第 34 卷 第 10 期 计 算 机 工 程 2008 年 5 月
Vol.34 No.10 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)10—0075—03 文献标识码：A 中图分类号：TP311

基于 EVMS 和 SNMP 的存储管理框架

苏智勇[1,2]，曲海平[1,2]，冯 硕[1]，许 鲁[1]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100039)

摘 要：提出一种基于企业卷管理系统(EVMS)和简单网络管理协议(SNMP)的存储管理框架，有效解决了 SonD 系统中存储服务器磁盘、分区、软件 RAID、LVM2 的 VG/LV 和 VSVM 卷的管理问题。给出该框架的 2 种实现方式。测试结果表明，subagent 方式在性能上比 MIB handler 方式优越 20%以上。

关键词：企业卷管理系统；简单网络管理协议；存储管理

Storage Management Framework Based on EVMS and SNMP

SU Zhi-yong[1,2], QU Hai-ping[1,2], FENG Shuo[1], XU Lu[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】 This paper proposes a kind of storage management framework based on Enterprise Volume Management System(EVMS) and Simple Network Management Protocol(SNMP). It can effectively solve the problem of how to manage the disks, segments, software RAIDs, LVM2's VG/LVs and VSVM volumes on the storage servers of the Service on Demand(SonD) system. This paper presents two ways to implement this framework. Test results show that the performance of subagent is superior to that of MIB handler by more than 20%.

【Key words】 Enterprise Volume Management System(EVMS); Simple Network Management Protocol(SNMP); storage management

1 概述

如何对存储系统中诸多物理或虚拟的存储设备进行有效管理，是每个存储系统必须解决的重要问题。telnet, ssh 远程登录等管理方式的操作很不方便，而且系统中存储设备的类型比较多，既有磁盘、分区，又有软件 RAID、LVM2 的 VG/LV 等，不可能要求管理员对每种设备的管理方式都非常熟悉。管理员需要的是一个直观、易用的操作界面(如 Web 页面)来对远程服务器上的存储设备进行管理。

企业卷管理系统(Enterprise Volume Management System, EVMS)是 IBM 开发的卷管理工具。简单网络管理协议(Simple Network Management Protocol, SNMP)是 Internet 工程任务组织为了解决 Internet 上路由器的管理问题而提出的。通过 SNMP，可以对网络中的设备进行有效管理。为了解决 Service on Demand(SonD)系统中存储设备的管理问题，本文通过整合 EVMS 和 SNMP 提供的功能，提出一种基于 EVMS 和 SNMP 的存储管理框架，并开发了相应的存储管理软件。

2 SonD 系统

SonD[1]是国家高性能计算机工程技术研究中心自主研发的用于快速部署集群服务器的系统。SonD系统由以下 4 个部分组成：(1)没有本地磁盘的客户机；(2)存储服务器，它在集中的物理存储资源上虚拟出VSVM卷(对内核而言，VSVM卷是一种虚拟的块设备)，这些VSVM卷被当作网络磁盘供客户机使用，客户机则利用网络块设备协议将这些网络磁盘映射成本地磁盘来使用；(3)管理服务器，它管理存储服务器上的存储设备，包括磁盘、分区、软件RAID、LVM2 的VG/LV以及VSVM卷；(4)高速互联网络，它将上述 3 种功能实体紧密地连接在一起从而构成一个完整的应用系统。

SonD 系统架构如图 1 所示。

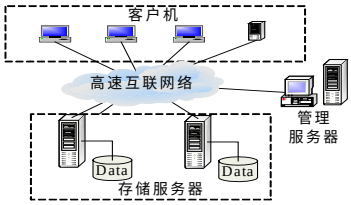

图 1 SonD 系统架构

3 EVMS 简介

EVMS[2]是IBM开发的开源卷管理工具。它在实现上采用完全模块化的方式，分为UI, Engine和Plug-in这 3 部分，如图 2 所示。

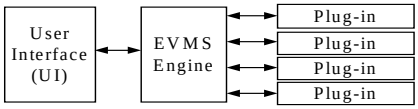

图 2 EVMS 的架构

EVMS UI 负责给用户一个直观易用的操作接口。每个 Plug-in 分别实现各自的管理功能，如 LVM2 Plug-in 实现了对 VG/LV 的管理。EVMS Engine 则负责在 UI 和 Plug-in 之间进行协调，为 Plug-in 提供服务，并且将各个 Plug-in 的功能进行抽象，提供一组统一的接口供 EVMS UI 调用，避免 UI 分别与各个 Plug-in 进行交互。

