# 基于对象存储系统的对象文件系统设计

冯 丹 史 伟 覃灵军 关 卿

(华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

摘要: 基于对象文件系统是建立在对象存储系统上的一种应用, 它利用对象的特点对数据进行存储和管理. 基于对象文件系统由客户端、元数据服务器和基于对象存储设备组成, 通过对系统的各部分进行分析, 设计并实现了基于对象文件系统. 通过对不同文件大小、不同文件分块大小条件下文件系统的性能进行测试分析, 找出系统的数据传输瓶颈, 使用以对象属性为导向的缓存和预取技术以及聚合读写的方法对文件系统进行了优化. 优化后读写吞吐率分别提高了 60 Mbyte/s 和 40 Mbyte/s.

关 键 词: 对象存储; 基于对象文件系统; 预取与缓存; 聚合读写
中图分类号: TP333 文献标识码: A 文章编号: 1672-4512(2006)12-0047-03

## File system design for object-based storage system

Feng Dan Shi Wei Qin Lingjun Guan Qing
(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: On the basis of object-based storage system (OBSS), object-based file system (OBFS), consisting of client sides, Metadata Server and object-based storage device store and manage data by using the characteristics of objects. This system was designed and realized by analyzing the compositions and interaction modes of each part in the system. The data transfer bottleneck in the system was determined after testing and analyzing the performance of the system under the different file sizes and file stripping sizes. Through cache, prefetch and aggregated Read/Write technology which is based on object attributes, the performance of OBFS is improved. The read and write throughput is increased by 60 Mbyte/s and 40 Mbyte/s respectively.

Key words: object-based storage; object-based file system; prefetch and buffer; aggregated read/write

## 1 对象存储系统

基于对象存储系统 OBSS 由高速网络将用户、元数据服务器(Metadata Server, MDS)和对象存储设备(Object-based Storage Device, OBSD)连接起来. MDS 提供全局名字空间, 管理文件到对象的映射, 提供身份验证等安全机制. OBSD 则向外提供对象接口, 以对象作为基本存取单元.

用户访问文件时先从元数据服务器获取文件的元数据信息(如文件由哪些对象组成以及对象所在的设备等)及访问证书, 然后用户直接对 OBSD 进行数据存取. OBSD 收到用户请求后, 对其进行身份验证, 然后进行对象读写. 用户向 OBSD 发送的 I/O 请求包括对象 ID、对象的偏移地址及长度. OBSD 包括 CPU, Memory, 网络接口及块设备, 管理对象存储空间的分配、数据组织及对象的属性. OBSS 的优点之一就是将底层的数据组织和同步操作交由 OBSD 进行, 大大减轻

收稿日期: 2005-11-30.
作者简介: 冯 丹(1970-), 女, 教授; 武汉, 华中科技大学计算机科学与技术学院 (430074).
E-mail: dfeng@hust.edu.cn
基金项目: 国家重点基础研究发展规划资助项目 (2004CB318201).

了用户端和元数据服务器的负担, 同时也提高了整个系统的并行性和可扩展性[1].

## 2 基于对象文件系统

基于对象文件系统(Object2Based File System, OBFS)是运行在 OBSS 各部分之上的一个分布式文件系统, 如图 1 所示. OBFS 提供 Windows 和 Linux 两种环境下的客户端. 客户端与

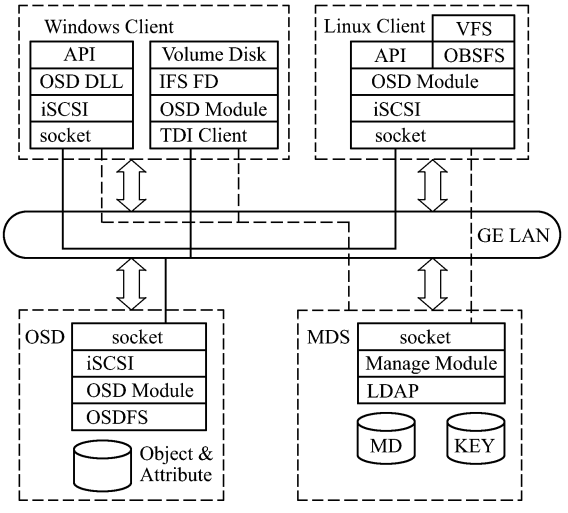


图 1 基于对象文件系统体系结构

OBSD 之间的数据传输路径在图中用实线表示, 图中虚线是客户端、OBSD, MDS 之间的进行身份认证、参数协商等操作时的数据传输路径.

