华中科技大学

硕士学位论文

存储虚拟化系统元数据管理的设计与实现

姓名：贾永洁

申请学位级别：硕士

专业：计算机系统结构

指导教师：金海

20040312

摘 要

随着现代社会中信息的高速增长，现有的数据管理体系已经无法有效管理广域环境中多达几十个 TeraByte 甚至 PetaByte 的数据，存储虚拟化技术的发展为解决这个问题提供了一条有效的技术途径。

在存储虚拟化系统中，元数据处于相当重要的地位。它是描述数据的数据，为系统提供单一全局用户空间、数据的定位和属性查找、数据的注册、数据访问控制信息和用户管理。为了方便有效的访问各种异构数据组成的数据集合，需要建立灵活的、可扩展的元数据管理机制。

在许多系统中比如 SRB（Storage Resource Broker），都采用层次目录结构管理元数据，由于元数据固有的层次性和结构单一性，这种管理方式有其方便简明的一面。但是，随着数据的激增，元数据变得越来越庞大，以前的组织结构和管理方式在扩展性、可用性和效率方面都不能很好的满足需求。

针对上述问题，在广域网虚拟化存储系统 GDSS（Global Distributed Storage System）中提出一种元数据层次管理模型 MDC（MetaData Controller）。MDC 提出一种基于匹配表的分级元数据管理方式，通过一次匹配就可以找到元数据所在的目录服务器，避免从根目录服务器的层层查询。同时，MDC 提供的 cache 模块保存热点元数据和正在被执行写操作的元数据，极大提高元数据读写效率。

基于 GDSS 系统已经实现了 MDC 元数据管理模型，通过测试证明，它可以提供并行元数据访问（快速搜索），对元数据稳定快速的定位，同时还可以提供一定的元数据容错能力，达到了很好的可用性和可扩展性，并更加易于管理。

关键词： 存储虚拟化，全局命名服务器，元数据，层次管理，匹配表

Abstract

Data-intensive, high-performance computing applications require the efficient management and transfer of TeraByte or PetaByte of information in wide-area, distributed computing environments. Storage virtualization integrates all kinds of high performance storage system to a unit one. It can't only share resources and make fully use of resources, but efficiently avoid conflict of data explode and limited storage ability. To resolve these conflict, After two years of research on storage virtualization, we developed GDSS(Global Distributed Storage System).

In GDSS, metadata is very important for it takes charge with characterization of data, so replica for metadata is a must. Although replica can fault tolerance and load balance, it also brings the problem of replica coherence. At the same time, In many systems, such as SRB, hierarchical directory structure is adopted. Generally when the metadata becomes enormous, the system employs several metadata servers. There are several limitations for directory servers. One is that it must keep the logic tree among the directory servers; the other is that the directory servers must cooperate and the result will return from the root node that adds the overload to the root server; the third is that when the root server is out of service the whole meta server will out of service too, so it is difficult to provide high availability; and the last is that it is difficult to expand the scale of meta servers.

Based on those problems, GDSS introduce a novel hierarchy model MDC(MetaData Controller)to manage metadata. In MDC, MatchTable module is charged with communication of SSP(Storage Service Point) and DS(Directory Server), and cache module is charged with increase of metadata access efficiency and as a assistant to maintain the coherence of metadata replica. MatchTable saves name and root of each DS, and cache saves hot metadata and metadata being updated. Update operation is only done on cache, and DS is updated by replica coherence module, which can avoid conflict of synchronously access to replicas. If cache doesn't hit, access a seemly DS through MatchTable. MDC can reach better efficiency, scalability and management.

Key words: storage virtualization, global naming server, metadata，hierarchic management，MatchTable

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：贾永洁

日期：2004年3月9日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在_____年解密后适用本授权书。

本论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：贾永洁　　指导教师签名：

日期：2004年3月9日　　日期：2004年3月9日

1 绪论

本章首先简述存储虚拟化技术的研究背景，接着介绍国内外类似系统当前的发展状况，然后解释几个存储虚拟化系统的相关概念，包括存储虚拟化、元数据和元数据副本等，最后说明元数据在存储虚拟化系统中的作用和本文将要做的研究工作。

1.1 研究背景

当今社会，信息正以超乎人们想象的速度增长。随着现代大型科学工程研究、信息服务和数字媒体等应用中的数据呈爆炸式的增长，数据已经成为一个重要的资源，例如：全球气候模拟、数字地球、电子商务、数字媒体、数字化图书馆、企业数据中心、生物信息、地理信息系统、航空航天、军事仿真等等应用，它们的数据量将达到几十个 TeraByte 至 PetaByte 的级别，这对信息存储系统的容量和速度提出了空前的要求，单个的存储设备已经远远不能满足这些科学应用的需求，现有的网络存储技术也受到巨大的挑战。人们对信息数据日益广泛的需求导致存储系统的规模变得越来越庞大，管理越来越复杂，信息资源的爆炸性增长和管理能力的相对不足之间的矛盾日益尖锐。同时，这种信息资源的高速增长也对存储系统的可靠性和扩展性提出了挑战，信息资源的共享也显得越来越重要。

在广域网中存在大量相互独立的数据孤岛，它们之间的数据资源不能共享，存储空间不能得到有效使用，数据的传输性能不足，同时地理上广泛分布的用户都希望能够访问、分析和使用这些庞大的分布数据，而相关的分析往往计算复杂和计算量大，这种结合海量数据集合、地理上分布的用户和资源以及计算密集型的分析处理应用导致了现有的数据管理体系结构、方法和技术已经不能满足高性能、大容量分布存储和分布处理能力的要求，如何存储、分发、组织和管理、高性能处理、分析和挖掘海量分布数据成为许多应用的首要问题。

存储虚拟化[1-3]技术的发展为解决这个问题提供了一条有效的技术途径，它通过集成网络上分布的多个数据集等资源，形成单一虚拟的数据访问、管理和处理环境，为用户屏蔽底层异构的物理资源，建立分布海量数据的一体化数据访问、存储、传输、管理与服务架构。

1.2 国内外概况

1.2.1 国内外类似系统概况

存储虚拟化技术的发展非常迅速，在此研究领域，美国和欧洲处于领先地位，他们的研究范围和规模都比较大，并且已经推出了一些试验系统，其中最著名的以欧洲数据网格项目、数据网格系统工具 Globus 中的数据网格支撑模块和美国圣地亚哥超级计算中心 SDSC（San Diego Supercomputer Center at UCSD）的 SRB 系统。

Globus[4-8]系统主要由美国 Argonne 国家实验室和南加州大学联合研制。它借鉴了因特网和 Unix 的开发路线，不构造一个完整的系统，而只构造一套底层的开发工具，采用模块化设计方式，可升级或替换，是一个中间件系统。Globus 对资源的管理、安全、信息服务、数据管理等网格计算的关键技术和方法进行研究，提供了一整套 SDK 和 API，用户可以任意选择其中的工具模块进行高层次的应用开发。Globus 系统最初是面向计算网格的，后来由于数据网格应用的需求迫切，Globus 系统使用了标准的协议实现了数据文件的移动和远程访问 GASS[9]和数据的高速传输 Gridftp[10-12]基本机制，在此基础上实现数据复制、元数据目录的管理和复制，为数据网格系统提供了一个较好的底层系统开发平台。

欧洲数据网格[13,14]的目标是以欧洲粒子中心(CERN)的从 TeraByte 到 PetaByte 规模数据处理为中心，为世界范围内分布的科研团体提供数据分布存储、传输和计算密集型分析处理能力，以进行科学研究，开展面向高能物理学、地球观测、生物信息学等应用的研究工作，研究内容主要包括：数据访问、数据副本管理、元数据管理、数据安全、查询优化、资源调度和管理等，采用 Globus、面向对象数据库、网格数据库服务系统等技术，构建一个包括软硬件的网格环境。其元数据管理服务注重以下几点：

1. 目录包含了原始文件与副本文件的文件名和位置，就像一个索引；
2. 监控信息包括错误状态、当前与历史流量、查询与存取类型；
3. 网格配置信息描述了网络、交换设备、集群、节点与软件情况；
4. 能够灵活动态的配置策略。

元数据服务通过标识符来管理大量的对象。至于松耦合的网格环境，必须在局域网和广域网上维持分区自治和高性能。因此，服务建立在全分布的层次模型与通用统一的协议之上，如 LDAP 协议。

美国 GriPhyN[15]系统提出应用虚拟数据的概念和语言，描述如何通过计算获得并使用派生信息和数据，这是为系统访问远程数据还是通过计算获得，或者获取他人计算处理过程符合自己需求的数据等情况提供决策依据，为数据的自动生成和再生成提供较完整的系统方法。

SDSC 的 SRB[16-19]提供了一套在分布环境下统一访问异构存储系统上数据的中间件系统，包括文件系统、数据库、文档系统等，为上层应用/用户提供透明的数据服务，SRB 采用集中式的元数据目录 MCAT（Metadata Catalog）服务广域的数据访问和管理，最初并不支持网格环境下使用，为了支持数据网格的特点，已经进行了改进，正在进行分布式设计和实现，对多域管理环境进行支持，主要以对文件的访问为主。Punch Virtual File System（PVFS）[20]采用代理机制接受 NFS Client 的请求，经过处理分析，访问 NFS 系统的服务端数据，实现了多个 NFS 系统的数据统一访问。

SRB 3.0 中 MCAT 的实现由单一 MCAT 向联邦（多个）MCAT 系统发展。单一 MCAT 结构包括联合中间件系统，单一 MCAT 服务器（提供元数据目录，使用传统的数据库管理系统），联合资源服务器（提供单一登陆，统一的存取所有资源，健壮的服务器到服务器操作）。单一 MCAT 的特点是提供单一全局用户空间、逻辑名空间和资源虚拟化等。多个 MCAT 提出了 MCAT ZONE，定义了由单一 MCAT 控制的 SRB 资源的联合，每个 ZONE 里都有自己的系统管理者控制本地用户与资源；实现对等的 ZONE 体系结构，每个 ZONE 完全独立于其他 ZONE；实现跨 ZONE 的数据与资源共享。

Avaki[21]数据网格系统采用了面向对象的方式实现对多个域环境下的 NFS 文件系统的数据进行访问，提供了统一的安全认证，支持数据复制管理。

1.2.2 相关技术简介

1. 存储虚拟化

存储虚拟化是指将用户看到的存储资源同具体的物理存储设备分隔开来，为存储用户提供统一的虚拟存储池。它通过构建虚拟存储池形成分布式系统的单一存储映像，将地理上分布的各种高性能存储系统集成为一体，形成庞大的分布存储空间，展示给用户一个逻辑视图，同时将应用程序和用户所需要的数据存储操作以及具体的存储控制分离。

虚拟化存储管理方式能够有效解决存储数据的爆炸性增长和存储管理能力的相对不足之间的矛盾。它主要完成三个任务：首先是在多个物理存储设备上创建

一个抽象层；其次要屏蔽存储系统尤其是在异构环境中的复杂性，简化管理；最后对存储资源进行优化。

从技术上说，存储网络，包括 NAS（Network Attached Storage）[22]和 SAN（Storage Area Network）[23-25]，都是一种很大的革新，但是它们有很多不尽如人意的地方，虽然目前已经实现了一定的存储整合和自动化的存储管理操作，但并没有实现真正的透明存储，存储管理用户仍然需要分别掌握不同存储设备的物理特性，才能对存储池进行有效的管理。

2. 元数据

在一个广域存储系统中，这些海量的存储资源必须被有效的管理,从而引入元数据的概念。元数据[26-32]是描述数据的数据，它为系统提供对象物理位置与其逻辑名字之间的映射，一个逻辑文件可以对应多个物理文件副本。此外，元数据还包括文件目录信息、文件信息、存储设备信息及相关的系统信息等等。

3. 元数据副本

元数据在存储虚拟化系统中占据重要地位，一旦丢失，影响全局。所以必须为元数据建立副本。副本可以起到容错与负载平衡的作用。对于某一个对象，如果在多台机器上都有副本可用，就可以共同承担客户端的请求。如果共享数据没有副本，那么就不存在存储一致性问题，但是这样就会产生系统瓶颈。为了提高系统性能，系统通常复制共享数据。复制导致了一致性协议的复杂性。 存储虚拟化系统所要解决的一个关键问题就是元数据多个副本之间的一致性[33,34]。

4. 目录服务

目录服务[35,36] 是基于目录的数据查询服务,它按照树状信息组织模式，是实现信息管理和服务接口的一种方法。目录服务系统一般由两部分组成：第一部分是数据库，且拥有一个描述数据的规划；第二部分则是和访问和处理数据库相关的访问协议。

与关系型数据库不同，目录服务一般不支持批量更新所需要的事务处理功能，而只执行简单的更新操作，适合于进行大量数据的检索；同时，它具有广泛复制信息的能力，不仅能缩短响应时间，而且能提高可用性和可靠性。目前，目录服务技术的国际标准有两个，即较早的 X.500 标准和近年迅速发展的 LDAP（Lightweight Directory Access Protocol，轻量级目录访问协议）标准。本系统的元数据管理采用后者。

5. LDAP 技术

LDAP[37-41]由 IETF 和密歇根大学联合开发，是目前流行的标准目录服务协议，它源自于 OSIX. 500 目录服务协议中的 DAP（Directory Access Protocol，目录访问协议）协议，但比 DAP 简单，负载小，并且是基于 TCP/ IP 的，满足了 Internet 上目录服务的需要。

LDAP 协议是基于客户机/服务器模式的、运行于 TCP/IP 之上的应用层协议，其模块结构如图 1.1 所示。一个或多个 LDAP 服务器组成了 LDAP 目录树。LDAP 服务器由目录服务模块、复制服务模块和管理模块三个模块组成。

(1) 目录服务模块主要由两部分组成，前端部分负责通常的客户机和服务器之间的网络通信，完成协议解析和分析，后端部分负责目录数据库管理；

(2) 复制服务模块负责 LDAP 服务器之间的目录数据复制；

(3) 管理模块负责目录信息管理，以确保用户在期望的反应时间、完整性、安全及一致性层次上取得准确的目录信息。

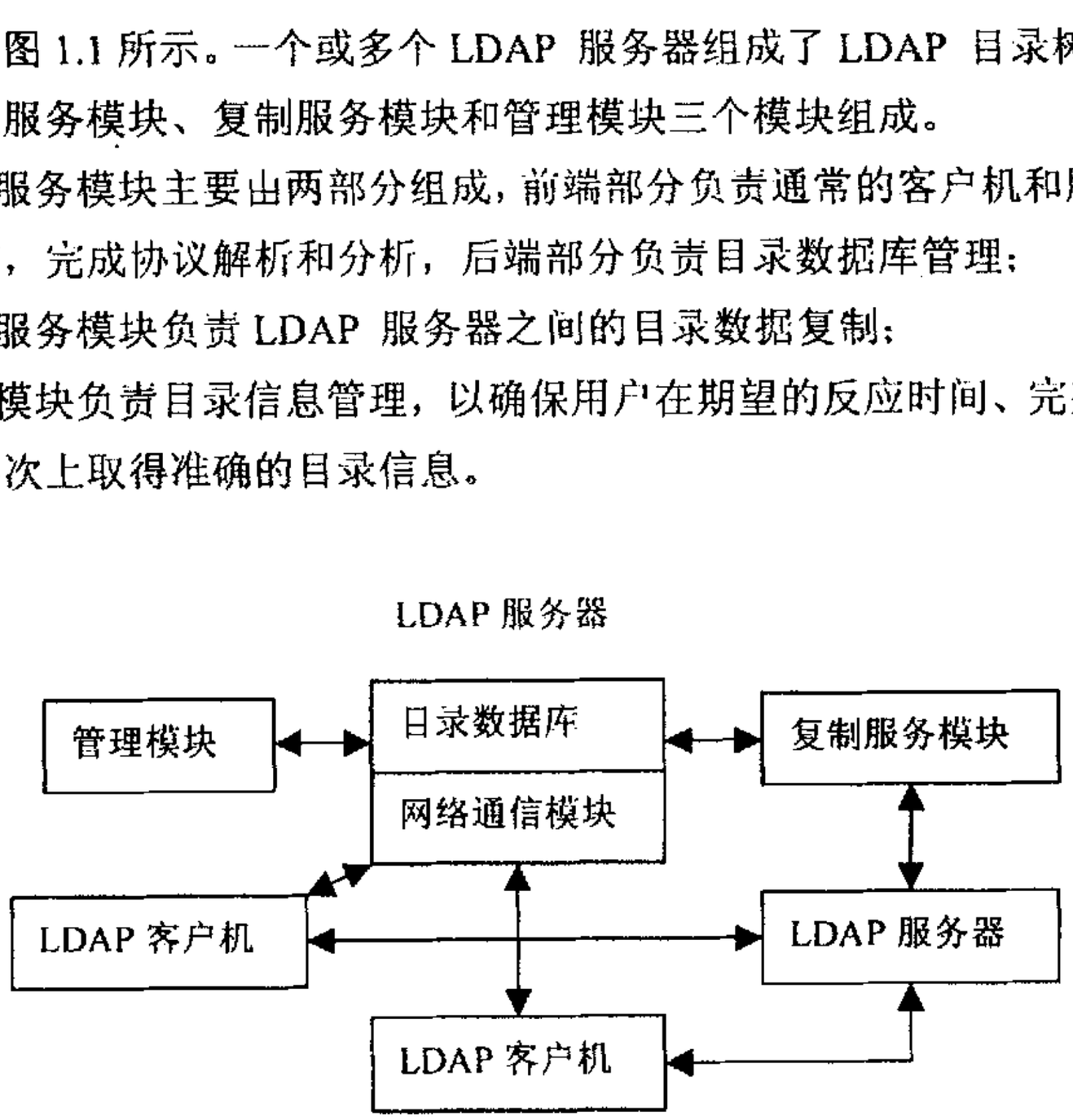

图1.1 LDAP目录服务客户机/服务器结构

LDAP 提供一个层次性的信息目录视图，将系统中所有的实体抽象成相应的对象（object class）。对象被定制在一个目录信息树中，每一个对象在目录树中都对应一个目录标识名（DN，Distinguished Name），对象的一个或多个属性值则表示为相关标识名（RDN，Relative Distinguish Name）。用户可以根据对象的属性信息查询相关的资源，而不一定要使用资源名称。对象和属性都是由相关的 SCHEMA 来定义的。SCHEMA 定义了属性类型，对象类别和相关信息的集合，以及是否允许添加、删除操作。在系统开发过程中，很重要的就是要设计一个合理的 SCHEMA，以便对广域网环境的资源进行合理的描述。

