虚拟计算环境的运行时资源监控与内存泄漏检测技术

肖如良◎ 著

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

福建师范大学软件学院学术著作出版基金资助出版

虚拟计算环境的运行时资源监控与内存泄漏检测技术

肖如良 著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

在云计算相关的虚拟计算环境中，不仅要对系统的资源占用进行运行时监控调整，而且要对系统所发生的内存泄漏进行检测与分析，这种可靠性保障技术极具挑战性。本书在介绍了相关研究工作及基础内容之后，主要针对资源监控调整与内存泄漏检测问题，构建了 Xen 虚拟机管理器中的自定义超级调用体系，研究了虚拟计算资源，包括内存资源与 CPU 资源的监控与调整机制、运行时内存泄漏检测等方面的关键技术，提出了基于自省机制的内存泄漏检测确认规则、虚拟计算环境的资源监测收集策略和资源调整策略，基于 Xen 虚拟机管理器设计并实现了虚拟计算环境的资源监控与调整子系统 XResMonitor 与虚拟计算环境的内存泄漏检测分析系统 MLDA，最后介绍了 GDI 内存泄漏检测的工作。

全书包含了以上主要关键技术的实现思路与技术细节，可供计算机专业相关工程技术人员、研究人员参考。

图书在版编目 (CIP) 数据

虚拟计算环境的运行时资源监控与内存泄漏检测技术 / 肖如良著. —北京：电子工业出版社，2015.3

ISBN 978-7-121-25264-8

I. ①虚… II. ①肖… III. ①电子计算机—资源控制 ②电子计算机—泄漏检测 IV. ①TP306

中国版本图书馆 CIP 数据核字（2014）第 303433 号

策划编辑：王晓庆

责任编辑：底　波

印　　刷：三河市鑫金马印装有限公司

装　　订：三河市鑫金马印装有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　　邮编：100036

开　　本：787×980　1/16　印张：16　字数：333 千字

版　　次：2015 年 3 月第 1 版

印　　次：2015 年 3 月第 1 次印刷

定　　价：58.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010)88258888。

前　言

随着云计算技术的不断发展，现代软件的规模越来越大、复杂性越来越高，其可靠性也越来越难以保障，这种挑战性问题与云计算的虚拟化技术密切相关。在云计算的虚拟计算环境中，为了提高系统服务的可靠性，不仅要对系统的资源占用进行运行时监控调整，而且要对系统所发生的内存泄漏进行检测与分析，这种可靠性保障技术尤其是运行时资源调整与不停机服务器应用程序的内存泄漏检测极具挑战性。

内存泄漏是导致系统可靠性降低的重要因素之一，它不仅会使应用程序申请动态内存失败，导致服务中止，严重时会导致整个应用系统因资源耗竭而崩溃。对于长期运行在服务器上的操作系统及大量服务，发生内存泄漏会带来严重的后果，可能会导致系统级服务中止。内存泄漏问题长期以来一直困扰很多开发人员，常常使项目交付使用的时间大幅延期，不仅降低了开发效率，而且影响了最终软件产品的质量，进而给软件企业造成了直接的经济损失。

内存泄漏是指已经被申请的内存资源没有被合理地释放，从而导致这部分资源不能被系统重新利用的一种现象。内存泄漏广泛存在或者隐藏于程序之中，被泄漏的内存资源不能被重新利用且不再被访问。如果被泄漏的是虚拟内存，则内存泄漏会使得应用程序的虚拟内存空间耗竭，导致任务的中断和失败；如果被泄漏的是物理内存，则整个系统减少了能够使用的物理地址。内存泄漏会使应用程序申请动态内存失败，导致服务中止，严重时会导致整个系统因资源耗竭而崩溃。内存泄漏不仅会减少可用的内核虚拟内存地址空间，而且还会不断蚕食整个操作系统，系统可用的物理内存页面，导致整机不得不重新启动。对于运行时间很短的程序，内存泄漏一般不是问题，但是对于长期运行的程序，如通信领域软件、嵌入式敏感软件系统等行业性软件、运行在服务器上的服务和操作系统本身，内存泄漏会带来严重的后果，可能会导致系统服务中止而造成非常严重的经济损失，并可能带来严重的社会问题。