基金项目：国家“973”计划基金资助项目“下一代互联网存储按需部署模型与服务质量研究”(2004CB318205)
作者简介：苏智勇(1979－)，男，硕士研究生，主研方向：网络存储，存储资源管理；曲海平，博士研究生；冯 硕，研究实习员；许 鲁，研究员、博士生导师
收稿日期：2007-06-15 **E-mail：**suzy@ict.ac.cn

EVMS提供的插件机制使得它具备良好的可扩展性。利用该机制，本文开发了用于管理VSVM卷的VSVM Plug-in[3]，将VSVM卷的管理纳入到了EVMS的框架内。

4 基于 EVMS 和 SNMP 的存储管理框架

在SonD系统中，存储服务器和管理服务器之间必须进行通信，这就涉及到它们之间的通信协议问题。SNMP[4]协议具有简单、高效的特点，并且协议本身就是为管理网络设备而设计的，因此，本文采用它作为存储服务器和管理服务器之间的通信协议。

从具体实现看，SNMP 基于 C/S 架构，服务器端(snmpd)接收客户端的 snmp 请求，对相应的设备执行相应的操作，从而实现对设备的管理。在这个过程中，snmpd 充当客户端和设备之间的中间人(代理)，因此，称服务器端为 agent。

SNMP 将需要管理的网络设备抽象成一个对象(Object)，并把这些对象的集合称为 MIB(Management Information Base)，将所有需要管理的对象组织成一棵树，称该树为 MIB 树，并根据每个对象在树中的位置赋予它一个编号，称为该对象的 OID(Object ID)。

snmpd 只能实现对已定义 MIB 节点的处理，当定义了新的 MIB，即在 MIB 树上添加了新的 MIB 节点以后，需要扩展 snmpd 的功能来实现对该部分 MIB 节点的处理。因此，SNMP 提供了 2 种方式来扩展 snmpd 的功能：

(1)MIB handler 方式。MIB handler 是用 Perl 等写的脚本。管理员在 snmpd 的配置文件中规定该 MIB hander 所能处理的 MIB 节点，snmpd 在启动时根据配置文件 fork 一个子进程来运行该 MIB hander，父子进程通过管道进行通信。

(2)subagent方式。subagent是用C实现的程序。它在启动时向snmpd完成注册，告诉snmpd自己所能管理的MIB节点。snmpd和subagent之间通过Unix Socket进行通信，采用的通信协议是AgentX[5]。

无论采用 MIB handler 方式还是 subagent 方式，最终都需要跟所要管理的存储设备进行交互。EVMS 提供了管理系统中各种存储设备的统一接口，因此，可以在 MIB handler 或者 subagent 中调用 EVMS 提供的功能来实现对各种存储设备的管理。这样，只须定义存储设备的 MIB 信息，在存储服务器端实现相应的 MIB handler 或 subagent，就可以在管理服务器端使用 snmp 客户端工具来实现对存储服务器上存储设备的远程管理。可以在管理服务器端运行 WWW 服务器，并使它通过 CGI 脚本来调用 snmp 客户端工具，从而让管理员可以通过 Web 页面来对存储服务器上的存储设备进行远程管理。本文的存储管理框架如图 3 所示。

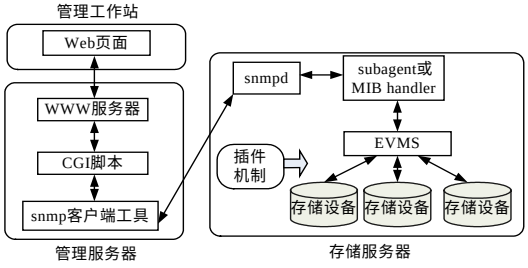

图 3 基于 EVMS 和 SNMP 的存储管理框架

该存储管理框架利用 EVMS 的插件机制实现了对各种类型的存储设备的统一管理。利用 subagent 或 MIB handler 来扩展 snmpd 的功能，并在 subagent 或 MIB handler 中调用 EVMS 提供的功能来实现对各种存储设备的管理，利用 SNMP 协议实现了对存储设备的远程管理。

5 MIB handler 和 subagent 的设计与实现

无论是采用 MIB handler 方式还是 subagent 方式，都须调用 EVMS 提供的功能来最终实现对存储设备的管理。最简单的办法是通过 system()这样的系统函数直接调用 EVMS 提供的命令行工具来完成。但每执行这样一次调用都要启动一次 EVMS，而 EVMS 启动时要有一个在内存中构建出各个管理对象映像的过程，当 EVMS 管理的对象较多时，这个过程比较慢。

