中文摘要

随着计算机技术的飞速发展，当前的计算机系统对于存储容量的要求逐步提高，存储系统逐步向磁盘虚拟存储的方向发展。磁盘虚拟存储技术作为一种面向磁盘的虚拟化技术，向用户屏蔽了物理存储的细节问题，使得用户的使用更方便。由于虚拟存储涉及到众多的磁盘以及网络性能等相关问题，所以有必要为其开发专门的监视系统，对系统的运行状况进行监视。

考虑到系统专注于性能监视（CPU 利用率,内存等）的特点，在比较 MRTG 和 GANGLIA 两种技术的基础上，采用了 MRTG 作为其基本的监视技术，并辅以 RRDTOOL 数据库弥补 MRTG 本身的缺点。

为了实现性能监视与磁盘监视的统一，在磁盘监视方面，系统采用了 SMART 作为基本的监视技术，并解决了不同磁盘厂商的 SMART 标准不统一的问题，加入了对于磁盘监视项的具体分析，从而更好的实现了系统的监视功能。系统也实现了对于虚拟存储中的系统性能监视（CPU 利用率，内存利用率）等和磁盘监视（磁盘温度，磁盘容量，磁头飞行高度）的统一。

总之，本系统在磁盘监视方面对现有的监视软件与技术进行了扩展，并且将用来作性能监视的 MRTG 和用来作磁盘监视的 SMART 进行了统一，使得系统可以同时对系统的性能以及磁盘信息进行监视。

关键词： 虚拟存储　监视系统　GANGLIA　MRTG　SMART

ABSTRACT

As long as the computer science develops, modern computer system requires a bigger storage capacity, after the development from single disk to RAID system, the storage system today has enter a new time—virtual storage system. Virtual storage system is a kind of virtual technology based on disks, it covers up the details of physical storage media, which makes the storage system easier to use for people. Virtual storage system is generally a huge array of disks, which involves many problems of disks and network flow, so a monitoring module, which can make the system run correctly, is needed.

This system focus on the performance monitoring (CPU load, memory load, for example). There are two kinds of technologies that are widely used in performance monitoring, after some careful comparing, we decided to use MRTG as our monitoring technology, we also added RRDTOOL together with MRTG, which can remedy some drawbacks of MRTG. In this way, we can develop a better monitoring system.

There are not any monitoring technologies that can monitor both the performance of the system and the disk together, so we use SMART technology to monitor the disks. Since there are many drawbacks in current disk monitoring software, this system makes an improvement and innovation over current system, which includes: making a single interface of different disk manufacturer and analyzing the specific reasons of different disk problems. These two tasks can help us make a better monitoring system.

This monitoring system combines the performance and disks monitoring, and the architecture and methods that are used in this system can also be used to add more monitoring items to the system, in this way, this system can be extended easily.

KEY WORDS: Virtual storage system Monitoring system GANGLIA MRTG SMART

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：[signature]　　签字日期：2007 年 6 月 19 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：[signature]　　导师签名：[signature]

签字日期：2007 年 6 月 19 日　　签字日期：2007 年 6 月 14 日

第一章 绪论

1.1 问题的提出

虚拟化技术并不是一件很新的技术，它的发展，应该说是随着计算机技术的发展而发展起来的，最早是始于 70 年代，最典型的应用就是虚拟内存技术。随着计算机技术以及相关信息处理技术的不断发展，人们对存储的需求越来越大。这样的需求刺激了各种新技术的出现，比如磁盘性能越来越好，容量越来越大。但是在大量的大中型信息处理系统中，单个磁盘不能满足需要，这样的情况下存储虚拟化技术就发展起来了。在这个发展过程中也由几个阶段和几种应用：首先是磁盘条带集(RAID，可带容错)技术，将多个物理磁盘通过一定的逻辑关系集合起来，成为一个大容量的虚拟磁盘。而随着数据量不断增加和对数据可用性要求的不断提高，又一种新的存储技术应运而生，那就是存储区域网络(SAN)技术。SAN的广域化则旨在将存储设备实现成为一种公用设施，任何人员，任何主机都可以随时随地获取各自想要的数据[1]；二来的是体系结构的变化：从由小规模固定数目的存储介质组成的存储系统向由大规模磁盘阵列组成的虚拟存储系统的转变。这种体系结构的革新使得虚拟存储的设计者们对设计规模，可靠性，层次结构以及管理模式这些基础的考虑因素有了重新的认识：规模，可靠性成为系统设计的关键因素。另外，由于今天的存储系统以及网格系统都是由几百台，上千台节点组成的，管理功能也已经成为非常重要的一个部分，而系统的监视功能作为系统管理的重要功能之一，其重要意义更是不言而喻[2]。

虚拟存储系统面临的一个主要挑战是可扩展的系统性能监视系统。由于现在的虚拟存储系统由大规模的存储节点组成，其上频繁的 I/O 操作与网络需求，使得虚拟存储系统越来越频繁地发生错误。为了减少节点的损耗，以及确保整个系统的可靠运行，监视系统软件必须能够迅速诊断出错误发生的现场，从而使得系统可以自动修复或者手动重新启动。在大规模的虚拟存储系统中，控制台与存储设备间的相互操作非常复杂，监视系统捕捉这些相互操作的部分信息并且把它们显示出来，使得管理员可以轻松地了解系统的行为。总之，当虚拟存储系统规模扩大以及更加分散的时候，系统瓶颈可能出现在多个部分。一个效率高的监视系统可以为管理员提供一个系统的整体视图，帮助管理员定位错误，优化系统。

虚拟存储是一个庞大的系统，里边涉及到的许多硬件设备的状态都会对整个

虚拟存储产生很大的影响，那么如何有效的对这些设备信息进行监视，从而不仅能够直观的看到系统当前的运行情况，更重要的是一可以根据设备的监视信息估计到可能出现的问题，做到防患于未然，那就可以极大的提高系统的可用性。

1.2 本领域研究概况

为了比较清楚的介绍当前的虚拟存储监视技术的发展状况，我们有必要先简单的介绍一下整个虚拟存储系统的架构方式以及其体系结构的特点，以便于我们理解虚拟存储的监视系统的技术特点。所谓虚拟存储，就是把多个存储介质模块(如硬盘，RAID)通过一定的手段集中管理起来，所有的存储模块在一个存储池(Storage Pool)中得到统一管理，从主机和工作站的角度，看到就不是多个硬盘，而是一个分区或者卷，就像是一个超大容量(如 1T以上)的硬盘。这种可以将多种，多个存储设备统一管理起来，为使用者提供大容量，高数据传输性能的存储系统，就称之为虚拟存储[3]。目前虚拟存储的发展尚无统一标准，从虚拟化存储的拓扑结构来讲主要有两种方式：即对称式与非对称式。对称式虚拟存储技术是指虚拟存储控制设备与存储软件系统、交换设备集成为一个整体，内嵌在网络数据传输路径中；非对称式虚拟存储技术是指虚拟存储控制设备独立于数据传输路径之外。可见，虚拟存储是一个对磁盘阵列以及网络情况紧密依赖的系统，所以当前大多的虚拟存储监视系统也专注于以上两个方面。

1.2.1 当前的主要监视领域

由于监视对象的多样性，在当前的监视技术中，各种不同的监视技术所监视的领域以及监视的具体实现方式也是多种多样的。

从监控领域来说，分为系统监控、性能监控、应用程序监控、进程监控四个方面，其监控参数、实现方法也各有千秋。

系统监控是比较重要的，它相当于一个虚拟的系统操作员，根据预先定义的非正常事件，监视一切潜在的系统故障，且撰写系统日志，以页面、邮件、信息的方式通知显示程序和系统管理员，并自动采取一些应对管理措施，（如重启进程）在保证系统环境的可靠和完整性方面扮演着重要的角色。Netsaint、NCSA 的监控脚本 Checksys 都属于系统监控范畴。

性能监控专注于计算机系统的整体性能表现，监控参数包括：

① 各节点的 CPU 使用率(idle、nice、system、user)。

② 各节点的内存使用率 (used、cached、buffered、swapped、in-core、shared)。

③ 各节点磁盘输入输出状况、用量和空间。

④ 各节点打开、缓存文件数。

⑤ 各节点网络状况， 读写包、字节数。

⑥ 管理网络使用率 (接收/发送的字节)[4]。

如 Parmon、Supermon、Ganglia、MRTG 等都是性能监控软件。

实际监控中不仅要收集各个节点、部件的状态性能信息，也应该能反映应用程序、作业队列的运行等待情况，如各个作业的状态、所属队列、所属用户、所在节点、等待时间，以及不同的节点上作业占用资源情况以及资源利用率等。Clumon（Open PBS）、NWPerf 涉及到了应用程序监控。

进程监控与安全关系反映最明显，因为入侵者最少需要运行一些进程才可以进入系统的节点，一些非预期进程是安全危机的指示。但是进程监控在目前的研究中还没有作为单独的领域由特定的软件来实现。

1.2.2 当前的主要监视方式

根据系统的不同要求，监视也可以有不同的实现方式，总的来说不外乎三种：软件实现、硬件实现和软硬件结合实现。

硬件实现方式是通过独立的硬件模块（如监控卡）和硬件传输网络进行信息的采集、判别、故障报告以及加电与断电的控制操作等。其优点在于不占用任何系统资源，同时也可获取一些底层硬件信息，缺点在于很难获取操作系统层或应用层的一些状态信息，同时成本较高。

软件实现方式一般都是通过基于操作系统之上的后台守护程序采集系统内部各个部件的状态信息，然后通过管理网络传递给运行在特定的节点机器或者专用的监控机上的监控程序，最后以某种特定的方式展现出来。其优点在于它能比较全面的获取系统信息和应用程序运行状态信息，同时成本相对较低；缺点在于无法获取底层硬件信息（电流、电压、风扇等），并占用了一定的系统资源，影响用户程序的运行。一般研究机构和商业团体所做的只是软件监控的设计。

软硬件结合的方式，将两者优点集与一身，弥补了单纯使用软件或者单纯使用硬件监控方式的不足，是理想监控系统的实现方式，但实现复杂性比较大，同时带来的成本增加也远高于前两者[5]。

1.2.3 当前主要监视技术

下面我们列出性能监视方面当前一些主要的监视技术的各自优缺点：

Supermon & Clumon

1．没有自带数据库。

2．持续进行数据收集的时候可能会出现中断。

CARD

1．不是很稳定。

2．有很大的系统负载。

PARMON[6]

1．系统数据收集经常容易被中断。

2．客户端程序比较复杂。

MRTG

1．每幅图上只限画两个参数。

2．监视项的数据以 log 形式存储，耗费资源。

GANGLIA

1．主要面向集群，结构复杂。

2．采用自己的私有协议来收集数据，通用性不好。

SMART

SMART技术与上述面向性能监视的监视软件不同，它主要专注于磁盘的全面监视。所谓SMART（Self Monitor Analysis Report Technology），即自检测诊断分析与报告技术，它是ATA/IDE和SCSI环境下都可使用的一种可靠性预测技术，可以用来获得磁盘的各项参数的数值，并对磁盘进行各种粒度的自检扫描 ，它是随着磁盘标准ATA的不断发展而引入的，并且在随着ATA标准的不断发展而改善[7]，目前的许多监视软件都是在SMART技术的基础上包装而成的。SMART技术也有自身的一些不足，主要表现在：

- SMART 标准虽然是统一的，但是各个不同的利用 SMART 技术的磁盘监视软件却没有完全按照标准来设计，所以引起了不一致。
- 磁盘监视软件仅仅有磁盘各项参数的监视，却没有针对不同参数项的分析，不能很好的发挥监视的作用。

1.3 研究的主要问题

针对以上对于当前虚拟存储监视系统的研究现状的概述，我们看到当前的虚拟存储监视软件还存在着很大的改进余地，正是出于以上的目的，本文所描述的是一个更完善，更实用的监视系统，该系统的主要优势如下：

- 本系统利用 MRTG+SNMP 协议架构，不单可以监视系统性能（CPU 利

用率，网络流量等）方面的参数，而是把它和当前的磁盘监视软件结合起来，使它也可以监视磁盘温度等磁盘的参数信息，实际上，利用本系统的思路，还可以加入许多 SNMP 标准库中并没有的监视项，例如风扇转速等。

- 针对当前磁盘 SMART 信息过于简单的事实，将提取出来的 SMART 信息分类并根据不同信息的重要程度进行报警，并且在分析不同参数出错原因的基础上给出可能的解决办法。
- 针对不同厂家的磁盘 SMART 信息格式不统一的现象以及磁盘阵列中有磁盘通道卡或者是 RAID 卡存在的情况，采取配置文件的形式对磁盘信息进行解析，使得用户不必关心底层的数据获取情况，从而统一了对于磁盘信息的提取与处理。

第二章 文献综述

本章主要介绍当前的监视领域一些主流的监视技术与监视软件，并在对这些软件与技术的分析与总结的基础之上，提出了本监视系统的架构方式，具体的包括 SNMP 协议，SOAP 协议，MRTG，RRDTOOL，SMART 技术等。

