



一种面向云平台的虚拟机内存 SLA 审计机制^{*}

李攀攀, 张宏莉, 邓会敏, 周志刚

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要: 针对云计算的服务模式屏蔽了云租户的物理硬件视图, 不可信的云服务提供商 (cloud service provider, CSP) 可能利用廉价的硬盘资源通过虚拟化技术, 违背服务等级协议约定 (service level agreement, SLA) 按物理内存定价标准为云租户提供服务这一问题, 为了审计 CSP 提供内存服务的 SLA 合约性, 提出了由 Xen 层到物理硬件层的内存轻量级测量的 SLA 合约性审计方案。同时引入可信启动机制和 HyperSentry 用于保证审计系统的可信启动和完整性运行, 提出了带云租户签名机制的 Diffie-Hellman 密钥交换协议支持策略安全和可信告警。实验结果表明, 在虚拟机运行环境下该方法能高效地进行内存 SLA 合约性审计, 同时具有较高的云租户自定义策略扩展性和较低的性能开销。

关键词: 云计算; 内存 SLA; 审计; 虚拟化; Xen

doi: 10.3969/j.issn.1000-0801.2013.06.012

SLA Audit Mechanism of Virtual Machine Memory on Cloud

Li Panpan, Zhang Hongli, Deng Huimin, Zhou Zhigang

(School of Computer Science and Engineering, Harbin Institute of Technology, Harbin 150001, China)

Abstract: Cloud service style has shield physical hardware view to cloud tenant, thus untrusted CSP (cloud service provider) may replace expensive physical memory by cheaper hard disk resource, which violates the SLA. Therefore, in order to audit memory SLA of cloud, a novel scheme for auditing physical memory of VM was proposed. This scheme is based on light-weight memory measurement SLA auditing by Xen layer to physical layer. Meanwhile, trust boot mechanism and HyperSentry module to ensure trust boot and integrity guarantee at running time were introduced. Then, digital signatures-based Diffie-Hellman key exchange protocol was also proposed to support strategy security exchange and trust alarm. The experimental results indicate that the proposed module can effectively audit VM memory SLA, and also support strong expansibility of cloud tenant customize strategy with low overhead.

Key words: cloud computing, memory SLA, audit, virtualization, Xen

1 引言

云计算服务作为一种经济高效的服务模式和新兴计算模式, 近年来日益受到学术界和工业界的广泛关注。区别于传统的服务模式, 云计算服务是基于互联网和虚拟化

技术提供相关服务的租用和交付模式。云计算将计算或存储任务分布在大量计算机构成的资源池上, 将有限的资源集中起来按照一定模式提供可靠、安全的服务。一方面各种应用能够根据需要获取计算力、存储空间和信息服务, 极大地提高软硬件资源的利用率; 另一方面这种服务模式

^{*} 国家自然科学基金资助项目 (No.61100188, No.61173144), 国家重点基础研究发展计划 (“973”计划) 基金资助项目 (No.2011CB302605), 国家高技术研究发展计划 (“863”计划) 基金资助项目 (No.2011AA010705)

使得云租户能够将资源切换到需要的应用上,根据需求使用计算资源和存储资源。

虚拟化技术是支撑云计算基础设施虚拟化的核心技术,也是推动云计算服务发展的主要动力之一。但是硬件虚拟化技术屏蔽了云租户的硬件视图,云租户无法知道CSP是否按照约定SLA标准提供服务。诸如不可信CSP用廉价的硬盘资源,采用内存虚拟化技术,将其按照物理内存的定价标准提供给云租户,云租户无法察觉CSP这种不可信的行为。但是从物理特性上,内存虚拟化技术的内存访问时间为毫微秒级,硬盘级虚拟的内存访问时间为毫秒级^[1],无论是从SLA合约性保障方面还是从经济性上都会造成云租户利益的损失。云租户的这种SLA合约性上的顾虑已经在一定程度上影响了云计算商业化应用的推广^[1]。因此,虚拟机运行环境下内存SLA合约性审计成为当前云计算安全审计研究的热点。

Xen作为采用半虚拟化技术的虚拟机监控器(virtual machine monitor, VMM)的典型代表,以其优越的性能被广泛地部署到现有的商业化云计算平台,例如Amazon EC2^[2,3]。本文针对较为流行的Xen平台上客户虚拟机的物理内存合约性审计进行研究。目前针对云计算平台Xen平台上的虚拟机SLA审计的研究有了一定的成果^[4-6],但是,传统的用户云计算平台虚拟机运行环境下内存SLA审计方案仍存在如下问题:

