高技术通讯 2009 年 第 19 卷 第 3 期:224 ~ 229

doi:10.3772/j.issn.1002-0470.2009.03.002

在 Linux 内实现基于内容存储的驱动器(CASD)①

鞠大鹏②** 刘川意* 汪东升*** 刘 宏*** 汤志忠*

(*清华大学计算机科学与技术系,北京 100084)
(**清华大学信息技术研究院,北京 100084)
(***清华大学威视数据安全研究所,北京 100084)

摘 要 针对目前基于内容存储(CAS)系统原型绝大多数因建立在已有的各种文件系统上而使其性能和应用范围受到限制的情况,设计和实现了一种在 Linux 内核中用作块设备驱动器的 CAS 驱动器(CASD),它可作为 Linux 的基本构件组成各种 CAS 系统以支持不同应用,并且可以用作 Linux 支持的单机存储系统和客户-服务器网络或对等网络存储系统。论述了此 CASD 主要功能部件的设计选择、在 Linux 内的实现方法及其性能评价。在 iSCSI 存储网络上的性能测试结果表明,此 CASD 的写性能比 Ext2 更高,且读性能相近。进一步提高其性能瓶颈在于对文件内容的 Hash 计算和通过 Hash 索引查找对象元数据这两个环节。

关键词 存储技术,基于内容存储(CAS),Linux 块设备驱动器,CAS 驱动器(CASD),性能评价

0 引 言

作为一种面向对象的存储设备,基于内容存储(content aware storage,CAS)器使用文件内容的哈希(Hash)值来存储和检索文件。由于不同数据块的 Hash 值相异,因而使得 CAS 设备具有以下优点:重复文件或文件其中的重复部分可只存一份,从而节省存储空间;定位迅速,也节省网络带宽,系统整体性能得到提高;对数据可进行有效的完整性检验,增强数据的安全性;名字空间的全局唯一性提高了系统共享的可管理性。因此,这样的设备特别适用于存储固定内容(fixed content)的文件。

现已研究实现的用于各种目的的 CAS 原型多数建立在原有各种文件系统之上,这不仅影响了 CAS 性能,且通用性差。如 OceanStore[1] 的原型 Pond[2] 是使用 Java 在分级事件驱动的架构(staged event-driven architecture,SEDA)上实现的。在操作系统内核实现的 CAS 原型的性能要好,移植应用也更加方便,如 Venti[3] 在操作系统 Plan 9 内实现的 CAS 驱动器(CAS driver, CASD)等。与此不同,我们设计的 CASD 是在 Linux 内核[4] 中作为设备驱动程序实现的。本文讨论了其设计和实现方法,并将其与当前流行的存储系统进行了性能比较。

1 CASD 的设计

CASD 是个虚拟 CAS 块设备,与 Ext2 和 Ext3 同位于 Linux 的虚拟文件系统(VFS)层下,可看作使用文件内容摘要代替文件名的传统文件系统(图 1)。在 CAS 中文件由一个或多个数据块(对象)组成,每个数据块有自己的摘要。向磁盘写文件时,其数据块一个跟一个地送入 CASD。

Linux 支持字符和块两类设备文件。块设备利用一块系统内存作为缓冲区,响应读写请求时,先查看缓冲区中的数据,能满足要求就返回相应的数据,否则就调用相应的请求函数进行 I/O 操作,因此性能比字符设备好,这对于存储服务器来说非常重要,所以 CASD 基于块设备驱动器[5] 设计和实现。如图 2所示,CASD 由 4 个功能部件构成:Hash 计算(Hash)、元数据管理(Metadata)、空间管理(Space)和磁盘 I/O(Disk I/O)。

(1) Hash 计算

CASD 实现了 MD5[6] 和 SHA-1[7] 两种 Hash 算

① 国家自然科学基金(60273006)资助项目。
② 男,1967 年生,硕士,副研究员;研究方向:计算机系统结构,存储技术;联系人,E-mail:judapeng@tsinghua.edu.cn
(收稿日期:2008-01-23)