1.3 元数据在存储虚拟化系统中的作用

在广域网中，资源及其提供者是分布的。数据资源各种各样，其表示和存储形式也各不同。一些数据可能以文件形式存储；一些数据存储在数据库或数据仓库中；另一些存储在如同 HPSS（High Performance Storage System）[42]的 Archive 档案系统中；还有一些数据是由多个分布存储系统中的数据组成的。如何方便有效地访问各种异构数据组成的数据集合是存储虚拟化系统的一个主要功能和关键技术。因此，在存储虚拟化系统中，需要建立灵活的、可扩展的信息服务体系结构。这种体系结构应当保证资源信息提供者的广泛分布性和信息服务的分布特性，避免由于单个信息服务实体的失败导致其他资源信息服务不能正常提供服务，具体要求如下：

1. 为了实现命名的透明性，存储虚拟化系统需要有效管理数量繁多的名字和属性以及它们之间的关系，需要一种统一的全局命名方式；

2. 为了实现定位的透明性，存储虚拟化系统需要有效管理数据集的定位信息和数据资源的有关信息；

3. 为了实现访问的透明性，存储虚拟化系统需要有效管理系统资源的安全、授权、访问控制等信息。

实际上，这些信息就是用于描述资源、方法、数据集和用户的元数据。元数据为系统提供全局资源的信息服务、数据的定位和属性查找、数据的注册、系统资源信息的查询和维护、数据访问控制信息和用户管理，并为用户和系统提供一个元信息的访问接口和访问协议。

元数据的管理包括元数据的命名和访问，并为用户提供统一的访问接口。存储虚拟化系统中的所有元数据构成元数据目录。在元数据目录中应该尽量采用统一的结构来描述元数据。无论使用何种结构，元数据目录应当满足两点：其一，它应该是一种层次和分布式目录结构系统，如 LDAP；其二，它应当不破坏现有系统的元数据描述方法，并能与它们很好地交互、融合。

随着应用的不断发展,存储虚拟化系统也在不断发展，元数据在不断增多，其结构也日趋复杂。为了保证在存储虚拟化系统规模不断扩大的情况下，仍然提供高效的元数据服务，元数据目录应该采用具有良好可扩展性的层次式分布式结构，这需要一套合理的管理机制。

1.4 主要研究工作

在许多系统中比如 SRB[16-19]，采用层次目录结构管理元数据。随着元数据变得越来越庞大，系统将同时启用多个元数据服务器。这时元数据的根目录服务器的压力也会随之增大，因为系统必须通过根服务器自上而下的查询。这样，不仅系统的扩展性会变得十分困难，而且系统的高可用性也会受到影响，因为根目录服务器很容易成为系统的单一失效点。

为解决上述问题，需要为广域网虚拟化存储系统 GDSS（Global Distributed Storage System）提出新的元数据管理方法。本文提出一种元数据层次管理模型 MDC（MetaData Controller）。

MDC 管理模型中，GDSS 将被划分为多个域，每个域中有一个 GNS（Global Naming Server）和多个目录服务器 DS（Directory Server）。每个 DS 都是一个 LDAP 服务器，用于存放元数据；GNS 用于管理域内的这些 DS 服务器。在 GNS 上包含以下三个模块：

1. 匹配表模块：用于快速定位元数据。匹配表中记录了每个目录服务器的名字及存放在其上的元数据的根目录，通过一次匹配，找到元数据所在的目录服务器 DS，避免从根目录服务器的层层查询，减轻根服务器的压力。

2. 缓存模块：缓存中保存热点元数据和正在被执行写操作的元数据，提高元数据读写效率。写操作只在缓存上进行，然后把这些更新信息统一传送给域内的 DS，这样可以避免元数据副本被同时访问可能引起的冲突，维持元数据副本一致性。如果缓存没有命中，则查找匹配表，找到用户请求访问的文件/目录所在的 DS，选择一个进行访问。

3. 副本一致：用于确保存放在不同 DS 上的多个元数据副本之间的一致性。当某个元数据需要更新时，在缓存中记录元数据所要更新的内容和所有当前保存其副本的 DS。更新信息只传播给保存副本的 DS 而非所有的 DS，减少网络通讯负载。

1.5 文章的框架结构

第一章简要概述存储虚拟化的研究背景和国内外类似系统的发展状况；然后解释了几个存储虚拟化系统的相关概念；最后讲述元数据在存储虚拟化系统中的

作用和本课题将要研究的工作。

第二章首先介绍虚拟化存储系统 GDSS 的系统结构，包括其各个组成模块及模块的主要功能；然后结合系统结构图描述 GDSS 的工作流程；最后介绍 GDSS 的系统特色。

第三章介绍 GDSS 系统中元数据的组织结构，提出适合 GDSS 系统的元数据管理模型 MDC 的架构，简述其框架结构、工作流程以及由此带来的优越性。MDC 是一种分布式的基于匹配表的元数据管理模型，可以达到更好的扩展性和可用性。

第四章详细论述 MDC 包含的技术，有匹配表、缓存技术和元数据副本管理技术。匹配表模块用以快速定位元数据；副本一致模块根据匹配表中的记录把这些更新信息统一传送给域内的 DS；缓存模块用于提高元数据的访问效率，同时辅助副本一致模块保证元数据副本之间的一致性。

第五章介绍测试环境，然后按照既定步骤从系统功能和系统性能两个方面对 MDC 模型进行了测试。

第六章对全文进行总结并展望了未来工作；最后是致谢和参考文献。

2 存储虚拟化系统的框架与工作流程

参考前一章中介绍的国内外相关项目的研究成果，针对存储虚拟化需要解决的问题，本章提出一种新的虚拟化存储系统 GDSS，并简要描述其系统结构、工作流程及系统特色。

2.1 系统概述

GDSS 实现对广域范围内孤立存储资源的统一管理，它屏蔽底层数据资源的分散性、异构性、多认证性质，为用户提供统一的逻辑视图，实现数据的透明访问，减少存储系统的管理开销，实现存储系统的数据共享，提供透明的高可靠性和可扩展性，优化使用存储系统。其研究内容包括：多样的数据访问接口；全局范围内的数据共享与访问控制；全局统一的文件名字空间；动态的副本管理机制；灵活的元数据管理机制；高性能数据分片传输技术；全局可视化统一管理。

GDSS 的设计充分借鉴目前国内外的先进方法与技术，提出一整套完善的存储虚拟化解决方案，整体设计到达了国际先进水平。经过 2 年的研究开发，目前已经形成了一套完整系统，经过测试和使用表明系统的设计是成功的。

2.2 系统结构

GDSS 主要包括存储服务点 SSP（Storage Service Point）、全局命名服务器 GNS（Global Name Server）、资源管理器 RM（Resource Manager）、认证中心 CA（Certificate Authority）、客户端、存储代理 SA（Storage Agent）以及可视化管理，其系统结构图如图 2.1 所示。

SSP 是整个系统的入口，对系统所有模块的访问都通过 SSP，它主要提供 FTP 接口、CA 接口、RM 接口和 GNS 接口；在整个系统中 SSP 不是唯一的，系统可以根据需要动态增加，SSP 实现了传统方案中命名服务器的部分功能，减轻了 GNS 的负载，提高了系统的可扩展性。

GNS 采用一种基于匹配表的分级层次模型对元数据进行管理，主要包括匹配表模块、缓存模块和副本管理模块，实现元数据的存放、快速定位、搜索和元数

据副本管理。RM 主要负责资源的申请和调度，同时提供透明的副本策略，主要包括资源调度器和副本管理模块。副本技术减少了访问延迟和带宽消耗。通过创建同一数据的多个副本，副本机制有助于改善负载平衡和可靠性。

客户端目前支持通用 FTP 客户端、文件访问接口和特制客户端。用户通过系统提供的特制客户端，不但能够进行用户组操作，具有搜索和共享等功能，还可以获得更高性能的服务。CA 包含证书管理系统，主要负责系统的安全性和数据的访问控制，同时它记录了用户的注册信息。

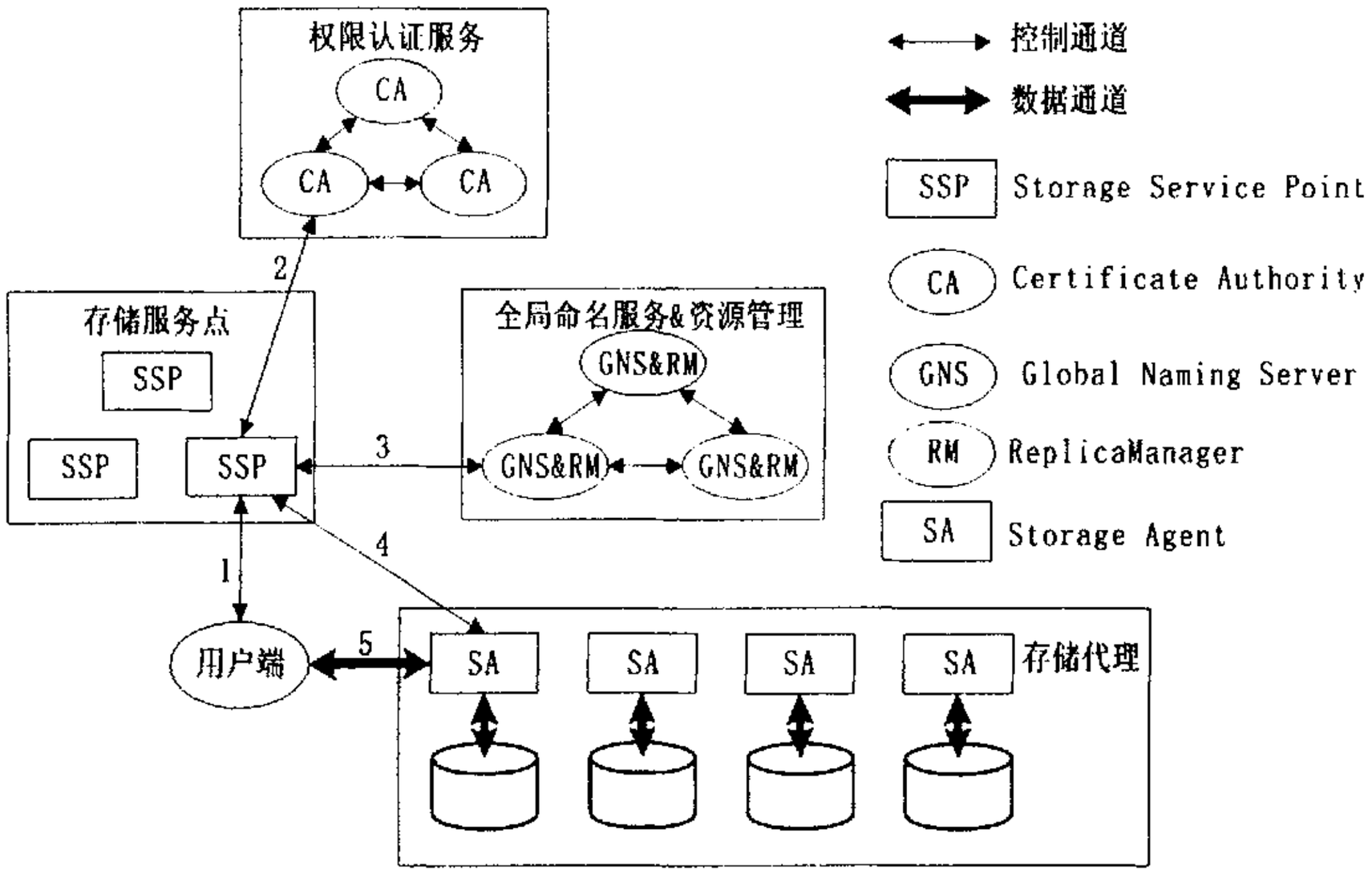

图 2.1 GDSS 系统结构图

SA 屏蔽了存储资源的多样性，为系统提供统一存储访问接口，同时提供了文件操作方式和扩展的 FTP 操作方式，另外它对文件复制管理操作提供支持，为高效传输提供服务。同时 SA 这一级实现了局域存储资源的虚拟化，包括了统一 SAN 和 NAS、分布式的磁盘虚拟化、磁带库虚拟化和 SAN 内部共享管理等。

这种高性能存储系统从 3 个方面保证了虚拟化存储方式。首先，SSP 的用户接口为存储用户提供了单一存储映像，对存储用户而言，SSP 将广域存储资源虚拟为一个逻辑存储池供其方便使用；其次，SA 对广域存储服务屏蔽了本地存储系统的具体特性，保证了虚拟化存储能够适应异构存储环境；最后，GNS 保证了逻辑存储池到具体存储系统的对应关系，提供了虚拟化存储系统的内部实现机制。

2.3 工作流程

基于图 2.1 所示的系统结构图，对 GDSS 的工作流程解释如下：

1. 用户向SSP发出存储请求；

2. SSP向CA提供用户的身份信息和服务请求信息，CA根据访问权限控制机制向SSP反馈是否接受用户请求以及用户的访问权限；

3. 认证通过后，SSP将用户服务请求交给GNS解析，通过GNS返回的信息得到相应的存储任务；

4. SSP将存储任务发给能够满足用户请求的最合适的SA，并将相关服务信息反馈给用户；

5. 用户与SA建立连接，在SA的控制下进行直接数据传输。

2.4 系统特色

GDSS 主要特色包括以下几点：

1. 广域的虚拟化分布式存储架构：GDSS系统将分布在广域范围内的数据资源统一管理，屏蔽了底层数据资源的异构性、分散性，采用多个分布的存储服务点为用户提供一个透明的全局数据视图；

2. 多样的数据访问接口：GDSS系统为终端用户提供形式多样的访问方式，包括标准FTP协议的服务、特定的GDSP协议的服务、兼容MPI I/O文件读写接口、客户端的类Java文件I/O和GDSS图形客户端；

3. 全局统一的文件名字空间：GDSS系统的名字空间结合了单一名字空间和多名字空间的特点，建立了一个全局统一的文件命名空间，对所有的用户提供相同的文件命名、定位和访问机制，同时针对不同用户的兴趣，为他们提供不同的数据视图；

4. 面向用户特性的数据共享：GDSS系统根据用户使用数据资源的特性将用户分为个人用户和组用户，并且相应的提供两种数据共享方式，分别是简单数据共享和带目录一级ACL（Access Control List）访问控制的数据共享；

5. 灵活的元数据管理方式：采用层次管理的方式，把元数据分布在域内多个目录服务器上，并按照类似UNIX文件系统的组织方式形成一棵“本域元数据逻辑树”；采用基于匹配表的方式将这些分散在多个DS上的信息组织成一个统一的逻

辑视图，将用户请求快速准确的定位到对应的DS上；

6. 动态的副本管理机制：在GDSS系统中，通过创建同一数据的多个副本可以有效减少访问延迟和带宽消耗，同时数据副本技术有助于改善系统负载均衡和可靠性；

7. 高性能数据分片传输技术：存储虚拟化涉及大量数据的移动、传输和复制，在GDSS中初步建立起一个广域网高效数据传输机制，具体包括分片传输、部分数据传输、分布式合作传输、断点续传和Socket复用；

8. 全局可视化统一管理：GDSS通过分布的、全局统一的Web服务器来提供全局一致的服务管理，管理员可以通过支持Java plug-in的浏览器方便从远程对整个系统进行监控管理，监控管理主要包括全局信息和证书的管理、SSP管理、GNS管理和RM管理，所有的管理模块通过HTTP服务的方式进行链接，统一在全局信息服务器上；

9. 灵活的安全认证机制：GDSS系统采用PKI/CA的身份认证和Kerberos的访问控制相结合的底层安全架构，PKI/CA的CA中心采用网状和层次相结合的布局方式，极大提高了安全架构的灵活性和动态可扩展性。