- Xen的Balloon Driver内存管理机制使得CSP能动态地为虚拟机分配物理内存,这就导致虚拟机所配额的物理内存具备一定的动态变化性^[3,7],所以策略性不定时抽检才更具备可信性;
- 虚拟机运行环境下内存SLA审计的排他性,即在审计过程中无法同时进行常规应用,导致在现实场景中不具备通用性;
- 基于物理硬件访问时间特性的审计精确度低,受网络时延、Guest OS硬件使用率等各种因素的制约,使物理硬件测量时间的波动变化较大。

鉴于上述问题,如何帮助云租户对虚拟机运行时环境内存进行SLA审计成为亟待解决的问题,需要提出一种更精确的虚拟机运行环境内存SLA审计方法。本文针对Xen的半虚拟化应用,提出了基于Xen平台的内存SLA合约性审计机制的原型系统。为了提高审计的效率,本文提出了由Xen层到物理硬件层的“自上而下”的虚拟机运行环境下内存SLA审计的框架机制。然而,在虚拟机运行环境下

内存SLA审计存在如下一系列技术挑战:

- Xen的内存Balloon Driver管理模式能动态调整Guest OS物理内存分配额度,这种机制使得单纯的审计方式不太适用;
- 在Xen运行时环境中, Guest OS内存映射页表的高速动态性,如何对特定的时刻进行审计,成为亟待解决的问题。

本文的主要贡献如下:

- 提出了一种基于层次化虚拟机运行环境的内存测量方案;
- 本方案具备轻量级和较高的SLA审计结果精度;
- 提出了一种适应云计算环境的密钥安全交换协议,有效地支持云租户策略安全传输与审计结果的可信警告。

2 设计

本文针对Xen的半虚拟化应用场景,提出了内存SLA合约性的审计机制,下面就设计目标和系统架构做详细阐述。

2.1 设计目标

虚拟机运行环境内存SLA审计的目的是通过掌握虚拟机当前内存使用情况以及虚拟机物理内存占用的快速测量策略,采用“自顶向下”层次化的轻量级内存SLA审计方案,提高SLA审计结果的精度,并在不导致Guest OS服务性能严重下降的前提下,在一定精度范围内,审计Guest OS所分配物理内存SLA的合约性。虚拟机运行环境内存SLA审计机制的目标是能够按照云租户对内存SLA合约性审计的要求,策略性地不定时抽检和审计CSP分配虚拟机物理内存的SLA,同时必须满足审计过程及结果不能被CSP所预先获知,并且审计结果也不能被CSP篡改,为此需要实现以下基本目标。

- 审计结果的精确性:本审计机制的核心目标,在一定精度范围内,满足常规化应用需求的能力。
- 轻量级:审计模块在不干扰系统正常运行的情况下,不能引入过多的额外性能开销。
- 审计模块的安全性:对待审计的云租户虚拟机运行环境不能引入新的安全隐患。
- 审计模块具备顽健性:指审计系统在异常情况下能正常运行的能力,包含稳定顽健性和性能顽健性两大方面。



- 审计模块自我保护的能力:审计模块具备防御被不可信 CSP 篡改的能力。
- 不可旁路性:审计模块和云租户的策略传输和审计结果的传输等需要使用 CSP 具有绝对控制权的通信链路,完备性的审计方案必须具备双方通信内容不被 CSP 旁路监听的功能。

本文提出了一种轻量级、面向虚拟机运行环境的内存 SLA 审计方案。部署在 Xen 内核空间的 Guest OS 物理内存的测量模块及审计模块不与现有的各种技术冲突,是对云计算平台内存虚拟化技术的安全性审计的一个很好的补充。

2.2 审计系统框架及其原理

Xen 在内存虚拟化的同时,也支持内存的分段和分页机制。利用内存分段保护机制,使 Xen 和 Guest OS 共存于一个内存地址空间中,大大简化了 Xen 对 Domain 内存的分配和管理过程;利用内存分页机制,使 Xen 能够保证各个 Domain 在内存上的有效隔离^[8]。本文审计系统有效地利用 Xen 系统的内存分页式管理机制,避免了内存视图的二次转换,审计过程更加高效。