法。MD5 输入以 512 位分组,输出 16 字节 Hash 值,SHA-1 则输出 20 字节 Hash 值。Hash 计算部件可配置、可扩展,由上层应用来决定使用哪种 Hash 算法。加入新的 Hash 算法很容易。

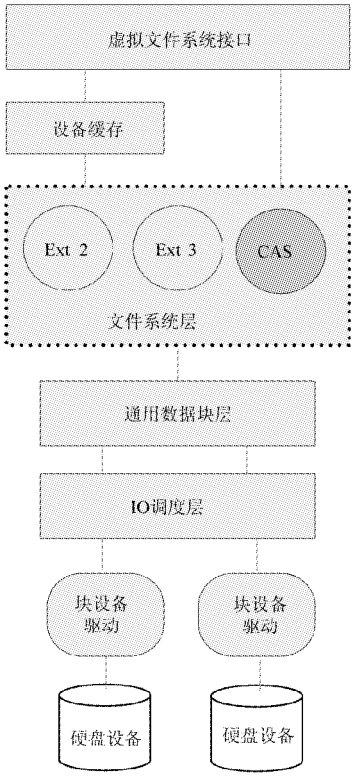

图 1　CASD 在 Linux 内核中的位置

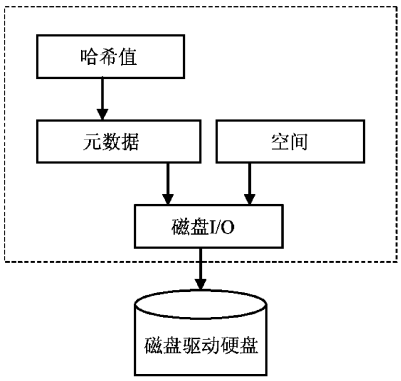

图 2　CASD 的组成

(2) 元数据管理

类似于文件控制块,磁盘上每个对象都与含有存储位置等信息的一个元数据(metadata)一一对应。每个对象再对应一个 Hash 值,由此产生一个 Hash-metadata 映射表,CASD 通过它存取对象。

每个 Hash-metadata 映射表也是一个文件,所以也有自己的元数据,它像传统文件系统的超级块一样存储在磁盘的固定位置。CASD 加载时,Hash-metadata 映射表被读入内存,使用红黑树[8]方法在内存中建立一个索引来提高检索速度。

(3) 空间管理

CASD 使用连续链式索引分配法[9]为对象分配存储空间。每个对象的元数据包括 10 个 64 位指向数据块的直接指针。每个对象的大小不需要动态改变,所有需要的块都可以预先分配。为了提高 I/O 操作速度,CASD 在内存中设有一个元数据的高速缓冲区(cache)。每个元数据的结构如图 3 所示。

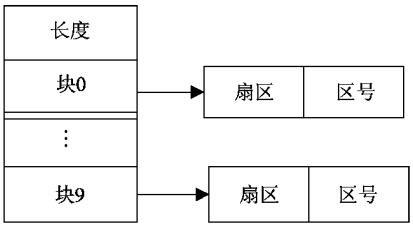

图 3　元数据结构

同时,CASD 使用空闲表法[9]建立一个空闲表(free-list)管理空闲磁盘存储空间。

(4) 磁盘 I/O

因安全性要求很高,对在内核空间(元数据)的数据进行磁盘操作采用同步 I/O 方式。而读写用户空间(对象数据)的数据因对性能和响应时间要求较高,因而采用异步 I/O 方式。

2　CASD 的实现

CASD 使用 C 语言编程并遵循动态核驱动器标准。加载到内存中的核代码很小,可以留给用户程序更多内存空间。

2.1　CASD 主要数据结构

(1)CAS 设备结构 *struct cas _ device _ struct* 表示设备组成,包括 CASD 名称及其相关的磁盘设备信息、Hash-metadata 映射表和空闲分区表 free _ list 的元数据和等待队列等。