比如，通信软件作为一类特殊领域的软件，不仅存在监控程序、系统服务程序和应用程序等多种形态，而且它们的部署平台不尽相同，其运行的软件环境也千差万别。这些都给通信软件的内存泄漏检测带来了一系列的挑战性问题。例如，针对运行在不同操作系统、硬件环境等异构平台下的各种不同形态程序，如何有效地判定内存泄漏嫌疑，并准确地进行故障定位？如何高效地监控这些程序对有限内存资源的申请、使用和释放？又如何针对动态变化的运行平台，制定出科学合理的监控策略等。因此，研究和开

发跨平台的服务器应用程序内存泄漏检测的关键技术具有重要的意义。

随着云计算、大数据时代的来临，软件系统和硬件环境的异构性将更为突出、更加明显，软件企业（当然包括通信软件生产商）不得不面对这一新的计算特点，也不得不适应这一新的计算环境。将虚拟化技术应用于跨平台内存泄漏检测之上，以虚拟化平台构建内存泄漏检测系统，可以利用虚拟机管理器的一致性，在不同操作系统之间保持同样的实现方式。某些系统内核的关键数据对于猜测内存泄漏现象是否发生具有重大的意义，但通常情况下因操作系统对这些数据加以保护而不容易取得。利用虚拟机管理器特权则可以很容易获得这些数据。对于构建内存泄漏检测框架过程中遇到的域间通信，虚拟化计算平台上也可以利用这些操作系统运行在同一套硬件上的特性提供高效率的解决方案。

全书共 11 章，第 1 章介绍虚拟计算环境可靠性问题两个方面的主要内容，即资源的监控调整与内存泄漏检测的背景信息。第 2 章从内存泄漏的研究现状、虚拟计算环境下的内存泄漏检测方法、虚拟计算资源监测与资源调整等方面进行了概述。第 3 章介绍基于 Xen 虚拟机的虚拟计算环境中与内存泄漏检测相关的虚拟化基础理论。第 4 章基于 CSP 理论构建一种基于 Xen 虚拟化平台的内存泄漏检测模型，给出其实现方法；讨论模型内系统成员之间的交互，并基于 CSP 进行了正确性证明。第 5 章介绍虚拟化体系结构中的监控机制与自省机制，然后提出基于自省机制的运行时内存泄漏检测模型，并构建原型实现与实验分析，给出实验结果与分析。第 6 章在原有工作 XMMC 框架上，基于 Xen 虚拟计算环境提出 XResMonitor 资源监测系统框架。首先对资源监测调整系统 XResMonitor 的设计思路进行简单介绍，并给出其部署结构和总体结构；详细描述其模块结构及各模块的功能及各模块之间的关系，并介绍整个系统框架及各模块的工作流程，总结了 XResMonitor 系统的主要特色。第 7 章介绍 XResMonitor 系统的设计思路和功能模块。首先对 Xen 虚拟机现有的资源信息收集的策略进行介绍，并引出适合 XResMonitor 系统的资源信息收集策略；其次介绍 XResMonitor 系统资源监测的工作原理和关键技术实现，包括对内存资源和 VCPU 资源的监测；最后给出 XResMonitor 系统对内存资源和 VCPU 资源调整的工作原理和关键技术。第 8 章介绍 MLDA 系统框架模块级层面的内存泄漏检测分析关键技术。第 9 章从系统级角度，对虚拟计算环境的内存泄漏检测分析系统 MLDA 的技术实现进行介绍。第 10 章介绍 GDI 内存泄漏检测分析，是本书相关内容的补充。该章内容并不是基于 Xen 层面的成果，而是运行于 Windows 层面的检测与分析，因为集成在基于 Xen 虚拟环境上作为一个统一的工作，故放在后面章节进行介绍。第 11 章对本书进行的全面总结。

