一个基于第三方数据传送的 SAN 数据共享系统

贾瑞勇，张延园
(西北工业大学 计算机科学与工程系，陕西 西安 710072)

摘　要：在 SAN 环境下实现了一个开放系统与大型机数据共享系统，克服了传统客户/服务器数据共享模型的缺点，能有效地提高系统的数据传送性能；提出了一种基于共享磁盘映射关系的第三方数据传送方法，由于共享磁盘映射关系是动态建立的，因而系统具有良好的可扩展性；最后讨论了冗余存储路径容错和访问负载均衡功能的实现策略。

关键词：存储区域网络；第三方数据传送；数据共享
中图法分类号：TP316.4　　**文献标识码**：A　　**文章编号**：1001-3695(2005)01-0179-03

A SAN Data Sharing System Based on Third-party Data Transfer

JIA Rui-yong, ZHANG Yan-yuan
(Dept. of Computer Science & Engineering, Northwestern Polytechnical University, Xi'an Shanxi 710072, China)

Abstract: A heterogeneous data sharing system for Storage Area Network (SAN) is presented. A method for dynamically creating the mapping relationship among metadata server is suggested, which shared disks and their storage paths. Third-party data transfer in our system is based on the mapping relationship. More advanced functions such as multiple paths failover and load balance are also implemented by extending the mapping relationship. Because the mapping relationship is created dynamically, the whole system can scale well.

Key words: Storage Area Network; Third-party Data Transfer; Data Sharing

基于 LAN(Local Area Network)的客户/服务器数据共享方案采用的是服务器连接存储(Server-Attached Storage)模型。在该模型中，文件元数据、控制信息和文件数据都要由服务器通过 LAN 转发给客户端，这样不但会消耗大量网络带宽，还会使服务器成为文件传输的瓶颈，尤其是在传输大文件时。

存储区域网络(Storage Area Network, SAN)把存储设备从服务器后端移到了存储网络上，允许计算机通过高速通道(如光纤)对网络共享磁盘直接访问[1]。第三方数据传送[2]方法利用了 SAN 的上述特征，将大量的文件数据流转移到了高速存储网络上，而文件元数据和控制信息仍在 LAN 上传输，从而可以减少对 LAN 带宽和服务器 CPU 资源的消耗，提高数据访问性能。

1　系统背景与体系结构

典型的企业计算环境是包括大型机在内的多服务器平台异构环境。大型机具有高可靠性、高可用性和强大的数据处理能力等优点，承担着关键业务的海量信息处理任务。开放系统具有灵活性强、应用软件丰富和应用广泛的优势，通常作为服务器来提供与客户相关的各种应用服务。实现大型机与开放系统间的数据共享，可以充分发挥两者的优势，高效地整合企业的计算资源。

ROSDFS 系统是一个面向记录访问的 SAN 数据共享系统。它的设计目标是在光纤通道 SAN 环境中实现开放系统对大型机顺序记录文件的高速访问，主要应用于数据仓库、数据分析和在线事务处理等领域。

收稿日期：2004-02-12；修返日期：2004-04-27

ROSDFS 系统的基本设计思想是采用第三方数据传送，分离文件数据和元数据的传输路径。大型机提供文件元数据和控制功能，元数据和控制信息通过 LAN 传输，而数据信息则通过 SAN 直接在开放系统和共享磁盘之间传输。

1.1　硬件构成

ROSDFS 系统的硬件构成如图 1 所示。开放系统与大型机由两个不同的网络互连：一个是控制网络，如 LAN；另一个是 SAN。开放系统与大型机通过网络共享磁盘实现数据共享。

1.2　软件结构

ROSDFS 系统的软件构架包括两个部分：一是运行在大型机上的元数据控制器(MetaData Controller, MDC)；二是运行在开放系统上的客户端模块。