2.1 SNMP 协议

简单网络管理协议(Simple Network Management Protocol)是目前网络上用最为广泛的网络管理协议。实际上，SNMP指网络管理的一系列标准，包括协议、管理信息结构和一系列管理对象的定义。具体而言就是被管网络实体中的每个被管理资源由一个对象来表示，通过对各种网络环境中所有可能用到对象的标准化和规范化，就形成了一个树型结构的数据库，并命名为MIB (Management Information Base，管理信息库)。为了满足协同操作的要求，SNMP协议还提供对MIB库的统一定义方法。通过SNMP报文的交换，网络管理站对代理进程中的相应MIB对象进行读(Get。)、写(Set)操作，就实现了监视被管理网络实体的系统资源和其运转状况，进而通过修改这些对象值来控制系统中资源的目的[8]，更为重要的一点在于，SNMP允许我们在一个指定的节点位置之下（通常为编号为 19040 的节点，也被称为私有节点）扩展标准的MIB库，从而加入我们自己关心的，希望能够监视的监视项[9]，图 2-1 是一个MIB库的结构示意图：

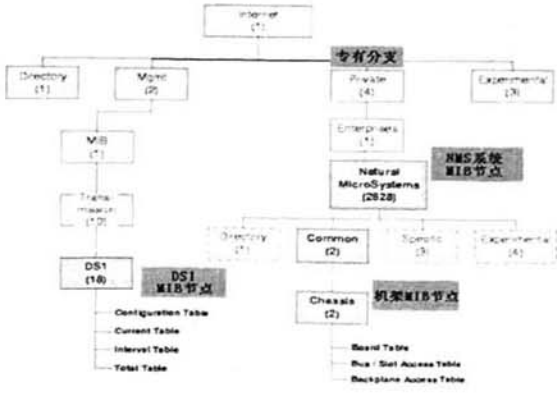

图 2-1 标准 MIB 库

我们可以在上述 MIB 库的一个指定的节点上加以扩展，从而加入我们需要监视的数据项（例如磁盘温度等），从而利用标准的 SNMP 来实现对设备的监视。

SNMP 管理的网络有三个主要组成部分：管理的设备、代理和网络管理系统。管理设备是一个网络节点，包含SNMP 代理并处在管理网络之中[10]。被管理的设备用于收集并储存管理信息。通过SNMP，网络管理系统能得到这些信息。被管理设备，有时称为网络单元，可能指路由器、访问服务器，交换机和网桥、HUBS 、主机或打印机。SNMP 代理是被管理设备上的一个网络管理软件模块。SNMP 代理拥有本地的相关管理信息，并将它们转换成与 SNMP 兼容的格式。网络管理系统运行应用程序以实现监控被管理设备。此外，网络管理系统还为网络管理提供了大量的处理程序及必须的储存资源。任何受管理的网络至少需要一个或多个网络管理系统 ，关于SNMP CLIENT以及SNMP AGENT的关系可以参照下图：

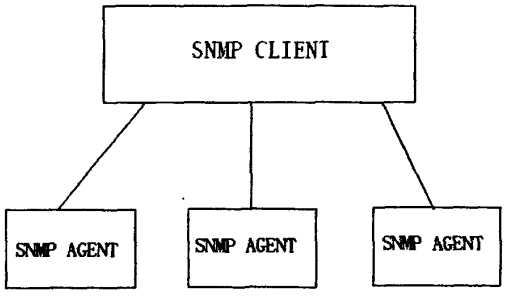

图 2-2 SNMP Client，Agent 示意图

2.2 MRTG 技术

MRTG(Multi Router Traffic Grapher ，MRTG)是一个监控网络链路流量负载的工具软件，它通过SNMP协议得到设备的流量信息，并将流量负载以包含PNG格式的图形的HTML 文档方式显示给用户，以非常直观的形式显示流量负载[11]。MRTG最初提出的时候确实是用于分析网络流量的一些问题，但是实际上任何你可以想到的与时间有关的参数都可以通过MRTG来监视，网络流量，CPU负载，温度，湿度，甚至是海潮等等[12]，MRTG的工作主要由它的配置文件mrtg.cfg来完成，mrtg.cfg指定了MRTG工作的方方面面的细节，灵活使用MRTG的过程，也就是逐步掌握mrtg.cfg的配置的过程，以下是一个利用MRTG实现的监视图像的

示意图：

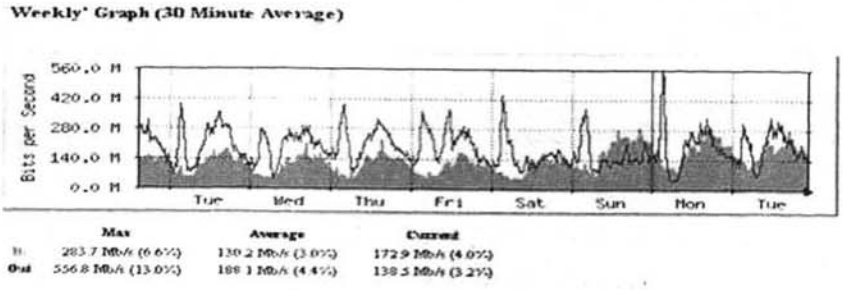

图 2-3 MRTG 显示图

MRTG 使用的是 SNMP 作为其底层数据收集的协议，它的不足表现为：

- 只有是在 SNMP 协议的 MIB（标准信息管理库）中定义的监视项才可以成为 MRTG 监视的内容，否则就要对 MIB 进行扩展。
- MRTG 的数据采取 LOG 的形式保存，随着数据量的不断增大，会急剧的消耗系统资源。
- MRTG 每张图中只可以显示两个监视项，不利于许多个数据的比较。

所以当前使用 MRTG 的监视系统总是和 RRDTOOL 一起使用，RRDTOOL 其实是一种特殊的数据库，它弥补了 MRTG 的一些重要的不足。

2.3 RRDTOOL 数据库

RRDTOOL代表 “Round Robin Database tool”，作者同时也是 MRTG 软件的开发者。

所谓的“Round Robin” 其实是一种存储数据的方式，使用固定大小的空间来存储数据，并有一个指针指向最新的数据的位置[13]。我们可以把用于存储数据的数据库的空间看成一个圆，上面有很多刻度。这些刻度所在的位置就代表用于存储数据的地方。所谓指针，可以认为是从圆心指向这些刻度的一条直线。指针会随着数据的读写自动移动。要注意的是，这个圆没有起点和终点，所以指针可以一直移动，而不用担心到达终点后就无法前进的问题。在一段时间后，当所有的空间都存满了数据，就又从头开始存放。这样整个存储空间的大小就是一个固定的数值。所以RRDTOOL 就是使用类似的方式来存放数据的工具，RRDTOOL所使用的数据库文件的后缀名是“.rrd”。

MRTG 的监视系统之所以总是和 RRDTOOL 一起使用，是因为它解决了 MRTG 的两个主要问题：

- RRDTOOL 的大小有一个最大值，随着数据库数据的不断增多，一些比较老的数据会进行响应的合并，从而为新的数据的插入腾出空间，这样，即使随着数据的不断增多，也不会使得数据库的大小无限制的增大，RRDTOOL 的这种特性，使得它很适合用来作为监视的数据库。
- RRDTOOL 的另一个优势在于，它有着比 MRTG 丰富的多的参数，从而可以绘制出更为细腻和个性化的图像，更主要的是 RRDTOOL 允许在同一张图上绘制多个图像，这样就便于图像的比较。

RRDTOOL 还有一些其余的优势，主要表现为：

- 首先 RRDTOOL 存储数据，扮演了一个后台工具的角色。但同时 RRDTOOL 又允许创建图表，这使得 RRDTOOL 看起来又像是前端工具。其他的数据库只能存储数据，不能创建图表。
- 其他数据库只是被动的接受数据， RRDTOOL 可以对收到的数据进行计算，例如前后两个数据的变化程度（rate of change），并存储该结果。
- RRDTOOL 要求定时获取数据，其他数据库则没有该要求。如果在一个时间间隔内（heartbeat）没有收到值，则会用 UNKNOW 代替，其他数据库则不会这样[14]。

也就是说，RRDTOOL（round robin database）本质上是一种数据库，其特色在于它的数据库的大小有一个最大值，随着数据库数据的不断增多，一些比较老的数据会进行响应的合并，从而为新的数据的插入腾出空间，这样，即使随着数据的不断增多，也不会是的数据库的大小无限制的增大，RRDTOOL的这种特性，使得它很适合用来作为监视的数据库，而且RRDTOOL的另一个优势在于，它有着比MRTG丰富的多的参数，从而可以绘制出更为细腻和个性化的图像[15]。不过需要说明的一点是，在MRTG和RRDTOOL合起来使用的时候，它们是公用MRTG的配置文件mrtg.cfg的，而关于RRDTOOL的具体的一些参数的控制是在程序中实现的，以下是利用MRTG+RRDTOOL所绘制出来的图像的示例，可以看到，它更加丰富，也更加美观：

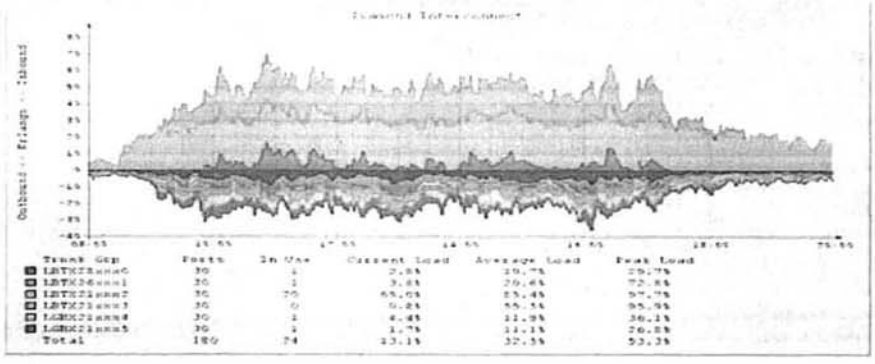

图 2-4 RRDTOOL 显示图

2.4 GANGLIA 监视技术

GANGLIA 与 MRTG 是当前监视领域使用最为广泛的两种监视技术，两种不同的技术有各自的长处与不足：

GANGLIA 是一种主要面对集群系统的监视软件，和 MRTG 相比，它的优势在于：

- GANGLIA 的使用与配置比较简单，它可以比较容易的集成在当前系统之中。
- 当系统的规模比较大的时候，由于 GANGLIA 有比较复杂的体系结构，所以比较适合于监视大规模的集群系统。

但是 GANGLIA 也有其显著的缺点，主要表现为：

- Ganglia 对于低层数据采集使用的是自己的私有协议，而 MRTG 采用的是通用的 SNMP 协议，所以其通用性更好。
- Ganglia主要是针对分布式监视开发的一个监视软件，它的体系结构比较复杂[16]，所以对系统资源的消耗也比较严重，把它作为对实时性要求很高的虚拟系统的监视软件，是不合适的。

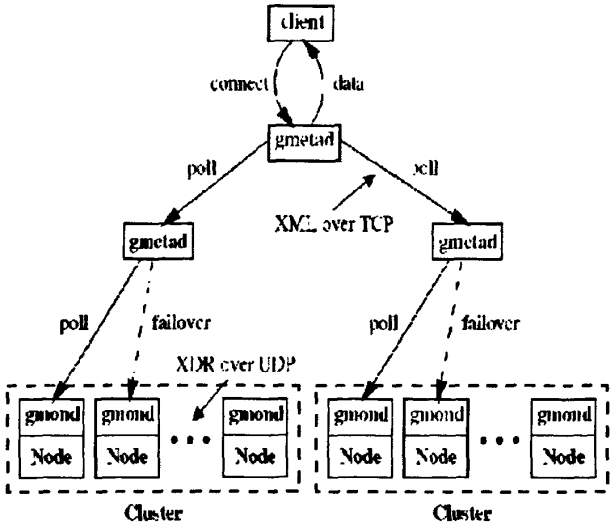

图 2-5 GANGLIA 结构图

2.5 磁盘监视技术

在磁盘监视方面，当前的磁盘监视与控制方式，主要可以分为以下几种：

- 对支持 SMART 功能的硬盘，可以定期通过 SMART 命令查询磁盘状态，一旦发现磁盘健康状况异常，就产生报警。
- 在磁盘空闲时，使用 STANDBY ，SLEEP 命令，使磁头停转，避免空转造成的磨损，在磁盘正常状态下，磁盘内部无论忙闲都处于高速旋转状态。
- 对支持命令队列的磁盘和磁盘控制器，可以使用命令队列，将执行命令的顺序改变，减少执行命令过程中磁头整体的运动。

