方面:首先,它给应用服务器提供一个全局的数据逻辑视图(单一命名空间层),包含文件列表和目录结构等;其次,它负责数据在物理存储媒体上的组织,也就是管理数据存储的物理视图.

为了解决元数据服务器的瓶颈问题,针对多媒体数据访问的特点,CMSS 系统将存储数据的逻辑视图同物理视图分离,管理存储数据全局逻辑视图的服务器称为 GMS(Global Metadata Server),而存储数据的物理视图由存储服务器上的 LMS(Local Metadata Server)负责管理,每个存储服务器上都有一个 LMS,它负责管理该存储服务器的物理视图,另外 LMS 也负责管理该存储服务器的局部逻辑视图,所有的 LMS 由 GMS 统一协调管理,共同合作完成元数据服务器的管理工作,如图 1.

CMSS 允许应用服务器直接访问存储服务器上的数据,从而多个应用服务器同存储服务器之间可以并行工作,从而提高系统的性能.另外,采用分布式的元数据管理解决了原来中央元数据服务器的瓶颈问题.因此,CMSS 为需要高性能和可管理性的多媒体应用提供一个完整的解决方案.

### 4.1　全局元数据服务器 GMS

CMSS 分布系统的元数据到各个存储服务器中,由各个存储服务器的 LMS 自主管理,从而降低了 GMS 的开销.不同于传统的存储系统,GMS 不管理各个存储服务器内部的元数据,仅仅管理跨多个存储服务器的元数据,从而减轻了

GMS 的负担，提高了系统的性能和可扩展性.

为了实现系统的单一命名空间的同时保证各个存储服务器的高度自治性，原则上一个目录下的文件和子目录都放置在相同的存储服务器，只有当各个存储服务器上的数据存放负载出现较大的不平衡时，才会将一个目录下的子目录移动或新建到其它存储服务器上. 另外，为了尽可能实现静态负载平衡，全局根目录下的 N 级目录尽可能分布在不同的存储服务器，在测试系统中，N 的值设定为 4.

另外，GMS 还负责管理整个系统的存储资源信息，用于数据放置算法使用，每个 LMS 维护自己的存储资源状态表，如表 1，并定时向 GMS 报告该表的状态改变.

表 1 存储服务器资源状态表

Table 1 Resource table of storage server

| 定义 | 说明 |
|---|---|
| int iResourceID | 存储资源 ID |
| char acResourceName [MAX_RESOURCE_NAME] | 资源名称 |
| char acServerIP[MAX_SERVER_IP] | 存储服务器 IP 地址 |
| long long llAllCapacity | 总存储空间的大小 |
| long long llAvailableCapacity | 剩余存储空间大小 |
| int iResourcePerformence | 存储资源的性能 |
| time_t zLastAccessTime | 最后访问时间 |
| int iAccessTimes | 访问次数 |

按照上面的数据放置策略，各个存储服务器上的元数据以多个独立的目录子树的形式存在，由LMS来管理这些目

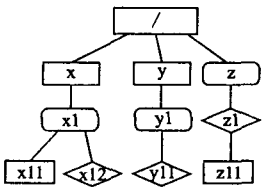


图 2 一个全局命名空间目录树实例

Fig. 2 Instance of global namespace directory tree

录子树，并提交给 GMS，由 GMS 汇总生成一个全局目录树，也就是命名空间目录树. 图 2 是命名空间目录树的一个实例，矩形框表示的节点存储在存储服务器 1 上，菱形框表示的节点存储在存储服务器 2 上，椭圆框表示的节点存储在存储服务器 3 上.

### 4.2 本地元数据服务器 LMS

在 CMSS 中，存储系统的物理视图和局部逻辑视图由各个存储服务器的 LMS 来管理. 它通过本地文件系统来管理存储在其上的文件或文件段. 因此 90%以上的元数据管理被分布到各个存储服务器上，由其自主管理，从而减少了应用服务器同 GMS 的通讯，这大大提高存储系统元数据管理的性能. 另外，因为元数据是分布式管理的，增加更多的存储服务器能够增加系统的性能和容量.