(1) MDC 是一个用户空间进程，其主要功能是存储空间的分配与释放、文件元数据管理、数据锁分配与释放等。

(2) 客户端模块全部在用户空间实现，包括资源管理模块、通信管理模块、系统监视模块以及一个共享磁盘访问函数库，如图 2 虚线框部分所示。

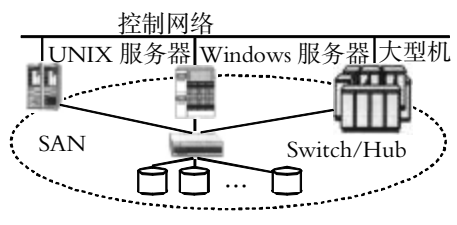

图 1　系统硬件构成图

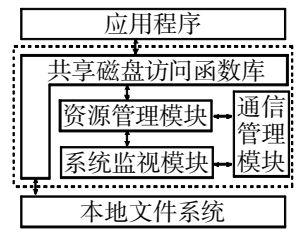

图 2　客户端模块结构

①共享磁盘访问函数库是应用程序访问大型机记录文件的接口，实现了大型机文件系统的面向记录访问语义[3,4]。它

包括两类函数: 访问函数(文件打开和关闭等) 和读写函数(读记录和写记录等) 。访问函数通过进程间通信机制获取或更新资源管理模块缓冲区中的文件元数据。读写函数的实现利用了文件系统提供的原始 I/O(Raw I/O) 接口, 它们直接对网络共享磁盘进行块访问, 并进行逻辑记录与块的相互转换。

②资源管理模块是共享磁盘访问函数库与 MDC 交互的中介, 主要功能是网络共享磁盘的自动识别与管理、处理来自共享磁盘访问函数库的文件元数据操作命令(如文件打开和关闭) , 向 MDC 请求或更新文件元数据, 文件元数据缓冲区管理等。

③系统监视模块的主要功能是监视应用程序和 MDC 的状态。当应用程序崩溃时, 通知资源管理进程清理应用程序占用的资源。当 MDC 崩溃时, 通知资源管理进程阻止应用程序对网络共享磁盘的访问, 从而确保数据的完整性和一致性。

④通信管理模块是其他模块与 MDC 通信的窗口, 负责管理与 MDC 的连接和通信。

2 基于共享磁盘映射关系的第三方数据传送

要实现第三方数据传送, 客户端模块必须知道如下关系:

(1) 共享磁盘与 MDC 的映射关系。不同的共享磁盘可能由不同的 MDC(或 MDC 集群) 来管理。当开放系统上的应用程序要打开或关闭某一共享磁盘上的文件时, 客户端模块必须确定管理该磁盘的 MDC, 向其请求或更新文件元数据信息。

(2) 共享磁盘与存储路径的映射关系。SAN 上的共享磁盘通常是在磁盘阵列上设定的逻辑卷, 磁盘阵列与计算机之间可能有冗余存储路径连接, 当开放系统上的应用程序要对共享磁盘进行读写时, 客户端模块从该磁盘的多条存储路径中选择一条进行访问, 通常选择负载最轻的那条路径。

2.1 共享磁盘映射关系动态建立方法

假设 SAN 上的存储系统是磁盘阵列, 从开放系统上来看, 磁盘阵列上的每一个 LUN(Logical Unit Number) 就是一个网络磁盘。ROSDFS 系统使用的网络共享磁盘就是标记有 ROSDFS Label 的网络磁盘。

由于客户端模块使用文件系统的原始 I/O 接口, 一个网络磁盘可用存储路径(或设备文件名) 来表示。当开放系统与 SAN 之间有多条物理连接时, 一个网络磁盘就有多条存储路径。假设客户端模块运行在 Windows 2000 Server 上, 那么, 一条存储路径就表示为 PhysicalDriveN, N 为 0 或正整数, 以下简称为 PD。

客户端模块在系统启动时建立上述映射关系, 整个过程可分为两个阶段: 第一个阶段建立共享磁盘与 PD 之间的映射关系; 第二个阶段建立共享磁盘与对应 MDC 之间的映射关系。

2.1.1 共享磁盘与 PD 之间映射关系的建立

这一映射由资源管理模块负责建立。映射的结果存放在该模块的一个内部表 PhysicalDriveTable(以下简称 PDT 表) 中。一个简化的 PDT 表项是一个二元组 < 共享磁盘名、PD > 。

该映射建立的主要步骤如下: ① 调用 Win32 API QueryDosDevice( ) 自动检测出所有的 PD。②从①的结果中取出一条 PD。③使用 Win32 API CreateFile( ) 打开该 PD, 判断它代表的磁盘是否是共享磁盘。若不是, 转到②执行, 直到遍历完所有的 PD; 否则, 继续。④在 PDT 表中增加一个 PDT 表项。⑤到②执行, 直到遍历完所有的 PD。

2.1.2 共享磁盘与 MDC 之间映射关系的建立

这一映射是由资源管理模块和 MDC 合作来建立。映射结果存放在资源管理模块的另一个内部表 SharedDiskTable (简称 SDT 表) 中。一个简化的 SDT 表项可表示为一个二元组 < 共享磁盘名、MDC 列表 > 。