如果采用的是 MIB handler 方式，可以对 EVMS 提供的 Command Line UI 进行适当修改，使 EVMS 作为一个后台进程运行，MIB handler 与 EVMS 则通过进程间通信机制(如 FIFO)进行通信。在最初的实现中，本文采用的就是这种方式。但后面的测试结果表明，这种实现方式的性能比较差。这是因为 MIB handler 是用脚本语言实现的，而脚本是解释执行的，另一方面进程间通信带来的开销也比较大。

为了减少上面两方面原因带来的性能上的影响，本文在 SonD 系统中采用 subagent 方式，并且将 EVMS 提供的 Command Line UI 修改成 EVMS UI for subagent。EVMS UI for subagent 提供了一组可供 subagent 直接调用的 API。这样，subagent 就可以直接调用这些 API 来实现对 EVMS 功能的调用，而无须通过进程间的通信机制。整个 subagent 的结构如图 4 所示。该 subagent 的前端负责与 snmpd 进行通信。它接收、整理来自 snmpd 的命令，然后调用 EVMS UI for subagent 提供的 API 通知 EVMS engine 执行相应的操作，最后将操作的结果返回给 snmpd。

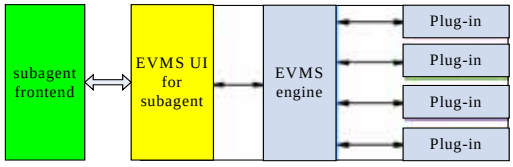

图 4 subagent 整体结构

在 SonD 系统中，本文的 subagent 需要具备磁盘管理、分区管理、软件 RAID 管理、LVM2 卷组管理及 VSVM 卷管理等功能，因此，笔者根据这些功能对 subagent 前端进行了模块划分。各个模块相对独立，可以单独进行开发和调试。进行模块划分后，该 subagent 的结构如图 5 所示。

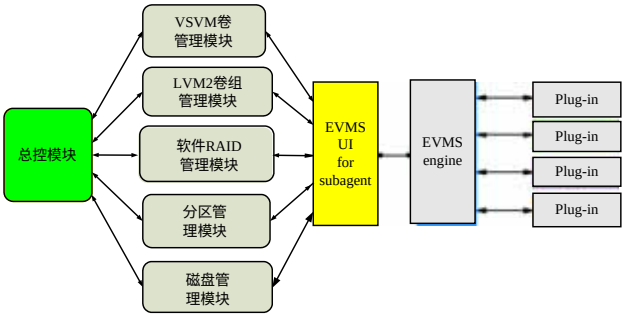

图 5 进行模块划分后的 subagent 结构

6 性能评价

管理员通过 Web 页面对设备进行管理时，不但要求相应的操作能够正确地被执行，同时也要求完成这些操作的时间尽可能短。对 SonD 系统而言，管理员经常要通过管理页面成批地创建、删除 VSVM 卷，因此，可以通过这个典型的、且最耗时的操作来考察性能。测试时使用的存储服务器的配置情况如表 1 所示。

表 1　存储服务器的配置情况

处理器	内存/MB	存储系统	操作系统
Intel(R) Celeron(R) CPU 2.66 GHz	512	160 GB Seagate Barracuda SATA 硬盘	RedHat Fedora Core4 2.6.11 内核

表 2 给出了在采用 MIB handler 方式时，管理服务器端通过 snmp 客户端工具在存储服务器上创建、删除 VSVM 卷所需的时间。

表 2　采用 MIB handler 方式时创建、删除 VSVM 卷所需时间

VSVM 卷/个	创建所需时间/s	删除所需时间/s
20	6	7
50	14	21
80	25	39
100	33	53

可以看出，这种实现方式的性能比较差。这是因为 MIB handler 是用脚本语言实现的，而脚本是解释执行的，并且进程间通信的开销也比较大。在进行删除操作的时候，MIB handler 与 EVMS 后台进程需要进行的交互次数要比进行创建时多，进程间通信开销带来的影响也就越凸显，因此，所需的时间也就明显比进行创建所需时间多。随着删除卷数量的增长，上述因素带来的瓶颈效应也就越明显，所需的时间呈现明显的非线性增长。

在采用 subagent 方式时，在同样的平台上，通过 snmp 客户端工具在存储服务器上创建、删除 VSVM 卷所需的时间如表 3 所示。

表 3　采用 subagent 方式时创建、删除 VSVM 卷所需时间

VSVM 卷/个	创建所需时间/s	删除所需时间/s
20	5	4
50	13	11
80	20	17
100	26	23

对照表 2 和表 3 可以看出，在采用 subagent 方式后，性能得到了明显的提高，尤其是在 VSVM 卷数量比较多的情况下。创建 100 个 VSVM 卷所需时间比采用 MIB handler 方式减少了 20%左右，删除 100 个 VSVM 卷所需时间比采用 MIB handler 方式减少了 50%以上。而且，创建、删除所需的时间与卷的数量基本成线性关系。