(2)块设备操作 *struct block _ device _ operations cas _ bdops* 表示 CASD 在 VFS 层注册的数据读写和控制操作,通过这些操作对 CASD 中的对象进行数据读写。

(3)元数据结构 *struct CAS _ Metadata _ Struct* 包括对应对象的总长度和存储位置。

(4)红黑树 *Node _ * hash _ rbtrees* 用于存储 Hash-metadata 映射表以提高检索速度。

(5)空闲分区表 *struct List free _ list* 用来记录空

闲存储空间的情况。

2.2 CASD 接口函数

CASD 上层通过调用下面的 CASD 接口函数对对象进行操作：

(1)CREAT 在磁盘上创建一个新的 CAS 对象，并将对象的数据写到磁盘上。

(2)READ 从磁盘上读取指定对象的数据。

(3)QUERY 查找对象是否存在。

(4)GET _ ATTR 获得对象的属性。

(5)REMOVE 删除不再需要的对象以释放磁盘空间。

(6)FORMAT 格式化 CASD。CASD 需要在使用前对磁盘进行逻辑格式化，将 CASD 需要的数据结构写到磁盘上。

3 CASD 性能评价

3.1 评价方法

可以使用单机和网络两种环境测试 CASD 的性能。在单机上可以直接比较基于小型计算机系统接口(SCSI)的 CASD 与 Ext2 或 Ext3 文件系统性能，也可以在 iSCSI 网络上间接比较 CASD 与 Ext2 的性能(此时附加网络开销)。当前流行的 iSCSI 存储网络[10,11]是存储技术今后发展的方向，所以下面使用它比较 CASD 和 Ext2 文件系统。作为参考，也测试了基于 Ext2 的传统的网络文件系统(NFS)性能。使用相同的环境和方法，从与实现方式无关的用户应用的角度对这三种存储系统性能进行公平的比较。

测试平台由一个客户端、一个应用服务器和一个存储服务器组成，服务器通过单独的千兆以太网互连。存储服务器使用一个 Sun Fire V40z 服务器，具有 4 个超微设备(AMD)双核处理器(2.6GHz、1MB L2 缓存)、16GB RAM 和 6 个 Ultra-320 SCSI 磁盘(每个磁盘 146GB)。应用服务器使用一个 SUN 工作站，具有 2 个 AMD 双核处理器(2.4GHz、1MB L2 缓存)、8GB RAM 和一个 250GB SATA 磁盘。以太网由 1Gb/s D-Link DGE550T 适配器互连。客户端共享存储服务器的 876GB 存储器。

基于 Ext2 和 CASD 的 iSCSI 存储网络使用开源的 Intel iSCSI v2.0[10] 实现。为区别，以下将使用 Ext2 的 iSCSI 存储系统简称 iSCSI，将使用 CASD 的 iSCSI 存储系统简称 CASN。图 4 示出 iSCSI 存储网络，它是本地 SCSI 设备在互联网上的扩展。客户端(client)通过文件系统运行应用程序(application)，向存储服务器发出请求，由后者响应。此图也用于 CASN，不过图中的块接口(block interface)要换成 CAS 接口(CAS interface)。此图用于 NFS 存储网络时要换成文件接口(file interface)，不使用 iSCSI 协议但仍使用 Ext2。

