一种新的网络对象存储设备研究

张悠慧,郑纬民
(清华大学计算机系,北京 100084)

摘 要: Internet 的飞速发展对于存储系统的可扩展性提出了很高的要求,也带来了由数据模型与存储模型的不一致问题引起的服务器性能瓶颈现象.针对这些情况,本文提出了网络对象附属存储设备(NAOSD)的概念,其利用设备处理器的能力直接支持结构化数据存储.这一设计减少了存储系统中数据服务器的负载,增加了系统吞吐量.同时,研究了该设备原型在集群环境中的应用,提出了数据/元数据统一存储与查询式数据定位机制.分析表明,这些机制能够较显著地提高系统扩展性——数据访问时间随系统规模的扩大呈对数增长,优于传统的映射定位机制.我们已经模拟实现了 NAOSD,并在性能比较测试中取得了较好的效果.

关键词: 对象存储设备; 集群存储; 定位机制; 网络附属存储
中图分类号: TP391 **文献标识码**: A **文章编号**: 0372-2112 (2003) 05-0679-04

Research on a New Type of Network-Attached Object Based Storage Device

ZHANG You-hui, ZHENG Wei-min
(Department of Computer Science and Technology, Tsinghua Univ, Beijing 100084, China)

Abstract: With the ever-increasing development of Internet, today's data-centric applications require storage platforms to possess high scalability. It causes a data-model-mismatch problem between applications and current storage systems. This paper designs a new network-attached object-based storage device (NAOSD), which provides storage interfaces for structured data to solve this problem. The usage of NAOSD in cluster storage environments is proposed, which introduces the uniform storage of data and meta-data as well as query-locating mechanism. The performance analysis shows that its access time increases with the system scale logarithmically, which is better than the conventional systems. The simulator of NAOSD have been implemented, and experiments show it improves the performance remarkably.

Key words: object based storage device; cluster storage; locating mechanism; Network-attached storage

1 引言

随着 Internet 的发展,Internet 信息服务所存储的数据量呈爆炸式增长.这些服务所管理的数据,具有一定的规则结构,不属于通用的数据类型,被称为结构化数据.对这些数据,往往进行比较简单的保存、读取、查询小数据集等操作,不适合传统的文件系统与数据库来保存数据.

为解决这些问题,许多相关研究机构与公司提出实现结构化数据存储平台这一思想,包括以下几个典型的研究项目:

Lore[1]是 Stanford 大学研发的 XML 数据管理系统.

见 Porcupine 中的 Transactional Record Store (TRS)[2]. Porcupine 是华盛顿大学研发的超大规模的集群 email 系统,TRS 是其中的单结点存储子模块.

DDS[3]是加州大学 Berkeley 分校的 Ninja 计划的一部分,是一个面向 Internet 服务的基于集群的存储层.

较早期的项目还有 Shore[4]等.但这些存储系统的体系结构仍属于传统的块设备——文件系统——应用接口这一模式,这样底层存储设备所支持的数据模型与应用数据模型的不匹配而带来的"服务器瓶颈问题[5]"依然存在.则根据存储设备智能化发展[6,7]的趋势,设计了一种新的网络对象附属存储设备——Network Attached Object based Storage Device (NAOSD);利用内部 CPU 的处理能力来直接支持结构化数据的存储以减少服务器的工作负载,并提高了系统可用性.我们研究了 NAOSD 在集群存储环境中的应用,提出了数据/元数据统一存储机制与查询式数据定位与访问机制.通过 Open Queuing Network 模型对系统可扩展性进行了分析,表明这两个机制提供了较高的扩展性,其数据访问时间随系统规模扩大呈对数增长,要优于传统的映射定位机制.

2 网络对象存储设备——NAOSD

芯片性能的突飞猛进,为设计功能更强的存储设备奠定了硬件基础.此类存储设备与传统块设备的最大差别在于前

收稿日期:2002-06-10;修回日期:2002-12-05
基金项目:国家自然科学基金(No. 60203023);国家 863 高科技发展计划资助项目(No. 2002AA1Z2103)

者提供了更为高级的数据抽象与访问接口. CMU 的 NASD[8], HP 的 Attribute based Storage[9]是这方面的典型研究.

我们借鉴了这些设计思想,更进一步提出了直接支持结构化数据的网络附属对象存储设备的概念.

2.1 设备结构

网络附属对象存储设备充分利用设备处理器能力,提供结构化数据存储接口,在内部实现非定长结构化数据的物理定位与存取,还可以分析相关的数据属性——实际上是将传统的对象存储服务器上的部分负载移至存储设备.