另外，LMS 还负责管理它所在存储服务器上的文件和目录的局部逻辑视图，这个局部逻辑视图是全局逻辑视图的一部分. 图 3 是图 2 实例在各个存储服务器上的部分命名空间目录树，图4中的(a)、(b)和(c)分别是存储服务器1、2和3

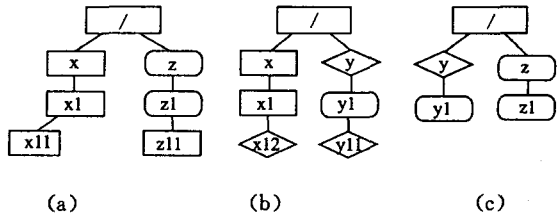


图 3 部分命名空间目录树实例

Fig. 3 Instance of part namespace directory tree

的部分命名空间目录树. 为了能够独立对外提供服务，每个 LMS 除了保存自己的文件和目录信息外，还保存一些跟它紧密相关的目录信息，如图 4(a)中，目录 z11 的 2 个父目录 z 和 z1 虽然存储在其它的存储服务器上，为了保证 z11 目录的路径完整性，在存储服务器 1 的 LMS 中也会保持这 2 个父目录的基本信息.

### 4.3 GMS 的双机热备技术

采用两级元数据管理，GMS 的负载大大减轻，不需要物理的服务器作为 GMS，而是采用可以运行在任何存储服务器上的逻辑 GMS. 为了提高 GMS 的可靠性，GMS 采用双机热备的方式来实现，系统初始化时在两台存储服务器上启动 GMS 例程，其中一台服务器上的为活动 GMS，另外一台为备份 GMS，活动 GMS 上的所有元数据都在备份 GMS 上进行实时备份. 一旦活动 GMS 失效，备份 GMS 将接管活动 GMS 的任务，从而避免了系统的单点失效，提高了系统的可用性.

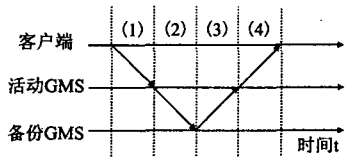


图 4 GMS 元数据的修改时序图

Fig. 4 Scheduling for modify GMS metadata

GMS 上的元数据修改过程如图 4，当客户端需要修改全局元数据时，它首先向活动 GMS 发送修改请求，如步骤(1)；活动 GMS 修改它的元数据，并将元数据的改变通知备份 GMS，如步骤(2)；备份 GMS 接到通知后，也修改自己的元数据，从而跟活动 GMS 的元数据保持一致，并将结果返回活动 GMS，如步骤(3)；活动 GMS 在将最终的结果返回客户端，如步骤(4)，整个的修改过程才最终完成. 因为在数据放置时采用高内聚低耦合的策略，不同 LMS 之间的元数据比较少，并且因为是多媒体数据，其修改很少，所以 GMS 上的元数据很少改变，因而其修改开销对整个系统的影响很小.

### 4.4 全局元数据 Cache 及其一致性算法

GMS 只保存命名空间的上层的几级目录和同父目录不在同一个服务器的目录或文件信息，这样 GMS 保存的全局

命名空间信息很少,且相对静态.为了提高性能,GMS 收集完成所有的全局命名空间信息后,将它发给所有的存储服务器.用户节点代理程序只是在第一次请求处理时访问 GMS 服务器,并将 GMS 的全局命名空间信息 Cache 在客户端,用户节点代理程序有这个全局命名空间信息,就能够直接计算出请求的文件和目录所在的存储服务器,不需要访问 GMS,从而缩短了请求的处理路径,提高了系统的性能.

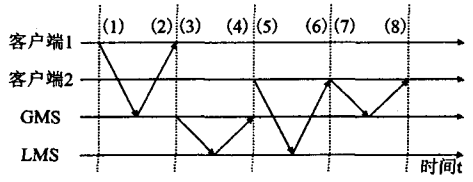


图 5　全局元数据 Cache 一致性算法图

Fig. 5　Consistency arithmetic for GMS cache

因为多媒体应用对数据一致性要求不高,CMSS 采用元数据弱数据一致性算法,一个客户端修改了某个存储服务器的元数据,可能需要几秒或几十秒的时间在其它客户端节点上更新.