对于以上后两条控制方式，有必要做以下说明：

- ATA 硬盘通常有四种运行状态[17]（见表 1-1）：

表 1-1 磁盘状态表

状态	含义
ACTIVE	磁盘用电正常，盘片始终处于高速旋转状态
IDLE	磁头的读写电流关闭，但是磁盘并没有停止旋转，耗电和磨损比 NORMAL 状态低一些

STANDBY	读写电流关闭，磁盘也停止了旋转，耗电和磨损更低，通常在 1 瓦以下
SLEEP	磁盘基本已经不工作，只保留唤醒电工作，处于最低耗电状态

磁盘的四种工作状态可以通过一组电源管理命令相互转换，也就是说磁盘的电源管理功能本来是为了减少功耗设置的。但是，我们可以用它来降低磁盘的磨损，延长磁盘的使用寿命。因为关闭磁头的读写电路，可以减少磁盘的耗电，而且降低电磁方面的干扰，停止磁头的转动，可以减少磁盘的磨损。当然，这样做的前提是磁盘有一段比较长的空闲时间。如果磁盘一直都很忙，停止磁盘工作只会使磁盘的磁头频繁地启动——加速——停止，反而增加了磁盘的机械磨损，降低磁盘的使用寿命。

- 命令队列技术

命令队列的方法是从SCSI的TCQ 命令借鉴发展起来的。这项技术的核心就是将一组命令按照磁头运动最小的原则排成队列，在队列中进行。这样，命令队列技术可以提高磁盘的读写性能（磁头的机械运动相对于电子读写消耗更长的时间），同时，由于减少了磁头的运动，客观上降低了对磁盘的磨损，有助于提高磁盘的使用寿命。但是，由于PATA协议本身的局限，执行命令队列的效率很低，所以很少有PATA磁盘支持命令队列功能。SATA 2 的标准扩展了命令队列的支持，提出了NCQ的命令集，将会受到磁盘厂商比较普遍的支持。但是从现在的情况看来，使用这种方式还是具有很大的局限性的[18]。

可以看到，以上的两种磁盘的控制方式，从本质上来讲都是由一些单独的管理命令来实现的，这种监控方式的缺陷在于它们并不能全面的反映磁盘的各种参数的数值大小，而且由于电源管理功能的使用必须要求磁盘有一定时间的空闲时间之后，而如何判别磁盘是否空闲，这本身就需要磁盘SMART技术的支持，而且我们也提到，命令队列的方法由于其协议的局限性，也没有被普遍使用。所以当前的磁盘监控系统主要还是以基于SMART的磁盘监视的软件众多，比较常用的有ACTIVE SMART，SMART MONTOOLS等，但是各种各样的软件其基本原理都是一样的，就是利用SMART技术将磁盘的各种参数提取出来，然后以页面的形式显示出来，所谓SMART(Self-Monitoring ，Analysis and Reporting Technology) 技术，是一种磁盘的自我监视，分析与报告技术，由于硬盘的容量越来越大，为了保证数据的安全性，硬盘厂商都在努力寻求一种硬盘安全监测机制，SMART 技术便应运而生。SMART 即“自我监测、分析及报告技术”。它

可以监控磁头、磁盘、电机、电路等部件，由硬盘的监测电路和主机上的监测软件对被监控对象的运行情况与历史记录和预设的安全值进行分析、比较，一旦出现安全值范围以外的情况，它就会自动向用户发出警告。而更先进的技术还可以自动降低硬盘的运行速度，把重要数据文件转存到其它安全扇区，通过 SMART 技术可以对硬盘潜在故障进行有效预测，提高数据的安全性，它是ATA/IDE和SCSI环境下都可使用的一种可靠性预测技术，可以用来获得磁盘的各项参数的数值，并对磁盘进行各种粒度的自检扫描 ，它是随着磁盘标准ATA的不断发展而引入的，并且在随着ATA标准的不断发展而改善[19]，下面我们简单的介绍一下SMART技术（见表 1-2）：

SMART 技术有各种各样的命令，比较重要的有以下几条：

表 1-2 SMART 命令表

状态	含义
SMART ENABLE OPERATIONS	使能所有 SMART 操作
SMART DISABLE OPERATIONS	禁止所有 SMART 操作
SMART READ DATA	读取设备的 SMART 数据，在这个命令中，可以通过 PIO-DATAIN 的协议获得 512 字节的 SMART 信息
SMART READ LOG	读取设备的 SMART 错误日志
SMART ENABLE OPERATIONS	使能所有 SMART 操作
SMART RETURN STATUS	可以直接判断硬盘运转是否正常

利用 SMART 命令，可以得到磁盘自检测的结果，配合监控报警系统，可以检测磁盘的状态，形成报警，提醒用户及时换盘。但是，SMART 信息各个厂商定义的不同，需要分别获取。

现在市场上有不少利用SMART技术做成的软件，用来对磁盘的运行状况进行监视，以下是一种叫做Active SMART的软件的运行界面[20]：

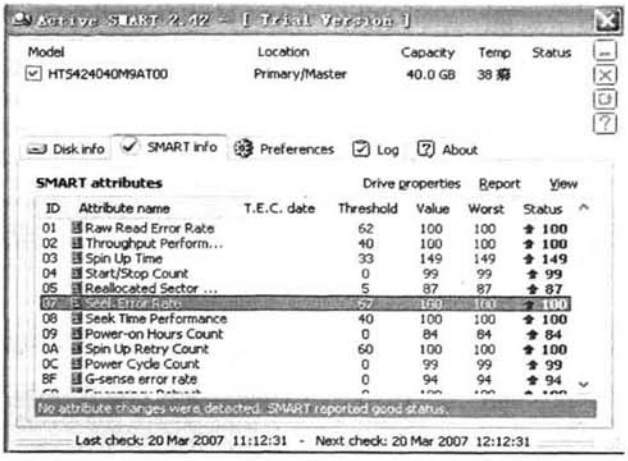

图 2-6 Active Smart 运行图

从上边的运行界面中我们可以看到，Active SMART 列出了磁盘的许多重要属性的当前值以及参考值，这些值就是我们进行磁盘信息提取与解析的基础。

但是与此同时我们也发现当前比较流行的 SMART 工具（例如 ACTIVE SMART）基本上都具有如下的缺陷：

- 当前磁盘工具只是给出了简单的参数信息名称，而对于这个参数的具体含义并没有说明。
- 磁盘的信息是可以根据其不同的含义而进行分类的，并且对于不同的磁盘信息，我们实际上应该在出错之后采取不同的措施。
- 各个磁盘制造商之间虽然可能都采用了 SAMRT 技术作为其磁盘监控保护的一种方式，但是磁盘商之间的标准并不统一，可能对于磁盘的同一个属性却采用了不同的名称，不利于对各类的磁盘进行统一处理。
- 虚拟存储往往是一个 RAID 系统，所以有磁盘通道卡或者是 RAID 卡存在，这类卡片禁止了外界通过 SMART 命令与 RAID 阵列里的磁盘直接交互，而是提供了自己的一套磁盘监控保护的命令，外界的必须使用磁盘卡的命令来得到阵列内磁盘的 SMART 信息，这就更造成了不统一。

本系统正是要从这些不足之处入手，通过对磁盘阵列的底层接口的统一以及对于不同的 SMART 参数的具体分析，来构建一个更为有效的监视系统。

2.6 SOAP 协议

简单对象访问协议（SOAP）是一种轻量的、简单的、基于 XML 的协议，它被设计成在 WEB 上交换结构化的和固化的信息。SOAP 可以和现存的许多因

特网协议和格式结合使用，包括超文本传输协议（HTTP），简单邮件传输协议（SMTP），多用途网际邮件扩充协议（MIME）。它还支持从消息系统到远程过程调用（RPC）等大量的应用程序。简单对象访问协议是在分散或分布式的环境中交换信息的简单的协议，它包括四个部分：

SOAP 封装(envelop)，封装定义了一个描述消息中的内容是什么，是谁发送的，谁应当接受并处理它，以及如何处理它们的框架。

SOAP 编码规则(encoding rules)，用于表示应用程序需要使用的数据类型的实例。

SOAP RPC(RPC representation)，表示远程过程调用和应答的协定。

SOAP绑定(binding)，使用底层协议交换信息[21]。

本系统中为了在整个系统中保持数据的一致性，防止数据冗余，在初始化磁盘对象的时候，采用的都是 SOAP 协议的远程对象方式，这样就很好的保持了数据的一致性。

2.7 CGI 技术

CGI(Common Gateway Interface)是 HTTP 服务器与你的或其它机器上的程序进行"交谈"的一种工具，其程序必须运行在网络服务器上，绝大多数的 CGI 程序被用来解释处理来自表单的输入信息，并在服务器产生相应的处理，或将相应的信息反馈给浏览器，也就说 CGI 程序使网页具有交互功能。CGI 的具体处理步骤如下：

⑴通过 Internet 把用户请求送到服务器。

⑵服务器接收用户请求并交给 CGI 程序处理。

⑶CGI 程序把处理结果传送给服务器。

⑷服务器把结果送回到用户[22]。

本监视系统所要实现的功能之一就是允许管理员在 WEB 页面上选择监视图像的各种类型（日，月，年等），所以需要 CGI 技术来在服务器上运行相关的程序来返回管理员所希望得到的图像。需要强调的一点是：CGI 程序不是放在服务器上就能顺利运行，如果要想使其在服务器上顺利的运行并准确的处理用户的请求，则须对所使用的服务器进行必要的设置，在本系统中采用的是 APACHE 服务器，所以也有必要对 APACHE 服务器的一些参数进行具体的配置与修改。

第三章 系统的基本架构

3.1 监视模式的选择

3.1.1 监视的实现方式

在监视的实现方式方面，采取软件监视的策略，主要依据如下：

- 考虑到对于虚拟存储的监视主要是对其系统的性能的监视，包括(CPU，MEMORY，DISK，NETFLOW 等)，而并不关心其底层的诸如电流、电压、风扇等硬件信息，所以并不需要采取硬件监视的方式。
- 虚拟存储是对实时性要求很高的系统，所以其监视系统必须是相对简单，消耗系统资源少的一个系统，虽然软硬件结合的方式在理论上来讲是理想的监控系统的实现方式，但是其复杂性太大，成本过高。
- 虚拟存储的监视系统只是其管理系统中的一个子模块，必须考虑与管理系统中的其余模块的兼容性，由于本监视子系统将依附的虚拟存储管理模块完全采取了软件实现的方式，所以本监视子系统也采取了软件的实现方式。

3.1.2 监视领域

在监视领域的选择上，我们专注于性能监视。因为虚拟存储系统是一个对节点的 CPU 使用率，网络流量情况和磁盘阵列情况比较敏感的系统，所以我们有必要重点监视系统各个节点的性能如何，以便于我们及时的发现系统的性能瓶颈，从而采取不同的应对措施。至于监视领域中的应用程序监控，进程监控等方面，由于并不是虚拟存储系统所关注的要点所在，所以在本系统中并没有实现。

3.1.3 监视软件

在监视软件的选择上，我们采取 MRTG+RRDTOOL 的架构方式。当前可以对于系统性能进行监视的比较流行的软件主要有两种：Ganglia 与 MRTG。在对比两种监视软件各自的特点之后，本系统采取 MRTG 作为系统的实现方式，具体原因如下：

- Ganglia 对于低层数据采集使用的是自己的私有协议，而 MRTG 采用的

是通用的 SNMP 协议，所以其通用性更好。

- Ganglia 主要是针对分布式监视开发的一个监视软件，它的体系结构比较复杂，把它作为对实时性要求很高的虚拟系统的监视软件不合适。
- 监视子系统只是虚拟存储管理系统的子模块，必须考虑到它与已有系统的兼容性，而已有系统的数据采集与数据交换，基本上是在 SNMP 的协议下开发的，所以采用 MRTG+SNMP 的策略，保证了监视系统与已有系统很好的兼容性。

3.1.4 监视对象

由于虚拟存储是对网络性能以及节点的CPU，内存，磁盘容量比较敏感的系统，所以我们选取上述参数作为本监视系统的监视对象[23]，更为重要的是，由于磁盘阵列是整个虚拟存储的重要组成部分，而磁盘温度的高低又往往反映了磁盘当前的状态是否正常，所以我们也选取磁盘的温度作为我们的监视对象。不同的是，网络流量，节点的CPU，内存，磁盘容量等值，我们都可以通过SNMP协议，从标准的MIB库中直接获取，而对于磁盘的温度，则必须由我们自己加以提取，这里我们选择磁盘工具SMART来获得磁盘的温度值，然后再将其通过SNMP协议加以封装，使其标准化，具体的细节问题将在下章中加以解释。