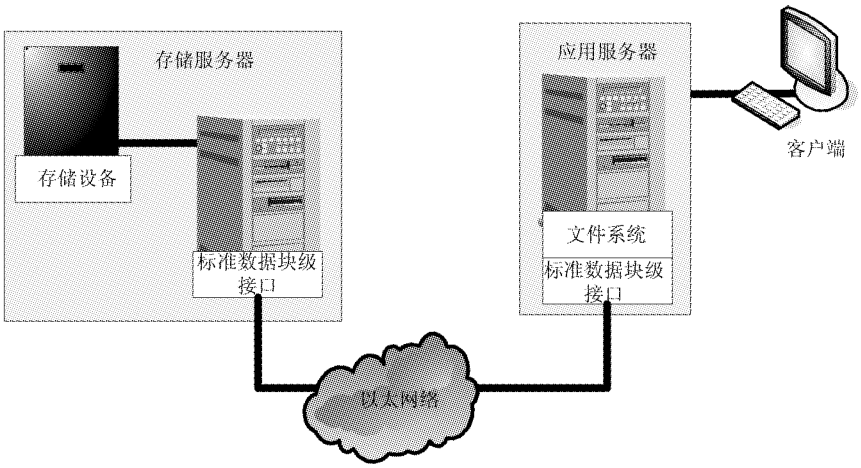

图 4 iSCSI 存储网络

服务器使用的操作系统是 RedHat Linux 9.0，内核版本 2.6.9。在 NFS(V.3)[12] 系统中，使用用户数据报协议(UDP)测量异步和同步模式下的性能，其它参数都是默认设置。Intel iSCSI v2.0 的参数是默认配置，块长度(blocksize)选为 4KB。CASD 将整个文件作为一个对象，其标识符用 MD5[13] 哈希算法产生。

测试程序是 IOmeter[13]，它仅能测试具有块接口的存储系统，所以我们作了一些修改。客户端通过应用服务器向存储服务器发出对不同长度(从 1kB 到 1GB)文件的读写操作请求，该程序从 Linux 内核文件中获取所需的状态信息，记下从开始到结束花费的时间，除以被传送数据的长度，就得到读写带宽(或称吞吐率，throughput)。

3.2 MD5 和 SHA-1 的性能

对采用 Hash 算法 MD5 和 SHA-1CASN 进行读写性能测试,结果见图 5 和图 6。可见 MD5 优于 SHA-1,以后的测试都采用 MD5。

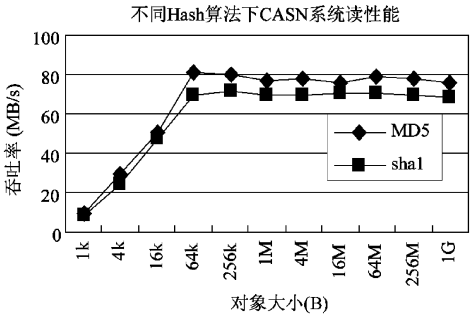

图 5 Hash 算法对 CASN 读吞吐率的影响

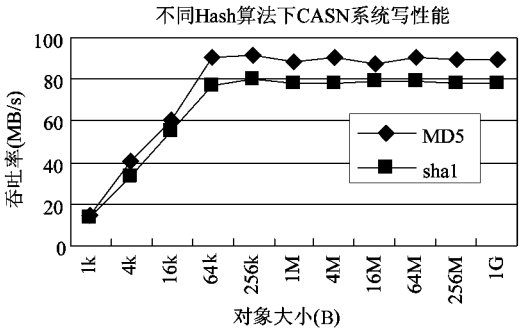

图 6 Hash 算法对 CASN 写吞吐率的影响

3.3 CASN 性能

图 7 是 CASN 写操作(write)的性能。吞吐率随着对象大小(object size)增长比较快,从 object size = 64kB 开始趋于平缓,可高达 90MB/s 左右。写重复数据(rewrite)性能比 write 好,因为这时不需要再执行写磁盘操作,同时节省网络带宽。rewrite 吞吐率基本上是计算 Hash 的带宽,最高为 100MB/s 左右。从计算 Hash 和网络带宽限制这两方面看,目前 CASN 系统的瓶颈是 Hash 的计算。