### 2.1 元数据服务器端

MDS 负责维护文件到对象的映射关系, 并实时掌握 OBSD 的各种信息(包括负载、可利用空间等), 实现负载平衡. MDS 还负责管理用户的身份验证和访问权限. MDS 对权限的管理以对象为基本单元, 粒度介于块和文件之间, 在易管理性和安全性上做了很好的折中. MDS 使用 LDAP(Lightweight Directory Access Protocol, LDAP)为每个用户维护一张文件与对象的对应表. 一个文件可分为多个对象, 多个文件也可聚合在一个对象中.

### 2.2 对象存储设备端

对象存储设备对外提供基于对象的访问接口, 负责管理磁盘的数据组织与空间的管理. 以读操作为例, iSCSI 模块负责接收网络上传来的数据, 并对接收到的 iSCSI 数据包进行解析, 解析后交由 OSD Module 进行处理, 产生对象操作命令由基于对象存储设备端文件系统(Object2based Storage Device File System, OSDFS)进行处理.

OSDFS 是 OBFS 中的重要组成部分, 负责对象数据及其属性在磁盘上的存放[2]. 通用的文件系统(如 Ext2)对大文件的支持不够理想, 为此 OSDFS 在 Ext2 基础上进行了改进, 将对象按照大小分为两种: 小对象(4 Kbyte 左右)和大对象(512 Kbyte 左右). 小对象仍使用 Ext2 原有的方式进行存放, 而大对象在存放时申请一个连续的磁盘空间.

由对象 ID 查找 inode 节点号的流程如图 2 所示. 为了保证读写的高性能和磁盘空间的较高

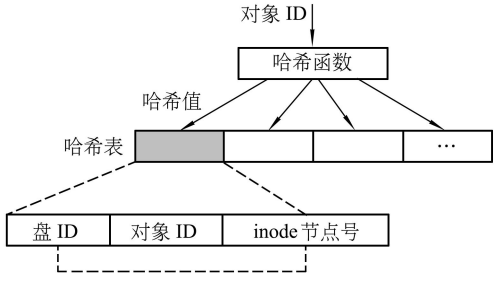


图 2 由对象 ID 查找 inode 节点号流程图

利用率, OSDFS 将磁盘空间按域管理. 每个域内的数据分块大小一定, 而各个域分块大小可以不同. 由上述分析, 域中数据分块大小分为两种, 4 Kbyte和 512 Kbyte. 各种长度的对象在合适的域中存放. 其结构如图 3 所示. 对象存放可以跨

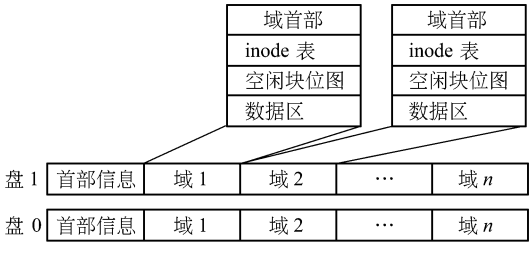


图 3 磁盘中域的结构图

域, 甚至可以跨盘. 这使得对象的并行读取成为可能. 创建对象时, 首先根据对象大小找到一个合适的域, 并根据域的空闲位图为对象申请空间. 域的申请和回收非常灵活. 当对象没有合适的域存放时, 系统可以申请一个新域; 如果域中所有对象被删除, 这个域也随即从系统中删除.

### 2.3 客户端

Windows 客户端提供两种接口: API 和虚拟逻辑盘. API 是一套根据 OSD 命令集规定的标准的基于对象的命令接口. 用户调用这些 API 后, 由 OSD DLL 模块对其进行解析, 并交由 iSCSI 模块进行发送. Windows 客户端还可以向用户提供一个虚拟逻辑盘, 用户可以像访问本地磁盘一样对其进行格式化和各种读写操作. 此逻辑盘是由一个 Windows 可安装文件系统的过滤驱动程序(Windows Installed File System Filter Driver,

IFS FD)实现[3]. 以读文件为例说明, 当用户读取虚拟磁盘中的某个文件时, IFS FD 模块就能得到此操作的 IRP( I/ O Request Package) . 而 IRP 中包含有对文件进行何种操作(读、写、重命名等)、文件名、读写数据的大小、偏移等信息. IFS FD 将此命令重定位给 OSD Module 模块. OSD Module 模块按照 OSD 命令集的要求进行重新封装, 通过 TDI Client 模块向元数据服务器请求认证信息及文件元数据(文件所在的 OBSD 设备号, 以及文件对应的对象号和偏移地址) . 之后客户端直接通过 TDI Client 向 OBSD 设备发送读对象命令读取相关的对象并附带元数据服务器传来的认证证书. OBSD 对客户端的请求进行认证, 并读取对象数据返回. 在客户端, 由 IFS FD 进行数据重组, 并返回给用户.