面向云计算平台的虚拟机运行环境内存合约性审计系统体系结构如图 1 所示,主要由静态度量模块(static measurement module, SMM)、Guest OS 内存测量 (Guest OS memory measurement, GOSPMM)、可信启动与生命周期完整性保证(trusted boot and lifecycle integrity assurance, TBLIA)和策略安全与可信告警 (strategy security and trusted warning, SSTW) 4 个部分组成。

静态度量模块部署在 Xen 运行空间,负责度量宿主系统静态参数,主要包含宿主操作系统的字长、宿主操作系统物理内存大小等关键参数。这些关键参数是整个系统进行 SLA 合约性审计的基础。

在每个 Guest OS 中均部署 GOSPMM 模块,是整个审计模型的关键模块,其主要功能是接收 SMM 的宿主操作

系统的关键参数,对 M2P 表进行快速镜像、测量和内存审计等工作。详细内容将在第 3.2 节中介绍。

可信启动与生命周期完整性保证、策略安全与可信告警两大功能模块起着保障审计系统安全性的功能,是审计过程及其结果可信的基础。

3 方法实现

为了验证所提的方案,本文实现了虚拟机运行环境内存合约性审计的原型。本节对原型系统的实现和主要关键技术进行阐述。本原型系统以模块的方式在 Xen 的内核空间运行,拥有较高的系统权限。

3.1 Xen 内存模型

在 Xen 体系结构存在着 3 个层次的地址空间定义:

- 客户机虚拟地址(guest virtual address, GVA),也称虚拟内存,指 Guest OS 提供给它应用程序使用的线性地址空间;
- 客户机物理地址(guest physical address, GPA),也称物理内存,指经过虚拟机监控器抽象的、虚拟机所看到的伪物理地址;
- 主机物理地址(host physical address, HPA),也称机器内存,指真实硬件的机器地址,即地址总线上应该出现的地址信号。

在引入物理内存后,对内存页面的描述,除了页框号 (page frame number, PFN) 和机器页框号 (machine frame number, MFN) 外,还增加了客户页框号 (guest page frame number, GPFN) 和客户机器页框号 (guest machine frame number, GMFN)。MFN 表示真正的机器物理内存的页框号,GPFN 则与伪物理地址对应,表示位于 Guest OS 地址空间的页框号,而 GMFN 则和 Guest OS 的体系结构相关,在 x86 半虚拟化框架中,GMFN 等同于 MFN^[3,10]。

从内存管理角度,在 Xen 半虚拟化场景中,每个 Guest OS 管理自己的页表,Xen 的虚拟内存管理单元实现了

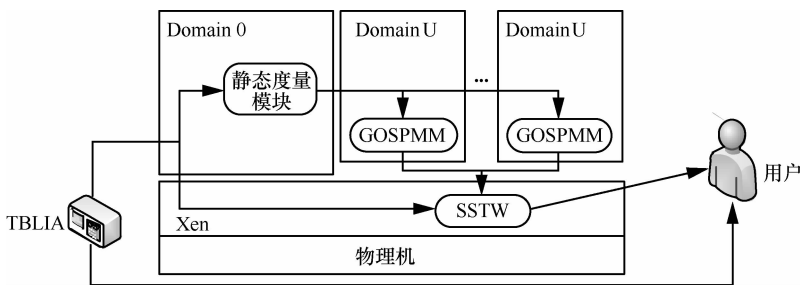


图 1 虚拟机运行时环境内存 SLA 合约性审计系统体系结构

Guest OS 完成特权级 MMU 操作。在半虚拟化工作模式下, Guest OS 使用自己的页表直接访问机器内存, 无需地址二次转化, 这种方式效率很高。Guest OS 在硬件 MMU 的参与下, 使用自己的页表直接访问机器内存, 在页表中的页表项保存的地址为机器地址, 即 MMU 的分页单元的地址转换可以直接获得内存页的机器物理地址。而在 Guest OS 的使用过程中, Guest OS 不能直接使用机器物理地址, 它操作的地址是物理地址, 因此, 在 MMU 的分页单元之外, 为了实现机器地址到物理地址之间的相互转换, Xen 通过表格的形式记录两者之间的关系。即机器地址到物理地址转换表 (machine-to-physical translation table, M2P 表) 和物理地址到机器地址转换表 (physical-to-machine translation table, P2M 表)。借助 M2P 表和 P2M 表, Xen 不仅负责管理每个域物理内存的分配, 还负责管理 Guest OS 在运行过程中物理内存的动态调整。