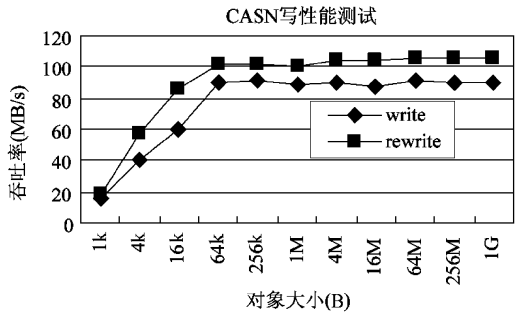

图 7 CASN 写性能

图 8 是 CASN 读(read)操作性能,吞吐率增长规律和写操作类似。重读(reread)性能比 read 要好也是因为 metadata 已经在 cache 中,节省了访问磁盘和网络传送时间。

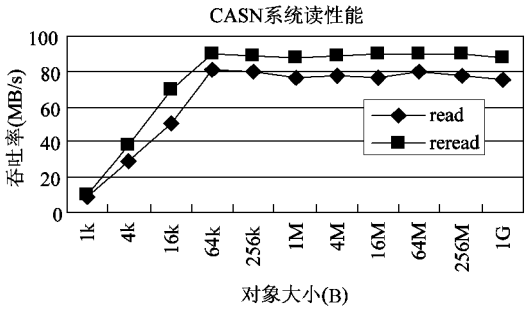

图 8 CASN 读性能

3.4 CASN 与 iSCSI、NFS 的性能比较

图 9 是三个系统在网络为千兆以太网上的写吞吐率比较。iSCSI 性能及其与 NFS 的比较见文献[11,14],本文仅侧重 CASD 与 iSCSI 的比较。

iSCSI 写吞吐率随数据传输单位大小增大而增大,最终稳定在 50MB/s 左右。CASN 系统开始增长比较快,从文件长度为 64kB 开始趋于平缓,达到 90MB/s,高于其它两个系统的 1 倍左右,这是因为写操作是对不变的对象执行的,只需向存储设备顺序添加,充分利用设备的顺序访问带宽,比随机访问带宽几乎高一个数量级。并且存储服务器和应用服务器的内存都特别大(大于传输的总量),CASN 采用异步 I/O 机制,充分利用了系统内存,应用程序和 I/O 操作可以并行执行,提高了系统的总体性能。

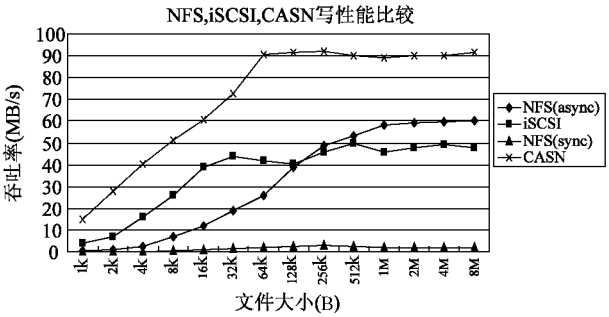

图 9 NFS、iSCSI 和 CASN 在千兆以太网上的写性能比较

图 10 是三种系统在网络为千兆以太网上的读吞吐率比较,都是随文件长度增大而增大。iSCSI 的读操作吞吐率可以达到 100MB/s,几乎是网络带宽的极限。CASN 在文件较小时,读性能比其它两个系统要好。随着文件长度的增加性能稳定在 80MB/s

左右，比 iSCSI 性能差 20%左右，这是因为 CASN 比 iSCSI 增加 CAS 接口的额外计算开销。

以上实验结果表明，CASN 与 iSCSI、NFS 的性能是可以比较的，不同存储操作的性能有所不同，iSCSI 具有最高的读性能，而 CASN 具有最高的写性能。这也说明，与 Ext2 相比，CASD 的写性能好而读性能总的说来差不多。CASD 的性能瓶颈是对文件内容的 Hash 计算和通过 Hash 索引查找对象元数据两个环节。