对象设备的内部结构设计如图 1. 其与 NASD 设备一样,集成了磁盘控制器与网络接口;关键是增加了结构数据管理模块来负责将结构化数据的读写转换为磁盘控制器的块数据读写命令,在此基础上实现了对数据进行解析与查询的功能模块.

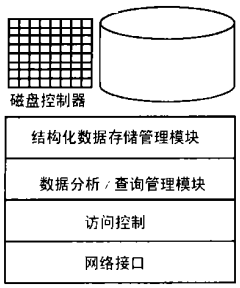

图 1 网络附属对象存储设备结构示意图

目前许多数据处理应用的特点是处理的数据量远小于最终结果的数据量,而且需要对数据进行分析. 如果在存储服务器直接处理原始数据之前,先在各个设备上进行数据预处理以降低数据传输量,这可能会带来整体性能的提高:因为其大大降低了数据传输开销,同时将数据处理并行化.

表 1 服务器处理能力与所配置的存储设备的处理能力比较

系统名称	服务器的处理器主频	服务器处理能力	磁盘个数	磁盘处理能力	I/b
Compaq TPC-C	4 *400MHz	1600MHz	141	3525MHz	2.2
Microsoft TerraServer	8 *440MHz	3521MHz	324	8100MHz	2.3
Digital TPC-C	1 *500MHz	500MHz	61	1525MHz	3.0
Digital TPC-D 300	12 *612MHz	7344MHz	521	13025MHz	1.3

表 1 是各种较大规模数据存储系统的服务器处理能力与所配置的存储设备处理能力的比较. 这些情况下,对象设备的处理能力之和均能够超过服务器的处理能力. 现在服务器的 CPU 主频与存储设备芯片主频的比例大概是 20:1～50:1,随着 IO 处理芯片的发展与网络存储设备的独立化,这一比例会越来越小,而利用 NAOSD 的优越性可能会随之增大.

2.2 设备的数据组织与接口

NAOSD 内部的存储单位是数据对象——称为 OItem, OItem 可以被独立地存取、查询、复制以及删除等.

OItem 由对象标识 OID、对象本身的数据 OValue 与对象类型 OType 这一个三元组来表示,结构如图 2.

OID 是对象的物理标识,包括 OItem 所存储的物理结点标识符——BID,即每个 NAOSD 设备都具有的唯一标识符;OType 标识符——CLID,是 OItem 的对象数据类型的标识. 最后 48 位是用于标识本地数据对象的序列号. 其中,OType 是一种特殊的数据对象——即数据类型本身也由此三元组构成,只是其数据类型为全 0.

当然,我们必须注意到 OID 是系统内部的标识,与应用程序所见到的名字是不一样的. 为此引入了位于存储服务器上的名字映射机制,具体的命名过程是这样的:(1) 存储服务器初始化,取得当前可用的 Brick 信息. (2) 存储服务器向最近的若干个 Brick 申请一段未用的本地序列号. 最近指的是本地 Brick(如果有的话),及集群中负载较轻的几个结点. (3) 存储服务器接收创建对象请求,根据请求提供的 CLID、最近的 Brick 标识及相应本地序列号构造 OID. (4) 进行 Brick 数据写入. (5) 写成功后,将用户提供的对象数据名与 OID 存于本地的非易失性的名字映射表内. (6) 如果最近 Brick 提供的本地序列号用尽,则重新请求未用的本地序列号. 转(3). 利用本身的处理能力,NAOSD 能够根据对象的类型描述数据来解析对象内容本身,取得各个域的值,并进行比较等操作.

3 基于 NAOSD 的集群存储

与一般的对象存储设备类似,NAOSD 通过服务器对外提供存储服务,其在集群环境下的应用模型如图 3.

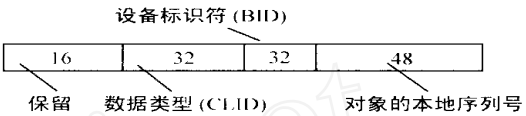

图 2 OID 的结构

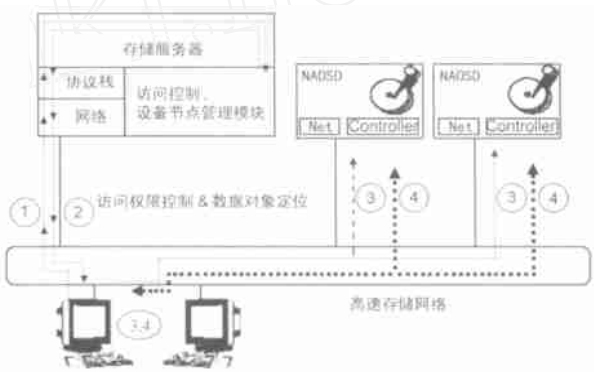

图 3 NAOSD 的集群应用模型

首先客户端向存储服务器发送对象数据的访问权限认证①;由后者判断后发回访问许可②;接着通过我们提出的查询式定位机制进行欲访问数据的定位操作③及数据访问操作④. 可以看到,存储服务器的功能主要集中于访问的权限控制,而由设备自主地完成数据定位与传送.