在半虚拟化 Xen 运行环境中, P2M 表和 P2M 表的内存映射关系如图 2 所示。

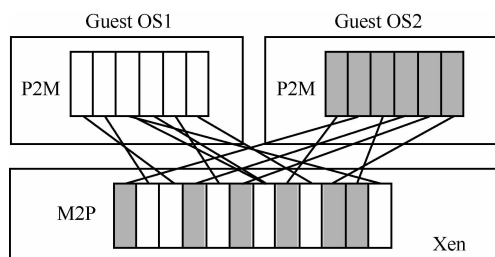


图 2 Xen 半虚拟化内存映射关系

M2P 表是一个映射机器地址到物理地址的数组的全局表, 位于 Xen 堆空间内, 其以机器页框号为数组索引, 数组内容对应的是客户操作系统物理页框号。由于在 Xen 系统中, Guest OS 负责分配和管理硬件页表, M2P 表对应的 P2M 表不是唯一的, 每个 Guest OS 都拥有一个自己的 P2M 表。P2M 表以 Guest OS 物理内存地址为索引, Xen 可以在该表中找到以 Guest OS 物理内存地址为索引查找所对应的宿主机机器地址, 该数组位于每个 Domain 的堆中^[11], machine_to_phys_mapping 是 M2P 表的具体实现^[12]。

本文方案高效地监控 Guest OS 是否使用硬盘级虚拟内存主要体现在如下两个方面。

- 轻量级监控: 审计系统和操作系统具有同一页面级内存视图, 在保证精度的同时又增强了审计模块的顽健性。
- 策略性调整: 审计模块提供用户更改审计策略的

接口。

3.2 Guest OS 物理内存测量

通过 Xen 内存虚拟化的 M2P 表和 P2M 表的双向映射机制实现的内存测量, 是基于页面级粒度的, 即所监控的最小内存单位是内存页, 通常大小是 4 KB。在进行监控时, 虚拟机对页面范围内的所有内存地址访问不会因为地址越界而导致陷入, 因此, 本方案具备顽健性的要求。采用监控超级调用带来的优点主要包括:

- 页面级精度监控, 与 Kernel 采用相同的页面视图, 无需二次转换, 因此监控过程中带来的额外开销较小;
- 处于较高的级别, 拥有较高的监控级别, 宿主机无法回避对内存的非合约性操作。

从 Xen 三层内存映射的角度看, 实现虚拟机运行环境物理内存的测量将涉及以下问题:

- 在虚拟机运行环境中, 物理内存和虚拟内存的映射关系始终在高速地变化, 任何高效的匹配算法都将带来一定的滞后性, 需解决遍历的滞后性问题;
- P2M 表包含了整个 Guest OS 内存映射的信息, 为了不给系统正常运行带来较大的性能惩罚, 要考虑如何轻量级遍历 P2M 表。

为了解决前一问题, 提出了离线型 P2M 表镜像, 即启动独立进程镜像 P2M 表到新开辟适量内存空间, 遍历完 P2M 表后, 由独立进程释放 P2M 表内存空间; 为了解决后一问题, 针对镜像 P2M 表, 采用基于前缀搜索的 KMP (knuth-morris-pratt) 精确串匹配算法扫描 P2M 表, 寻找特定虚拟机 ID 所映射的内存地址空间集合, 匹配算法时间复杂度 $O(mn)^{[13]}$, 其中, m 表示虚拟机 ID 字符串的长度, n 表示 P2M 表的长度。设 $Mem_{VM,t,p}$ 表示在 t 时刻虚拟机物理地址空间中, 由宿主机物理内存所提供部分的集合; $Mem_{VM,t,v}$ 表示在 t 时刻虚拟机物理地址空间中, 由宿主机硬盘级虚拟内存所提供部分的集合。

算法 1 P2M 表镜像扫描算法

输入: 用户审计策略

输出: Guest OS 物理内存地址空间 $Mem_{VM,t}$

```
{
    根据用户审计策略, 选定  $t$  时刻;
    启动独立线程, 镜像当前 P2M 表到新开辟内存空间;
    for ( P2M != Null )
    {
```