2.5 小结

本章介绍虚拟化存储系统 GDSS 的系统结构，工作流程及系统的特色。

GDSS 主要有六个模块组成，其中 SSP 模块是整个系统的入口，它为其它模块直接的通讯提供结构，使所有的模块形成一个统一的整体，对用户服务；GNS 负责管理系统的元数据；RM 主要负责资源的申请和调度；SA 为系统提供统一存储访问接口，为高效传输提供服务；CA 主要负责系统安全性和数据访问控制；同时，GDSS 还拥有自己的客户端，更好的为用户提供服务。

GDSS 采用十分清晰的工作流程。用户通过通用客户端例如 cuteFTP 或者系统客户端 GDSSClient 向单一入口点 SSP 发出存储请求，由 SSP 负责向其他模块申请资源，读写元数据，最后返回相关信息给用户。最后，用户与 SA 建立连接，在 SA 的控制下进行直接数据传输。

通过以上几大模块的有机联合，GDSS 可以对用户提供统一访问点，屏蔽系统底层操作，提供一套完整的元数据管理机制、副本管理机制、权限访问控制机制和高效的传输机制，并拥有自己的客户端，为用户提供统一的逻辑视图，实现数据的透明访问。

3 元数据管理模型

为了快速准确的定位元数据，方便其扩展和管理，提出一种元数据管理模型MDC。本章将讨论GDSS中元数据的组织方式及其在LDAP数据库中的定义，并在此基础上描述MDC模型的框架和工作流程。

3.1 元数据的结构和组织

元数据管理涉及的关键研究内容包括：定义元数据信息，对存储资源、文件、用户等给出相应的资源描述SCHEMA；元数据服务器的扩展，基于用户的资源分配方案，系统提供基于用户的资源存储视图，实现逻辑文件名与物理文件位置的分离及映射功能；元数据信息与其他相关应用的兼容性问题；元数据的准确定位并提供基于元数据属性的搜索；元数据服务器性能的优化；元数据服务器容错功能。

3.1.1 元数据信息系统的实现

虚拟化存储系统GDSS借鉴网格[43]中的虚拟组织的思想，通过系统中域的划分，实现域内的自治和域间的协作，能较好的实现分布资源的管理使用。

图3.1给出了系统单个域中的各个功能组件之间的连接关系，终端用户通过SSP获取GNS上储存的用户存储资源分配信息，GNS的所有元数据信息保存在LDAP目录服务器中。

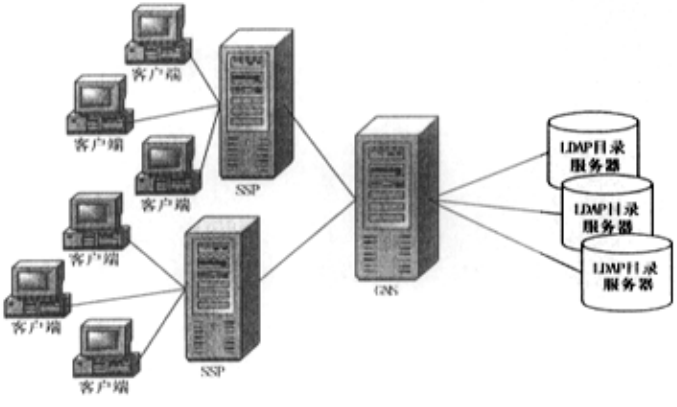

图3.1 单个域中元数据服务器系统架构

元数据服务器的重要职能就是实现用户逻辑文件名到物理文件名的转换，保存用户的文件存储信息。元数据服务器的后台是 LDAP 服务器，它存储用户文件的逻辑目录信息，数据的条目格式信息由 LDAP 的 SCHEMA 决定。在 LDAP 服务器的支持下，将查询信息进行转化，与 SSP 交互并编制合适的 API, 实现用户存储资源相关的基本操作，以及为文件分片、文件资源注册、快速拷贝等操作提供相关信息。

3.1.2 元数据目录组织结构

广域网虚拟化存储系统 GDSS 提供基于多用户的名字空间服务，每个用户都有自己的逻辑存储视图，同时还提供用户之间的资源共享，所有的这些存储信息和控制信息都以条目的形式记录在目录服务器中。遵循 LDAP 协议，为了实现相应的目录管理功能，必须定义一定的模式来满足虚拟化存储系统管理的需求，有效的组织目录服务器的内容。模式是对条目信息组织格式的定义，定义属性类型、对象类别和相关信息的集合。为了适应于存储虚拟化目录管理的需要，系统定义了相应的文件、目录，以及用户、组等模式，在以后的功能扩展上，将定义更多的模式，来适应目录组织管理的需求。

在 GDSS 系统中，单个域内按照目录信息树的层次结构安排条目信息。例如，图 3.2 所示的目录信息树由两个大的分支组成，其中一部分以“DC=FAT，DC=ICCC”（目录信息树的条目 DN 表示方式，由每个节点的相对区别名自下而上组成）指定一个整体的系统文件目录信息树，另外一个分支“DC=SUBJECT，DC=ICCC”保存用户信息和用户组信息。在系统的目录树结构中，采用类似于 Unix 文件系统的目录树结构组织系统的目录文件结构，与用户信息和用户组的信息分

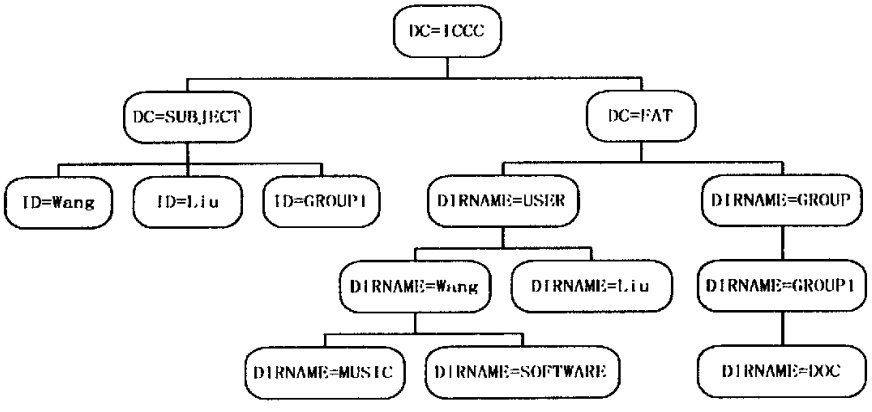

图 3.2　目录服务器目录信息树

开。在用户访问过程中，通过用户信息条目中的属性信息指定的用户根目录以及用户及组之间的共享目录信息，在系统的统一目录树中映射生成用户的逻辑目录信息视图，从而实现多用户的目录空间，并附加文件访问的权限控制。

由于广域网范围内的用户数量可能不断提升，目录服务器的负载以及规模都会越来越大，必须采用一定的措施减轻目录服务器 DS 的负载。在 GDSS 系统中，采用分布目录服务器的方式来减轻单个目录服务器的负载，尽量保证服务质量。利用 LDAP 协议规定的参考指针 (Referral)，将系统的文件目录信息树进行分割，分布在多个目录服务器上，这样就可以缓解单个目录服务的压力。此外，在目录服务器与 SSP 的交互过程中，尽量简化操作，采用线程的工作方式。同时，尽可能提高 LDAP 目录数据库的查询速度，对经常需要使用的属性建立相应的索引（如文件名、目录名、用户等），以达到提高性能的能力。

3.2 元数据的定义

在 GDSS 系统中，元数据主要包含用户元数据和文件/目录元数据。用户元数据主要是关于用户的信息，比如用户的一些基本属性，用户和组之间的关系等等；文件/目录元数据主要是关于数据的信息，比如文件与具体存储系统之间的映射关系，文件的大小，目录的访问权限等等。

3.2.1 组元数据的定义

组元数据在 LDAP 数据库 SCHEMA 中的定义代码如下：

```
#Group ObjectClass
objectclass ( 8.2.8.2 NAME 'GroupObject'
    DESC 'Group objectclass'
    MUST ( ID $ passwd $ GroupName $ CreateTime )
    MAY ( Discription $ GroupType $ UserCount $ StorageType $ TotalSpace $
UsedSpace $ ChildGroup $ ParentGroup $ GroupMember $UserMember $
ApplyUser $ApplyGroup $ GroupAdded $ RootDir $ ShareOutDir $ ShareInDir
$ShareInInviting ) )
```

其中主要的关键字解释如下：

1. 第一个 objectclass：表示所定义的是一个对象；
2. 8.2.8.2：对象标识符（Object Identifiers）；

3. NAME 'GroupObject'：所定义对象的名字为“GroupObject”；
4. DESC 'Group objectclass'：对于对象的描述；
5. MUST (ID $ passwd $ GroupName $ CreateTime)：对象必须包含的属性；
6. MAY (Discription $ GroupType $ UserCount $ StorageType $ TotalSpace $ UsedSpace $ ChildGroup $ ParentGroup $ GroupMember $UserMember $ ApplyUser $ApplyGroup $ RootDir $ ShareOutDir $ ShareInDir $ShareInInviting))：组对象的可选属性，依次为：对组对象的描述，组类型，组内用户数目，存储类型，组所有空间大小，已用空间大小，子组，父组，申请加入的组，申请加入的组内用户，正在申请加入的用户，正在申请加入的组，组根目录，组输出共享目录，组输入共享，组得到邀请的输入共享。

表 3.1 是一个组元数据用例。其中，objectClass 属性表示出此条元数据是关于组的，ID、passwd、GroupName、CreateTime 四项属性是必须的，其他属性是可选的。

表 3.1 组元数据用例

CreateTime	1058317524148
Discription	datagrid group at iccc
GroupName	The Data Grid Group
ID	datagrid@iccc.hust.edu.cn
objectClass	GroupObject
passwd	griddata
RootDir	/group/d/datagrid
ShareInDir	gdss:wzp@iccc.hust.edu.cn:gdss:STATE_OK
StorageType	agent
TotalSpace	1234234
UsedSpace	132
UserCount	12
UserMember	wzp@iccc.hust.edu.cn:Team

3.2.2 用户元数据的定义

用户元数据在 LDAP 数据库 SCHEMA 中的定义代码如下：

User Objectclass

objectclass (8.2.8.1 NAME 'UserObject'

DESC 'user objectclass'

MUST (ID $ passwd $ UserName $ CreateTime)

MAY (Discription $ StorageType $ TotalSpace $ UsedSpace $ RootDir $ ShareOutDir $ ShareInInviting $ ShareInDir $ GroupAdded))

其中相应的关键字解释和组元数据相同，MAY 可选属性中的关键字依次为：对用户对象的描述，存储类型，用户所有空间大小，已用空间大小，用户根目录，用户输出共享目录，用户得到邀请的输入共享，用户输入共享。

表 3.2 是一个用户元数据用例。其中，objectClass 属性表示出此条元数据是关于用户的，ID、passwd、UserName、CreateTime 四项属性是必须的，其他属性 RootDir、StorageType、TotalSpace 和 UsedSpace 是可选的。

表 3.2 用户元数据用例

CreateTime	1066114110015
ID	ajia@iccc.hust.edu.cn
objectClass	UserObject
passwd	123456
RootDir	/user/ajia@iccc.hust.edu.cn
StorageType	1100
TotalSpace	1000
UsedSpace	1000
UserName	ajia

3.2.3 文件元数据的定义

文件元数据在 LDAP 数据库 SCHEMA 中的定义代码如下：

#File objectclass

objectclass (8.2.8.3 NAME 'FileObject'

DESC 'File objectclass'

MUST (FileName $ CreateTime $ ModifyTime $ FileLength)

MAY (FATID $ FileCreator $ Discription $ FileType $ AdditionAttribute))

表 3.3 是一个文件元数据用例。其中，objectClass 属性表示出此条元数据是关

于文件的，FileName、CreateTime、ModifyTime、FileLength 四项属性是必须的，FATID 和 FileCreator 是可选的。

表 3.3 文件元数据用例

CreateTime	1064629379671
FATID	825252380
FileCreator	ajia
FileLength	0
FileName	ajia
ModifyTime	1064629379671
objectClass	FileObject

3.2.4 目录元数据的定义

目录元数据在 LDAP 数据库 SCHEMA 中的定义代码如下：

#Dir ObjectClass

objectclass (8.2.8.4 NAME 'DirObject'

DESC 'Dir objectclass'

MUST (DirName)

MAY (FATID $ CreateTime $ ModifyTime $ FileCreator $ Discription $ FileLength $ AdditionAttribute $ ACL $ACLBoolean))

表 3.4 是一个目录元数据用例。其中，objectClass 属性表示出此条元数据是关于文件的，DirName 属性是必须的，其他属性都是可选的。

表 3.4 目录元数据用例

AdditionAttribute	Tianjin:KFC
CreateTime	1060157190873
DirName	d
FATID	825241664
FileCreator	system
FileLength	0
ModifyTime	1060157190873
objectClass	DirObject

3.3 元数据管理模型简述

在许多系统中比如 SRB[39]，都采用层次目录结构管理元数据。当元数据变得庞大时，系统会同时启用多个元数据服务器，这样会导致以下问题的产生：首先，逻辑树必须分布在这些协同操作的元数据服务器上，因为要返回的元数据可能分布在不同的元数据服务器上，这样就对元数据的根目录服务器造成了压力。其次，如果根目录服务器出现故障，整个目录服务系统也就不能再正常运转了，很难做到系统的高可用。最后，这种结构使得元数据服务器的扩展也变得很困难。在 GDSS 系统中，为了快速准确的定位元数据，提高广域网内的元数据访问效率，更好的保证元数据副本一致性，并克服上述缺陷，我们建立了元数据层次管理模型 MDC，并基于虚拟化存储系统 GDSS 实现了该模型。

在 MDC 模型中，GDSS 被划分为多个域，在每个域中设有目录服务器 DS（Directory Server）用以存放元数据，设置一个全局命名服务器 GNS（Global Naming Server）管理域中的多个 DS，GDSS 系统中的 GNS 和 DS 部分的示意图如图 3.3 所示。

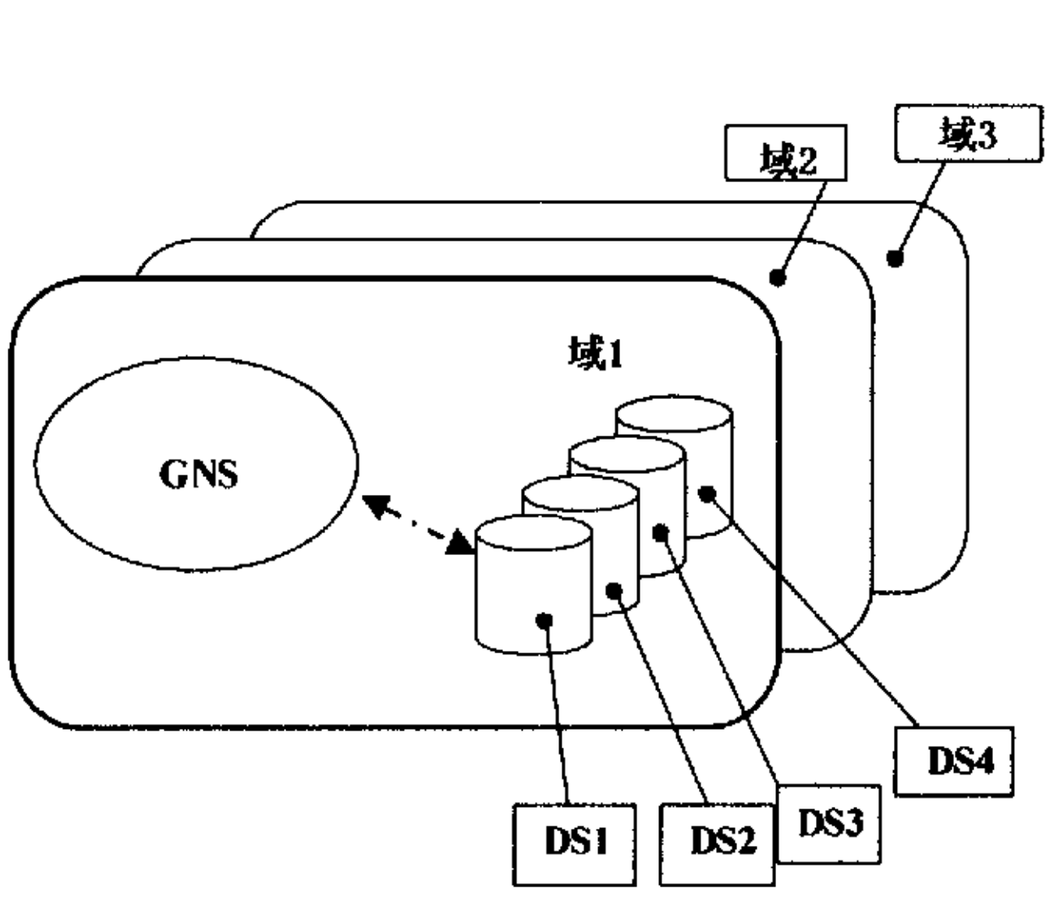

图 3.3　GDSS 系统中的 GNS 和 DS 示意图

图 3.4 是 GNS 的模块组成示意图。在 GNS 中，由匹配表模块来快速定位元数据，缓存模块用于提高元数据的访问效率，同时辅助副本一致模块保证元数据副本之间的一致性。具体解释如下：