3.1 数据统一存储机制

NAOSD 实现了对于结构化数据的直接存储支持,这样就无需像传统的存储体系那样通过服务器上实现数据的定位与存储管理. 利用这一点,我们提出对象元数据/数据的统一存储. 这里的元数据包括两类:

® 与文件系统中的类似,如数据名称、创建与访问时间以及存放位置. NAOSD 将其归结为对象的属性,即每个 OValue 的前几个域是固定的,分别表示该对象数据的名字(字符串)、属主、创建时间等,而位置信息直接体现在 OID 的 BID 上.

® 对象数据的类型描述,即 OType. 系统处理类型描述的方式与处理数据本身一样. 这样,在基于 NAOSD 的系统中已

不存在传统意义上的元数据,所有的数据,包括类型描述、对象数据及属性,都被同样对待.每个 NAOSD 可以完全独立地处理本地数,这带来了两个优点:

® 首先减少了网络通信量,及与数据定位相关的处理环节,可以增加系统扩展性;

® 其次增强了数据可用性 ——其只与设备有关.

3.2 查询式定位

与对象数据统一存储相适应的是查询式数据定位机制,NAOSD 提供了根据 CLID 的助记符,或者根据相应的数据类型内部某个(些)域的条件,通过查询操作取得数据对象集合的功能.查询条件由多个 Filter,即最基本的数据过滤条件,以"与或"关系符连接而成,一个 Filter 的形式为:

OItem. OType = "某 CLID 的助记符" && OItem. OType FieldName. < > = "某值";

即取得属于特定对象类型的,且某个域值满足某条件的所有对象实例;查询返回的是符合条件的全部对象实例的 OID 集合.将这种查询操作分布到各个 NAOSD 设备上,可以减少结果数据的传输量,这方面的性能分析在 2.2 节中给出.

4 基于 Open Queuing Network 模型的扩展性分析

我们给出了基于 NAOSD 的数据统一存储与查询式定位的可扩展模型.系统中有 N 个存储服务器,与 n 个 NAOSD 设备,以高速网络连接,单个存储服务器与设备的处理能力之比为 m,一般 $m > 1$.每个设备上存储有总量固定的对象数据.对于定位对象用的元数据,有两种分布方式:一是与对象数据统一管理,即分布于 NAOSD 设备上;二是传统的管理方法,即与数据相分离,均匀分布于存储服务器上.

利用已被证明对大规模存储系统的性能分析相当有效的 Open Queuing Network 模型[11],可以给出两种方式下访问数据的可扩展模型.传统映射方式的数据定位步骤如图 4.查询式定位步骤如图 5.

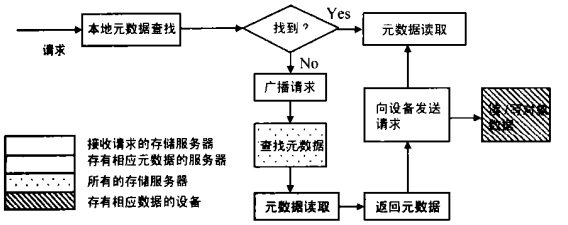

图 4　元数据与数据分离的映射方式数据定位模型

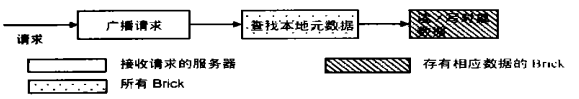

图 5　NAOSD 的查询式数据定位模型

设 T_{lookup} 为 NAOSD 查询本地元数据的时间,与对象数据的个数成正比,而与结点的处理能力成反比.这样存储服务器上查询本地元数据的时间为 $T_{lookup} * \frac{n}{N * m}$;

T_{lookup} 为读/写本地数据的时间,假设读/写元数据与读/写对象数据的用时相同;T_{remote} 为存储服务器之间,存储服务器与 NAOSD 间的点对点通信时间;$T_{ip\ broadcast}$ 为存储服务器间的网络广播用时;$T_{lb\ broadcast}$ 为存储服务器向 NAOSD 进行网络广播的用时;