该映射建立的主要步骤如下: ①资源管理模块根据 PDT 表生成共享磁盘列表。②资源管理模块向所有的 MDC 分别发送该共享磁盘列表。各 MDC 的地址是预配置参数。③每一个 MDC 接收到来自资源管理模块的共享磁盘列表报文后, 判断其中列出的共享磁盘是否由自己管理; 然后向资源管理模块发送应答报文, 指出哪些共享磁盘属于它管理, 哪些不属于它管理。④资源管理模块分析接收到的应答报文, 对于每一个被 MDC 管理的共享磁盘, 以共享磁盘名为关键字, 在 SDT 表中查找: 若不存在, 则增加一个 SDT 表项; 若存在, 则将该 MDC 加入该 SDT 表项的 MDC 列表中。

SDT 表的大小事先难以确定, 采用大的静态表将造成内存空间的浪费, 而采用线性链表虽然节省空间, 但平均查找长度大。因此, 我们采用了哈希表与线性链表相结合的方法实现 SDT 表。

采用除余法构造哈希函数: Hash(key) = key mod P, Key 为共享磁盘名各字符对应的整数码之和, P 为 7(共享磁盘名的长度加 1) 。哈希表是长度为 P 的指针数组, 每一个指针指向一个由 SDT 表项构成的线性链表。解决冲突的方法如下: 将所有关键字为同义词的 SDT 表项链接在同一个线性链表中。

2.2 第三方数据传送过程

在建立了共享磁盘与对应的存储路径、MDC 之间的映射关系后, 就可以用第三方数据传送的方式进行数据访问了。

假设 Vol 为一个共享磁盘, 文件 File1 在 Vol 上的路径为 /dir1 /dir2 /file1, 那么, 在开放系统上, 采用第三方数据传送访问 File1 的典型过程如下:

(1) 应用程序调用共享磁盘访问函数库中的文件打开函数向资源管理模块发出打开文件 /vol/dir1 /dir2 /file1 的命令。

(2) 资源管理模块根据共享磁盘与 MDC 的映射关系, 向管理该磁盘的 MDC 发送打开文件 /dir1 /dir2 /file1 命令。

(3) MDC 返回该文件的元数据信息(文件逻辑块与磁盘物理块的对应关系等) 给资源管理模块。

(4) 应用程序通过资源管理模块返回的文件描述符对文件 File1 进行读记录操作(调用共享磁盘访问函数库中的函数) 。

(5) 共享磁盘访问函数库根据共享磁盘与存储路径的映射关系, 选择一条存储路径以原始 I/O 方式读取 File1 的物理块, 将其转换为逻辑记录返回应用程序。

(6) 应用程序调用共享磁盘访问函数库中的文件关闭函数向资源管理模块发出关闭文件 /vol/dir1 /dir2 /file1 的命令。

(7) 资源管理模块通知相应的 MDC 关闭该文件。

3 共享磁盘映射关系的扩展应用

利用共享磁盘映射关系还可以实现冗余存储路径容错与

访问负载均衡功能。

3.1　冗余存储路径容错

冗余存储路径容错功能在共享磁盘访问函数库中实现,该库内的记录读写函数调用本地文件系统的原始 I/O 接口函数访问共享磁盘。例如,在 Windows 2000 Server 上,读/写原始磁盘(Raw Disk) 的方法是 ReadFile(PD, …) 或 WriteFile(PD, …) 。由于 PDT 表中包括了一个共享磁盘的所有存储路径信息,当访问一条存储路径失败时,查询对应的 PDT 表项,找到一条冗余的可用路径,可以继续对同一共享磁盘进行访问,从而实现对上层应用程序的透明容错。

3.2　访问负载均衡

在 SDT 表中,增加一个域 Vol_use,记录每一个共享磁盘上正在被访问的文件数。那么,每一个 MDC 上所打开的文件数,记为 Host_use,可表示为

$$host\_use = \sum_{i=1}^{n} vol\_use[i]$$

其中 vol_use[i] 是一个 MDC 所管理的第 i 个共享磁盘上正在被访问的文件数。

Host_use 表示了一个 MDC 的负载状况,客户端模块在发送文件打开请求时,可以根据 MDC 的集群信息(SDT 表中的 MDC 列表) ,选择一个 Host_use 数最小的 MDC 来服务该请求,从而实现 MDC 访问负载均衡。