1. 缓存（Cache）：保存热点元数据和正在被执行写操作的元数据，提高元数据读写效率。更新操作只在缓存上进行，然后统一传送给域内的 DS，这样可以避

免元数据副本被同时访问可能引起的冲突，维持元数据副本一致性。同时，由于只保留少量元数据，缓存并不占用太大空间。

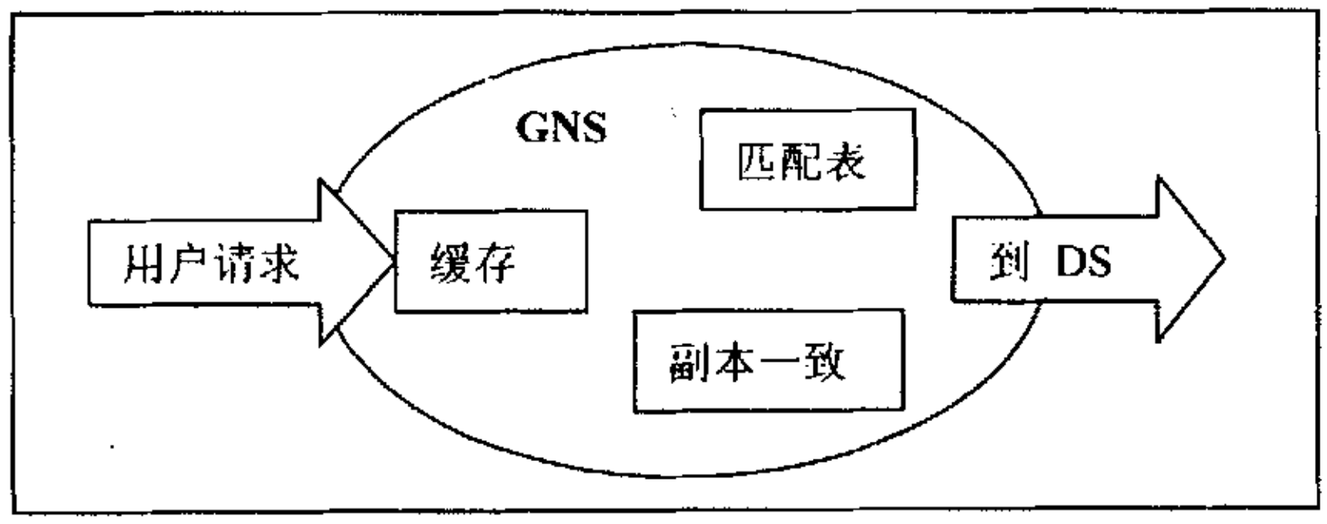

图 3.4 GNS 的模块组成示意图

2. 匹配表（MatchTable）：匹配表保存每个目录服务器的名字及存放在其上的元数据的根目录。当读操作在缓存中未命中时，查找匹配表，找到用户请求访问的文件/目录所在的 DS，选择一个进行访问。

3. 副本一致（Coherency）：当某个记录需要更新时，在缓存中记录元数据所要更新的内容和所有当前保存其副本的 DS。然后根据缓存中的记录同步更新元数据副本。更新信息只传播给保存副本的 DS 而不是所有 DS，以便减少网络通讯负载。

3.4 元数据管理模型的工作流程

GDSS 中元数据管理的处理流程如图 3.5 所示。包括编制相应的模式文件，完成目录服务器的初始化配置工作，启动目录服务器，绑定目录服务器、启动网络监听进程、等待 SSP 的操作请求或者特定的客户端的请求，解析 SSP 相关的文件、目录或用户等操作请求，进行相应的目录操作请求，增删相应的目录条目，返回操作结果给 SSP。

图 3.6 所示为当缓存中存在文件/目录信息时系统的工作流程。如果缓存中存在用户所请求访问的元数据，则直接对缓存中的元数据进行操作。

图 3.7 所示为当缓存中不存在文件/目录信息时，读操作的工作流程。如果用户想要读文件，而缓存中不存在这个文件的元数据，则操作步骤如下：

1. 根据用户的请求查找缓存；
2. 如果缓存中没有，则根据用户的请求查找匹配表；
3. 根据匹配表找到元数据所存放的 DS；

4. 在 DS 中找到元数据后，返回缓存；
5. 返回用户。

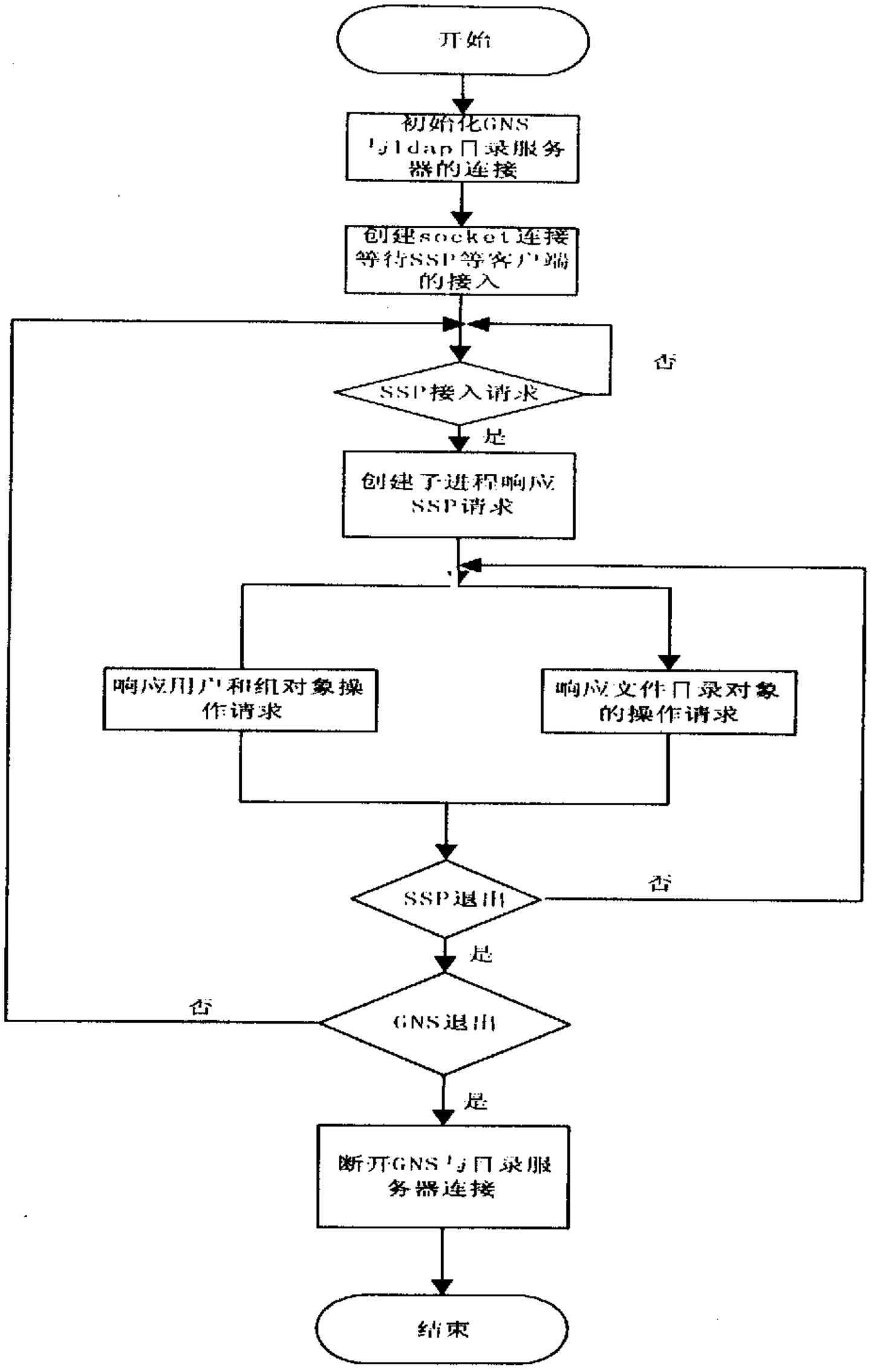

图 3.5 GDSS 中元数据管理的处理流程

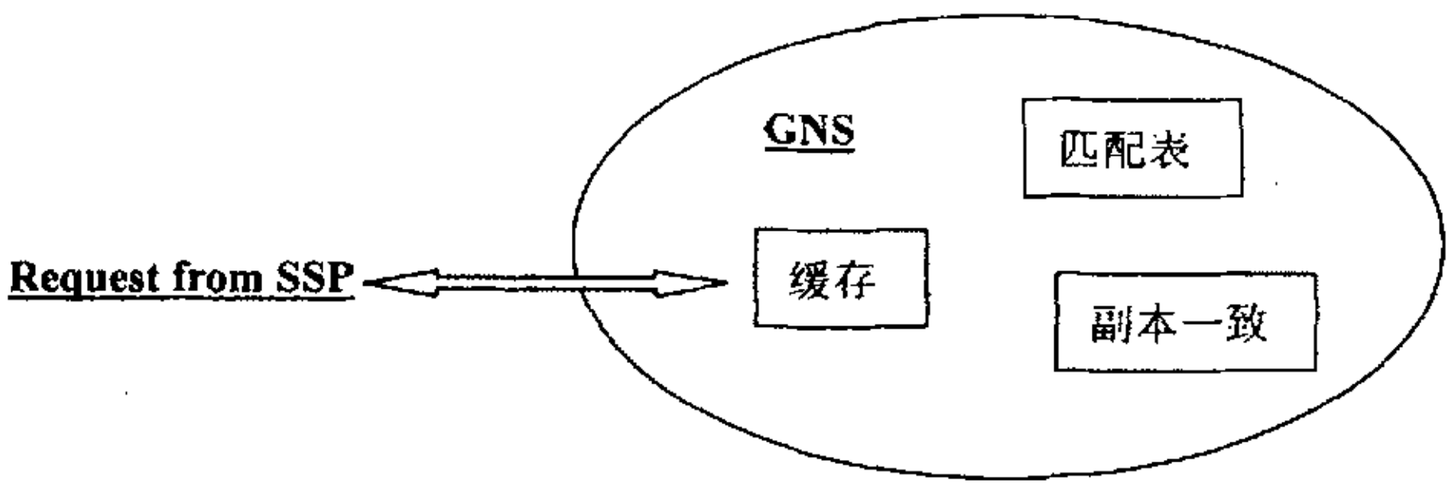

图 3.6　缓存中存在文件/目录信息时的流程

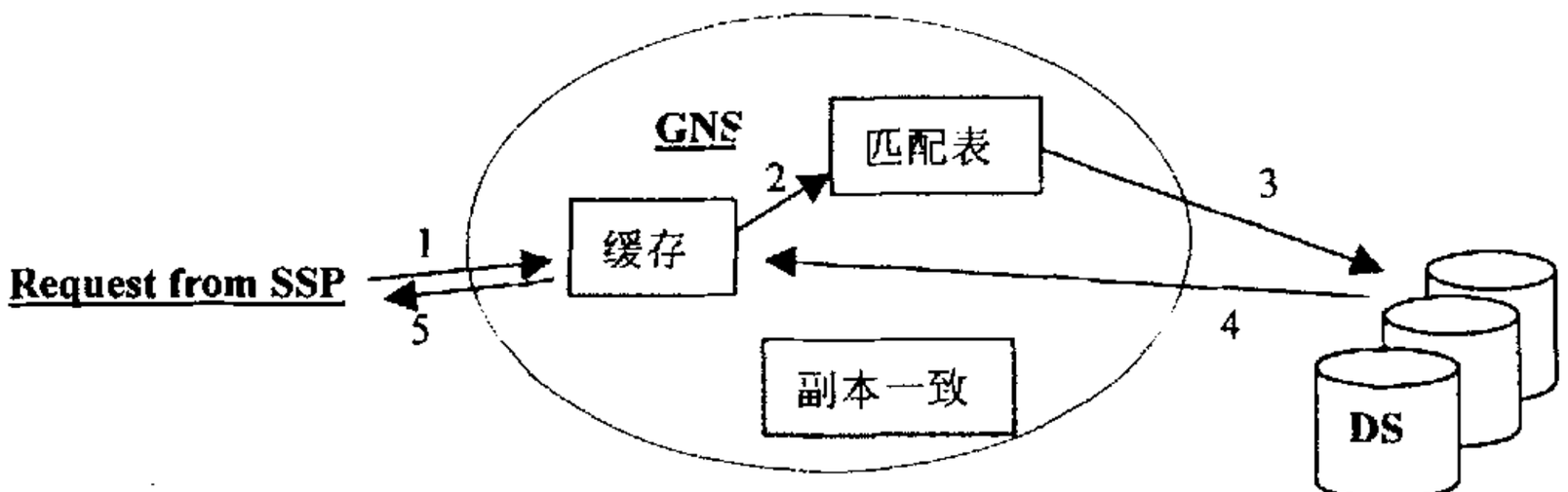

图 3.7　缓存中不存在文件信息时的读流程

写操作分为首次写入和更新操作两种。图 3.8 所示为首次写入时，缓存中不存在文件/目录信息的操作流程。如果用户首次写文件，而缓存中不存在这个文件的

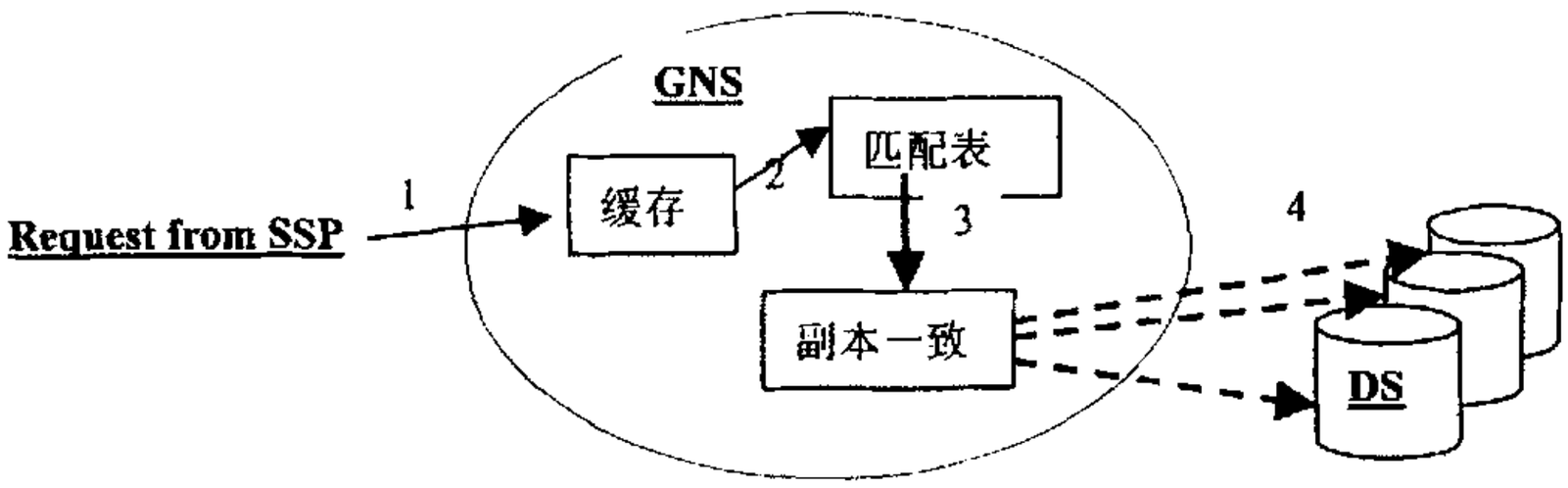

图 3.8　首次写入时，缓存中不存在文件/目录信息的写流程

元数据，则操作步骤如下：

1. 把用户所要更新的内容记录在缓存中；
2. 根据匹配表找到元数据所存放的 DS；
3. 在 DS 中找到相应的元数据更新之；
4. 所有的 DS 更新完毕后，在 cache 中把元数据从“写”状态转成“读”状态。

对于更新操作，一般情况下遵循“读－修改－写”的步骤，所以可以分步采

取以上几种流程。

3.5 小结

本章介绍 GDSS 系统中元数据的组织结构，提出适合 GDSS 系统的元数据管理模型 MDC 的架构，简述其框架结构，工作流程以及由此带来的优越性。

为了提高元数据的访问效率和搜索效率，加强元数据的可用性，在 GDSS 中提出一种元数据管理模型 MDC。在 MDC 模型中，GDSS 系统被划分为多个域，每个域中设置一个 GNS 和多个 DS，每个 DS 都是一个 LDAP 服务器，元数据就分布在这些 LDAP 服务器中。GNS 包括缓存模块、匹配表模块和副本一致模块。MDC 是一种分布式的基于匹配表的层次管理方式，可以达到更好的扩展性和可用性。

4 元数据管理模型的主要技术

上一章对元数据管理模型 MDC 的整体结构做了描述，在此基础上，本章将详细讨论 MDC 的实现细节，包括匹配表技术、缓存技术和元数据副本管理技术。

4.1 匹配表技术

4.1.1 匹配表的设计原理

在 GDSS 中，通过划分逻辑域对整个系统进行层次管理。在每个域内，元数据分布在多个目录服务器上面，并按照类似 Linux 文件系统的组织方式形成一棵“本域元数据逻辑树”，图 4.1 是一个域元数据逻辑树示例。图中，DS-A，DS-B，DS-C，DS-D 表示本域中的目录服务器；root 表示域内元数据逻辑树的根目录；A1，A2……，表示域内元数据逻辑树中的目录。