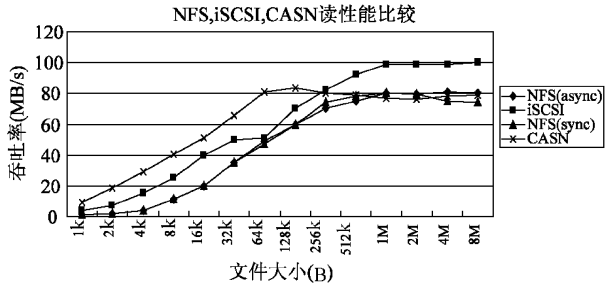

图 10 NFS、iSCSI 和 CASN 在千兆以太网上的读性能比较

4 结 论

本文讨论了 CASD 在 Linux 内核中作为块设备驱动器的设计和实现方法，并和当前流行的 iSCSI 存储系统、传统的 NFS 文件系统进行了性能比较。由于 CASD 是 Linux 内核的一部分，可组成各种运行 Linux 的存储系统应用于不同场合，具有可与 Linux 自身文件系统相比较的性能，并且具有更高的写性能。下一步的工作是增加 CASD 功能并改进性能，开发在单机存储和网络存储系统上的应用。

致谢：感谢清华大学计算机系宋铭，侯纬玮，喻强，顾瑜，生拥宏等同学对 CASN 的实现、部署和性能测试等方面做出的贡献；感谢清华大学计算机系陈渝老师提供的支持和帮助。

参考文献

[1] Kubiatowicz J, Bindel D, Chen Y, et al. OceanStore: an architecture for global-scale persistent storage. In: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, 2000. 190-201.

[2] Rhea S, Eaton P, Geels D, et al. Pond: the OceanStore prototype. In: Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03). San Francisco, CA, USA, 2003. 1-14

[3] Quinlan S, Dorward S. Venti: a new approach to archival storage. In: Proceedings of the Conference on File and Storage Technologies. Monterey, California, USA, 2002. 89-102.

[4] Bovet D, Cesati M. Understanding the Linux kernel. 3rd edition. Sebastopol: O'Reilly & Associates Inc, CA, USA, 2005

[5] Rubini A, Corbet J. Linux Device Drivers. 3rd edition. Sebastopol: O'Reilly & Associates Inc, CA, USA, 2005

[6] MD5 Homepage. http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html, 1991

[7] John Halleck's SHA implemented in C. http://www.cc.utah.edu/nahaj/c/sha/, 2001

[8] Black P E. Red-black Tree [EB/OL]. http://www.nist.gov/dads/HTML/redblack.html, 2006

[9] Silberschatz A, Galvin P, Gagne G. Operating System Concepts. Hoboken: John Wiley & Sons, Inc, New Jersey, 2002

[10] Mesnier M, Nair S. Intel's Open Storage Toolkit. http://sourceforge.net/projects/intel-iscsi, 1999

[11] Xinidis D, Bilas A, Flouris M D. Performance evaluation of commodity iSCSI-based storage systems. In: Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies, Monterey, CA, 2005. 261-269

[12] Pawlowski B, Juszczak C, Staubach P, et al. NFS version 3 design and implementation. In: Proceedings of the Summer Usenix, Berkeley, CA, USA, 1994:137-152

[13] Intel Server Architecture Lab. Iometer: The I/O Performance Analysis Tool for Servers. http://www.intel.com/design/servers/devtools/iometer/index.htm, 2003

[14] Radkov P, Yin L, Goyal P, et al. A Performance comparison of NFS and iSCSI for IP networked storage. In: Proceedings of the 3rd USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, 2004:101-114

[15] Rivest R L. RFC 1321: The MD5 message-digest algorithm. IETF(Internet Engineering Task Force). 1992

[16] NIST(National Institute of Standards and Technology). Secure Hash Standard (SHS). FIPS(Federal Information Processing Standards) Publication 180-1. 1995

[17] Dabek F, Kaashoek M F, Karger D R, et al. Wide-area cooperative storage with CFS. In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP) conference, Chateau Lake Louise, Banff, Canada, ACM Press, 2001. 202-215