3.2 系统的基本结构

虚拟存储的具体架构可能不同，但是无论采取何种架构方式，从用户的角度来看，其架构方式是基本固定的[24]。一般即为下述的构造方式：

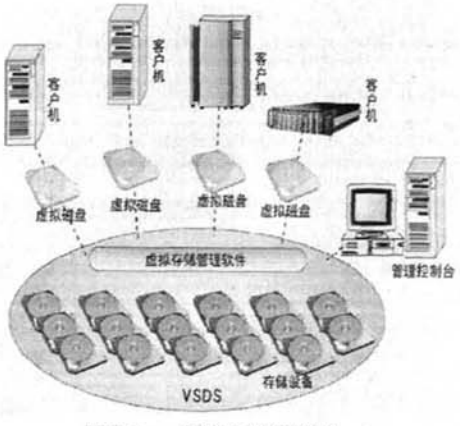

图 3-1 虚拟存储结构图

其中的虚拟存储管理软件是对虚拟存储系统进行全面管理的重要部分，正是在虚拟存储管理软件的作用下，用户才能看到一个与物理存储介质无关的存储平台，虚拟存储管理软件的功能模块有很多，包括卷的创建与删除，RAID 管理，设备的动态添加与删除，设备监视等，而本文的要解决的核心问题，就集中在了虚拟存储的监视系统中。

虚拟存储的监视系统从总体上来讲应该完成监视数据的采集与分类，监视图像的呈现以及监视错误的报警等功能，基于这一基本理念，设计了如下的虚拟存储监视系统的基本架构图：

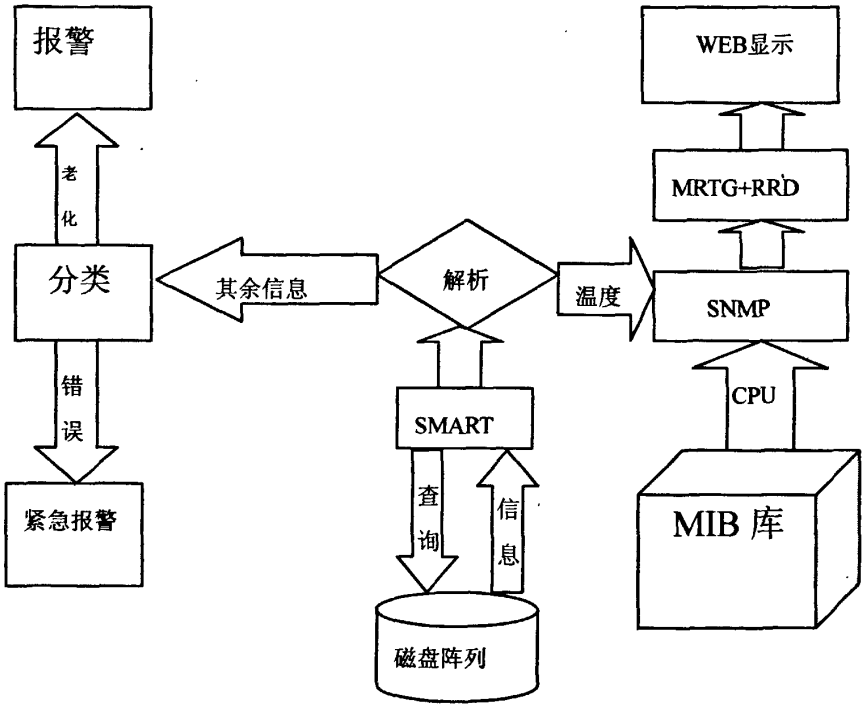

图 3-2 监视系统结构图

以上就是本虚拟存储的监视系统的整体架构方式，按照其各自功能的不同，系统可以分为以下几个模块：

- 信息的提取与查询。
- 信息解析的分类与报警。
- 利用监视数据形成监视图像。
- 监视页面的实现。

以上四个模块即为本监视系统的四个基本的模块，这些模块之间相互耦合，

完成了虚拟存储监视系统的主要功能。

第四章 系统的具体实现

接下来我们将从本系统的四个不同的模块入手介绍各个模块的结构以及模块之间的耦合方式。

4.1 信息的提取与查询

4.1.1 信息的提取

本监视系统所要监视的监视项主要有五种：CPU 负载，网络流量，磁盘容量（总量和已使用量），内存容量（总量和已使用量），磁盘相关信息。这类信息根据其数据来源的不同可以分成两类：

- CPU 负载，网络流量，磁盘容量（总量和已使用量），内存容量（总量和已使用量）的数据可以直接从 SNMP 的 MIB 库中得到。
- 磁盘相关信息则需要利用 SMART 工具得到。

从之前的关于 SMART 的介绍我们可以知道，有许多的磁盘监视软件可以列出磁盘的许多重要属性的当前值以及参考值，由于我们的系统建立在 LINUX 的操作系统上，基于上述考虑，我们设计如下方案的获取磁盘温度的数值：

- 利用 Smartmontools 获取磁盘的 SMART 信息。Smartmontools 是运行在 Linux 平台下的获取磁盘 SMART 信息的软件，它将磁盘的信息以一定的格式显示出来。
- 由于不同的磁盘厂商在 SMART 信息的使用方面并没有完全的达成一致，但是有一点，即在 SMART 信息的 ID 方面是严格遵照 SMART 标准的，所以我们以磁盘信息的 ID 为准，利用 Perl 脚本解析通过 Smartmontools 获取的磁盘信息，获取磁盘的温度值。

4.1.2 信息的查询

信息的查询是指系统能够根据管理控制台的查询要求得到对应磁盘的 SMART 信息，在经过 Perl 脚本的解析处理（主要是对不同磁盘厂商的信息格式进行规整）之后，将处理后的磁盘 SMART 信息返回给控制台。

4.2 信息解析分类与报警

这里的信息，主要是指通过磁盘 SMART 工具得到的磁盘信息，如果我们仔细分析 SMART 工具得到的磁盘信息，我们就会看到当前的 SMART 工具都有着很大的局限性：

- 当前磁盘工具只是给出了简单的参数信息名称，而对于这个参数的具体含义并没有说明。
- 磁盘的信息是可以根据其不同的含义而进行分类的，并且对于不同的磁盘信息，我们实际上应该在出错之后采取不同的措施。

基于以上两点，本系统对得到的磁盘 SMART 信息进行解释，进而进行分类，对于不同类的信息，采取不同的处理方式，具体分析如下：

第一类磁盘信息，我们称之为错误（pre-fail）信息，这些信息具有如下的特点：

- 它们反映的是磁盘运行中很重要的一些参数的数据情况，一旦这些数据发生变化，则往往意味着磁盘已经发生了严重的错误，可能会在 24 小时内崩溃，需要马上更换。
- 对于这些信息，我们采用更高级，更紧急的报警方式。

表 4-1 磁盘 pre-fail 属性描述表

事件名称	事件描述	产生原因	应对措施
Raw_Read_Error_Rate	RawReadErrorRate 发生变化	读数据错误	及时备份数据
Spin_Up_Time	磁盘运转发生异常现象	磁盘中部件有异常	及时备份数据
Reallocated_Sector_Ct	磁盘表面可能出现坏块	磁盘表面发生退磁或磨损	及时备份数据或更换磁盘
Seek_Error_Rate	磁盘寻道出错率上升	电机等机械部件有异常	及时备份数据
Spin_Retry_Count	磁盘启动重试的次数变化	磁盘休眠后启动旋转有问题	及时备份数据或更换磁盘
Calibration_Retry_Count	磁头校准发生异常	电机等机械部件可能有异常	更换磁盘
Multi_Zone_Error_Rate	Multiple Zone 出错率发	未知	及时备份

	生变化		数据或更换磁盘

第二类磁盘信息，我们称之为老化（old-age）信息[25]，这些信息具有如下的特点：

- 它们反映的是磁盘运行中相对次要重要的一些参数的数据情况，一旦这些数据发生变化，则往往意味着磁盘已经使用了很长时间，发生了比较严重的老化，但是不会马上崩溃。
- 对于这些信息，我们采用不是那么紧急的报警方式。

表 4-2 磁盘 old-age 属性描述表

事件名称	事件描述	产生原因	应对措施
StartStopCount	磁盘启停次数	老化	及时备份数据
PowerOnHours	磁盘累计加电时间	老化	及时备份数据
PowerCycleCount	磁盘加电次数	老化	及时备份数据
TemperatureCelsius	磁盘温度	老化	及时备份数据或更换磁盘
ReallocatedEventCount	磁盘表面损伤	老化	及时备份数据或更换磁盘
CurrentPendingSector	当前出现问题的扇区数目	磁盘表面发生损伤	及时备份数据或更换磁盘
OfflineUncorrectable	离线扫描后发现不可修复错误的百分比	未知	及时备份数据或更换磁盘

针对以上两类不同的磁盘信息，我们定义了不同的事件类别分别对其进行报警，现示意如下：

第一类，pre-fail 信息的事件模型

```
30001 => {
        name        => "RawReadErrorRateChanged",
        description => "smart 信息 RawReadErrorRate 发生变化",
        cause       => "读数据错误",
        action      => "及时备份数据",
    }
```

第二类，old-age信息的事件模型[26]

```
30009 => {
        name        => "PowerOnHours",
        description => "磁盘累计加电的时间",
        cause       => "老化",
        action      => "及时备份数据",
    }
```

对于以上两类事件，我们可以按照其紧急程度定义不同级别的报警处理。

4.3 利用监视数据形成监视图像

4.3.1 基本架构

本系统的最终目的是将得到的系统信息以监视图像的方式呈现出来，采用的基本架构为SNMP+MRTG+RRDTOOL[27]，具体的流程可以用下图来表示：

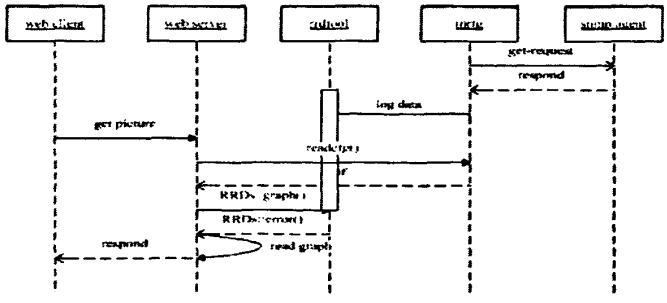

图 4-1　图形呈现流程图

- 运行在设备上的 SNMP AGENT 在接收到 MRTG 请求后构建 PDU 返回给 MRTG，MRTG 作为一个后台进程每隔一定的时间请求一次数据。
- MRTG 收到数据后调用 RRDTOOL 存储数据。

- 一个 WEB 客户请求一张监视页面图。
- WEB 服务器读取 MRTG 的配置文件，并将其转换为 RRDTOOL 的参数格式。
- WEB 服务器调用 RRDTOOL 生成一张新的图片，存储在特定的位置。
- WEB 服务器将图片读取，转发给 WEB 客户。

4.3.2 系统搭建的要点分述

在简单的介绍了上述的基本软件的相互配合使用的情况之后，我们专注于本系统中个性化的需求。我们知道，由于虚拟存储是对网络性能以及节点的 CPU，内存，磁盘容量比较敏感的系统，所以我们选取上述参数作为本监视系统的监视对象，更为重要的是，由于磁盘阵列是整个虚拟存储的重要组成部分，而磁盘温度的高低又往往反映了磁盘当前的状态是否正常，所以我们也选取磁盘的温度作为我们的监视对象。不同的是，网络流量，节点的 CPU，内存，磁盘容量等值，我们都可以通过 SNMP 协议，从标准的 MIB 库中直接获取，而对于磁盘的温度，由于它并没有在标准的 MIB 中定义，所以 SNMP 协议并不直接支持对于磁盘温度的获取，而 MRTG 的使用是严格的建立在 SNMP 协议的基础上的，所以我们需要在 MIB 库中动态添加磁盘温度项，并且对于与其相关的一些配置文件做相应的修改，我们需要分几个步骤，来完成上述的目的：

1. 编写 MRTG 配置文件

我们给出本系统配置文件的一个片断如下：

Mrtg.cfg

```
Interval: 5
LogFormat: rrdtool
Refresh: 300
Language: gb2312
PathAdd: /usr/local/bin/
.....
Target [10.10.113.15_network]: 2: public@10.10.113.15
PageTop [10.10.113.15_network]: Traffic Analysis
Maxbytes [10.10.113.15_network]: 25000000
Monitor*dontshowindexgraph [10.10.113.15_network]: yes
.....
Target[10.10.113.15_disktemp8]: `/usr/local/vsds/var/disk.sh 10.10.113.15 8`
PageTop [10.10.113.15_disktemp8]: Disk Temperature
```

Options [10.10.113.15_disktemp8]： growright，gauge，nopercent
Maxbytes [10.10.113.15_disktemp8]： 200
Title [10.10.113.15_disktemp8]： Disk Temperature