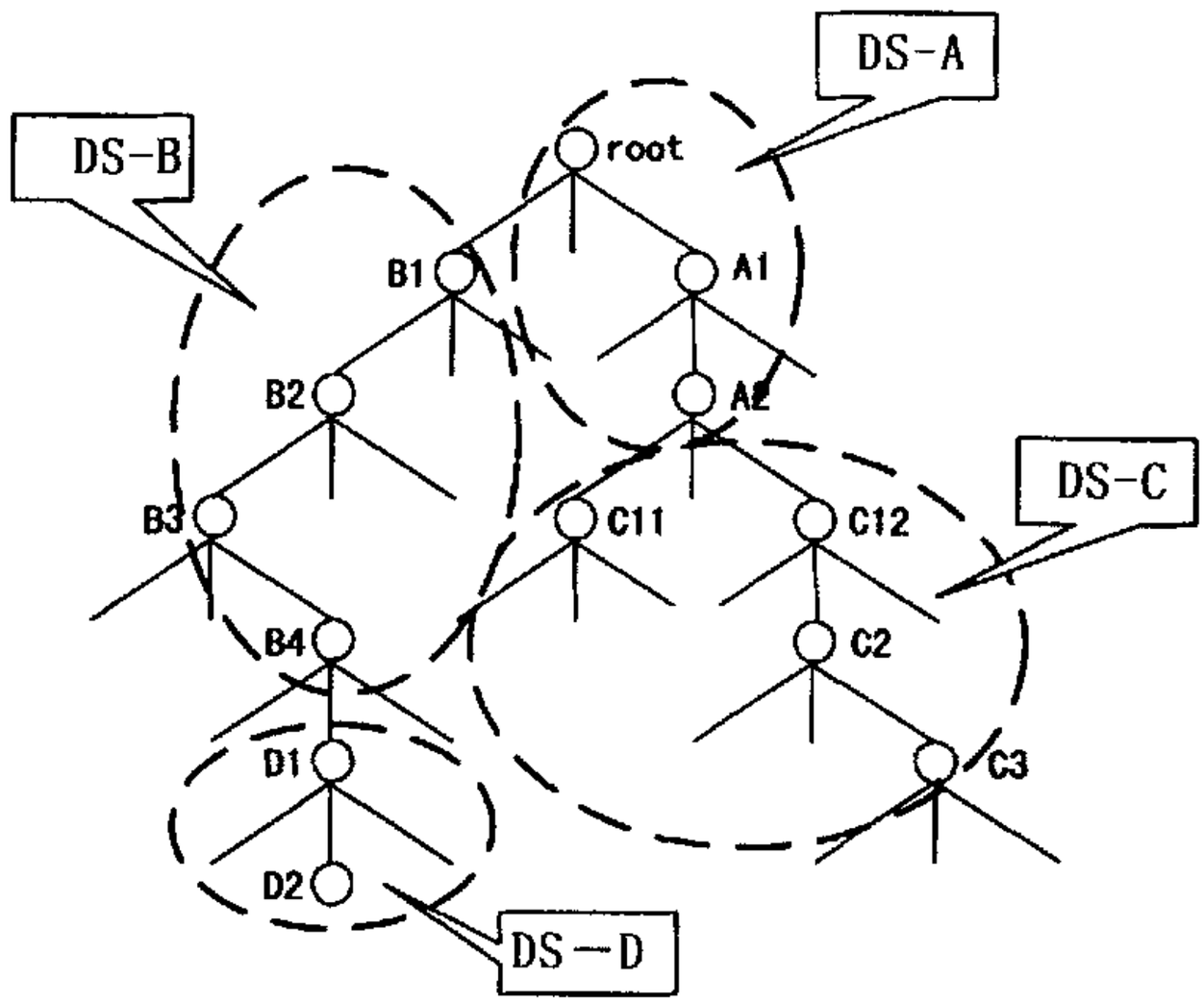

图 4.1 一个域元数据逻辑树示例

每个用户都隶属于一个域，用户的根目录绑定到域内一个子目录上，用户登陆时，系统自动定向到该用户的根目录去。

建立基于目录树结构的元数据目录服务器 DS（Directory Server）来存储元数据及其副本。但是在广域网的范围内，随着系统中资源的增长，文件和目录的信

息会变得越来越庞大，很显然，目录服务器组织得不好，就会成为系统的瓶颈。所以如何快速准确的定向到用户的根目录就是一个十分突出的问题，这将影响到元数据的搜索速度进而影响整个系统的效率。在我们的系统中，引入匹配表来解决上述问题。

在每个域中设置一个元数据服务器管理域中的多个目录服务器，匹配表保存在元数据服务器上。匹配表是一个常驻内存的数据结构，记录了每个目录服务器的名字及存放在其上的元数据的根目录；如果一个目录服务器上面保存了某个目录的副本，也把这个目录放到匹配表中去。

当系统收到用户的请求时，首先在元数据服务器上遍历匹配表，根据用户所请求访问的路径对匹配表包含的所有路径做一个最长匹配，找到用户要访问的目录所在的目录服务器，直接访问这个目录服务器，而不用通过域内元数据根目录所在的目录服务器依次查找；如果元数据存在副本，还可以从匹配表中选择一个合适的目录服务器给用户访问。

图 4.1 列举了一个逻辑树的例子。这个逻辑树代表一个域内的文件目录结构，分布在四个目录服务器上。其中，逻辑树的根/root 存放在目录服务器 DS-A 上，目录/root/B1 存放在 DS-B 上，目录/root/A1/A2/C11 和目录/root/A1/A2/C12 放在 DS-C 上，目录/root/B1/B2/B3/B4/D1 放在 DS-D 上。表 4.1 显示了与图 4.1 逻辑树对应的匹配表。

表 4.1　图 4.1 所示逻辑树对应的匹配表

根目录路径	目录服务器
/root	DS-A
/root/B1	DS-B
/root/A1/A2/C11	DS-C
/root/A1/A2/C12	DS-C
/root/B1/B2/B3/B4/D1	DS-D

举个例子，假设用户想访问/root/B1/B2/B3/B4/D1/D2。如果按照现存的元数据访问方式，首先会从域内逻辑树的根目录所在的目录服务器 DS-A 开始查找，历经 DS-B，到达 DS-D，然后再把结果返回给用户,如图 4.2 所示。但是如果采取匹配表方式，系统收到用户的请求后，将首先查找匹配表，根据用户请求的路径做一个最长匹配，这样就可以得到元数据在 DS-D 上面。系统将直接访问 DS-D，而不需要再通过根目录入口 DS-A，如图 4.3 所示。

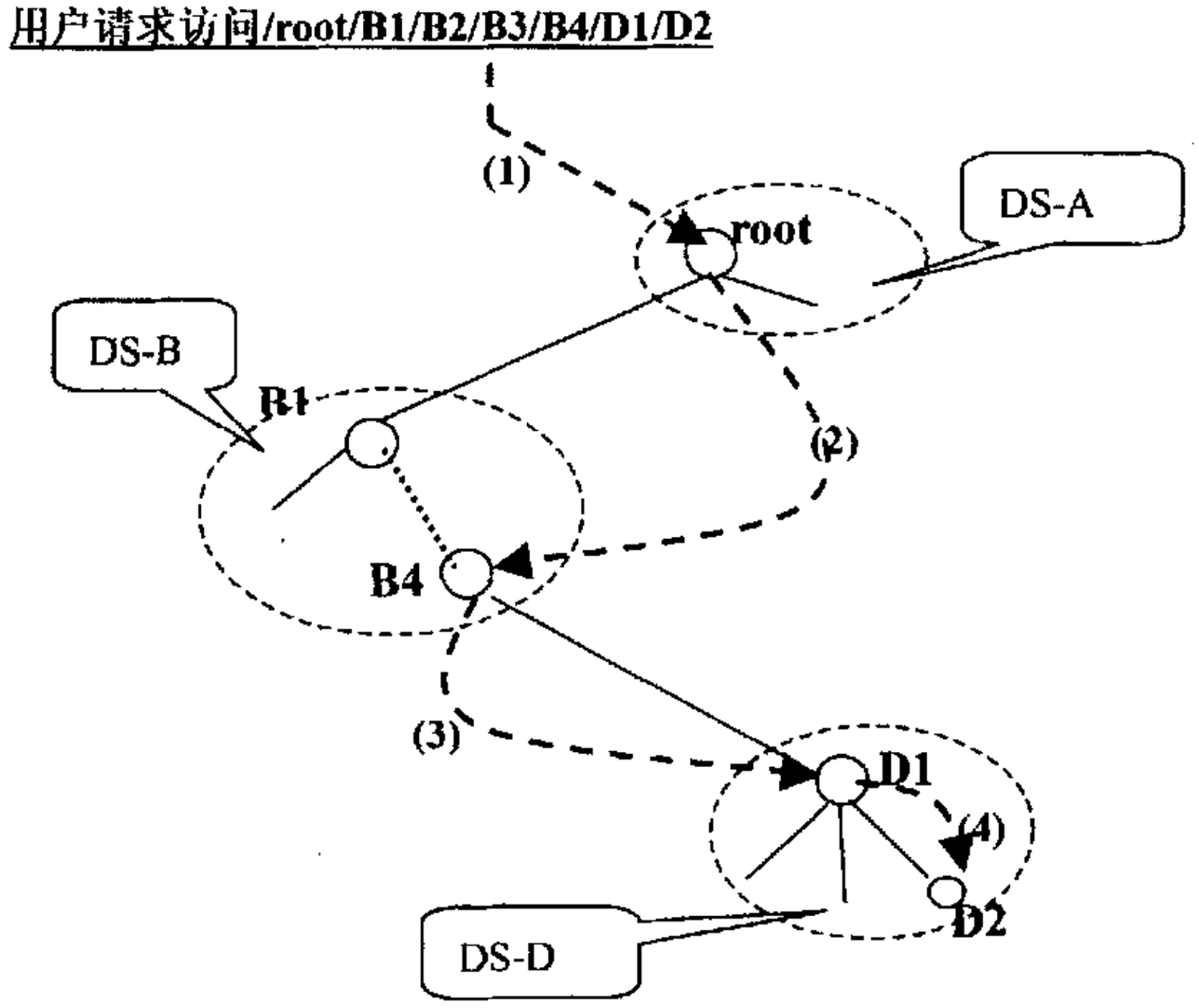

图 4.2　现存的元数据访问方式

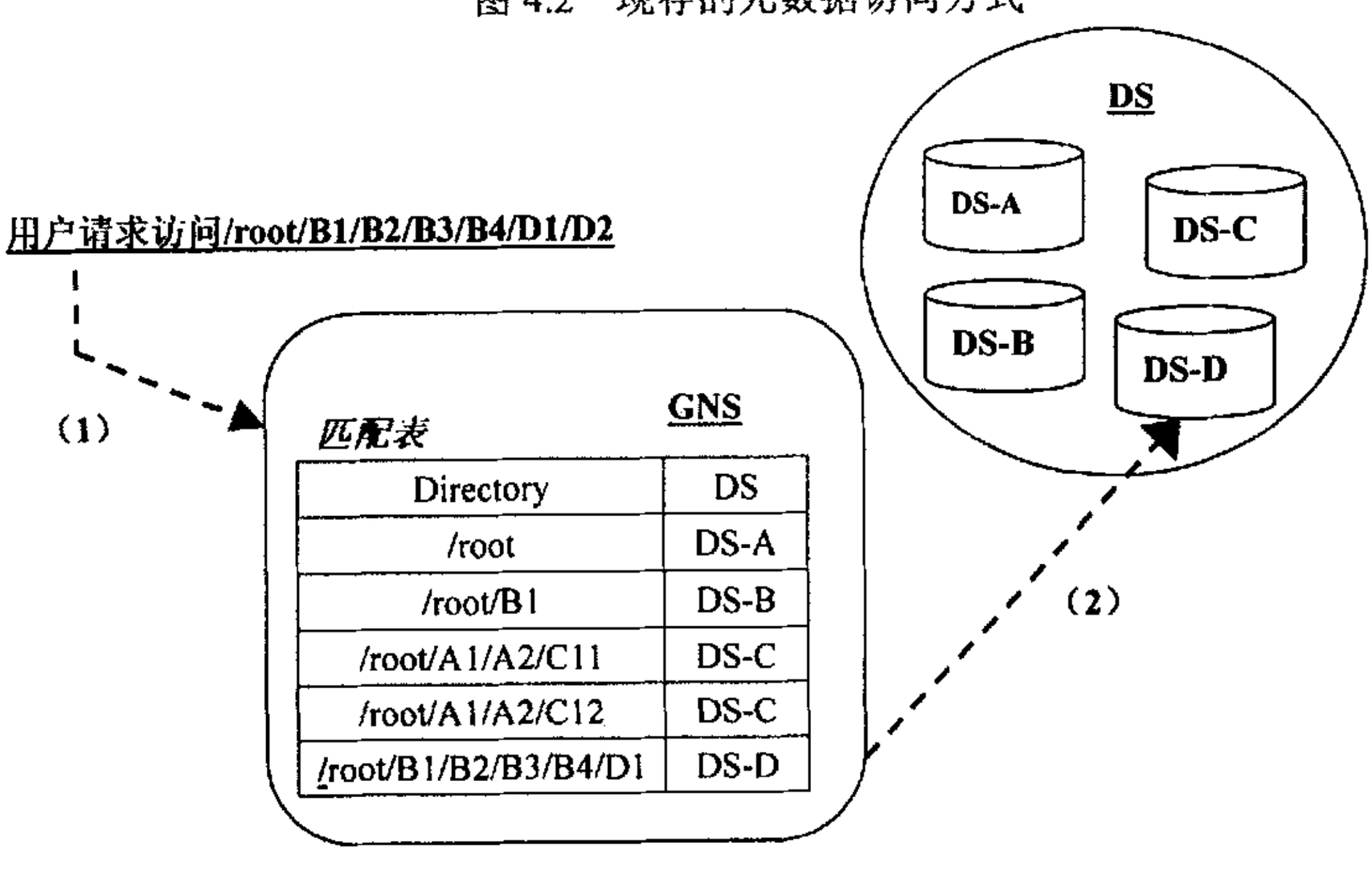

图 4.3　匹配表方式访问元数据

为了达到有效搜索的目的，所有目录服务器都必须自己保存域内元数据树的逻辑结构。例如，目录服务器 DS-D 仍然会保存域内元数据树的根，但它只是一个路径，并不保存根的内容，这样当一个搜索请求到达 DS-D 的时候就不需要对请求做任何变换。这个根条目出于内容为空，只需要占用极少的存储空间，并且维护

起来也很方便。

4.1.2 匹配表的实现

匹配表的内容存放在一个文件中，当整个系统启动的时候，解析文件把匹配表读入内存。匹配表的结构和功能决定了一般情况下都是在内存中对匹配表进行读操作，这就不会涉及到对匹配表文件的操作。只有在新增目录服务器或者现有目录服务器的根目录发生变化的情况下才对匹配表进行修改，进而对文件进行修改，并不需要频繁读写匹配表的文件，大大节省了访问匹配表的时间。

在元数据服务器中添加一个匹配表管理模块，对匹配表中的记录进行维护，具体操作包括记录的添加、删除、更新操作，匹配表文件的解析以及根据用户请求的路径匹配到合适的目录服务器等等，具体操作讨论如下：

匹配表记录的添加：当域内新增一个目录服务器的时候，向匹配表中添加一条记录；如果某个目录服务器中的一个子目录过大，需要迁移到另外一个目录服务器去，也要通知匹配表，增加一条相关记录，以记录这个子目录和其迁移到的目录服务器。

匹配表记录的删除：当某个目录服务器不再服务于某个域时，从匹配表上删除相应的记录。

匹配表记录的修改：当某个目录服务器上保存的目录树根目录发生变化时，修改相应的记录。

匹配用户请求的路径：在匹配表中查找用户请求访问的路径所在的目录服务器。

匹配表文件的解析：当系统启动的时候，解析保存匹配表的文件，读至以下三个字符串数组中，常驻内存：

Sindex[i]：保存匹配表中第 i 个记录的索引值，匹配表中的索引值递增。它并非必需，但是可以简化和方便程序的编写；

Spath[i]：保存第 i 个条目中字段“路径”的内容；

SDS[i]：保存第 i 个条目中字段“目录服务器名称”的内容。

Sindex[i]，Spath[i]，SDS[i]是相互对应的一套。从数组[1]开始记录文件。Sindex[0]的值为表中存在的条目数；Spath[0]为空；SDS[0]为空。

收到用户请求访问的文件或目录路径path后，首先遍历数组Spath，如果Spath[i]的值为 path，返回 SDS[i]；如果 Spath 中不存在 path，则返回 path 的父目录所在的所有目录服务器，或者继续向上匹配直至整个域的根目录。

4.2 缓存技术

4.2.1 缓存的设计原理

为提高元数据读写效率，辅助副本一致模块，在 MDS 模型中建立缓存。缓存模块存放在元数据服务器上，它保存热点元数据和正在被执行写操作的元数据，根据访问的局部性原则把目录服务器上的数据复制到元数据服务器上。由于只保留少量元数据，缓存并不占用太大空间。