[18] Bhagwan R, Tati K, Cheng Y, et al. TotalRecall: systems support for automated availability management. In: Proceedings of the 1st Symposium on Networked Systems Design and Implementation, San Francisco, California, 2004(1):337-350

[19] Muthitacharoen A, Morris R, Gil T, et al. Ivy: a read/write peer-to-peer file system. In: Proceedings of the 5th USENIX Symposium on OSDI, Boston, MA, USA, 2002. 31-44

[20] Weatherspoon H, Eaton P, Chun B. Antiquity: exploiting a secure log for wide-area distributed storage. In: Proceedings of the Second EuroSys Conference (EuroSys'07), Lisboa, Portugal, 2007.371-384

[21] Tolia N, Kozuch M, Satyanarayanan M, et al. Opportunistic use of content addressable storage for distributed file systems. In: Proceedings of the Usenix Annual Technical Conference. San Antonio, TX, USA, 2003. 127-140

[22] Annapureddy S, Freedman M J, Mazi'eres D. Shark: scaling file servers via cooperative caching. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, New York, 2005.(2):129-142

[23] Tolia N, Satyanarayanan M, AWolbach. Improving mobile database access over WAN without degrading consistency: [Technical Report CMU-CS-07-100]. Pittsburgh: Carnegie Mellon University, 2007

[24] Bolosky W J, Corbin S, Goebe D, et al. Single instance storage in windows-2000. In: Proceedings of the 4th USENIX Windows Systems Symposium, Seattle, WA, 2000. 13-24

[25] Muthitacharoen A, Chen B, Maziéres D. A low-bandwidth network file system. In: Proceedings of 18th ACM Symposium on Operating Systems Principles, Chateau Lake Louise, Banff, Canada, 2001. 174-187

[26] Dimitrov V. Content addressable storage provider in Linux. Granville: Mathematics and Computer Science Department, Denison University, Ohio, 2003

[27] Lin C. Build object-based filesystem into Linux: [Technical Report, LCX-SSRC-2002-09]. Santa Cruz: Storage Systems Research Center, Jack Baskin School of Engineering, 2002

[28] Mostek J, Earl B, Levine S, et al. Porting the SGI XFS file system to Linux. In: Proceedings of the annual conference on USENIX Annual Technical Conference, San Diego, California, 2000. 65-76

[29] Huang L, Hsu W, Zheng F. CIS: content immutable storage for trustworthy record keeping. In: Proceedings of the Conference on Mass Storage Systems and Technologies (MSST), College Park, Maryland, USA, 2006

[30] Wang Y, Zheng Y. Fast and secure magnetic WORM storage systems. In: Proceedings of the Second IEEE International Security in Storage Workshop, San Diego, 1993. 259-269

Implementation of a content aware storage driver (CASD) in Linux

Ju Dapeng[**], Liu Chuanyi[*], Wang Dongsheng[* **], Liu Hong[***], Tang Zhizhong[*]

([*] Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

([**] Microprocessor and SOC technology R&D Center, Tsinghua University, Beijing 100084)

([***] Tsinghua-Nuctech Data Security Institute, Tsinghua University, Beijing 100084)

Abstract

In consideration of the fact that the performance and applications of most existing content aware storage (CAS) system prototypes are limited due to their implementation based on original file systems, the paper gives the design and implementation of a CAS driver (CASD), which can be used as a block device driver in the Linux kernel, and a build block for building variety of CAS systems to support different applications. The CASD may be used in storage systems running Linux, including single instance storages and client-server or p2p network storages. The results of the experiments on an iSCSI storage network demonstrate the write operation performance of the CASD is much better than that of Ext2, and the read operation performance is similar to that of it. The performance bottleneck lies in computing hashes of file contents and looking up the hash-metadata map table.

Key words: storage technique, content aware storage (CAS), linux block device driver, CAS driver(CASD), performance evaluation