以上就是本系统的配置文件的一个非连贯的片断截取，我们可以利用表格来解释上述配置文件的具体格式：

- 一些共有的基本属性，它对配置文件中所有的项都起作用。

表 4-3 MRTG 公有属性配置表

Refresh	Browser(Netscape) reload page 的时间间隔，以 s 为单位(缺省 300)	Refresh： 600
LogFormat	设置该参数为 rrdtool，MRTG 就依赖 rrdtool 去做它的 logging，创建相应的数据库 rrd，并定期更新数据库	LogFormat： rrdtool
WorkDir HtmlDir ImageDir LogDir	指定了 logfiles 和 webpages 的存放位置。在全局关键字中 WorkDir 权限最高，即如果存在 WorkDir，后面三者的赋值就无效了，若不存在 WorkDir，就必须对后面三者全部赋值，缺一不可	Htmldir：/var/www//mrtg/ Imagedir：/var/www/mrtg/images Logdir：/var/www/mrtg/logs
LoadMIBs	指定要使用的 SNMP MIB 节点的说明	LoadMIBs： /usr/share/snmp/mibs/UCD-SNMP-MIB.txt

- 一些特有的绘图规则，它针对每个不同的监视项起作用，所以需要对每个监视项进行不同的配置[28]。

表 4-4 MRTG 专有属性配置表

PageTop	想要加到生成的HTML page 顶端的文字	PageTop[myrouter] : <H1>Traffic Analysis for ETZ C95.1</H1>
MaxBytes	两个变量所能达到的最大值，这个值要根据所监视的项的实际值小心设定	Maxbytes[10.10.113.15_network]: 25000000

- 针对自身需求自定义的规则，MRTG 允许自写扩展规则，使得我们可以根据自己的要求编写规则，这类规则一般在程序中用作分支语句的判断条件，用来针对不同的监视项，做出不同的处理，以下是本系统中自定义的一些规则：

表 4-5　MRTG 自写规则配置表

Monitor*dontshowindexgraph[10.10.113.15_cpu]: yes	不显示综合负载图
monitor*memused[10.10.113.15_memory]: yes	显示内存已使用量
monitor*cpu[10.10.113.15_cpu]: yes	标识监视项是 CPU
monitor*stackgraph[10.10.113.15_disk]: yes	使用堆栈的形式绘图
monitor*addtotal[10.10.113.15_disk]: yes	显示总量
monitor*next[10.10.113.15_cpu]: 1	标示是否还有后继参数需要在本图显示

需要特殊强调的是配置文件中的以下这条语句：

Target[10.10.113.15_disktemp8]: `/usr/local/vsds/var/disk.sh 10.10.113.15 8`

这条语句中有两点需要强调：

- /usr/local/vsds/var/disk.sh 表示接到 10.10.113.15_disktemp8 的请求的时候，需要调用 /usr/local/vsds/var/disk.sh 来返回结果，请求

10.10.113.15_disktemp8 在我们的程序中将被解析成如下请求：得到 IP 为 10.10.113.15 的设备上的磁盘位置号为 8 的磁盘的温度信息（disktemp）。这个信息的解析是由我们设计的程序 MONITOR.PM 来完成的，这个模块主要就是用来根据上层监视页面传来的命令与参数，来绘制相应的图像，并返回给上层界面。

- Disktemp8 中的 8 并不是固定的，可能是 1－15 内的任何一个数值，它代表的是本设备的磁盘槽中有哪些槽中存在磁盘，由于我们的虚拟存储系统应该是一个自动化的系统，所以 MRTG 的配置文件也是在整个管理系统加载的时候就自动生成的，所以在系统加载的时候，将对磁盘的槽位进行读取，并记录下那些在确实存在磁盘的槽位，然后利用这些信息，形成我们的 MRTG 配置文件。

2．编写 disk.sh 文件

Dish.sh 主要是用来通过 SNMP 向 MRTG 返回所需要的信息，它需要两个参数来完成其功能，在`/usr/local/vsds/var/disk.sh 10.10.113.15 8`中，10.10.113.15 是第一个参数，指示要获取的信息的 IP，8 是第二个参数，指示要获取该设备上具体哪块磁盘的信息，由于 dish.sh 程序很短，我们可以将其列出：

Disk.sh

```
#! /bin/bash
tempth=0
uptime="time"
if
    answer=`snmpget -v 1 -c public -m all $1   .1.3.6.1.4.1.1.9.0.4.0.1.1.1.2.$2
2>/dev/null`
then
    echo cat $answer | awk '/[0-9]+$/ {print $5}'
else
    echo 'UNKNOWN'
fi
echo $tempth
echo $uptime
hostname
```

需要解释的几点：

- 语句 answer=`snmpget -v 1 -c public -m all $1 .1.3.6.1.4.1.1.9.0.4.0.1.1.1.2.$2 2>/dev/null`表示，通过 snmpget 命令返回MIB的OID(MIB库中某个节点所对应的序列号)为.1.3.6.1.4.1.1904.打头的监视项的数据，这个监视项也就是我们所要监视的磁盘的温度，不过需要注意的一点是，.1.3.6.1.4.1.19040.所代表的磁盘温度的项并不是 SNMP 的 MIB 库中固有的，而是需要我们自己扩展定义，接下来我们会有详细的介绍。
- echo cat $answer | awk '/[0-9]+$/ {print $5}'是从得到的结果中解析出磁盘温度的数值，然后返回给 MRTG 来画图。

3．扩展 MIB 库，加入磁盘温度监视项

我们之前讲过，磁盘温度不是SNMP固有的MIB库中的一项，要想使用MRTG监视磁盘的温度，则必须对 MIB 库进行扩展，主要步骤如下：

- 增加 MIB 节点，扩展 MIB 库

我们看到在未扩展之前的MIB库的结构如下所示[29]：

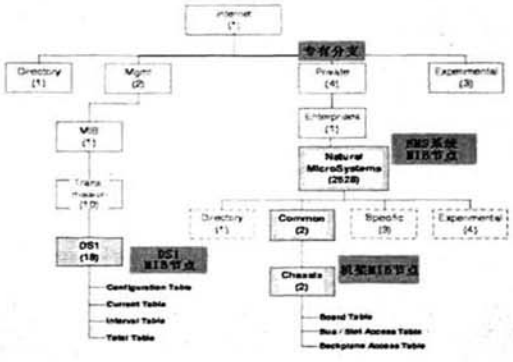

图 4-2　未扩展前的 MIB 库

我们在节点 19040 下对其进行扩展，扩展后的结构如下：

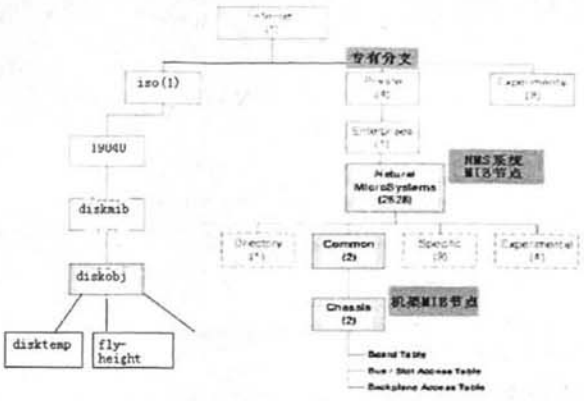

图 4-3 扩展后的 MIB 库

扩展的具体方法是按照 SNMP 定义的规则编写一个文本文件，将此文件置于 SNMP 运行时的 MIB 文件夹中：

- 修改 SNMP 配置文件，在其中添加如下语句，

pass_persist .1.3.6.1.4.1.19040.1 /etc/snmp/mib_disk.pl

这句话的作用在于 OID 为 .1.3.6.1.4.1.19040.1 打头的请求将通过 /etc/snmp/mib_disk.pl 来返回参数的数值

- 解析参数，返回结果

Mib_disk.pl 的作用就在于返回上述请求所要得到的参数的数值，需要注意的是它必须严格的按照 SNMP 规定的一些格式来编写。

经过上述的一些工作，我们就基本上可以实现对于包括磁盘温度在内的设备各项参数的监视，但是作为一个系统，我们还有一些问题需要考虑，主要是 MRTG 配置文件的生成时机。

4. MRTG 配置文件的自动生成

由于系统必须实现自动化，所以 mrtg.cfg 必须能够自动生成，而且要满足对系统的消耗最小，由此引入了 MRTG 配置文件的自动生成的问题，由于讲清楚这个问题需要我们对于监视系统的程序调用关系有一定的了解，所以我们先简单的介绍一下本系统的程序构造体系：

本系统的程序主要可以分为两个部分：MONITOR.PM 以及 Monitor.pl。其中 Monitor.pl 负责图像界面部分；MONITOR.PM 负责根据 Monitor.pl 传来的参数命令，结合 mrtg.cfg 中相关的规则调用 RRDTOOL 绘图，可以说 MONITOR.PM 负责的是产生图像的底层部分。当用户点击监视页面的某个按钮时，他就调用了 Monitor.pl 的某个函数，而 Monitor.pl 的函数又调用 MONITOR.PM 中的函数生成图像，这种界面显示与功能分离的程序结构，有利于程序的扩展，在后面我们将

比较详细的介绍 Monitor.pl 以及 MONITOR.PM 的一些具体细节，接下来我们先考虑配置文件的自动生成的相关问题了。

- 生成位置

配置文件的生成位置可以有以下三种方式：

①做一个 demon 程序，安装后开机后台运行，设定每隔一段时间检测一次，如发现系统中与当前配置监控节点有不同，就重新生成一次配置文件。

②把这个检测的程序嵌入到页面程序 Monitor.pl，随着每执行一次页面程序就执行一次检测，发现不同就重新生成配置文件。

③在client模块下新增一个client时候，提交就触发重新检测配置配置文件，达到检测目标与系统实际的一致[30]。

①②两种方式主要需要考虑怎样对系统资源耗费最少，因此需要考虑多长时间检测一次比较合适的问题。③方式可能和其它模块产生耦合，不利于以后的程序维护了。

经过对比分析，从最小化耗费资源角度考虑，利用方式③。在进行 insert、delete、update 的时候就调用配置文件生成程序，具体的操作为：

Insert：直接改动配置，增加该设备。

Delete：除了改动配置，删除该设备，还有删除相关的数据库。

Update：改动配置，还要判断 IP 是否改动，如果改动 IP，还需要把相应的数据库改名，以保证图形的连续性。

- 更新时机

一旦 mrtg.cfg 有变化，就需要重启 MRTG，这主要是由 MRTG 的特定决定的，命令“mrtg mrtg.cfg”的作用是在第一次时启动对相关监控设备的 snmpget，创建相关的数据 rrd，并在随后的时间更新这些数据库。这样一旦在 MRTG 运行过程中，对 mrtg.cfg 进行改动，如新增了一个监控设备，就无法创建对应 rrd，只能重启 MRTG。而删除一个设备，虽然设备图像在界面上没有了显示，但仍然继续在更新相关数据库，影响效率。至于更新操作，我们可以理解 insert and update，因此都需要重新生成 mrtg.cfg 并重启 MRTG。

至于何时重新生成 mrtg.cfg 及重启 MRTG 命令，考虑以下几个方案：

方案一：对原有数据库进行改动，新建一个表 flag，只有一个属性为时间的字段，有且仅有一个记录。再添加一个触发器，一旦对表 client or storage 进行 insert，delete，update 操作，就触发，将 current_time 加到 flag 的记录中。每 5 分钟运行一次的 WEB 界面，开始就比较 flag 中的时间与 mrtg.cfg 的最近修改时间，若前者>后者，就完成以下操作：

- make (“mrtg.cfg”)，将 mrtg.cfg 重新生成。

- remrtg (“mrtg.cfg”)，MRTG 重启。

方案二：改动的地方全部集中在 Monitor.pl 程序中，其实现逻辑如下：

- make(“mrtg2.cfg”)生成一个新的配置文件。
- if diff (mrtg.cfg mrtg2.cfg)与原有文件进行对比。
- 如果两个文件不同，则 remrtg (“mrtg2.cfg’)，重启 MRTG。

优劣比较：

方案一：更经济一些，不用每 5 分钟就生成一个新的文件，但需要对数据进行改动。

方案二：优缺点正好相反。

另外，两方案生成新的配置文件都是从数据库中获取监控对象，如果监控对象比较多，频繁的从数据库读取数据是对系统有比较明显的影响的，因此考虑以下方案：

方案三：只在对 client 和 storage 进行了 insert，delete，update 操作，mrtg.cfg 备份才进行更新，然后比较两个文件替换完成新的配置，这样解决了前两个方案对系统性能影响的最大问题。

方案四：将 MRTG 这部分全部放在 client and storage 完成，即每次做 insert，update and delete 操作，都修改 mrtg.cfg，重启 MRTG，而 Monitor.pl 不做任何操作。

综合考虑后决定采取方案四，理由如下：insert，update and delete 操作不是经常发生的动作，而且一般都是批量操作，Monitor.pl 却是 5 分钟运行一次，因此我们没必要把判断 mrtg.cfg，重启 MRTG 放在其中。

4.3.3 磁盘对象以及程序结构设计

在经过上述的分析之后，设计出本系统的磁盘对象以及 MONITOR.PM 的结构具体结构如下：

- **磁盘对象**

对磁盘 SMART 信息的管理分为如下几个部分：

1. 磁盘 SMART 信息的查询。
2. 根据磁盘 SMART 信息产生事件。
3. 磁盘温度的监视，包括 MRTG 和 SNMP 的配置。
4. 原有实现兼容通道卡环境和 Areca 卡环境，其中包括对原有实现的相关部分的修改[31]。