CMSS 系统采用命名空间版本号的技术来解决客户端命名空间同服务器命名空间的一致性问题,客户端和服务器上都保存有命名空间的一个 32 位版本号,服务器的版本号从 1 开始计数,每当命名空间发生修改,版本号就加一,因为命名空间修改的频率不高,就算一分钟修改一次,32 位的版本号要 8171 年才能够溢出.全局元数据 Cache 一致性算法如图 5,当客户端 1 修改了 GMS 的元数据,如步骤(1)和(2);GMS 服务器会更新所有的 LMS 服务器,如步骤(3)和(4);当客户端 2 访问 LMS 服务器时,LMS 服务器在返回数据本身的同时捎带上服务器的命名空间版本号,如步骤(5)和(6);客户端收到这个版本号后,就跟自己的版本号比较,如果不一致,就从 GMS 服务器上更新自己 Cache 的命名空间,并更新版本号,如步骤(7)和(8);这样就能够保证命名空间的一致性.

## 5　性能测试与结果分析

因为多媒体数据的特点是存储的文件很大,文件数目比较少,因此元数据比较少,用户访问主要集中在数据的本身,而且用户对存储的数据很少修改,所以,多媒体存储系统的性能关键在于对多媒体数据的读请求处理上,因此我们的测试重点是多媒体的读性能.

### 5.1　性能测试平台

为了验证 CMSS 系统的性能,在 1000Mb 以太网环境对其进行了性能测试.采用的测试工具为 XDD,XDD 是在单个系统或者集群系统上测量存储系统 I/O 的工具.它原来是用在 UNIX 系统上基于命令行的工具,现在也可以在 windows 平台上使用.它有三个基本的组件:XDD 程序自身、一个全局时间服务器和一个同步时间的程序.全局时间服务器和同步时间的程序用来同步多个运行 XDD 的计算机系统的时钟.

测试是在 22 台通过 1000Mbps 的 IP 交换机连接起来的 PC 机上进行的,其中 1 台是元数据服务器,其中 1 台用于 XDD 同步,10 台是客户机,另 10 台是服务器.各个计算机的配置大致相同:技嘉主板 GA945G 主板;Intel Celeron 2.8GHZ CPU;512M DDR2 RAM;2 个 Seagate 7200 转 300GB SATA 硬盘做 RAID0,单盘最大数传率 60MB/s;Broadcom 1000Mb 网卡;交换机为 Cisco 3750 千兆以太网交换机;操作系统分别为 Fedora Core 5.0.为进行对比,分别在此平台上分别测试了 CMSS 文件系统、Lustre 文件系统和 PVFS 文件系统的性能.

### 5.2　测试数据及分析

对于 CMSS 的分布式并行放置策略,因为系统采用两级元数据管理方式,各个存储服务器能够管理自己的元数据,从而能够独立对外提供服务,整个 CMSS 系统的性能是各个服务器的性能之和,当各个客户端的请求平衡分布到各个服务器上时,整个 CMSS 系统能够获得最好的性能,当各个客户端的请求集中到一个服务器上时,系统性能降到最低.因此用户的请求分布情况对 CMSS 系统性能影响很大,动态负载平衡是 CMSS 系统下一步要解决的问题.本文主要测试的是 CMSS 的并行性能.

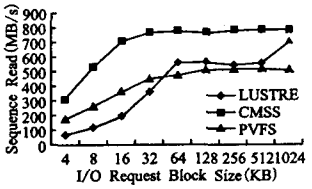


图 6　顺序读数传率曲线

Fig. 6　Sequence read curve of data transmission rate

顺序读的性能测试结果如图 6,可以看出顺序读 CMSS 服务器性能要明显高于 LUSTRE 和 PVFS 服务器.因为 CMSS 采用 GMS 客户端 Cache 技术,只是在第一次请求时访问GMS服务器,并将GMS的数据Cache在客户端,以后的

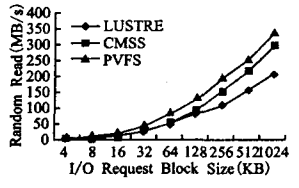


图 7　随机读数传率曲线

Fig. 7　Random read curve of data transmission rate

请求就不需要访问 GMS 服务器,而 LUSTRE 和 PVFS 要经常访问元数据服务器,因此影响了系统的性能.另外 CMSS 针对的多媒体的顺序读的特点进行了 Cache 优化,也可以提高顺序读的性能.