本书较全面地介绍了国内外内存泄漏检测的相关研究和成果，即虚拟化技术实现过程中需要解决的若干关键技术，这些技术支撑着虚拟机管理器功能的核心技术。本书还

主要研究了 Xen 虚拟计算环境下资源监控系统 XResMonitor 的资源监测收集策略和资源调整策略，设计并实现一种基于 Xen 虚拟计算环境的资源监控与内存泄漏检测的 MLDA 系统。

我们与中邮科通信技术股份有限公司合作，受福建省科技计划重大项目资助，开展了“跨平台内存泄漏检测系统的关键技术研发及产业化”（2011H6006）的立项研究，本书全面总结了本项目研究及其拓展的主要内容。

本书获得福建师范大学软件学院软件工程省级重点学科建设的资助，该资助为本书的顺利出版提供了非常重要的支持。在此，也特别感谢福建师范大学科研处的有关领导、软件学院姚志强院长、蔡声镇副院长、倪友聪副教授、杜欣副教授等有关领导和老师的支持和帮助，感谢软件学院提供了相关研究开发所必需的各种条件；姜军、胡耀、陈明珍、韩佳、李鹏澎等研究生同学，他们为本项目做了大量的研究工作，进行了编程实现，付出了艰辛和努力，为本书部分内容的撰写也提供了很重要的帮助。电子工业出版社的王晓庆编辑为本书的出版做了大量的工作，对此深表感谢。最后，特别感谢我的家人对我的支持。

本书错误之处在所难免，敬请各位读者批评指正。

肖如良

xiaoruliang@fjnu.edu.cn

2014 年 11 月 5 日

目　录

第1章 绪 论

云计算是近几年来一项划时代的IT技术，无论是产业界还是学术界都在深入开展研究。虚拟化技术已有较长的历史，虚拟计算环境正成为云计算的重要基础设施。基于虚拟化技术的云计算研究已经产生了很多重要的理论成果和应用成果，并取得了巨大的社会效益与经济效益，但虚拟计算环境的可靠性一直让人困扰，促使产业界与学术界在这一领域开展广泛的研究工作。

1.1 虚拟计算环境

云计算的目标是将各种IT资源以服务的方式通过互联网交付给用户。计算设施不在本地而在网络中，用户不需要关心它们所处的具体位置，于是像画网络图一样，用“一朵云”来代替了。所有的计算资源、存储资源、软件开发、系统测试、系统维护、游戏引擎和各种丰富的应用服务都可存在于“云”中，用户按需获取计算力、存储空间和信息服务[1]。在这种背景下，促使计算系统的基础设施要具有足够的可靠性，同时也需要尽可能降低计算系统的管理成本。

虚拟计算环境构成了云计算系统的基础设施，虚拟化（Virtualization）技术已成为云计算关键技术之一。虚拟化提供了一种灵活的划分硬件资源的方法，可以对一台机器上的真实硬件资源重新进行抽取、划分、配置，形成多个彼此隔离的虚拟分区，向上层的客户机提供一个虚拟的运行环境，并负责控制管理客户机之间的切换和通信等，从而实现在一台物理机上运行多个操作系统和应用程序，以便更好地利用服务器。基于x86体系架构虚拟化技术的代表性产品主要有VMware、KVM和Xen，它们也是目前普遍使用的虚拟机技术。

VMware公司推出了面向云计算的一系列产品和解决方案，基于VMware虚拟化技术提供云基础架构及管理、云计算应用平台、终端用户计算等多层次上的解决方案，主要支持企业级利用服务器虚拟化技术，实现云计算的虚拟计算环境。

KVM是Kernel-based Virtual Machine的简称，是一个开源的系统虚拟化模块，自2007年2月被导入Linux 2.6.20之后集成在Linux的各个主要发行版本中，也被引入FreeBSD。它使用Linux自身的调度器进行管理，其核心源码很少。KVM的虚拟化需要硬件支持（如Intel VT技术或者AMD V技术），是基于硬件的完全虚拟化。因此，KVM是基于x86