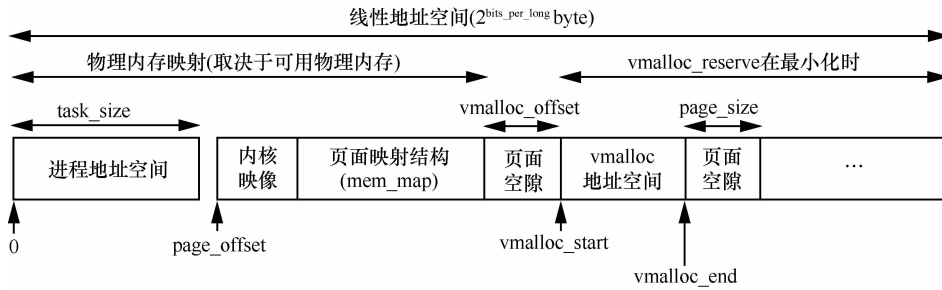


图3 内核地址空间

KMP 匹配算法,遍历 P2M 表;

Mem_{VM}.t = & (P2M);

P2M++;

}

释放 P2M 镜像表内存空间;

}

从 Kernel 角度观察内存视图,内存地址空间分为如下两个部分:

- 随着上下文切换而改变的用户空间部分;
- 保持不变的内核空间部分。

两者的分界线由 page_offset 决定^[4]。对于系统字长为 bits_per_long 的宿主机,从 Kernel 角度看线性虚拟地址空间如图 3 所示。

物理内存映射由宿主机上所有虚拟机所共享,审计模块拥有探测宿主机物理内存的能力。审计模块静态度量的客体是加载至平台的物理内存大小。在多数情况下,通过读取/proc 即可确定宿主机物理内存大小,“getconf long_bit”可获得系统字长。在图 3 中,page_offset 到 vmalloc_start-vmalloc_offset 是由宿主机物理内存映射的部分,该区域的大小由宿主机物理内存所决定。Xen 通过页表项把物理内存映射到 page_offset 开始的虚拟地址。在 32 位系统字长 0x86 平台架构中,page_offset 的地址为 0xC0000000。

在图 3 中,对于物理内存为 Memhost 的宿主机上所有的 Guest OS 来说,审计模块根据机器地址的范围进行判断,其是由物理内存提供还是由宿主机硬盘级虚拟内存提供的。当 Guest OS 物理内存地址空间小于 vmalloc_start-vmalloc_offset 时,则该虚拟内存由宿主机物理内存提供;当物理内存地址高于 vmalloc_start 时,则该虚拟内存由宿主机硬盘级虚拟内存提供。

算法 2 审计算法

输入:虚拟机的机器地址空间 Mem_{VM}.t;

宿主机物理内存大小 Memhost;

宿主机系统字长 bits_per_long

输出:Mem_{VM}.t_p, Mem_{VM}.t_v

{

While (MemVM.t != Null)

{

if (page_offset < \$MemVM.t)

MemVM.t_p = MemVM.t

if (vmalloc_start < \$MemVM.t)

MemVM.t_v = MemVM.t

MemVM.t++;

}

3.3 可信启动与生命周期完整性保证

由于 CPS 对云平台的物理上的绝对控制权,审计模块运行于 Kernel 内核,保证审计模块的可信启动在其生命周期内防篡改,是审计模块结果可信的前提。可信启动与生命周期完整性保证如图 4 所示。

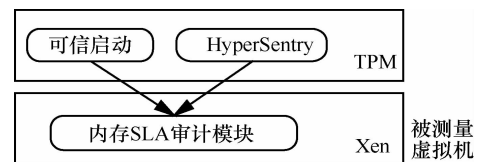


图4 可信启动与生命周期完整性保证框架

(1)可信启动

审计模块初始化之前如何保证审计模块系统的安全性,不可信 CSP 可能在审计模块启动之前破坏或者禁止其启动。鉴于此,本文引入可信启动(trust boot)机制用于支持审计模块的可信启动,在审计模块启动过程中对其度量与检测事件,当异常发生时,可信启动模块有权决定是否要终止整个启动过程,或者仅终止度量以及审计。这是审计模块可信运行的基础。

(2)生命周期完整性保证

HyperSentry^[4]是用于 Xen 完整性度量的安全模块,在Xen

中以模块化运行,引入该模块监控本文审计模块生命周期的完整性。根据第2.2节所述,审计模块作为嵌入式模块运行于Xen和Domain内存空间,在HyperSentry的安全防护框架之内。HyperSentry在Xen和审计模块处于运行状态时就能检查其是否被第三方入侵,同时HyperSentry也能够检查Xen和审计模块的内存空间以及宿主平台CPU的寄存器,能进一步发现被感染程序的确切位置。HyperSentry的部署是审计模块运行过程完整性保证的保障。