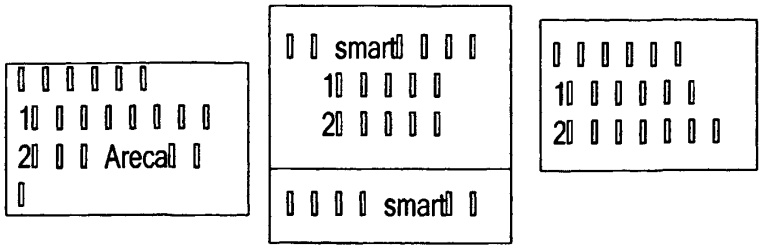

图 4-4　磁盘对象设计图

根据所要实现的功能，设计中定义了如下对象：

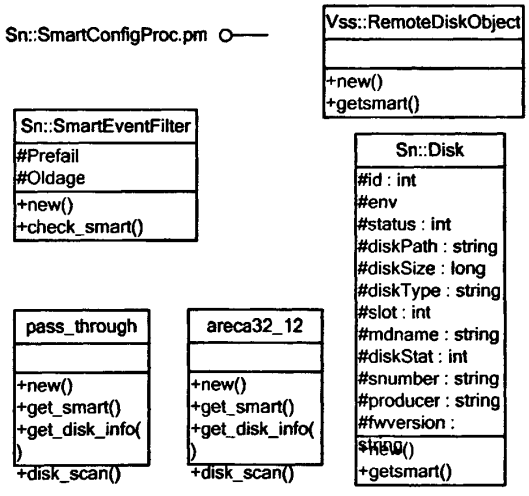

图 4-5　磁盘对象详细设计图

其中，以上各对象的功能为：

- areca32_12 和 pass_through 对象分别用于安装 Areca 卡和通道卡的环境。
- Sn::Disk 用来封装与磁盘相关的信息和方法，可通过该对象调用上述环境对象。
- Vss::RemoteDiskObject 实现在 Vss 端用于导出远程对象 Sn::Disk。
- Sn::SmartEventFilter 用于在 Sn 上产生与磁盘 SMART 信息相关的事件。
- Sn::SmartConfigProc.pm 中提供用于读取配置文件的方法。

出于数据一致性的考虑，本系统在磁盘对象的处理方面采取并行编程，具体为 SOAP 远程调用的方式，在管理端只初试化对象 Sn::Disk，具体的实际初始化任务交给设备端的 areca32_12 和 pass_through 对象完成。

对上述模块的一些简单的解释：

- VSS端也就是控制端，我们的虚拟存储的管理系统就运行在控制端，所以在控制端运行的程序和软件有Monitor.pl，MONITOR.PM，MRTG，RRDTOOL，SNMP CLIENT等；SN称为设备端，也就是各个具体的物理设备，运行在设备端上的程序有Mib_disk.pl，SNMP AGENT等，负责得到实际物理设备上的监视项的数值[32]，具体如图所示：

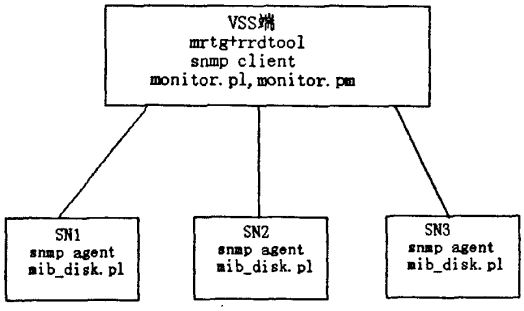

图 4-6　SNMP Client，Agent 示意图

- 我们在初试化磁盘对象的时候采用了远程调用的方式，在 VSS 端初始化磁盘对象的时候实际上并没有真正的构建一个磁盘对象，而是把这个任务交给具体的 SN 端来完成，这么做的原因主要是出于数据一致性考虑，避免 VSS 端和 SN 端出现冲突，但是由于不同的设备上的磁盘阵列通过不同的卡连接到设备上（areca32_12 卡，或者是 RAID 卡，也可能是直接连接），所以我们考虑在设备端保留一个配置文件，这样在 VSS 初始化磁盘对象的时候，就可以通过读取这个配置文件来知道这个磁盘阵列的具体环境，配置文件的另外一个优势就是其可扩展性，在有新的通道卡加入的时候，只要简单的修改其设备端的配置文件即可，下面列出了本系统所使用的磁盘配置文件：

env_for_disk.conf

#当前环境信息
package：　areca32_12
#时间间隔，单位分钟
interval：　60
#Pre-fail 事件列表
smartevent：1：0：3：0：5：0：7：0：10：0：11：0：200：0
#Old-age 类型 smart 信息的最大标准值*10%
maxnormvalue：4：10：9：10：12：10：194：15：196：20：197：20：198：20：

199：20

- **MONITOR.PM 模块**

前面我们已经提过，本系统所有的绘图工作实际上是通过一个MONITOR.PM 的程序来完成的，事实上 MONITOR.PM 充当了一个桥梁，它把MRTG，RRDTOOL 与本系统的界面程序 Monitor.pl 联结在了一起，使得整个监视系统成为一个实现了自动化的整体，其具体工作原理如下所示：

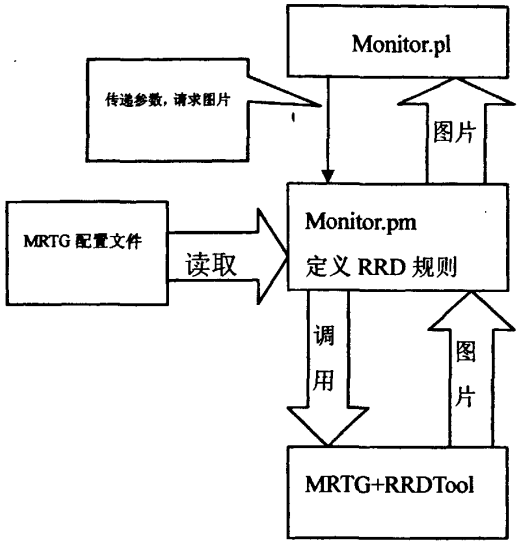

图 4-7　MONITOR.PM，Monitor.pl 工作示意图

利用监视数据形成监视图像这一功能其实开源软件 14ALL.CGI同样具有，14ALL.CGI 和MRTG 处理同一个配置文件mrtg.cfg，利用其中的信息来创建相应的index页面及图像页面[33]，但是考虑到与我们监视页面的特殊要求以及与整个虚拟存储管理系统的需求，我们设计了自己的MONITOR.PM程序。接下来我们给出MONITOR.PM的程序流程图：

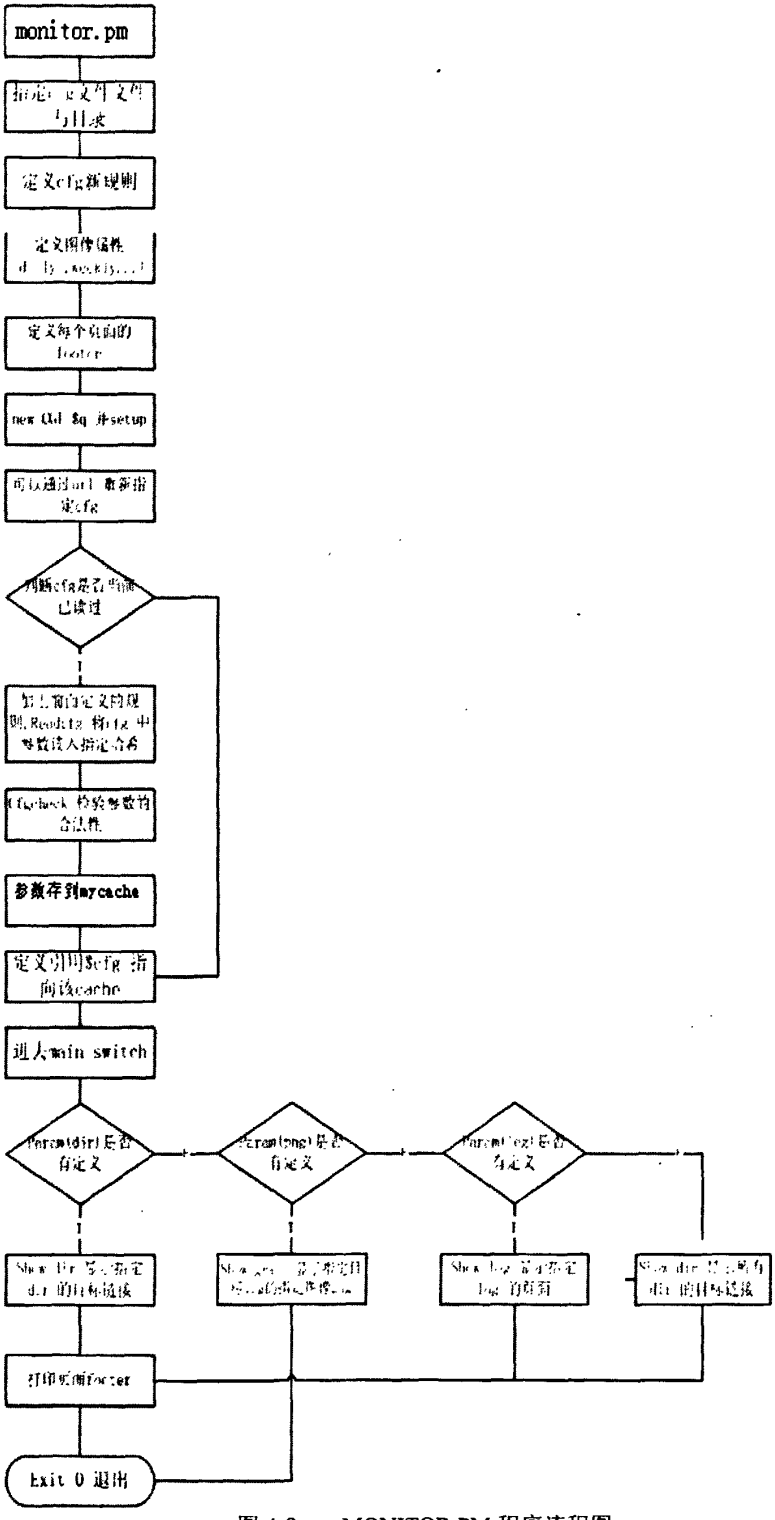

图 4-8 MONITOR.PM 程序流程图

MONITOR.PM是负责绘图部分的重要模块，绘图的实际部分是由MRTG调用RRDTOOL完成的，RRDTOOL具有非常丰富的绘图规则以及自己的一些绘图命令，包括RRDGRAPH，RRDUPDATE，RRDFETCH[34]等，但是MONITOR.PM作为更高级的程序接口，并不涉及上述具体调用，下表列出了其主要的函数接口：

表 4-6 MONITOR.PM 接口列表

函数名称	函数说明	输入参数	输出结果
new()	构造函数		
init_configurations()	初始化工作，把配置文件里的内容读到一个哈希结构中	MRTG配置文件	由配置文件得到的哈希
get_graph_name()	根据MRTG的配置文件调用RRDTOOL绘图	设备IP，监视项以及图片类型	绘制得到的图形
add_device()	用来添加一个监视设备	设备IP	修改后的MRTG配置文件
modify_device()	用来修改一个监视设备	设备IP	修改后的MRTG配置文件
remove_device()	用来删除一个监视设备	设备IP	修改后的MRTG配置文件
add_item()	添加监视项	@items：想要添加的监视项的数组	修改后的MRTG配置文件
modify_item()	修改监视项	@items：想要修改的监视项的数组	修改后的MRTG配置文件
remove_item()	删除监视项	@items：想要删除的监视项的数组	修改后的MRTG配置文件

init_restart()	每次添加，删除，修改设备或者监视项的时候调用，用来重起MRTG服务		

4.4 监视页面的结构

监视页面是监视系统与系统管理员之间的接口，一个美观，简单，实用的监视页面能够大大的简化管理员的系统管理工作，基于对本系统自身特征的考虑，设计如下的二层监视页面结构：