架构且硬件支持虚拟化技术（如 intel VT 或 AMD-V）的 Linux 全虚拟化解决方案。

Xen 是由剑桥大学开发由 XenSource 管理的一个开源 GPL 项目，也是 openSUSE 和 Novell 主要支持的虚拟化技术，它能创建更多的虚拟机，每一个虚拟机都是运行在同一个操作系统上的一个实例程序，主要有两种运行模式：全虚拟化与半虚拟化。前者是指一种完全模拟所有硬件设备的虚拟化模式，而后者是指一种选择性的模拟硬件设备的虚拟化模式。一个标志性的进展是在 2013 年 04 月 16 日，开源虚拟机 Xen 成为 Linux 基金会项目，且在 2014 年 03 月 11 日 Xen 发布 4.4 版本，更好地支持 ARM 架构。

Xen 以高性能、占用资源少著称，赢得了 IBM、AMD、HP、Red Hat 和 Novell 等众多世界级软硬件厂商的高度认可和大力支持，已被国内外众多企事业用户用来搭建高性能的虚拟化平台。著名的商业虚拟化软件 VMware ESX 系列也是基于软件模拟的 Para-Virtualization。

Xen 早期则是基于软件模拟的 Para-Virtualization，新版本则是基于硬件支持的完全虚拟化。但 Xen 本身有自己的进程调度器，存储管理模块等，所以代码较为庞大。

1.2 虚拟计算环境的可靠性

虚拟机（VM）是指在一个硬件平台上模拟多个独立的、ISA 结构和实际硬件相同的虚拟硬件系统，在每一个虚拟硬件系统上都可以运行不同的操作系统，这些客户操作系统通过虚拟机管理器 VMM 访问实际的物理资源。因而整个虚拟计算环境系统是以 VMM 为中心，来实施资源的管理与控制。在各种虚拟计算环境中运行的系统大多是长时间不停机系统，如通信领域软件、嵌入式敏感系统等行业性软件、运行在云集群服务器上的服务及各种应用程序，存在一个非常严重的挑战：尚缺少可以在虚拟计算环境下保障系统可靠性的可行方法。

GB/T 11457—2006 将软件可靠性的定义为：

（1）在指定的约束条件下，在指定的约束时间内软件不引起系统发生失效的可能性；

（2）在指定的约束时间周期内所述约束条件下软件程序执行所要求达到功能的能力。

其中，（1）是软件可靠性的定量定义，而（2）是软件可靠性定性定义。Musa 将系统可靠性定义为：在规定的运行环境中和规定的时间内系统无失效运行的概率。

基于虚拟化技术的可靠性研究，本书仅从最典型的内存资源与 CPU 资源的系统资源使用的角度来考察，也从内存泄漏的角度来考察。而内存泄漏会带来严重的后果，导致系统提供服务延迟缓慢，经常会发生资源耗竭而导致整个系统崩溃。

为了解决这一困难问题，需考虑虚拟化计算环境兼具开放性、可拓展性和动态性的特征[2, 47]，而 Xen 正好具备这些重要特征。Xen 起源于 2002 年由剑桥大学主持的开源

虚拟化项目，2007年Citrix公司发布了Xen 3.1版，标志着Xen真正走向成熟，Xen已获得了业界的广泛认可与应用。传统的运行时资源监控与内存泄漏检测方法受到虚拟计算平台应用复杂、硬件需求千差万别等各种条件的约束，本项目在研究现有资源监控方法、内存泄漏动态检测方法的基础上，采用Xen虚拟化技术，构建出一种面向Xen虚拟计算环境的资源监控与按需调整以及内存泄漏检测的基本方法，并进行了真实系统的设计与实现[3]。

1.3 资源监控与动态调整

近年来，硬件辅助虚拟化技术（Virtualization）的功能日渐完美与成熟，使x86架构在服务器领域有了广泛的应用。虚拟化技术为各种应用提供高性能和高可靠的廉价服务器，也为云计算、数据中心提供了可行的技术支持，如VMware公司、Citrix公司的各种云计算及数据中心的解决方案。