这样,以传统方式访问时,有:

$$T_{traditiond} = T_{lookup} * \frac{n}{N * m} + T_{access} * \frac{1}{N} + (T_{ip\ broadcast} + T_{lookup} * \frac{n}{N * m} + T_{remote} + T_{remote}) * \frac{N-1}{N} + (T_{remote} + T_{remote})$$

第一项为 NAOSD 的查找本地元数据时间,第二项则是读取元数据时间(元数据在本地的概率为 1/ N),而下一项则是在未找到的前提下,向其余的 NAOSD 转发请求的开销;最后是向存有对象数据的 NAOSD 发送请求与处理的时间,有:

$$T_{traditional} = \frac{2nN - n}{m * N^2} * T_{lookup} + 2 * T_{access} + \frac{N-1}{N} * T_{ip\ broadcast} + \frac{2N-1}{N} + T_{remote};$$

而以查询方式访问时,有:

$$T_{naosd} = T_{lb\ broadcast} + T_{lookup} + T_{access};$$

第一项为 NAOSD 以广播方式向 NAOSD 发送请求的开销,第二、三项分别是各个 NAOSD 进行本地元数据查询与读取数据的时间开销.

图中的通信是小数据量操作,为讨论简单起见,可以认为 T_{remote} 是常数;同时使用性能较好的 binomial tree 算法[11]作为广播通信算法:

这样 $T_{ip\ broadcast} = (\lfloor \log_2(N-1) \rfloor + 1) * T_{remote}$;

$T_{lb\ broadcast} = (\lfloor \log_2(n-1) \rfloor + 1) * T_{remote}$;

设 $T_{remote} = 5$, $T_{lookup} = 1$, $T_{access} = 10$(时间单位), $N = 5$, $m = 5$(即单个存储服务器与设备的处理能力之比为 5);有:

$T_{traditional} = 9n/125 + 20 + 0.8 * 5 * 3 + 9 = 9n/125 + 41$;

$T_{naosd} = (\lfloor \log_2(n-1) \rfloor * 5) + 16$

图 6 给出了 $T_{traditional}$ 与 T_{naosd} 的函数图,可以看出,随着 n 的扩大,$T_{traditional}$ 是呈线性增长,而 T_{naosd} 则以对数形式增长.所以,查询式定位的可扩展性要优于传统的分离元数据管理、及采用映射机制作为定位机制的方法.

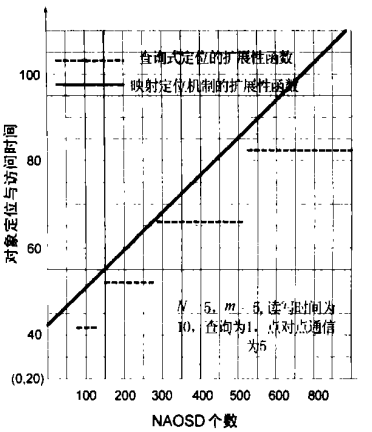

图 6　查询式定位与传统映射定位方式的扩展性

5 性能测试

我们在 PC 机上的 Linux 系统中模拟实现了 NAOSD,其在文件系统之上提供了结构化数据存储与查询接口,分别就启用 NAOSD 与否进行数据查询的性能比较.测试环境如下:一台 PC 服务器,带有四个 P Ⅲ700MHz 处理器与 1G 内存,运行

应用服务器;NAOSD 由较为低档的 PC 机模拟,共六台,主频为 500MHz,各带有一个 SCSI 硬盘,容量 10G;全部由百兆以太网(交换型)互连.

低档 PC 上运行模拟设备程序,该程序向外提供数据对象的插入与查询接口,在内部实现了对于数据对象的文件存取.当然,模拟设备程序也可以读取全部的对象,不加筛选地传送给应用服务器,这就是不启用 NAOSD 时所作的数据读取操作.为测试简单起见,数据对象的结构是固定的,如图 7,其中域 Others 用于调节数据记录的大小,而 *ID* 则标识数据对象,用于查询.应用服务程序可以向这 6 台发送数据对象插入与查询请求.

```
Class TestData{
Int ID;  String Name
String Addr; Int age
String others,}
```