4.4.1 监视页面整体结构

点击监视菜单进入监视系统的一级页面：

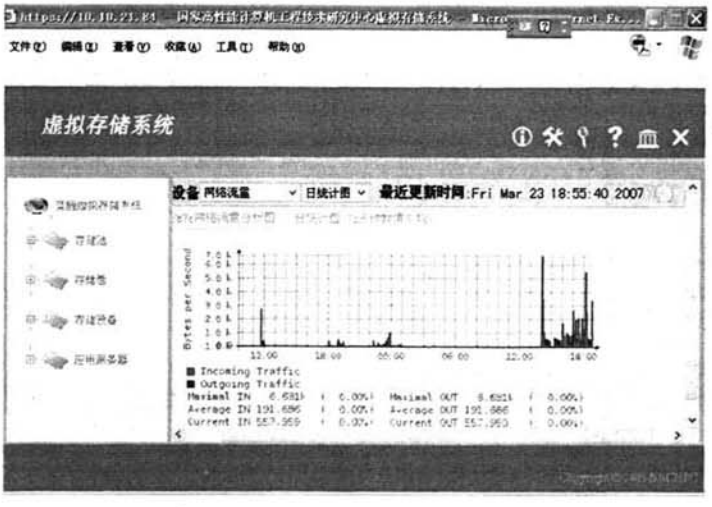

图 4-9　一级监视页面示意图

本监视页面默认显示的是各个设备的综合负载情况，便于进行各个设备之间的对比，下拉菜单提供了对于不同监视项以及监视图类型的选择，以便于我们纵向比较各个设备之间的同一监视项之间的运行情况。

- 初始进入页面显示的是当前系统的综合负载图像。
- 监视的最新更新时间，设计为RRDTOOL数据库的数据每五分种更新一次，WEB页面每也是5分钟更新一次。
- 右上方是两个超链接：刷新和返回，其中返回是返回其逻辑上的上一层。

- 左边显示设备名，右边显示监视图片，监视图片所代表的具体监视项以及监视图的类型（日，周，月，年）可以由两个下拉菜单来调整。
- “设备名”超链接：用来链接到一个页面，来显示这个设备的详细信息。
- 图像“超链接”，点击主界面中的图片，则进入一个二级子界面，此界面显示的是该设备的所有监视项的日图，同样设计一个下拉菜单，可以选择各个监视项的不同类型（日，周月，年）的图。同样设计刷新和返回两个超链接。点击一级界面中某个图像则进入监视系统的二级页面：

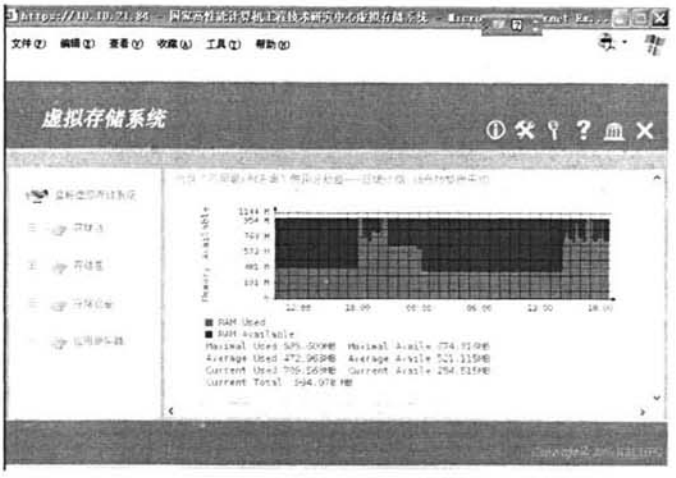

图 4-10　二级监视页面示意图

二级结构显示的是本监视设备的所有监视项的监视图像，便于对单个设备的不同监视项之间进行横向比较。

监视页面整体的结构即如上面所示，但是考虑到监视项的各自不同特点，我们有必要针对不同的监视图像做一些个性化的考虑。

4.4.2 监视项的图形呈现

- **网络流量**

取值方式：MRTG通过节点ifInOctets.2 & ifOutOctets.2 取值[35]。
图像类型：不封顶，area 显示 IN，LINE 显示 Out。
图形显示如下：

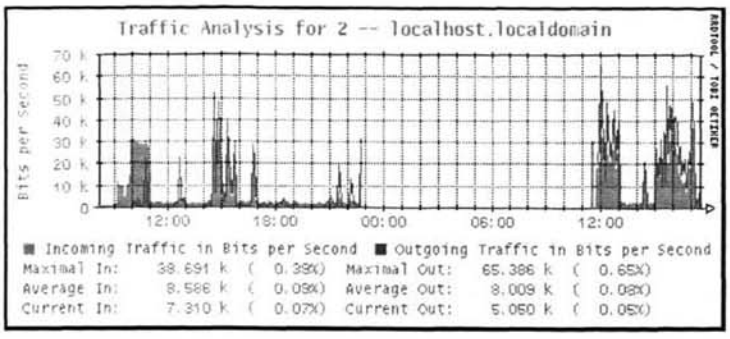

图 4-11　网络流量生成图（不封顶）

- **CPU Load**

取值方式：使用两个 TARGET 实现 (MRTG 一次只能取两个值，一大缺陷)，MRTG 把 ssRawCpuUser.0，ssRawCpuSystem.0 放入一个数据库，再把 ssRawCpuNice.0，ssRawCpuIdle.0 放入另一个数据库。

图像类型：用 100 封顶，将上面两个数据库画到一个图像中，一个绘图方式为 AREA，另外三个以 STACK 方式依次堆叠。

图形示例如下：

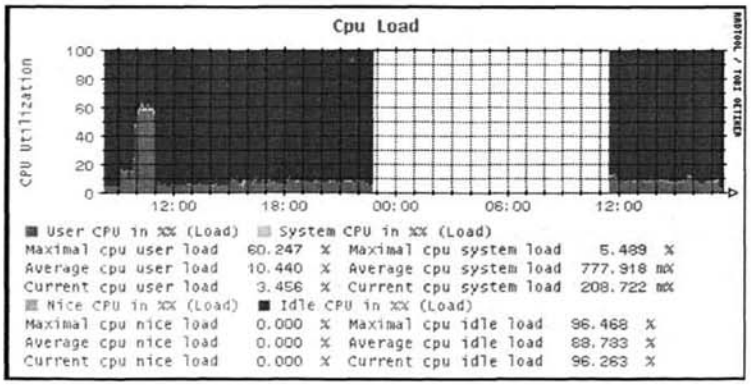

图 4-12　CPU 生成图（封顶）

- **内存使用情况**

取值方式：MRTG 通过节点 memTotalReal.0 & memavailReal.0 取值。

图像类型：不封顶，AREA 显示 avail，LINE 显示 Total，之间部分为 Used。

- **磁盘使用情况**

取值方式：MRTG 通过节点 dskUsed.1 & dskAvail.1 取值。

图像类型：不封顶，AREA 显示 Used， STACK 堆叠显示 Avail。

- **综合负载的引入**

在得到上述四组参数的值之后，我们考虑到引入综合负载这一指标来权衡系统当前的综合负载，具体方式是将 disk percent，net percent，memory percent 以及 cpu percent 四个值求平均，这样可以大概的表现当前系统的负载情况。

取值方式：利用已经取得的各个值，计算出各自的百分比，然后再求平均，上述计算需要在 RRDTOOL 的语法规则下进行。

图像类型：用 100 封顶，曲线显示综合负载情况。

对于上述分析的一个简单说明：在上边的分析中我们看到了 AREA，LINE 等术语，这些其实是 RRDTOOL 绘图时的参数，AREA 代表绘图时将数值下方全部涂满，LINE 代表只绘制代表图形的曲线，STACK 代表采用堆栈叠加的方式绘图，即将新的图形绘制在已有图形之上，这里要指出的是 RRDTOOL 的绘图参数规则是非常复杂的，而根据不同的 RRDTOOL 参数绘图又是 MONITOR.PM 所作的主要工作之一，具体的流程大概如下：

第一步：MONITOR.PM 读取 MRTG 配置文件，将 mrtg.cfg 的内容读取到程序定义的一个哈西结构中。

第二步：MONITOR.PM 解析得到的哈西结构，根据不同的监视项来选取不同的 RRDTOOL 规则绘图，这里需要说明的一点是，要想利用 RRDTOOL 绘图，基本上有两个主要的步骤：

- 创建 RRDTOOL 数据库，并将需要绘图的数据插入到 RRDTOOL 数据库中。
- 根据 RRDTOOL 数据库中的数据，根据不同的绘图需要，采用不同的 RRDTOOL 参数绘图。

在以上两个步骤中，第一个步骤是通过配置mrtg.cfg中的参数Interval和LogFormat来完成的，前者指出了数据将被多久更新一次，而后者则告诉MRTG不使用MRTG本身自有的LOG方式存储数据，而是采取RRDTOOL的方式存储数据。第二个步骤则是我们需要在MONITOR.PM程序中完成，MONITOR.PM需要调用RRDTOOL的绘图命令画图[36]，RRDTOOL的绘图命令格式如下：

---

rrdtool graph filename[-s|--start seconds][-e|--end seconds] [-x|--x-grid x-axis grid and label][-y|--y-grid y-axis grid and label] [-Y|--alt-y-grid][-R|--alt-y-mrtg][-A|--alt-autoscale][-M|--alt-autoscale-max][-N|--no-minor][-v|--vertical-label text][-w|--width pixels][-h|--height pixels] ..... [DEF：vname=rrd：ds-name：CF] [CDEF：vname=rpn-expression] [PRINT：vname：CF：

format] [GPRINT：vname：CF：format] [COMMENT：text]

上述即为RRDTOOL绘制一幅图所需要的命令，我们可以看到：RRDTOOL有着非常复杂的参数，事实上也是这样，要利用RRDTOOL绘制一幅完整的图，需要对图形的各个方面进行配置，上述还不是完整的一条GRAPH命令，事实上需要的参数会更多，所以接下来简单的介绍一下在本系统中所用到的一些RRDTOOL的参数配置[37]：

表 4-7　系统中 RRDTOOL 规则列表

参数	参数说明
filename	给出图形名字
-s|--start seconds	绘图数据的起始时间，默认是一天前，单位是秒
-e|--end seconds	绘图截止时间，默认是当前时间
-A|--alt-autoscale	自动调整显示细微差别譬如象画 260 + 0.001 * sin(x) 这样的图时，避免几乎画出的曲线几乎是直线的情况
-v|--vertical-label text	图像左边垂直方向的标签文字，通常用来说明使用的单位
-w|--width pixels	画图区域的宽度，影响到图片尺寸。默认为 400 象素
-h|--height pixels	画图区域的高度，影响到图片尺寸。默认为 100 象素
DEF：vname=rrd：ds-name：CF	定义从 RRDTOOL 中得到的原始数据的别名或虚名
CDEF：vname=rpn-expression	通过一个计算表达式增加新的虚拟数据源，表达式中的数据必须是前面已经定义的 DEF

	或者是 RRD 中的 ds-name
GPRINT：vname：CF：format	以指定的格式把数据输出

上述说明只是一个示例性的说明，实际使用时候要使用的参数个数以及使用方式都远比上述图表中所列的复杂，可以说 RRDTOOL 中精华的部分在其 CDEF 中，正是由于这个参数，我们可以根据数据 RRDTOOL 数据库中的数据源来定义我们所希望绘制的新的数据，以下是本系统中为得到 CPU 负载这一数据项所定义的 CDEF：

CDEF：p1=in0，in0，in1，out0，out1，+，+，+，/，100，*

CDEF：p2=out0，in0，in1，out0，out1，+，+，+，/，100，*

AREA： p1#006600： cpu user load

STACK： p2#ffff00： cpu systerm load \\1

GPRINT： p1： AVERAGE：average cpu user load%831f%s%%

GPRINT：p2：AVERAGE：averagecpusystemload%8.3lf%s%%\\1

GPRINT：p1：LAST：current cpu user load%8.3lf%s%%

GPRINT：p2：LAST：current cpu system load%8.3lf%s%%\\1

GPRINT：p1：MAX：max cpu user load%8.3lf%s%%

GPRINT：p2：MAX：max cpu systems load%8.3lf%s%%\\1

对于上述 CDEF 定义我们做以下解释：

- CDEF：p1=in0，in0，in1，out0，out1，+，+，+，/，100，*是一个后缀表达式，写成我们正常的表达式为：P1=in0/(in0+in1+out0+out1)*100，也就是说 p1 表示的是 in0 占总数的百分比，而 in0 代表的是 CPU 时间片中的用户占用时间，in1 代表的是 CPU 时间片中的系统占用时间，out0 代表的是 CPU 时间片中的优先级调度占用时间，out1 代表的是 CPU 时间片中的空闲时间，CPU 的时间片基本就是分成这四块，所以 p1 就代表 CPU 的用户使用百分比，同理 p2 代表的是系统占用时间的百分比，以次类推。
- AREA： p1#006600： cpu user load表示采用AREA方式绘制曲线p1，也就是说，对于小于p1 的部分也用#006600 颜色填涂，STACK： p2#ffff00： cpu system load表示将p2 的图形并不是从 0 开始算起，而是从图形p1 算起，采取叠加的方式绘制，由于p1，p2，p3，p4 均是百分比表示，所以当我们把p1，p2，p3，p4 都绘制在同一个图形上时，正好是以 100 封顶的，也就是下面的图形[38]：