随着云计算的发展，虚拟化技术的应用越来越广泛。虚拟化技术一个重要的应用就是服务器整合，通过在服务器上安装虚拟机可以达到整合分散的服务器资源，最大限度地提高服务器硬件资源的利用率，降低资源开销，减少企业运行成本。服务器整合通常通过在一台服务器上运行多层服务来实现，而不同层次的服务通常运行在不同的虚拟机中[4]，此时这些虚拟机所需要的内存资源就各不相同。内存资源的多少是决定整个系统性能的关键因素之一，有些虚拟机可能需要更多的内存资源来处理更复杂的任务，而有些虚拟机则只需较少的内存资源就可以很好地提供服务。因此，为了更好地分配虚拟机的内存资源，提高内存资源的利用率，有必要对各个虚拟机内存资源的占用情况进行实时地监测与分析。

虚拟计算环境的资源是由虚拟机管理器VMM来实施管理的，因而资源监控与资源的动态调整必须通过VMM来进行。

虚拟机VM能在一个硬件平台上模拟多个独立的、ISA结构和实际硬件相同的虚拟硬件系统，在每一个虚拟硬件系统上都可以运行不同的操作系统，即客户操作系统（Guest OS）。而虚拟机管理器VMM位于计算机硬件和操作系统之间的软件层，负责管理和隔离上层运行的多个虚拟机。运行在VM中的Guest OS通过虚拟机管理器VMM访问实际的物理资源。因而整个虚拟机系统是以VMM为中心，来实施资源的管理与控制。在这个虚拟化体系结构中，位于上层的VM与位于下层的VMM对整个系统功能的支撑是由一种监控机制（Virtualization Surveillance，VS）来体现的[5]。

对应于体系结构的上层虚拟机，文献[6]提出的SIM系统实现了一种典型的VS安全监控机制，在虚拟机内加载内核模块来拦截目标虚拟机的内部事件。基本机制是被监控

的系统运行在目标虚拟机中，安全工具部署在一个隔离的虚拟域（安全域）中。文献[7，8，58]都实现了这种架构，支持在虚拟机内部署钩子函数，拦截某些事件，例如进程创建、文件读写等。文献[7]特别实现了一种安全的机制，当这些钩子函数加载到客户操作系统中时，向虚拟机管理器通知其占据的内存空间，由内存保护模块根据钩子函数所在的内存页面对其进行保护。这种框架结构如图 1.1 所示。

对应于体系结构下层的虚拟机管理器实现的监控机制，必须透过 VM，同时把 VM 与 VMM 二者结合起来实现。最典型的是 2002 年由斯坦福大学的 Tal Garfinkel 和 Mendel Rosenblum 首次在文献[9]中提出一种能够观察到被监控系统的内部状态、同时与被监控系统隔离的入侵检测架构。该架构利用虚拟机管理器能够直接观察到被监控系统的内部状态，通过直接访问其内存来重构出客户操作系统的内核数据结构来进行检测。这种在虚拟机外部监控虚拟机内部运行状态的方法称为虚拟机自省（Virtual Machine Introspection，VMI）。如图 1.2 所示为最初的一个基于 VMI 的入侵检测系统的框架设计。

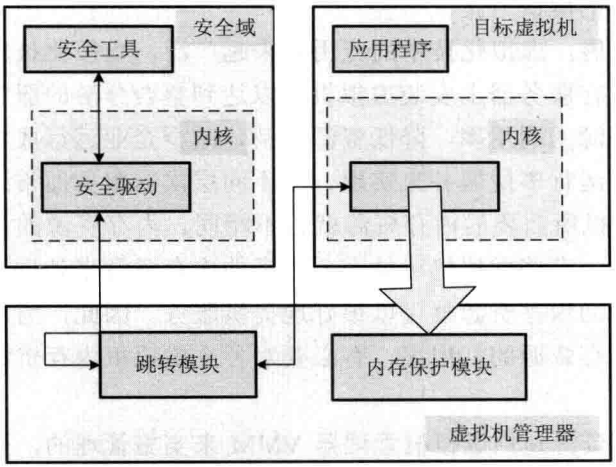

图 1.1 一种典型的 VS 安全监控框架[6]