4　结论

基于 SAN 的多平台数据共享具有很大的理论价值和广阔实用前景,在国外已成为研究热点[5,6],国内在这方面的研究成果还不多见。

以一个国际合作项目为背景,本文实现了一个 SAN 环境下的开放系统与大型机数据共享系统。该系统利用 SAN 的直接网络存储访问特征,采用第三方数据传送方法,克服了传统客户/服务器数据共享模型的缺点,提高了系统的数据传送性能。通过动态建立共享磁盘映射关系,实现了第三方数据传送,也使系统具有良好的可扩展性。通过对共享磁盘映射关系作进一步扩展,实现了系统的冗余存储路径容错和访问负载平衡等功能。

参考文献:

[1] Clark. Designing Storage Area Networks [M]. Boston: Addison-Wesley, 1999. 3-9.

[2] H Hulen, O Graf, K Fitzgerald, *et al*. Storage Area Network and the High Performance Storage System[C]. 10th NASA Goddard Confe～rence on Mass Storage Systems, 2002.

[3] Richard A Demmers. Distributed File for SAA [J]. IBM Systems Journal, 1988, 27(3): 348-361.

[4] Jay Shah. VAX/VMS: Concepts and Facilities [M]. New York: McGraw Hill, 1991. 12-20.

[5] IBM Corporation. IBM TotalStorage SAN File System Draft Protocol Specification [EB/OL]. http://www-5.ibm.com/storage/europe/uk/software/ virtualization/sfs/protocol.pdf, 2003-04-30.

[6] M Bancroft. Functionality and Performance Evaluation of File Systems for Storage Area Networks (SAN) [C]. Proceedings 17th IEEE Symposium on Mass Storage Systems/8 NASA Goddard Conference on Mass Storage Systems and Technologies, IEEE Computer Society Press, April 1999. 125-126.

作者简介:

贾瑞勇(1976-),男,博士生,主要研究方向为存储区域网络、分布式系统和软件工程;张延园(1954-),教授,主要研究方向为存储区域网络、软件工程。

(上接第 168 页)

```
GoTo hanErr
End If
…
hanErr:
If ErrDescription < > "" Then
   Err. Description = ErrDescription
   Err. Source = "vcDllFunction"
   Err. Number = ErrNumber
End If
Err. Raise Err. Number, Err. Source, Err. Description
```

在这段代码中,有可能出现两类运行时异常,一类是由 VB 程序本身所产生的,另一类是由 V C++编写的函数所产生的,通过判断 ErrDescription 是否为空,我们可以确定运行异常的来源,然后把运行异常的信息向上传递,交给统一的异常处理函数,这样就可以统一不同程序设计语言的异常处理机制。

在 VB 和 MATLAB 混合编程时,可能出现在实数之间的计算结果有复数的现象,如求实数矩阵的特征根和特征向量,这时可以通过判断虚数部分是否存在来确定运行时异常,通过扩展 VB 的错误代码表来描述运行异常。

4　结束语

针对混合编程时容易被忽略的几个问题,本文分析了它们产生的原因,并给出了相应的解决方法,在 FORSTAT 的设计和开发中得到了实际的应用,对于混合编程有一定的借鉴作用和指导意义。

参考文献:

[1] 周振红,颜国红,吴虹娟. Fortran 与 Visual C++混合编程研究[J]. 武汉大学学报(工学版),2001,34(2):84-86.

[2] 胡春生,秦石乔,王省书. C++ Builder和 Fortran PowerStation 的一种混合编程方法[J]. 计算机应用研究,2001,18(7):149-150.

[3] 朱国强,刘勇,洪嘉振. 32 位操作系统下的混合编程[J]. 计算机应用研究,2000,17(5):58-61.

[4] 张学胜. 用 VB 和 Fortran 混合编程开发科学计算软件[J]. 计算机应用,2003,23(S1):12-13.

[5] 常斌,李宁,黑新宏. 工程计算软件开发中 Fortran 混合编程的关键问题分析[J]. 计算机应用,2003,23(S1):56-58.

[6] 夏舒杰,谭建荣,陈洪亮,等. 基于文件操作的VC++和 Fortran 模块交互通信方法[J]. 计算机工程,2003,29(9):63-65.

[7] 付红勋. 与VC++混合编程时的几点 VB 编程经验[J]. 计算机应用研究,2001,18(6):140-141.

作者简介:

李海奎(1965-),男,副研究员,博士后,主要研究方向为数值计算、组件技术;郎璞玫(1970-),女,副研究员,博士,主要研究方向为信息技术在森林经理中的应用。