Linux 环境下的客户端和 Windows 环境下的类似, 向用户提供两种访问接口: API 和一个目录. API 接口的实现方式和 Windows 的基本类似. 与 Windows 不同的是, Linux 客户端在 VFS 注册一个新的文件系统 OBSFS. 用户可以使用 Mount 命令将此文件系统挂接到某个目录下. 系统对此目录的访问将被 OBSFS 进行重定位. 在 VFS 中, 用户对文件的访问是以页为单位, 通过结构体 struct page 与底层的各种文件系统进行交互. 而此结构体中包含有此文件的 inode、读写操作的长度和偏移等信息. OBSFS 将这些信息交给 OSD Module 模块进行处理, OSD Module 模块将这些信息封装成对象操作命令, 交由底层的 iSCSI 和 socket 模块发送到网络上.

## 3　性能与优化

OBFS 文件系统的测试环境如下: OBSD、客户端及 MDS 均采用相同配置的 PC 机(CPU: Intel P4 2. 4 GHz, 内存 512 Mbyte, 磁盘 ATA 120 Gyte 2 Mbyte 缓存), 通过 Cisco Catalyst 3750 series 千兆以太网交换机连接. OBFS 中的所有模块均在核态下实现, 对 OBFS 的性能测试是利用 iozone 在 Linux 环境进行的, 图 4( a) 和( b) 分别显示了 OBFS 在采用不同文件大小 f、不同文件分块大小 b 条件下的读速率 r 和写速率 w. 总体上来讲, 性能不够理想. 为此使用两种方法进行了优化: a. OBSFS 缓存与预取. 由于 OBSD 设备本身具有 CPU 和内存, 可将用户最近访问的对象缓存, 用 LRU 算法进行替换. 另外, OBSD 还可以根据用户的访问特征及对象的属性预取对象. 这里的预取和缓存以对象为基本单元. b. OBSFS 聚合读写. VFS 以 4 Kbyte 的页为单位读写文件, 故 OBSFS 每次都将此页中的数据发往网络, 这就使得系统频繁地进行数据包的封装和内存拷贝. 网络上传输较小的数据包并不能充分利用千兆以太网的带宽. 为此, 实现了 OBSFS 的聚合读写, 将小数据块的读写操作进行聚合, 当操作的数据块达到一定大小时再进行数据封装发往网络. 测试发现, 每次发送 64 Kbyte 的数据基本上能充分利用千兆以太网的带宽.

经过优化, 在同样的环境下对 OBFS 的性能进行了测试, 结果如图 4( c) 和( d) 所示. 可以看出 OBFS 的性能有很大改善, 优化后平均读性能提高了 60 Mbyte/ s, 平均写性能提高了 40 Mbyte/ s. 由于使用了聚合读写技术, 对小文件的读写性能提高, 加上 OSDFS 对大文件的读写时采用分配连续空间的方法, 因此使得大文件的读写性能保持在一个较高的水平上.

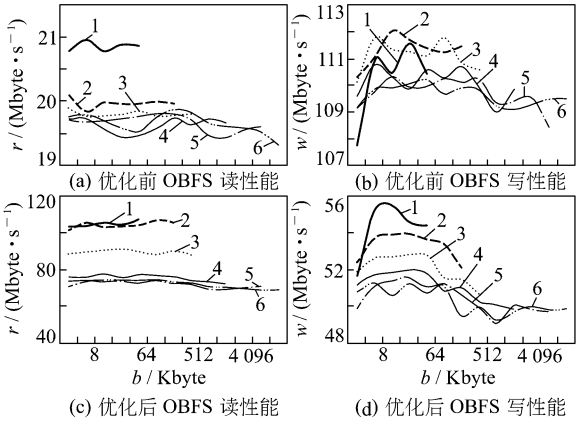


图 4　OBFS 读写性能

f / Kbyte: 1) 64, 2) 256, 3) 512, 4) 2 048, 5) 8 192, 6) 16 384

参　考　文　献

[1] Mesnier M, Ganger G R, Riedel E. Object2based Storage[J]. IEEE Communications Magazine, 2003, 41(8): 84290.

[2] Wang Feng. OBFS: a file system for object2based storage devices[C] M The 21nd IEEE/ 12th NASA Goddard Conference on Mass Storage Systems and Technologies: IEEE Computer Society, 2004: 8214.

[3] Rajeev Nagar. Windows NT file system internals [M]. Sebastopol, CA: OcREILLY, 2000.