资源监控与资源的动态调整是云计算平台必不可少的重要组成部分，以上所列举的两种基于虚拟计算环境下的资源监测模型，本质上也展示了这些模型在实现监控时所需要的多种关键技术。

云计算能够为用户提供基础计算设施服务，使得用户能够高效、可靠、经济地使用计算资源的同时，不用增加额外的购置、维护资源的开销，这在很大程度上要依赖对虚拟资源的使用。因此如何有效地管理虚拟资源，使其使用率最大化并保证用户对资源使用的有效性，已成了目前的攻关难题。

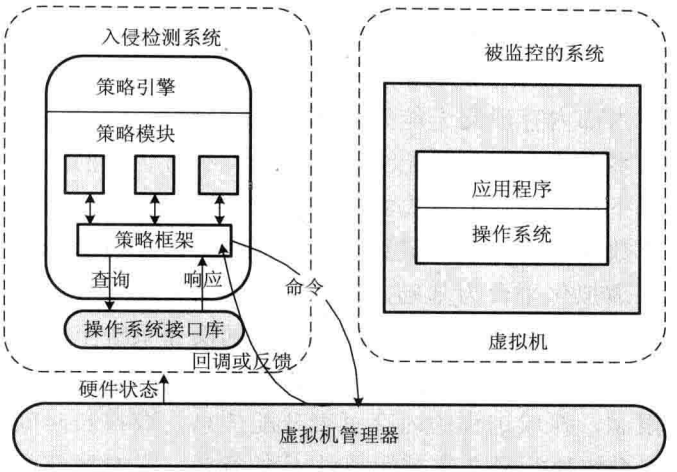

图 1.2 一个基于 VMI 的入侵检测系统的框架[9]

1.4 内存泄漏检测

现代软件的规模越来越大，复杂性越来越高。在这种情况下，软件的可靠性越来越难以保证。可靠性降低的一个重要因素是程序中的内存泄漏问题。它不仅会使应用程序在申请动态内存时失败，服务停止运行，严重时会造成操作系统频繁使用内存交换机制，硬盘负担极度加重，整个操作系统上运行的所有应用程序都受到影响，严重时几乎处于停滞状态，无法正常提供服务。

内存泄漏是指程序向运行时环境或者操作系统动态申请的内存块在使用完毕之后没有被正确释放，这部分内存块一直被该程序占据，操作系统或运行时环境无法将这些内存块重新利用，最终导致可用的内存空间不断减少直至没有内存块可用。

而对于内存泄漏检测而言，有一些棘手的问题需要解决。例如，通信软件在监控程序、服务器端、客户端等不同部分会有多种形态，甚至连最终部署的平台都有不同，运行的软件环境差别也很多。针对这样的系统进行内存泄漏检测的时候，这些不同会带来极大的挑战。因此研究和开发基于虚拟计算环境的内存泄漏检测关键技术可以实现跨平台进行内存泄漏检测分析，具有重要意义。

随着软件工程的发展，软件的复杂度越来越高，代码中不可避免地会出现各式各样的 Bug[10]。虽然新型编程语言如 C#、Java 提供了垃圾收集的机制来避免内存泄漏的出现[11, 12]，但是在注重性能的场合，或者资源有限的硬件环境下进行开发，主流仍然是使用 C/C++语言[13]。而在 C++开发中，一个非常典型而又常见的 Bug 就是内存泄漏。针对

代码中内存泄漏问题的工具也有不少[14]，但是这些工具有一定的局限性，例如，源代码级的内存泄漏静态检测工具因为能跟踪的部分有限，而且实现复杂，对于由程序外部状态的变化造成的程序内部内存泄漏无能为力。而动态检测工具又多依赖于操作系统服务，难以实现跨平台。大多数检测工具还无法针对 7×24 小时运行的服务器程序在不影响其正常提供服务的前提下进行内存泄漏检测。用户级的调试工具无法跨越操作系统的安全机制访问到一些内核数据，而这些内核数据又恰恰对于内存泄漏的分析很有价值。