3.4 策略安全与可信警告

在CSP对云平台拥有绝对物理控制权的情况下,特别是对云平台内部通信链路的绝对控制权,如何将云租户的审计模块策略发送到审计模块以及将审计结果返回给云租户具有一定的技术挑战,使传统的加密方案中对密钥的管理成本无法应用本文所涉及的场景。传统Diffie-Hellman密钥交换协议没有提供SSTW和云租户双方身份认证的有效信息,无法抵御CSP重演攻击^[15]。为此,本文采用带数字证书认证的解决方案。基于带身份认证功能的Diffie-Hellman的密钥交换协议,支持审计系统的策略安全与可信警告,身份认证用于抵御CSP的链路上的重演攻击,同时尽可能较少地引入TBLIA的工作负载。

SSTW模块和云租户密钥交换协议流程如下。

(1)TBLIA单元初始化全局公开参数:素数 q 、整数 α (是 q 的一个原根),同时颁发数字证书 $DigitCert_{TTSW}$ 和 $DigitCert_{CT}$ 到SSTW单元、云租户。

(2)SSTW随机选择私有密钥 $X_{SSTW}(X_{SSTW} < q)$,计算公开密钥 $Y_{SSTW} = \alpha^{X_{SSTW}} \bmod q$,发布数字证书 $DigitCert_{CT}$ 。

(3)云租户随机选择私有密钥 $X_{CT}(X_{CT} < q)$,计算公开密钥 $Y_{CT} = \alpha^{X_{CT}} \bmod q$,根据 $DigitCert_{SSTW}$ 验证SSTW身份合法性,同时生成秘密密钥 $K = (Y_{SSTW})^{X_{CT}} \bmod q$ 。

(4)SSTW根据 $DigitCert_{CT}$ 验证云租户身份合法性,同时计算秘密密钥 $K = (Y_{CT})^{X_{SSTW}} \bmod q$ 。

通过上述步骤,SSTW和云租户之间构建了安全的密钥交换通道,即 K 为SSTW和云租户间的秘密密钥,本算法的有效性依赖于计算离散对数的难度,当素数 q 足够大时,这个运算等于无解。由于引入了TBLIA的数字证书机制,能够抵御CSP的重演攻击。密钥的构建完毕,SSTW和云租户的策略安全和可信警告可以依赖于密钥 K 进行安全传输。

4 实验结果与分析

本文审计模块的有效性及其对Guest OS性能的影响

是实验关注的重点。

基于Xen-4.1.0开发了原型系统,审计模块使用insmod方式载入Kernel。为了进一步减小内存负载,启动独立进程对M2P表镜像。M2P表的查找算法采用基于KMP精确串匹配算法。实验环境采用Intel服务器和Xen3.0。服务器为8核2个Intel Xeon,E5506,2.13 GHz Quad-Core Processors,16 GB双通道1 333 GHz内存。

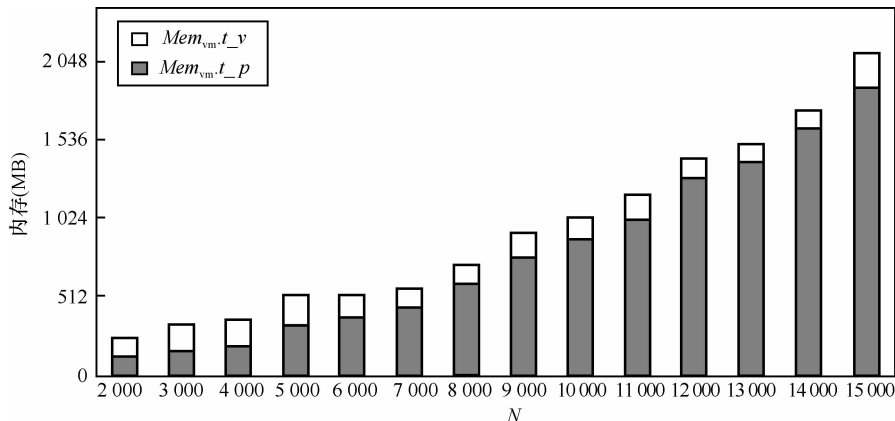
4.1 有效性

本文选择Linpack计算密集型负载测试不同的可用内存情况下审计系统的有效性^[16,17]。Linpack应用主要是用高斯消元法求解一元 N 次稠密线性代数方程组的测试,针对 N 次稠密线性代数组,矩阵的规模 N 越大,有效计算所占的比例也越大;但与此同时,矩阵规模 N 的增加会导致内存消耗量的增加,一旦系统物理内存空间不足,会触发Guest OS虚拟内存的使用。为了全面地验证审计模块的有效性,针对内存充裕状态和紧张状态,设计了3个场景对审计系统有效性进程测试。