缓存中的数据由系统自动决定在必要时有选择地删除，以让出空间进行其它数据的缓冲。需要注意的是，与副本数据不同，缓存中的数据不在复制信息表中进行登记，副本主要缓冲一个完整的数据文件，而缓存可能只缓冲一个数据文件的一部分。

缓存的工作流程：SSP 发送用户的读/写请求到元数据服务器。如果缓存中存在文件/目录信息，直接读写缓存，保证了用户访问的是最新更新过的元数据，因为写操作只在缓存上完成；不然，对于读操作，转到相应的 DS，对于写操作，把这些写操作以属性集方式记录在缓存中，然后统一更新 DS。由于根据匹配表，更新信息只传播给保存副本的 GNS 而非所有 GNS，所以极大的减少网络通讯负载。

4.2.2 缓存的实现机制

缓存机制的实现其实很简单，就是把所要缓存的元数据加到 HashMap 哈希映射表中，每一个缓存对象都用属性集 Attributes 表示。缓存中的元数据分为读对象和写对象两种，用两个 LinkedListedlist 单链表分别维持着读缓存对象和写缓存对象。读对象由单链表 R_lastAccessedList 组织，按照最后一次访问时间顺序排列读纪录；写对象由单链表 W_ageList 组织，按照创建时间顺序排列写纪录。如果一个缓存对象被访问到，那么就把它放到本链表的最前面。不定时的把要缓存对象的对象加入链表中，把过期对象删除，如此反复。

读对象的属性集包含的内容和 DS 中保存的元数据一样，但由于 DS 中并不保存文件或者目录的路径信息，所以缓存中的读对象多一个 path 属性。写对象的属性集相对而言比较复杂，主要包括：

1. 元数据的 path；
2. 对元数据进行的最近一次操作 Operation（insert, delete, updata, rename）；
3. 最近一次操作的时间 OPtimestamp；
4. 要更改/添加/删除的属性；

5. 文件/目录的名称（FileName / DirName）；

6. 文件/目录的类别（FileObject / DirObject）；

7. 元数据所在的 DS。

其中，第七条中的 DS 是一个多值 attribute，用于指示需要更新的 DS，其 value 的形式为“DSname:OPtimestamp”。其中，OPtimestamp 指出了操作的时间。如果上一次的更新 DS 操作还没有结束，这个元数据又需要有一次更新操作，那么根据 OPtimestamp 的不同来区分。例如，第一次操作为 insert，更新时间为 1，第二次操作为 updata，更新时间为 2，所在的 DS 名为“DS1”，“DS2”，那么这个属性的 value 们为：DS1:1，DS2:1，DS1:2，DS2:2。如果第一次的操作已经更新到 DS 上面了，那么就删除 DS1:1，DS2:1 两个 value。如果这个属性的 value 个数为 0，那就说明所有的更新都已经到达 DS 了，就把这个对象从写链表移到读链表中去。

每一个写对象加入缓存后，都要添加到写链表的尾部。当这个写对象被更新到其所在的每一个 DS 上以后，把写对象从写链表中删去，加入到读链表中去。缓存满时，只有读链表中的对象才可以从缓存中被置换出去。

缓存的主要操作如下：

1. 向哈希表中添加一个新文件/目录；

2. 向哈希表中添加一个属性集为 att 的读缓存对象；

3. 删除一个关键字为 path 的缓存文件/目录；

4. 更新哈希表中一个关键字为 updataPath 的缓存文件/目录对象；

5. 为哈希表中一个关键字为 path 的文件/目录重命名；

6. 从缓存中读取一个 path 文件/目录；

7. 当缓存满时，把读链表中处于表头的对象删掉，也即清理缓存；

8. 列出 path 的子目录/文件。

元数据服务器虽然是域内的单一入口，但由于没有存放大量的数据，而且及时分流只读访问，所以它并不会成为瓶颈；同时，其简单灵活的管理方式为元数据副本的一致性维护带来了极大的方便。

4.3 副本技术

元数据是描述数据的数据，一旦丢失，影响全局，因此需要对其容错功能进行研究。元数据的数据量与数据的数据量相比相对较小，数据格式也比较单一，所以本系统采用多副本对元数据进行容错。

在 GDSS 系统中，以目录为单位为元数据建立副本。副本之间是对等的。不同的用户有不同的权限，可以对其元数据要求不同的安全级别。对于安全级别比较高的元数据，可以分配较多的副本。

图 4.4 是一个域内 DS 存储元数据的例子。图 4.4（a）是 User a 目录结构的逻辑视图，图 4.4（b）是 User a 目录在域内 DS 上的物理存放结构。ROOT 是用户 a 的根目录。图 4.4（b）是一个普遍的情况，所有的这些信息可能都存放在一个目录服务器节点上，也可能分布在不同的目录服务器节点上。用户 a 的 mp3 目录存放在 node1 和 node3 上，这样如果 node1 不可用，还可以通过 node3 进行查询。

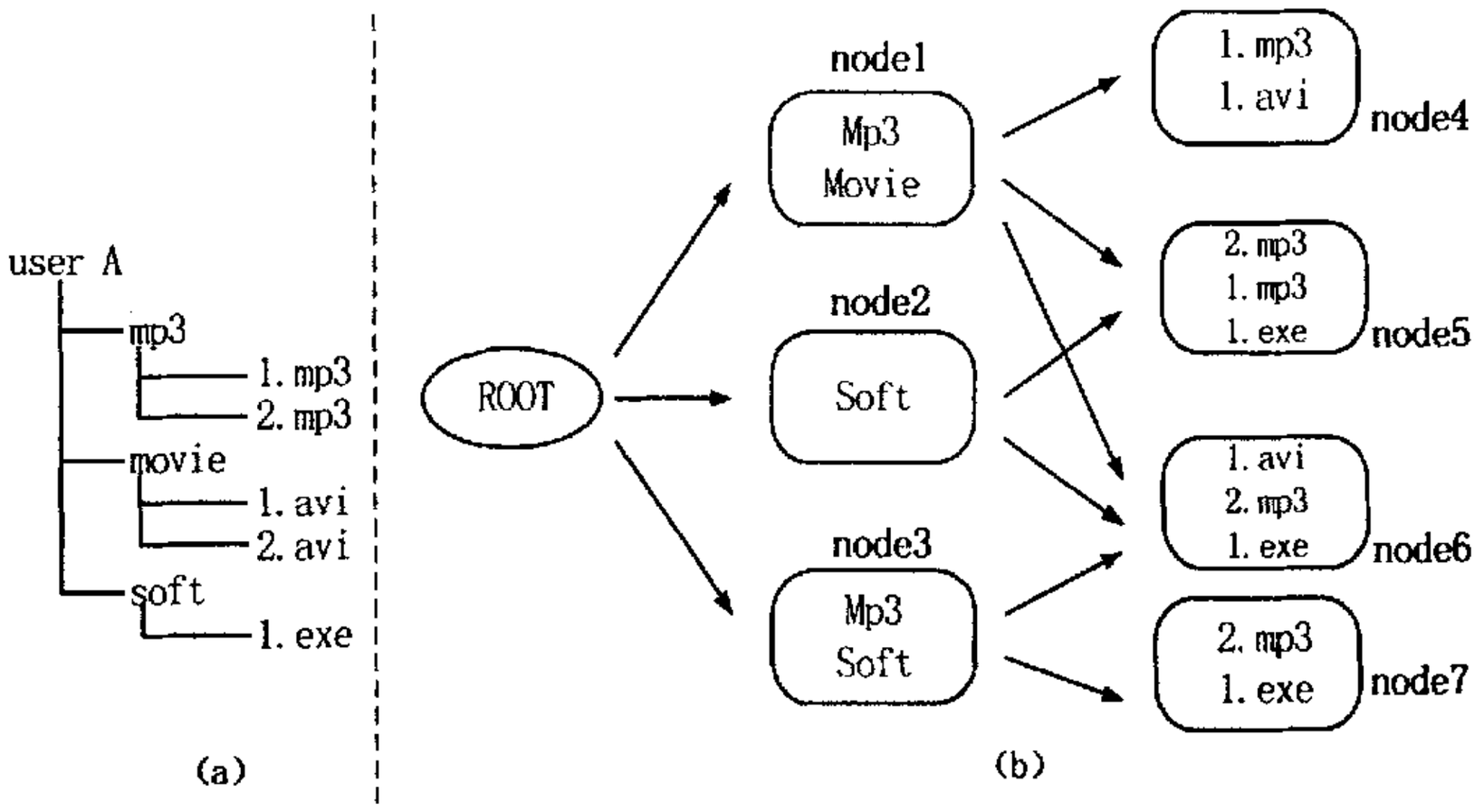

图 4.4　域内 DS 存储元数据示例

如果元数据有副本，在匹配表中相应的会存储其副本所在的所有 DS。系统访问匹配表时，能够得到这个所有 DS 的列表。这样，如果一个 DS 出错，系统可以平滑访问其他 DS。例如，在图 4.5 所示的域内元数据逻辑树中，目录/root/B1 存放在 DS-B 和 DS-C 上面，表 4.2 是图 4.5 所示逻辑树对应的匹配表。如果 DS-B 出错，可以访问 DS-C。

4.4 元数据管理模型的优势

具体来说，采用 MDC 模型具有以下优势：

1. 快速定位元数据：一次匹配就可以准确的定位元数据，避免了从根目录

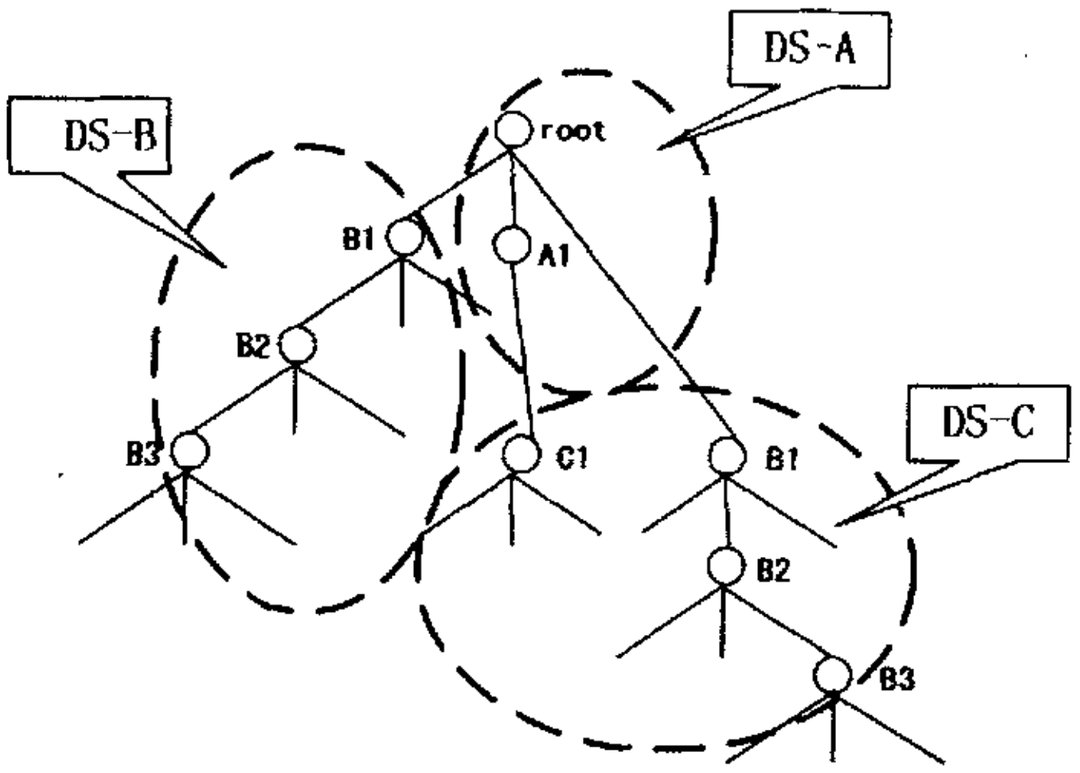

图 4.5 域内逻辑树示例

表 4.2 图 4.5 所示逻辑树对应的匹配表

根目录路径	目录服务器
/root	DS-A
/root/B1	DS-B
/root/A1/C1	DS-C
/root/B1	DS-C

服务器起的层层查询，极大的减轻了根目录服务器的负担，提高元数据的搜索速度。

2. 更高的效率：匹配表是常驻内存的，它保存的条目数受域内目录服务器数目所限，所以遍历匹配表对用户请求目录进行匹配的速度也很快。另外，当 GDSS 系统中增加元数据副本时，由于副本分布情况能够完全在匹配表中显示出来，可以通过匹配表选择一个合适的目录服务器，从而有效的利用副本，提高系统效率。

3. 更好的扩展性：分层的树型管理模式使维护和管理易于扩展；当系统负载上升需要增加目录服务器时，只需要在匹配表中增加几个条目，标示出系统中新增加的目录服务器及其对应的根目录即可。因为有了匹配表的调度，目录服务器可以达到相互独立，不需要感知域内其他目录服务器的存在。这样，系统效率不会随目录服务器数目的增多而下降，达到系统效率的可扩展性。

4. 更高的可用性：副本机制保证了更好的可用性；匹配表机制的使用，使得

当根目录服务器出现故障时，其他的目录服务器仍然可以正常工作。

5. 易于管理：匹配表的结构很简单，只记录了每个目录服务器的名字及存放在其上的元数据的根目录。所以其添加删除修改都很方便。

4.5 小结

本章详细介绍 MDC 包含的技术，有匹配表、缓存技术和元数据副本管理技术。

匹配表模块用于快速定位元数据，匹配表保存每个目录服务器的名字及存放在其上的元数据的根目录，它通过一次匹配就可以确定元数据所在的 DS，避免了在多个 DS 之间的搜索，极大的提高了元数据的搜索效率；缓存模块用于提高元数据的访问效率，同时辅助副本一致模块保证元数据副本之间的一致性；副本一致模块根据匹配表中的记录把这些更新信息统一传送给域内的 DS。

5 系统测试

本系统的测试主要包括功能测试和性能测试两个方面。其中功能测试主要包括针对元数据的一些读写操作，如：用户和组的注册、文件/目录的创建和删除、文件搜索等。性能测试主要包括元数据操作性能测试和搜索性能测试两个方面。

5.1 测试环境

如图 5.1 所示，本系统的测试环境分别安装在集群与网格计算湖北省重点实验室和教育部外存储系统重点实验室下属的几个实验室中的服务器上。集群与网格计算湖北省重点实验室测试环境包括50台主机、16个节点组成的集群和一个网关。外存储系统教育部重点实验室有多个磁盘阵列以及 SAN 和 NAS 设备。系统搭建起来两个资源域 iccc.hust.edu.cn 和 storage.hust.edu.cn。设置了两个 SSP 服务器 neverhood.ssp（192.168.1.62）和 node16.ssp（192.168.1.16），设置了多个 Agent 分别归属于两个资源域。

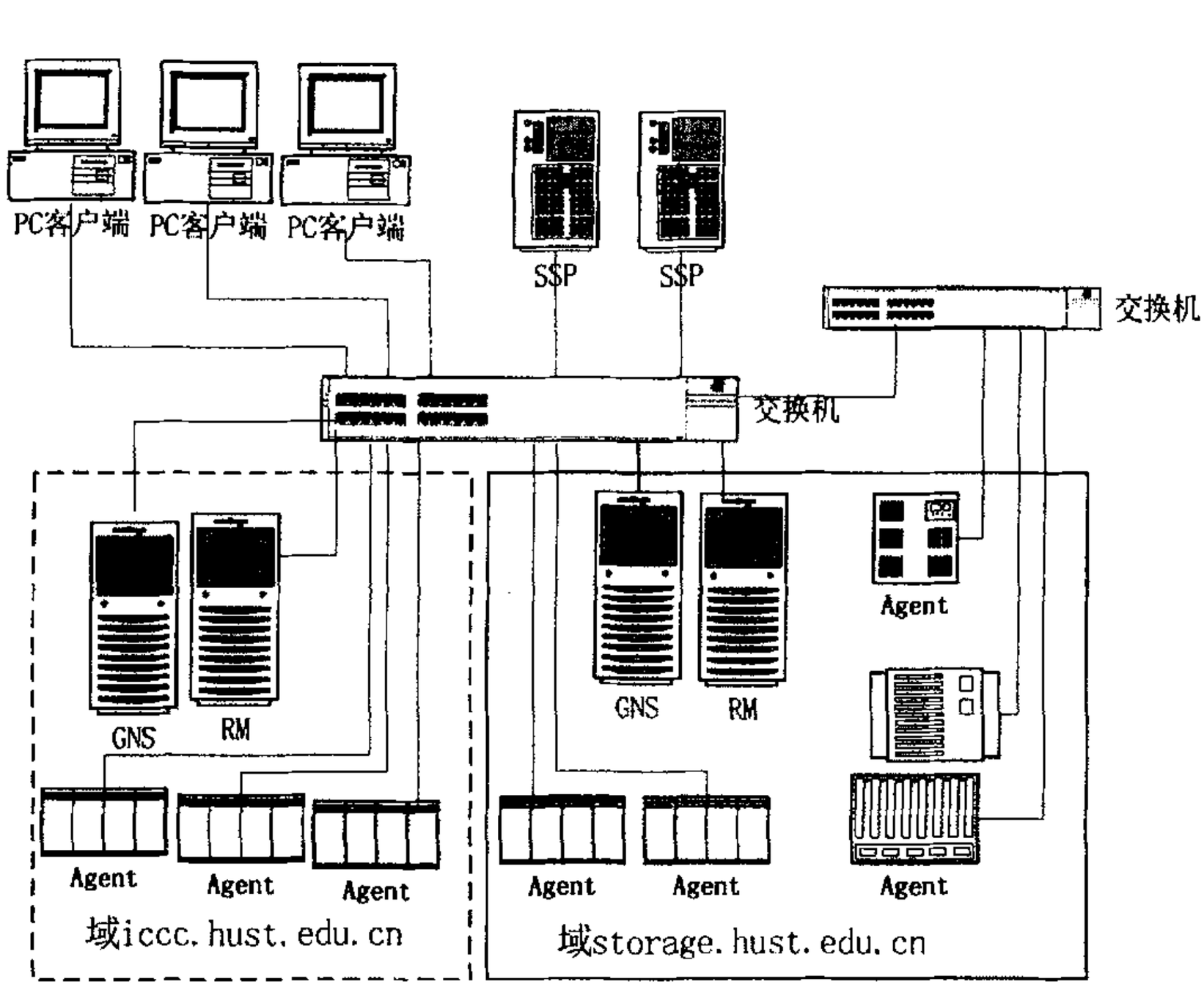

图 5.1 测试环境

5.2 功能测试

在 Windows 平台上采用流行的 FTP 客户端 CuteFtp、Flashxp、LeafFtp 和 Windows XP 自带 ftp 工具程序，在 Linux 平台上采用 ftp 程序、gFtp 进行测试，在 Windows 平台上采用系统自带的客户端 GDSSClient 登陆 GDSS 系统进行操作，实现以下功能：