随机读情况下,CMSS 服务器性能要高于 LUSTRE,但

是低于 PVFS 服务器,如图 7. 因为 CMSS 系统是针对多媒体应用的,随机读情况比较少,所以 CMSS 系统采用的是顺序读优化的策略,对随机读性能有一定的影响. 另外,GMS 客户端 Cache 的技术,对 CMSS 的性能有一定的提高.

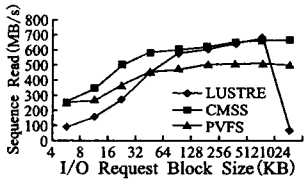


图 8 顺序写数传率曲线

Fig. 8 Sequence write curve of data transmission rate

写性能测试如图 8、图 9 所示,总的来说,CMSS 的顺序写性能比较好,随机写性能要差一些. 这个也是基于多媒体的应用,顺序写比较多,随机写比较少,从而设计上采用顺序写性能优化.

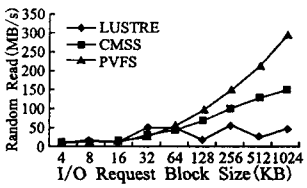


图 9 随机写数传率曲线

Fig. 9 Random write curve of data transmission rate

## 6 结束语

CMSS 系统主要是针对多媒体应用来设计的,采用高自治、自维护的 PC 集群结构,提高了系统的并行性和可扩展性;CMSS 采用两级的元数据服务器结构,通过分离存储数据的逻辑视图与物理视图,逻辑视图由专用的全局元数据服务器 GMS 来管理,物理视图由各个存储服务器上的本地元数据服务器 LMS 来管理. 采用 GMS 双机热备技术,即实现了单一命名空间,又避免了单点失效;采用全局元数据 Cache 技术,缩短了请求的处理路径,从而提高了系统的性能. 采用智能的动态放置策略是本项目的下一步研究工作,系统能够根据用户的访问记录,动态的将数据在两种放置策略之间转换,以提高系统的总体性能.

**References:**

[1] Huang Hua, Zhang Jian-gang, Xu Lu. Distributed layered resource management model in blue whale distributed file system [J]. Journal of Computer Research and Development, 2005, 42 (6): 1034-1038.

[2] Yang De-zhi, Huang Hua, Zhang Jian-gang, et al. BWFS: a distributed file system with large capacity[J]. High Throughput and High Scalabil ity. Journal of Computer Research and Development, 2005, 42(6): 1028-1033.

[3] Tang H, Gulbeden A, et al. Sorrento: a self-organizing storage cluster for parallel data-intensive applications[C]. Proceedings of the International Conference for High Performance Computing Networking and Storage, 2004.

[4] Tang H, Yang T. An efficient data location protocol for self-managing storage clusters[C]. Proceedings of the International Conference for High Performance Computing and Communications, 2003.

[5] Carns P H, Ligon Ⅲ W B, Ross R B, et al. PVFS: a parallel file system for Linux clusters[C]. Proc. of the Extreme Linux Track: 4th Annual Linux Showcase and Conference, October 2000.

[6] Avery Ching, Choudhary A, Liao Wei-keng. Noncontiguous I/O through PVFS[J]. Cluster Computing, 2002.

[7] Lustre: a scalable, high-performance file system cluster file systems [EB/OL]. http://www. lustre. org/docs/luster. pdf November, 2002.

[8] Braam B J. Selecting a scalable cluster file system cluster file systems [EB/OL]. http://www. lustre. org/docs/selecting-a-cfs. pdf November 2005.

[9] Kubiatowicz J, Bindel D, Chen Y, et al. Ocean store: an architecture for global-scale persistent storage[C]. Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 00), ACM, 2000.

[10] Rhea S, Wells C, Eaton P, et al. Maintenance-free global data storage[J]. IEEE Internet Computing, 2001, 5(5): 40-49.

[11] Rhea S, Eaton P, Geels D. Pond: the ocean store prototype [C]. File and Storage Technologies (FAST'02), CA, USA. Jan. 28, 2002, 43-59.

**附中文参考文献:**

[1] 黄 华,张建刚,许 鲁. 蓝鲸分布式文件系统的分布式分层资源管理模型[J]. 计算机研究与发展,2005,42(6):1034-1038.

[2] 杨德志,黄 华,张建刚,等. 大容量、高性能、高扩展能力的蓝鲸分布式文件系统[J]. 计算机研究与发展,2005,42(6):1028-1033.