图 7 测试程序中数据对象的结构

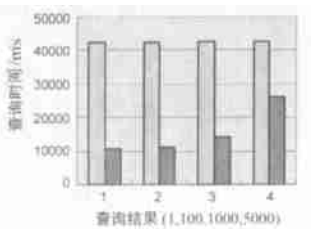

图8 基于对象设备的数据查询性能比较(对象大小:1024字节,人数:1000) ■ 一般存储设备; ■ 对象设备

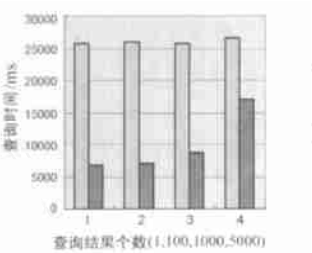

图9 基于对象设备的数据查询性能比较(对象大小:512字节,人数:10000) ■ 一般存储设备; ■ 对象设备

接着,收应用服务程序进行数据查询,查询的表达式是 $ID < N$; N 取不同值以控制结果集的个数.在不同数据对象长度与结果集个数、及启用或不启用 NAOSD 的情况下,我们记录了应用服务程序完成此查询的时间.

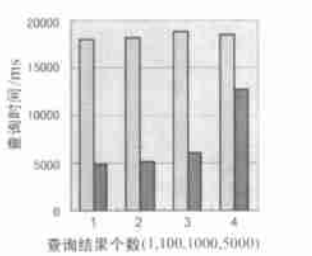

图10 基于对象设备的数据查询性能比较(对象大小:256字节,人数:10000) ■ 一般存储设备; ■ 对象设备

单个数据对象长度为 1024byte,如图 8,在结果集与原始数据集的数据个数比越小的情况下,NAOSD 的优势就越明显,最大能达到约 3.95 :1.单个数据对象长度为 512byte,如图 9,性能比分别为:3.78,3.69,2.97,1.55.

单个数据对象长度为 256byte,如图 10,性能比分别为:3.72,3.53,3.08,1.45.本试验中,六台 PC 机的处理能力总和与服务器近似相同,但其性能比均明显地大于 1.在后两种情况下,NAOSD 的优势要小一些,原因在于其数据量较小,使得网络传输这一性能瓶颈的影响不太显著.但从整体来说,使用 NAOSD 的优势还是明显的.

参考文献:

[1] R Goldman, J McHugh, J Widom. From semistructured data to XML: migrating the lore data model and query language [A]. Procedings of the 2nd International Workshop on the Web and Databases [C]. Philadelphia, USA: ACM Press, June, 1999. 100 - 105.

[2] Robert Grimm, Michael M Swift, Henry M. Levy. Revisiting structured storage: A transactional record store [R]. Technical Report UW-CSE-00-04-01, USA: University of Washington, Department of Computer Science and Engineering, April 2000.

[3] Steven D Gribble Scalable, Distributed data structures for internet service construction [A]. Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000) [C]. San Diego, USA: USENIX Press, October 2000. 181 - 190.

[4] M J Carey, et al. Shoring up persistent applications [A]. Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data [C]. Minneapolis, USA: ACM Press, May, 1994. 383 - 394.

[5] Garth A Gibson, David F Nagle, William Courtright II, et al. NASD scalable storage systems [A]. Proceedings of USENIX99 Extreme Linux Workshop. Monterey [C]. USA: USENIX Press, June 1999. 28 - 35.

[6] Gregory R Ganger. Blurring the Line Between OSes and Storage Devices [R]. CMU SCS Technical Report CMU-CS-01 - 166, December 2001. Available from http://www.ece.cmu.edu/ ganger/papers/recent.html.

[7] Christoforos E Kozyrakis, David A Patterson. A new direction in computer architecture research [J]. IEEE Computer, November 1998: 24 - 32.

[8] Garth Gibson. File server scaling with network-attached secure disks [A]. Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. Seattle [C]. USA: ACM Press, June 1997. 35 - 54.

[9] Elizabeth Shriver. A Formalization of the Attribute Mapping Problem [R]. USA: HP Labs Technical Reports, HPL-1999-127.

[10] Odysseas I. Pentakalos, Daniel A. Menasc é, Yelena Yesha. Pythia and Pythia/WK: Tools for the performance analysis of mass storage systems [J]. Software-Practice and Experience, 1997, 27(9): 1035 - 1054.

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest. Introduction to Algorithms [M]. MIT Press, 1990.

作者简介:

张悠慧 男,1974 年生于浙江宁波,博士,讲师,主要研究领域为网络存储、并行处理.

郑纬民 男,1946 年生于浙江宁波,教授,博士生导师,主要研究领域为并行处理、集群系统、网络存储、生物计算等.