基于此，以 Xen 虚拟化平台为基础，利用虚拟化平台的特权和提供的服务，构建一个内存泄漏检测分析系统的框架，能满足对多个不同操作系统上的应用程序进行内存泄漏检测分析的需求，对应用程序行为的内存操作信息捕获精度可以达到线程级，并能绕过操作系统权限的限制，抓取进程的内存页表分配信息，取得长时间未访问的内存页的相关信息，以及操作系统执行最久未使用算法[15]的行为。在对捕获的数据进行处理的方面，实现基于对象生命周期的内存泄漏预测算法。同时，该系统框架理论上可以应用于操作系统内核代码，甚至能在不修改框架的情况下直接添加对未知操作系统的支持。

针对 7×24 小时运行的服务器应用程序的内存泄漏问题，我们在本项目实施过程中，基于 CSP 理论构建了一种基于 Xen 虚拟计算环境的内存泄漏检测模型，对模型内系统成员之间的交互，建立了多目标监测时通信进程基于冲突管理的一个重要的解决方案，并基于 CSP 进行了正确性证明；这也是本项目的核心基础[16]。在效率方面，该框架设计为异步方式，以完成数据的处理工作，最大限度避免因监控而造成的性能瓶颈。

虚拟计算环境中系统性能的可靠性问题研究对于云计算相关技术的研究和应用具有重要的理论和实际意义。长时间不停机系统的内存泄漏可能给实际应用带来严重后果，在虚拟计算环境中检测运行时内存泄漏是一个极具挑战性的问题。针对该问题，我们在本项目研发实践中，对内存泄漏的现象进行了分类，基于 Xen 虚拟机的自省机制，构建并实现了一种面向 Xen 虚拟计算环境的运行时内存泄漏检测的 VMLD（Virtualization Memory Leak Detecting）模型[17]，提出了相应的检测规则，通过修改虚拟机管理器、设计超级调用，实现了内部缓冲区维护、控制、拦截、监视等模块[18, 19]。实验结果表明，VMLD 方法能有效地检测出运行时内存泄漏，并且具有较好的性能。

1.5 小　结

虚拟计算环境作为云计算的重要基础设施，资源的监控与调整是其核心工作之一；其服务器应用程序的内存泄漏会带来严重的可靠性问题，会导致系统提供服务延迟缓慢，系统性能越来越低，经常会发生资源耗竭而导致整个系统崩溃。本章概述资源监控与资源动态调整的意义，进一步叙述了虚拟计算环境下内存泄漏检测的重要性。

第 2 章　虚拟计算环境可靠性的相关研究

虚拟计算环境可靠性涉及很多的研究领域，对计算资源使用状态的监控是研究虚拟计算环境可靠性的一个基本途径，通过资源的人工调整或自动动态的调整，可使系统资源进行再分配，而使系统能平稳地提供服务。另外，在内存泄漏的情况下，对于长时间不停机的服务器应用程序，通过内存资源的使用状态的监测而进行内存泄漏的检测与分析，可以评估系统的可靠性。本章将从内存泄漏的研究现状、虚拟计算环境下的内存泄漏检测方法、虚拟计算资源监测与资源调整等方面进行概述。

2.1　内存泄漏国内外研究现状

国内外研究机构和软件企业一直高度关注内存泄漏检测的相关问题，开展了许多研究和开发工作，取得了大量的研发成果。检测内存泄漏的方法大致可以分为两类：动态方法和静态方法。它们最根本的区别在于是否需要执行目标程序。不需要执行程序的测试方法称为静态方法，需要执行程序的测试方法称为动态方法。现将一些具有代表性的工作归纳、总结如下几个方面。