1. 用户注册：进入 GDSSClient 注册单元，输入用户名（datagrid）和密码（datagrid），选择“用户”类别，选择所在域（iccc.hust.edu.cn），就会在域中建立用户 datagrid @ iccc.hust.edu.cn。如果用户 datagrid 已经存在，返回错误信息。

2. 创建目录：使用 datagrid @ iccc.hust.edu.cn 登陆，在用户根目录下创建目录 test，会在 iccc.hust.edu.cn 域目录树中 datagrid 用户根目录所指向的目录下建立目录 test。如果目录 test 已经存在，返回错误信息。

3. 创建文件：在 test 目录下创建文件 test.txt，会在 iccc.hust.edu.cn 域目录树中 datagrid 用户根目录所指向的目录下的 test 目录下面建立文件 test.txt。如果文件 test.txt 已经存在，返回错误信息。

4. 删除目录：删除目录 test，则 test 的元数据信息被删除。如果文件 test 不存在，返回错误信息。

5. 删除文件：删除文件 test.txt，则 test.txt 的元数据信息被删除。如果文件 test.txt 不存在，返回错误信息。

6. 上载目录：上载目录 upload 到用户 datagrid 的根目录下，会在 iccc.hust.edu.cn 域目录树中 datagrid 用户根目录所指向的目录下建立目录 upload，并相应的循环建立 upload 的子目录下的内容。如果目录 upload 已经存在，返回错误信息。

7. 下载目录：下载目录 upload，读取 datagrid 用户下关于 upload 目录及其子目录的元数据信息，返回。如果目录 upload 不存在，返回错误信息。

8. 上传文件：上传文件 upload.txt 到用户 datagrid 的根目录下，会在 iccc.hust.edu.cn 域目录树中 datagrid 用户根目录所指向的目录下建立文件 upload.txt。如果目录 upload 已经存在，返回错误信息。

9. 下载文件：下载文件 upload.txt，读取 datagrid 用户下关于文件 upload.txt 的元数据信息，返回。如果文件 upload.txt 不存在，返回错误信息。

10. 组注册：进入 GDSSClient 的注册单元，输入组名（group）和密码（group），

选择“组”类别，选择所在域（iccc.hust.edu.cn），就会在域中建立组 group @ iccc.hust.edu.cn。如果组 group 已经存在，返回错误信息。

11. 用户加入组：在“用户管理”单元，输入要加入的组名，如果组管理员批准，则加入成功。

12. 组中用户的管理：组管理员可以向组内添加删除用户和修改用户的权限。

13. 数据共享的建立、接受、拒绝：用户可以共享自己的目录给别的组或用户，后者确认之后就可以共享资源，也可以选择拒绝共享。

14. 对共享数据的访问：用户可以上传下载共享数据。

15. 文件搜索：用户可以在指定的目录中根据一定的条件搜索文件。

16. 元数据容错功能：在资源域 iccc.hust.edu.cn 中运行两个目录服务器 DS1 和 DS2，二者保存相同的元数据，运行正常。停止 DS1，iccc.hust.edu.cn 域内的用户再次访问自己的文件/目录，运行依然正常。

5.3 性能测试

5.3.1 元数据操作性能测试

在 Windows 平台上采用系统特定的客户端 GDSSClient 登陆 GDSS 系统，测试用户操作文件元数据信息所需要的响应时间，包括目录浏览响应时间、用户登录响应时间、读写文件响应时间和提交文件响应时间。

1. 浏览目录响应时间（每组 50 次）

分别用不同的用户登陆 GDSS 客户端浏览其根目录和子目录，每组模拟次数 50 次，记录系统平均响应时间。图 5.2 是一个测试用例，其中的 user 目录是用户 datagrid @ iccc.hust.edu.cn 在系统中指向的根目录，其下包含多级子目录和多个文件。测试结果如表 5.1 和图 5.3 所示。

表 5.1 浏览目录响应时间的测试结果

	第一组	第二组	第三组	第四组	第五组	第六组
响应时间（ms）	217.92	218.86	217.88	217.51	222.30	218.5

从测试结果可以看出，MDC 对用户操作的支持始终维持在一个稳定的时间范围内。

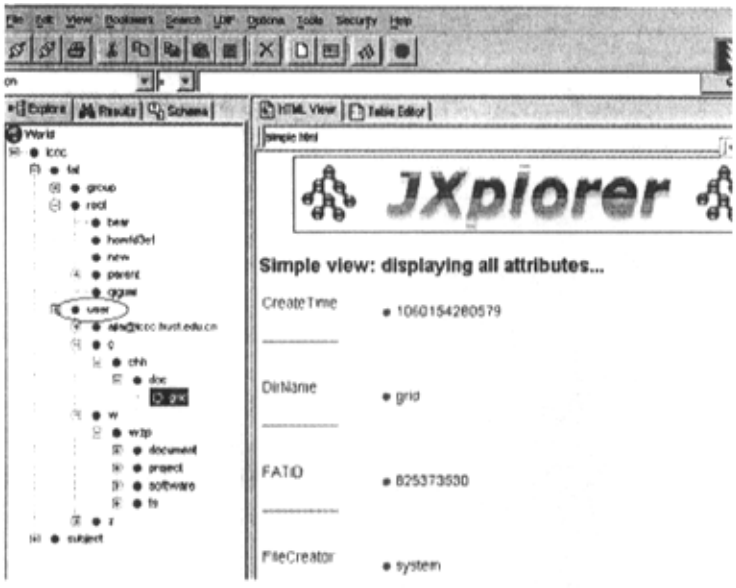

图 5.2 目录浏览测试用例示意

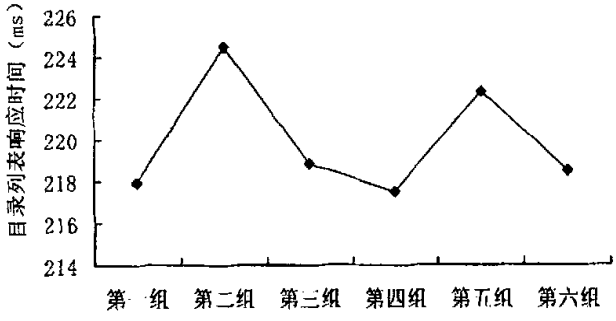

图 5.3 浏览目录响应时间测试结果示意图

2. 用户登录响应时间（每组 50 次）

用户登录响应时间指用户从开始登陆系统到进入自己的根目录之间的时间。分别用不同的用户登陆 GDSS 客户端，每组模拟次数 50 次，记录系统平均响应时间。测试结果如表 5.2 和图 5.4 所示。

表 5.2 用户登录响应时间的测试结果

	第一组	第二组	第三组	第四组	第五组	第六组
响应时间（ms）	247.5	224.49	227.35	217.80	225.45	242.92

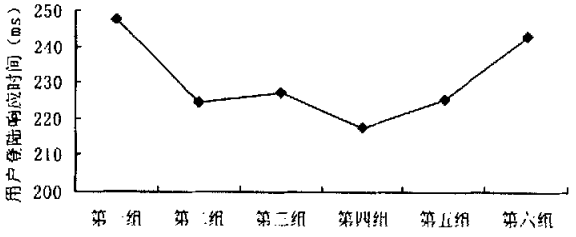

图 5.4 用户登陆响应时间

3. 读/写文件响应时间（每组 50 次）

读/写文件响应时间指用户从提交操作到文件开始传输之间的时间。分别用不同的用户登陆 GDSS 客户端进行上传和下载文件操作，每组模拟次数 50 次，记录系统平均响应时间。测试结果如表 5.3 和图 5.5 所示。

表 5.3 读/写文件响应时间的测试结果

	第一组	第二组	第三组	第四组	第五组	第六组
响应时间（ms）	1071.96	1176.65	933.41	1076.96	961.78	940.11

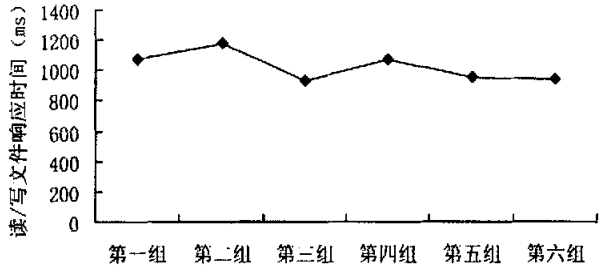

图 5.5 读写文件响应时间

4. 写文件提交时间（每组 50 次）

当用户新建（或更新）文件时，先建立新文件的元数据（或者更新文件的元数据），然后在 RM 上实际写文件，最后“回写”元数据，修改元数据中的文件长度属性值。写文件提交时间就是指这个“回写”的时间。分别用不同的用户登陆 GDSS 客户端，进行新建文件或者更新文件操作，每组模拟次数 50 次，记录系统平均写文件提交时间。测试结果如表 5.4 和图 5.6 所示。

表 5.4 写文件提交时间的测试结果

	第一组	第二组	第三组	第四组	第五组	第六组
响应时间（ms）	259.6	237.57	249.29	238.12	252.18	242.46

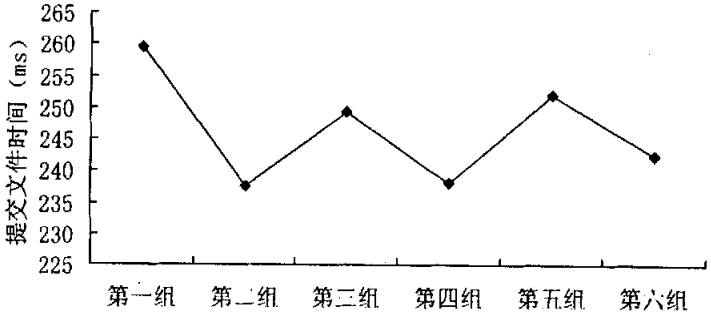

图 5.6 提交文件读写响应时间

5.3.2 对元数据的搜索性能测试

在客户端进行基于文件名子串查询功能的测试。单个用户一共包含文件10694，其中文件 9904 个，目录 790 个。目录服务器中共包含文件目录信息条目33900 个,共占用存储空间大小为 6M（包括索引信息），单个文件平均元数据信息大小为 185Byte。表 5.5 和图 5.7 为单个用户环境下的搜索响应时间测试结果（ms）（每组模拟次数 50 次）：

表 5.5 单个用户环境下的搜索响应时间测试结果

查询字符长度（Byte）	2	3	4	5	6	7	8
响应时间（ms）	1112	946	50	47	50	52	55

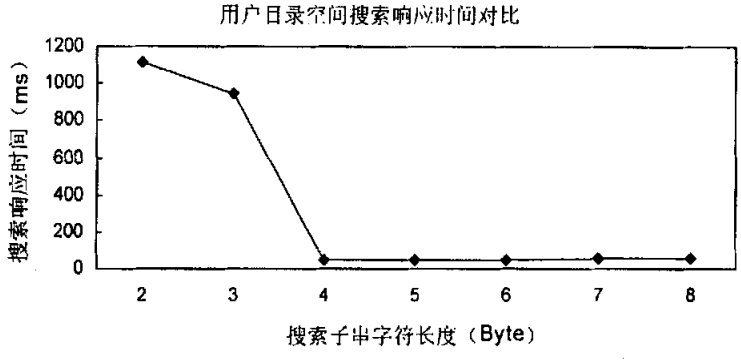

图 5.7 用户目录空间搜索响应时间比

由搜索测试结果可以看出，元数据信息量较小，单个条目信息仅使用 185Byte 左右的空间，保证了系统具有管理海量数据的能力。基于用户目录空间的搜索功

能得到了较好的实现，随着用户提交的查询范围、精度的不断提升，系统反应时间急剧下降。

5.4 结论

本章首先介绍了系统的测试环境，然后按照既定步骤从系统功能和系统性能两个方面对 MDC 模型进行了测试。

由功能测试可以看到，所设计的 MDC 模型可以完成用户/组注册、创建/删除目录或文件、上/下载目录或文件、数据共享、文件搜索等功能，并实现了元数据容错，已经达到既定的功能目标。

系统性能测试分为元数据操作性能测试和元数据搜索性能测试。操作性能测试了浏览目录响应时间、用户登录响应时间、读/写文件响应时间和写文件提交时间四个方面；搜索性能测试了单用户环境下基于文件名子串查询性能。从测试结果可以看出，MDC 对用户操作的支持始终维持在一个稳定的时间范围内，具有很好的可用性；最后，基于用户目录空间的搜索功能也得到了较好的实现。

6 结束语

在 863 项目“存储虚拟化及其文件系统研究”的资助下，经过两年的探索、研究和开发，成功研制了一个广域网范围内的虚拟化存储系统 GDSS，2003 年 12 月 26 日进行了系统测试，2003 年 12 月 28 日进行了系统鉴定。专家们一致认为系统已达到国际先进水平。

本文综合分析了现存的存储虚拟化系统中元数据管理方式的不足，并在 GDSS 系统中提出一种基于匹配表的元数据层次管理模型 MDC。在 MDC 模型中，元数据以类 UNIX 文件的层次方式存放在目录服务器中，采用全局命名服务器来管理域内的多个目录服务器。匹配表模块放置在全局命名服务器上，保存每个目录服务器的名字及存放在其上的元数据的根目录。系统通过匹配表来定位元数据。其优点在于：

1. 快速定位元数据：一次匹配就可以准确的定位元数据，避免了从根目录服务器起的层层查询，极大的减轻了根目录服务器的负担；

2. 更高的可用性：当根目录服务器出现故障时，其他的目录服务器仍然可以正常工作；

3. 更好的扩展性：增加一个目录服务器时，只需要在匹配表中添加一条记录，系统的效率不会因此而降低，因为并不增加根目录服务器的负载；

4. 提高效率：匹配表常驻内存，所以匹配速度会大大提升；同时，由于匹配表所拥有的条目并不多，搜索匹配表对用户请求目录进行匹配的效率也很高；

5. 简洁实用：匹配表只记录每个目录服务器的根入口地址，所占空间极小，管理方便。

由于 Globus 在网格领域已是事实上的标准，在以后的工作中，将结合下一个项目中国教育科研网格 ChinaGrid 计划，把 GDSS 和 Globus 相结合，构建 ChinaGrid 中的数据网格平台。

根据 ChinaGrid 系统的特性和要求，将 GDSS 中的 SSP、GNS 和 RM 合为一体，统一对外提供服务，同时调整各个子模块的功能使之更适合 ChinaGrid。系统整体结构的简化会使得模块之间的通讯开销降低，系统的稳定性得到进一步的提高。

在元数据管理方面，系统将沿用 Globus 中元数据的管理机制。但是由于 GT3

所提供的元数据管理只具有一个大体的框架，所以将沿用并改进 GDSS 中的元数据管理方法，进一步加强系统在元数据容错方面的功能，对元数据副本的一致性提出更加适合 ChinaGrid 的算法；另外在新的系统中，将提供基于属性的搜索，优化搜索算法，提高搜索的效率，使其更好的运行在 ChinaGrid 大平台上。

致 谢

CGCL 是个充满朝气的地方，在这两年多里，我非常幸运的伴随着 CGCL 实验室一起成长，很多的老师和朋友给了我热情无私的帮助，令我在研究生阶段过的十分充实，所以我要感谢这个地方。