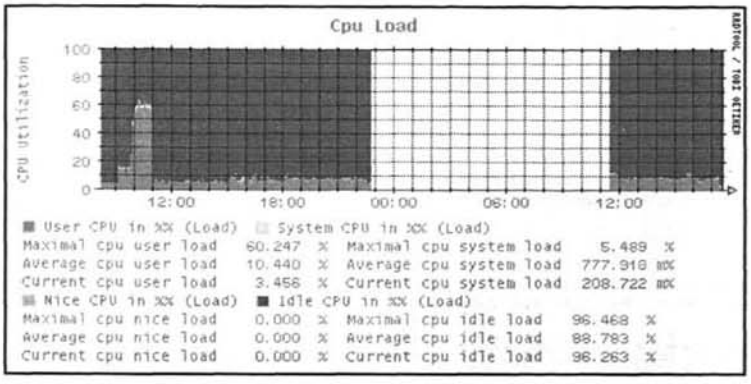

图 4-13 CPU 负载图

- GPRINT: p1: AVERAGE: average cpu user load%8.3lf%s%%是用来在图形底部加上注释的语句，AVERAGE表示要显示出CPU用户使用率的平均值。%8.3lf%s%%是对注释的左右空间的定义，同理GPRINT: p1: LAST: current cpu user load%8.3lf%s%%是要对CPU用户使用率的当前值做出注释，GPRINT: p1: MAX: max cpu user load%8.3lf%s%%是要对CPU用户使用率的最大值做出注释[39]，具体的注释效果可以参见上述图形。

4.4.3 页面监视程序结构

Monitor.pl 即为本系统的页面程序，Monitor.pl 采用的是 CGI（通用网关接口）技术，CGI就像是一座桥，把网页和WEB服务器中的执行程序连接起来，把HTML接受的指令传递给服务器，把服务器执行的结果返还给HTML页面 [40]，作为实现页面功能的Monitor.pl模块，它实现的一些基本函数如下：

表 4-8 Monitor.pl 接口列表

函数名称	函数说明	输入参数	输出结果
print_html_head()	主要用来生成html 的头信息	无参数	产生html头信息
show_index()	该函数在进入监视主界面的	无参数	显示各个设备的综合负载情

	时候被调用，用来显示一级界面		况
show_details()	该函数在点击“图片”的超链接的时候调用，用来显示这个设备的所有监视项的日（默认）或者月，周，年的监视图片	该图片所代表的设备的IP	设备的所有监视项的日（默认）或者月，周，年的监视图片
goto_device()	点下“设备名”链接的时候调用，用来转到一个显示该设备详细信息的页面去	设备IP	该设备详细信息

第五章 总结与展望

5.1 工作总结

虚拟存储是未来存储的发展方向，由于虚拟存储基本上是一个大规模的磁盘阵列，里边涉及到众多的磁盘以及网络性能等相关问题，所以有必要为其开发专门的监视系统，但是纵观当前的系统性能与磁盘监视技术，它们均存在一些明显的不足，本系统在现有的监视软件与技术的基础上进行了扩展于创新，现总结如下：

- 系统利用 MRTG+SNMP+SMART 协议的架构方式，通过扩展 SNMP 基本的 MIB 库（管理信息库）使得性能监视软件不单单是可以监视系统性能（CPU 利用率，网络流量等）方面的参数，而是把它和当前的磁盘监视软件结合起来，从而可以监视磁盘温度等磁盘的参数信息，实际上，利用本系统的思路，还可以加入许多 SNMP 标准库中并没有的监视项，例如风扇转速等。
- 在磁盘监视方面，系统将提取出来的 SMART 信息根据信息的不同特点分类并根据不同信息的重要程度进行不同级别的报警，并且在分析不同参数出错原因的基础上给出可能的解决办法，实现了监视与控制的统一。
- 本系统针对不同厂家的磁盘 SMART 信息格式不统一的现象，采取配置文件的形式，使得程序根据配置文件中的不同参数选择不同的处理方式，进而通过 Perl 脚本对返回的磁盘信息进行解析，并在此基础上形成统一的信息格式，使得用户不必关心底层的数据获取情况，从而统一了对于磁盘信息的提取与处理。
- 在系统集成的方面，本监视系统在加入到虚拟存储的管理软件后也基本运行正常，并且实现了与系统其余功能的很好的兼容，因此，本监视系统基本上在功能上以及实际运行中都基本的实现了预期的目标，也达到了预期的实际运行效果。

5.2 研究展望

虽然本系统基本上实现了预期的目标，但是在一些具体的方面仍然有继续深入的意义，可以作为后继的研究与开发的切入点，现分别列举如下：

- 在系统性能方面不单单局限于单纯的监视功能，可以加入分析与控制，例如若是监视系统显示系统的CPU利用率居高不下，则应该结合其余监视项的数值信息给出可能的原因，与此同时，还可以考虑到给系统添加上控制功能，当监视显示某些数值超出正常范围时，监视系统可以自动的做一些控制与调整。
- 而在磁盘监视方面，并不要单纯的采用SMART技术，可以考虑到把它和之前提到的磁盘状态控制以及磁盘命令队列的方式综合使用，SMART技术侧重于反映磁盘的运行状况，而磁盘状态控制技术以及磁盘命令队列技术则侧重于对磁盘阵列的运行状况进行调整与保护，通过这几种不同监视技术的综合使用，可以对磁盘阵列进行各种层次，各种粒度的监视与控制。
- 在监视项目方面，可以考虑扩充更多的监视项，SNMP的管理信息库（MIB）中提供的可监视项毕竟有限，不能很好的满足我们的要求，与虚拟存储的性能紧密相关的参数项有多项，包括机箱温度，风扇转速[41]等，这些对监视系统很重要的参数却没有定义在SNMP的管理信息库中，所以采用MRTG不能直接的得到其监视图像，但是本系统已经给出了添加监视项的一些基本步骤，虽然是以磁盘温度为例，但是其软件的架构方式与扩展MIB库的具体方法却具有普遍的意义，可以方便的用来扩展更多的监视项，这样，系统就可以方便自由的添加监视项，从而在各个侧面对虚拟存储系统进行监视。
- 系统采用的是MRTG+RRDTOOL的架构方式，但是这种架构方式一般只是适用于系统结构比较简单，磁盘数量也相对较少的情况，当需要监视的系统是一个很大的集群系统时，可以考虑之前提过的GANGLIA监视技术，GANGLIA具有层次化的体系结构，也正是为监视集群而设计的监视技术，所以它可以很好的满足大型集群的监视需求。

参考文献

[1]杨冬竹．虚拟存储技术及其应用探讨[J]．科技情报开发与经济，2006，16(15)：1—5．

[2]张丰毅．集群监视技术研究与系统实现[D]．北京：中国科学院高能物理研究所，2004．

[3]张春雅．存储虚拟化技术．中国信息导报[J]，2006，1(5)：1—3．

[4]陈得民．Scalable Distributed Monitoring- Ganglia[J]．CLUSTER/GRID 基础研习营，2006，1(1)：6—8．

[5]刘金哲．HPCS 监控系统的实现和改进技术研究[J]．超级计算机通，2006，l4(2)：17—18．

[6]Chichi(box) leangsuksun．NWPerf：Large Linux clusterMonitoring tool [R]．America： Chichi(box) leangsuksun，2006．

[7]Bruce Allen．Monitoring Hard Disks with SMART[EB/OL]．http://www.linuxjournal.com/article/6983，2004-01-01．

[8]吕斌斌．基于 SNMP 对服务器进行监管的研究和实现[J]．湖州师范学院学报．2006，28(1)：1—7．

[9]suningin．SNMP 协议详解[EB/OL]．http://suningin.blog.hexun.com/8324037_d.html，2007-03-20．

[10]黄珍，叶水生．基于 SNMP 协议的安全审计代理模型的设计与实现[J]．计算机与现代化．2006，9(1)：1—4．

[11]孙元军，童磊．MRTG 在校园网中的应用[J]．网络通信与安全，2006，7(1)：90—91．

[12]Oetiker，MRTG Index Page[EB/OL]．http://www.stat.ee.ethz.ch/mrtg/，2007-01-23．

[13]江魁，黄云森．基于 RRDTOOL 的网络性能监测系统实现[J]．中山大学学报，2002，41(1)：17—20．

[14]史国水，陈碧容．MRTG，RRDTOOL 在流量统计分析中的应用[J]．江西通信科技，2003，3(1)：16—17．

[15]Oetiker，RRDtool home page[EB/OL]．http://oss.oetiker.ch/rrdtool/，2007-05-23．

[16]何丽萍，刘立程．改进得基于 ganglia 的网格监控系统[J]．广东工业大学学

报，2006，23(1)：1—2.

[17]T13， Project 1532D，AT Attachment with Packet Interface – 7[S]，2004.

[18]马一力．减少磁盘磨损的方法[R]．北京：国家高性能计算机工程技术研究中心，2006.

[19]Bruce Allen．smartmontools Home Page[EB/OL]．http://smartmontools.sourceforge.net/，2007-05-06.

[20]王光临，用 Active SMART 将硬盘崩溃防患于未然[EB/OL]．http://tech.sina.com.cn/c/2003-01-24/17548.html，2003-01-24.

[21]简单对象访问协议[EB/OL]．http://wiki.ccw.com.cn/SOAP，2006-12-11.

[22]什么是 CGI[EB/OL]．http://www.cndw.com/tech/perl/2006032926231．asp，2006-03-29.

[23]陈得民．Scalable Distributed Monitoring- Ganglia[R]．北京：国家高速网络与计算中心，2006.

[24]郝志敏．虚拟设备存储系统介绍[R]．北京：国家高性能计算机工程技术研究中心．2006.

[25]PalickSoft team．SMART attribute meaning[EB/OL]．http//www.siguardian.com/products/siguardian/on_line_help/s_m_a_r_t_attribute_meaning.html，2000.

[26]卜庆忠．磁盘 SMART 信息管理设计[R]．北京：国家高性能计算机工程技术研究中心，2006.

[27]周利．Mrtg rrdtool integrating VSDS[R]．北京：国家高性能计算机工程技术研究中心．2006.

[28]Oetiker MRTG home page[EB/OL]．http://oss.oetiker.ch/mrtg/doc/mrtg-reference.en.html，2007-05-23.

[29]suningin．SNMP 及 MIB 相关知识[EB/OL]．http://suningin.blog.hexun.com/8324037_d.html，2007-03-20.

[30]曲海平，李健．CLUSTER 监控模块设计[R]．北京：国家高性能计算机工程技术研究中心，2004.

[31]卜庆忠．磁盘 SMART 信息管理设计[R]．北京：国家高性能计算机工程技术研究中心，2006.

[32]虚拟存储研究小组．VSS 结构设计[R]．北京：国家高性能计算机工程技术研究中心，2005.

[33]周利．MRTG RRDTOOL integrating VSDS[R]．北京：国家高性能计算机工程技术研究中心，2006.

[34]Oetiker，RRDtool home page[EB/OL]. http://oss.oetiker.ch/rrdtool/，2007-05-23.

[35]Sourceforge NET-SNMP group[EB/OL]，NET-SNMP home page[EB/OL]. http://net-snmp.sourceforge.net/，2007-05-01.

[36]Oetiker. RRDGraph[EB/OL]. http://oss.oetiker.ch/rrdtool/，2007-05-23.

[37]Oetiker. RRDtool home page[EB/OL]. http://oss.oetiker.ch/rrdtool/doc/index.en.html，2005-04-18.

[38]曲海平，李健. CLUSTER 监控模块设计[R]. 北京：国家高性能计算机工程技术研究中心，2004.

[39]Oetiker. RRDGraph[EB/OL]. http://oss.oetiker.ch/rrdtool/，2007-05-23.

[40]罗淳榕. 基于 CGI 的嵌入式远程控制系统[J]. 测控技术，2006，25(8)：50－52.

[41]马一力. 减少磁盘磨损的办法[R]. 北京：国家高性能计算机工程技术研究中心，2006.

发表论文和科研情况说明

本人于 2006 年 5 月至 2007 年 2 月在中国科学院计算技术研究所 国家高性能计算机工程技术研究中心从事客座研究生工作，期间在虚拟存储组(VSDS)从事虚拟存储的监视系统的设计与开发工作，并参与完成了国家 863 子软件专项－虚拟化网络存储功能软件的部分工作，主要包括：

- VSDS1.5：为虚拟存储管理软件加上了基本的性能监视功能。
- VSDS1.6：为监视系统加入了磁盘监视功能。

在经过上述两个项目之后，虚拟存储的监视系统的基本功能的设计与开发已经基本完成。

致 谢

本论文的工作是在我的导师冯志勇教授的悉心指导下完成的，冯志勇教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢两年来冯志勇老师对我的关心和指导。

冯志勇教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向冯志勇老师表示衷心的谢意。

冯志勇教授对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在实验室工作及撰写论文期间，邵彬、魏博等同学对我论文中的 SOAP 研究工作给予了热情帮助，在此向他们表达我的感激之情。

同时要感谢国家高性能计算机中心，正是在那里我完成了本系统的开发工作。

另外也感谢家人，他们的理解和支持使我能够在学校专心完成我的学业。