2.1.1　与内存泄漏静态检测相关的研究工作

国外已有许多内存泄漏静态检测相关的研究。最具代表性的是 PREfix[20]、Matal[21]等静态工具，基于内存分配点开始的分析，识别出可能发现内存泄漏的路径，相应地定义了检测的基准，依据对内存漏泄这一现象本质的剖析，总结得到可能出现泄漏的错误模式。Stanford 大学的 Suhabe Bugrara 等研究者所开发的 SATURN 工具[22]，基于所谓逃离分析（Escape Analysis）假设进行内存泄漏检测分析。其基本原理是任何在进程 P 中分配的内存，没有在进程 P 中释放，也没有传递到其他进程，那么可以判定，这个分配的内存产生了泄漏。Stanford 大学 MonicaS.Lam 教授领导的小组开发的 Clouseau[23]，是一个检测 C++和 C 代码中内存泄漏和对象多次删除的工具，是基于隶属关系模型（Ownership Model）进行分析的半自动工具。

国内方面的相关研究相对要晚。中国科学技术大学龚育昌等人[24]针对应用程序安全分析的实际需求，设计并实现了一个针对可执行代码的内存泄漏分析框架 MLAB。该框架首先从可执行代码中恢复控制流和数据流信息，再依据恢复的控制流图建立程序的有

限状态自动机，最后运用模型检测算法分析程序可能存在的内存泄漏。北京邮电大学网络与交换技术国家重点实验张威、宫云战等人[25]引入了指针映射代数的概念，用于全面地反映指针与内存之间的映射关系，并以此为基础，给出了面向不同故障的指针映射集的构造规则，建立了动态内存故障模型，提出了一种基于指针分析的内存泄漏故障测试方法。中科院计算所张广梅等人[26]针对内存泄漏、空指针引用等动态内存错误在C、C++等支持动态内存操作的程序中普遍存在这一现象，在对程序进行静态分析的基础上，提出了一种动态内存错误的静态检测方法。该方法借助已有检测工具，按照较为规范的流程定义了动态内存错误的具体检测过程。南开大学涂奉生等人[27]开发了在面向函数定位框架中嵌入动态分析的内存泄漏监方法。该方法先建立程序的函数动态调用树，其中包含程序分配释放内存的信息，再在调用树中总结程序的静态性质，为内存泄漏定位提供有价值的信息。华中科技大学黄瑞光[28]等人描述了一种在Sun Solaris环境下，由于ucb库函数fopen造成的隐蔽的内存泄漏现象，提出了检测的方法和解决、替代方案。中兴软件技术有限公司成都研究所柳长安[29]等人给出了Windows Mobile平台上基于AppVerifier工具，监测设备内存泄漏的方法。

2.1.2 与内存泄漏动态检测相关的研究工作

就国内外而言，当前已有许多关于运行时动态检测内存泄漏的研究工作。最典型的工作有IBM研究开发的Purify[30]、威斯康星大学麦迪逊分校的Lu Shan等人开发的SafeMem[31]等，在程序运行时记录程序动态分配的内存资源和释放信息，然后分析是否存在内存泄漏现象。

国内外已研发了许多用于动态检测内存泄漏的工具，可以分为四类。

第一类以监测内存使用为主[32]，如Jprobe[33]、Jprofiler[34]。这些工具提供了丰富的观察视角选项，可以从不同的角度描述程序运行时内存使用情况的变化。专业的程序员通过比较这些视图，可以比较清楚地观察泄漏的对象。国内所开展的相关研究工作基本上属于此类。南京大学徐宝文等人[35]给出了基于源代码插桩的C程序内存使用错误动态检测方法，并研发相应的支撑工具。西安交通大学冯博琴等人[36]实现了一个Linux平台下针对C/C++语言的动态内存检测模块，该模块可以检测内存泄漏、内存写溢出、释放野指针和内存管理函数的不匹配等问题。中国科学技术大学孔德光等人[37]实现了一个轻量级的堆内存泄漏检测工具。针对C++的开源代码，通过重载new和delete运算符，采用红黑树管理所分配的堆内存，动态跟踪程序在执行过程中堆内存块的分配释放情况，并在程序运行结束时给出内存泄漏的检测结果。南京大学计算机软件新技术国家重点实验室王勇等人[38]提出了一种统一的内存错误检测模型和接口，使内存错误检测处理过程规范化，便于扩展。华南理工大学刘发贵等人[39]实现一个嵌入式Linux平台下数据采集