首先我要感谢我的导师金海教授。金老师给我们带来最先进的研究课题，并时刻跟踪国际潮流调整研究的方向。他渊博的学识，敏锐洞察力和不折不挠的进取精神令人敬佩。在这里，金老师提倡的学术自由极大的激发了我们的研究热情和创新欲望。另外，金老师对待学生犹如朋友一般坦诚相待，他是我在学习和生活中的榜样。对于他一直以来对我的关怀与支持，在此致以深深的谢意。

同时，我幸运的得到了实验室其他几位老师共同的关心和指导。自进实验室以来，我一直得到韩宗芬老师的悉心指导。在工作中，韩老师是良师益友，她认真和执着的精神已深深的感染了我，在生活中，韩老师是可敬可亲的长辈，她待人热情，经常毫不吝啬的给予我鼓励与支持。还记得韩老师不顾眼疾对我的第一篇文章细心的修改，韩老师，衷心的感谢您。其次要感谢庞丽萍教授，庞老师非常关心我的科研情况和日常生活，经常对我嘘寒问暖，给予有效的指导。另外要感谢李胜利教授、章勤教授和石柯博士。李老师待学生热情大度，章老师对工作认真负责，这一切都感染着实验室的每一位同学。

特别感谢我们存储虚拟化小组的指导老师吴松博士和每一位小组成员，他们是冉龙波、王志平、陈勇、黄琛、周润松、官象山、刘志坤、熊幕舟、王庆春和谢超。我们的项目得以成功鉴定并取得比较好的成绩是大家的辛苦劳动换得的。吴松博士作为项目指导老师，在项目的进度等工作上付出了大量的工作，带领我们顺利完成了项目的鉴定。也特别感谢他在日常生活中对我的关心，对于人生的选择主动和我谈心，为我指明方向。还要非常感谢冉龙波和王志平两位组长，他们丰富的专业知识和熟练的编程技能是我学习的榜样。同时感谢周润松同学对我在系统完善中的帮助。再次感谢我们组的各位兄弟姐妹，大家辛苦了，希望大家记得我们曾经奋斗的日子。

在 CGCL 实验室的三年的生活中，很多师兄师姐师弟师妹也都给予了我无私的帮助和深厚的友情，真心的祝福大家今后幸福快乐。

感谢我的室友姜沙沙和徐彬，三年的相处中我们彼此带来了很多欢笑，希望

沙沙在宝洁的工作一帆风顺，也希望徐彬能够顺利在上海找到如意的工作，并祝你们两个爱情甜蜜。

还要感谢我的父母，他们从小到大给了我最无私的关怀，他们的一生辛辛苦苦，为了儿女奉献了自己的一切，我无法用言语来表达我对他们的感激，祝福他们身体健康、安享晚年。

还要特别感谢我的男朋友易川江，他在学习上帮助我，在生活上关心我，在我遇到困难时鼓励我，给了我极大的支持。在以后的日子里，我也会认真的珍惜他。

感谢所有关心支持我的每一个人，祝福他们健康快乐。

最后，衷心感谢各位评委的批评和指导！

参考文献

[1] C. Milligan, S. Selkirk. Online Storage Virtualization: The Key to Managing the Data Explosion. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences. Hawaii, 2002. Washington DC: IEEE Computer Society, 2002. 3052~3060

[2] B. Tierney, W. Johnston, J. Lee, et al. A Data Intensive Distributed Computing Architecture for Grid Applications. Future Generation Computer Systems, 2000, 16(5): 473~481

[3] Hu Yiming, Nightingale Tycho, Yang Qing. RAPID-Cache - A reliable and inexpensive write cache for high performance storage systems. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 290~307

[4] Foster, I. The globus toolkit for grid computing. In: Proceedings of the 1st International Symposium on Cluster Computing and the Grid. Brisbane. 2001. Washington DC: IEEE Computer Society, 2001. 2~2

[5] I. Foster, C. Kesselman, J. Nick, et al. Grid Services for Distributed System Integration. Computer, 2002, 35(6): 37~46

[6] Patrick C. Moore, Wilbur R. Johnson, Richard J. Detry. Adapting globus and kerberos for a secure ASCI grid. In: Proceedings of the 2001 ACM/IEEE conference on Supercomputing. Denver. 2001. New York: ACM Press, 2001. 21~21

[7] Ellert M., Konstantinov A., Konya B., et al. The NorduGrid project: Using Globus toolkit for building GRID infrastructure. In: Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. Moscow. 2003. Moscow: Elsevier, 2003. 407~410

[8] Sudharshan Vazhkudai, Steven Tuecke, Ian Foster. Replica Selection in the Globus Data Grid. In: Proceedings of the 1st International Symposium on Cluster Computing and the Grid. Brisbane. 2001. Washington DC: IEEE Computer Society, 2001. 2~2

[9] J. Bester, I. Foster, C. Kesselm an, et al. GASS: A data movement and access service for wide area computing systems. In: Proceedings of the Sixth Workshop on

Input/Output in Parallel and Distributed Systems. Atlanta. 1999. New York: ACM Press, 1999. 78~88

[10]B. Allcock, J. Bester, J. Bresnahan, et al. Data Management and Transfer in High Performance Computational Grid Environments. Parallel Computing Journal, 2002, 28 (5): 749~771

[11]Sunil Thulasidasan, Wu-chun Feng, Mark K Gardner. Optimizing GridFTP through Dynamic Right-Sizing. In: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing. Seattle. 2003. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2003. 14~23

[12]Vazhkudai Sudharshan, Schopf, Jennifer M. Using regression techniques to predict large data transfers. International Journal of High Performance Computing Applications, 2003, 17(3): 249~268

[13]Stockinger H, Samar A., Allcock B., et al. File and object replication in Data Grids. In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing. San Francisco. 2001. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2001. 76~86

[14]Wolfgang Hoschek, Francisco Javier Jaén-Martínez, Asad Samar, et al. Data Management in an International Data Grid Project. In: Proceedings of the First IEEE/ACM International Workshop on Grid Computing. Bangalore. 2000. London: Springer-Verlag, 2000. 77~90

[15]Clark, Elizabeth. Storage virtualization. Network Magazine, 2003, 18(6): 22~24

[16]Rajasekar A., Wan M., Moore R. MySRB and SRB - components of a Data Grid. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. Edinburgh. Edinburgh. 2002. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2002. 301~310

[17]Krauter Klaus, Buyya Rajkumar, Maheswaran Muthucumaru. A taxonomy and survey of grid resource management systems for distributed computing. Software - Practice and Experience, 2002, 32(2): 135~164

[18] Wan M., Rajasekar A., Moore R., et al. A simple mass storage system for the SRB data grid. In: Proceedings. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies. San Diego. 2003. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2003. 20~25

[19]Jacobsen Kjetil, Zhang Xianan, Marzullo Keith. Group membership and wide-area master-worker computations. In: Proceeding of the 23th IEEE International

Conference on Distributed Computing Systems. Providence. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2003. 570~579

[20]R. J. Figueiredo, N. H. Kapadia, and J. A. B. Fortes. The punch virtual file system: Seamless access to decentralized storage services in a computational grid. In: Proceeding of the 10th IEEE International Symposium on High Performance Distributed Computing. San Francisco. 2001. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2001. 334~344

[21]Natrajan Anand, Humphrey Marty A., Grimshaw Andrew S. The legion support for advanced parameter-space studies on a grid. Future Generation Computer Systems, 2002, 18(8): 1033~1052

[22]路川, 张堃. 存储区域网技术及其应用. 指挥技术学院学报, 2001, 12(2): 70~73

[23]H. Kai, J. Hai, R.S.C Ho. Orthogonal Striping and Mirroring in Distributed RAID for I/O-centric Cluster Computing. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(1): 26~44

[24]A. Jagatheesan, R. Moore, A. Rajasekar, et al. Virtual Services in Data Grids. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing . Edinburgh. 2002. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2002. 420~420

[25]P. Wang, R. Gilligan, H. Green, et al. IP SAN - from iSCSI to IP-addressable Ethernet Disks. in: Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies. San Diego. 2003. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2003. 189~193

[26]Atarashi R.S, Kishigami J, Sugimoto S. Metadata and new challenges. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops. Orlando. 2003. Washington DC: IEEE Computer Society, 2003. 395~398

[27]Steinacker A, Ghavam A, Steinmetz R. Metadata standards for Web-based resources. Multimedia, IEEE, 2001, 8(1): 70~76

[28]Vaduva A., Dittrich K.R. Metadata management for data warehousing: between vision and reality. In: International Database Engineering and Applications Symposium. Grenoble. 2001. Washington DC: IEEE Computer Society, 2001. 129 ~135

[29]Ruixin Yang, Xinhua Deng, Kafatos M, et al. An XML-based distributed metadata server (DIMES) supporting Earth science metadata. In: Proceeding of the 13th International Conference on Scientific and Statistical Database Management.

Fairfax, VA. 2001. San Francisco: Institute of Electrical and Electronics Engineers Computer Society, 2001. 251~256

[30]Andres F., Noureddine M., Ono K., et al. Metadata model, resource discovery, and querying on large scale multidimensional datasets. The GEREQ project. In: 2000 International Conference on Digital Libraries: Research and Practice. Kyoto. 2000. Washington DC: IEEE Computer Society, 2000. 312~319

[31]Widener P., Eisenhauer G., Schwan K. Open metadata formats: efficient XML-based communication for high performance computing. In: Proceeding of the 10th IEEE International Symposium on High Performance Distributed Computing. Mesa. 2001. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2001. 371~380

[32]Ruixin Yang, Kafatos M., Wang X.S. Managing scientific metadata using XML. Internet Computing, IEEE, 2002, 6(4): 52~59

[33]Ganesha Beedubai, Udo Pooch. Naming Consistencies in Object Oriented Replicated Systems. Texas: Texas A & M University, 2001. 5~30

[34]Bill Allcock, Joe Bester, John Bresnahan, et al. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In: Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies. San Diego. 2001. Washington DC: IEEE Computer Society, 2001. 13~13

[35]徐志大, 白鹏, 南相浩. 目录服务协议分析、比较与实现. 计算机工程与应用, 2001, 03 期: 51~53

[36]Fabián E. Bustamante, Patrick Widener, Karsten Schwan. Scalable directory services using proactivity. In: Proceedings of the 2002 ACM/IEEE conference on Supercomputing. Baltimore. 2002. Washington DC: IEEE Computer Society, 2002. 1~12

[37]Gietz, P. LDAP and the grid. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. Berlin. 2002. Washington DC: IEEE Computer Society, 2002. 4~4

[38]徐建波, 李仁发, 蒋云霞. 基于 LDAP 的目录服务分析与实践. 湘潭矿业学院学报, 2002, 17(1): 79~82

[39]陈小弟, 李长河, 张熙等. 基于 LDAP 的 Internet 分布式目录服务的分析与实现. 计算机工程, 2002, 28(8): 271~273

[40]Pete Loshin, Bill McCarthy. Big Book of Lightweight Directory Access Protocol (Ldap) Rfcs. Edition: 1st. Orlando: Academic Press, Inc, 2000. 1~53

[41]Lizmari Brignoni. LDAP directory basics. Sys Admin, June, 2000, 9(6): 23~31

[42]Keith Bell, Andrew Chien, Mario Lauria. A High-Performance Cluster Storage Server. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. Edinburgh. 2002. San Francisco: Institute of Electrical and Electronics Engineers Inc, 2002. 311~320

[43]John Kubiatowicz, David Bindel, Yan Chen, et al. OceanStore:an architecture for global-scale persistent storage. In: International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS, 2000. Cambridge. 2000. New York: Association for Computing Machinery, 2000. 190~201

附 录 1（攻读学位期间发表论文目录）

[1] 贾永洁，金 海，易川江，韩宗芬，吴 松，肖 侬. 存储虚拟化系统的元数据副本一致性管理模型. 计算机工程与科学，已录用，文章编号 23254. 署名单位：华中科技大学计算机科学与技术学院和国防科学技术大学计算机学院

附 录 2（攻读学位期间申请专利目录）

[1] 金海，贾永洁，吴松，冉龙波，王志平，周润松，“一种存储虚拟化系统的元数据层次管理方法及其系统”（申请号为 200310111436.x，申请日期：2003 年 11 月）

存储虚拟化系统元数据管理的设计与实现

作者：贾永洁
学位授予单位：华中科技大学

参考文献(44条)

1. 参考文献

2. C Milligan. S Selkirk Online Storage Virtualization:The Key to Managing the Data Explosion 2002

3. B Tierney. W Johnston. J Lee A Data Intensive Distributed Computing Architecture for Grid Applications 2000(05)

4. Hu Yiming. Nightingale Tycho. Yang Qing RAPID-Cache - A reliable and inexpensive write cache for high performance storage systems 2002(03)

5. Foster Ⅰ The globus toolkit for grid computing 2001

6. I Foster. C Kesselman. J Nick Grid Services for Distributed System Integration 2002(06)

7. Patrick C Moore. Wilbur R Johnson. Richard J Detry Adapting globus and kerberos for a secure ASCI grid 2001

8. Ellert M. Konstantinov A. Konya B The NorduGrid project:Using Globus toolkit for building GRID infrastructure 2003

9. Sudharshan Vazhkudai. Steven Tuecke. Ian Foster Replica Selection in the Globus Data Grid 2001

10. J Bester. I Foster. C Kesselm an GASS:A data movement and access service for wide area computing systems 1999

11. B Allcock. J Bester. J Bresnahan Data Management and Transfer in High Performance Computational Grid Environments 2002(05)

12. Sunil Thulasidasan. Wu-chun Feng. Mark K Gardner Optimizing GridFTP through Dynamic Right-Sizing 2003

13. Vazhkudai Sudharshan. Schopf Jennifer M Using regression techniques to predict large data transfers 2003(03)

14. Stockinger H. Samar A. Allcock B File and object replication in Data Grids 2001

15. Wolfgang Hoschek. Francisco Javier Jaén-Martinez. Asad Samar Data Management in an International Data Grid Project 2000

16. Clark Elizabeth Storage virtualization 2003(06)

17. Rajasekar A. Wan M. Moore R MySRB and SRB - components of a Data Grid 2002

18. Krauter Klaus. Buyya Rajkumar. Maheswaran Muthucumaru A taxonomy and survey of grid resource management systems for distributed computing 2002(02)

19. Wan M. Rajasekar A. Moore R A simple mass storage system for the SRB data grid 2003

20. Jacobsen Kjetil. Zhang Xianan. Marzullo Keith Group membership and wide-area master-worker computations 2003

21. R J Figueiredo. N H Kapadia. J A B Fortes The punch virtual file system:Seamless access to decentralized storage services in a computational grid 2001

22. Natrajan Anand. Humphrey Marty A. Grimshaw Andrew S The legion support for advanced parameter-space studies on a grid 2002(08)

23. 路川. 张堃 存储区域网技术及其应用[期刊论文]-指挥技术学院学报 2001(2)

24. H Kai. J Hai. R S C Ho Orthogonal Striping and Mirroring in Distributed RAID for I/O-centric Cluster Computing 2002(01)

25. A Jagatheesan. R Moore. A Rajasekar Virtual Services in Data Grids 2002

26. P Wang. R Gilligan. H Green IP SAN - from iSCSI to IP-addressable Ethernet Disks 2003

27. Atarashi R S. Kishigami J. Sugimoto S Metadata and new challenges 2003

28. Steinacker A. Ghavam A. Steinmetz R Metadata standards for Web-based resources 2001(01)

29. Vaduva A. Dittrich K R Metadata management for data warehousing:between vision and reality 2001

30. Ruixin Yang. Xinhua Deng. Kafatos M An XML-based distributed metadata server (DIMES) supporting Earth science metadata 2001

31. Andres F. Noureddine M. Ono K Metadata model,resource discovery,and querying on large scale multidimensional datasets.The GEREQ project 2000

32. Widener P. Eisenhauer G. Schwan K Open metadata formats:efficient XML-based communication for high performance computing 2001

33. Ruixin Yang. Kafatos M. Wang X S Managing scientific metadata using XML 2002(04)

34. Ganesha Beedubai. Udo Pooch Naming Consistencies in Object Oriented Replicated Systems 2001

35. Bill Allcock. Joe Bester. John Bresnahan Secure,Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing 2001

36. 徐志大. 白鹏. 南相浩 目录服务协议分析、比较与实现[期刊论文]-计算机工程与应用 2001(3)

37. Fabián E Bustamante. Patrick Widener. Karsten Schwan Scalable directory services using proactivity 2002

38. Gietz P LDAP and the grid 2002

39. 徐建波. 李仁发. 蒋云霞 基于LDAP的目录服务分析与实践[期刊论文]-湘潭矿业学院学报 2002(1)

40. 陈小弟. 李长河. 张熙. 杜江杰 基于LDAP的Internet分布式目录服务的分析与实现[期刊论文]-计算机工程 2002(8)

41. Pete Loshin. Bill McCarthy Big Book of Lightweight Directory Access Protocol(Ldap) Rfcs 2000

42. Lizmari Brignoni LDAP directory basics 2000(06)

43. Keith Bell. Andrew Chien. Mario Lauria A High-Performance Cluster Storage Server 2002

44. John Kubiatowicz. David Bindel. Yan Chen OceanStore:an architecture for global-scale persistent storage 2000