因此，本文实现的 subagent 有效地消除了早期采用的 MIB handler 方式中的性能瓶颈，明显减少了完成创建/删除 VSVM 卷所需的时间，为通过 Web 页面快速完成这些操作奠定了很好的基础。

7　结束语

针对如何有效管理存储系统中诸多存储设备的问题，本文提出一种基于 EVMS 和 SNMP 的存储管理框架，并把它应用于 SonD 系统中，介绍了相应 MIB handler 和 subagent 的设计与实现。测试结果表明，subagent 的性能比 MIB handler 优越得多。该 subagent 与 SonD 系统存储管理软件的其他部分密切配合，使管理员可通过 Web 页面对存储服务器上的磁盘、分区、软件 RAID、LVM2 的 VG/LV 及 VSVM 卷进行直观且有效的管理。

参考文献

[1] 刘振军, 许　鲁, 尹　洋. 蓝鲸 SonD 动态服务部署系统[J]. 计算机学报, 2005, 28(7): 1110-1117.

[2] Enterprise Volume Management System(EVMS)[Z]. (2003-05-06). http://evms.sourceforge.net/architecture.

[3] Yin Yang, Liu Zhenjun, Xu Lu. VSVM-enhanced: A Volume Manager Based on the EVMS Framework[C]//Proc. of the 5th International Conf. on Grid and Cooperative Computing. Changsha, China: [s. n.], 2006.

[4] SNMPv2[Z]. (1996-01-02). http://www.ietf.org/rfc/rfc1901.txt.

[5] AgentX[Z]. (2000-01-05). http://www.ietf.org/rfc/rfc2741.txt.

(上接第 74 页)

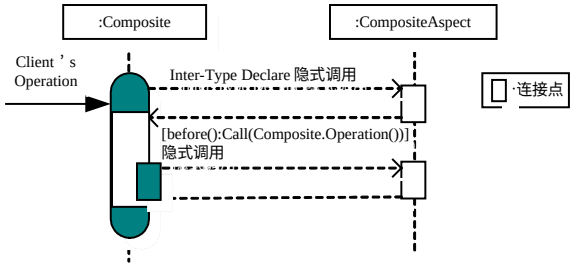

图 3　Composite 对象和 CompositeAspect 方面交互的时序图

4　2 种实现方案的比较

通过对比 2 种实现方法可以得出，基于 AOP 实现的 Composite 模式具有如下优点：(1)模块职责单一化。把 Composite 类特有的操作方法封装在 CompositeAspect 方面中，使 Component 类的职责单一化，只为用户提供一致的操作接口，同时 CompositeAspect 方面的职责也变得单一，只提供 Composite 类特有的操作，如 Add(Component)。因此，这种实现方式具有较好的模块性，实现了接口隔离原则，消除了基于 OOP 实现时的接口纠缠现象。(2)可扩展性好。因为把用户感兴趣的操作接口统一在 Component 中、把 Composite 类特有的操作封装在 CompositeAspect 中，所以无论是添加或删除用户感兴趣的操作还是 Composite 类特有的操作，都很方便，只须在相应的类中作修改。

5　结束语

利用 AOP 思想、采用 AspectJ 实现的 Composite 模式具有较好的模块结构，可以有效消除 OOP 中的接口纠缠现象，使模式具有更好的扩展性。这主要得益于 AOP 具有处理横切关注点的特性，利用该特性可以改进和完善在 OOP 中具有横切关注点的设计。

参考文献

[1] Gamma E, Helm R, Johnson R, et al. Design Patterns Elements of Reusable Object-oriented Software[M]. Boston, USA: Addison Wesley Professional, 1995.

[2] Wu Xiaoqing. Pattern Transformation for Two-dimensional Separation of Concerns[J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(4): 218-219.

[3] Filman R E, Elrad T, Clarke S, et al. Aspect-oriented Software Development[M]. Boston, USA: Addison Wesley Professional, 2004.

[4] Colyer A, Clement A, Harley G, et al. Eclipse AspectJ: Aspect-oriented Programming with AspectJ and the Eclipse AspectJ Development Tools[M]. Boston, USA: Addison Wesley Professional, 2004.

[5] Laddad R. AspectJ in Action — — Practical Aspect-oriented Programming[M]. Greenwich: Manning Publications Co., 2003.

[6] 徐　昊, 张　逸. AOP 的本质和意义[J]. 程序员, 2005, (5): 84-86.