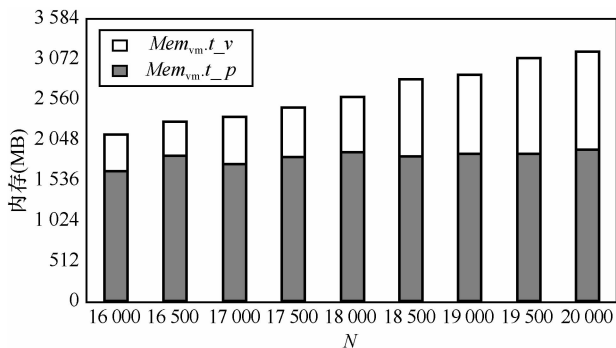
场景(1)中,初始化Guest OS物理内存为2 GB,验证审计系统在物理内存充裕状态下、不同求解规模的基准测试用例下内存使用量的测试结果。对于Linpack测试用例逐渐增加 N 的大小,Guest OS求解过程所消耗的内存显著增加,此外,为了保证不过多地启用系统守护进程,实验过程中屏蔽不相关宿主系统daemon进程,同时每次测试前清理宿主机物理内存,保证宿主操作系统内存管理机制对本文实验结果的影响最小。实验结果如图5所示。

从图5中显然可以看出,随着 N 由2 000~15 000按1 000为规格单位递增,Guest OS所消耗的物理内存急剧增加,这很好地吻合了Linpack求解过程消耗内存大小一般估算为 $(N \times N \times 8) \text{ byte}$ ^[16,17]。另一方面,还可以看出 Mem_{vmt_p} 具有一定的波动性,这与系统运行时内存占、管理和回收等有一定的关系。由于宿主系统中守护进程和内存回收的时机问题等诸多不确定因素, Mem_{vmt_v} 虚拟内存的大小具备一定的随机波动性。Xen为自身的内核和堆分配了12 MB的物理内存,并以Ring0级运行在虚拟地址最高端64 MB内存中。需要指出的是在高端内存线性地址空间除了映射物理内存空间之外还要映射处理器内部外设寄存器空间等I/O空间,这也导致了在内存充裕状态下 Mem_{vmt_v} 仍然被使用的情况。

场景(2)中,设计当Linpack基准程序所需要的物理内存大于所分配的内存时。在 N 从16 000到20 000逐渐增

图5 内存充裕状态下, $Mem_{vm}.t_p$ 和 $Mem_{vm}.t_v$ 随 N 递增的变化示意

大的过程中分别测试 $Mem_{vm}.t_p$ 和 $Mem_{vm}.t_v$ 的大小,实验结果如图6所示。

图6 内存紧张状态下, $Mem_{vm}.t_p$ 和 $Mem_{vm}.t_v$ 随 N 递增的变化示意

从图6可以看出, $Mem_{vm}.t_p$ 接近 Guest OS 可用物理内存的大小, $Mem_{vm}.t_v$ 随着 N 的增大迅速增加。 $Mem_{vm}.t_p$ 和 $Mem_{vm}.t_v$ 之和接近于 Linpack 的估算大小,验证了本审计模块在不同内存负载状态下的有效性。但是试验中采用的计算密集型测试用例对虚拟机运行环境的物理内存要求严格,在物理内存不足时将会引发大量磁盘 I/O,降低系统性能,影响到算法的性能。

场景(3)中,使用不同的物理内存和虚拟内存组合,测试在不同内存组合情况下的审计模块的输出结果。3组内存组合环境分别为:

- 512 MB 物理内存, 1 536 MB 虚拟内存;
- 1 024 MB 物理内存, 1 024 MB 虚拟内存;
- 1 536 MB 物理内存, 512 MB 虚拟内存。

使用两组负载用例分别为 $N=10\ 000$ 和 $N=15\ 000$ 。

在图7中,对相同负载的不同内存配比情况下,审计模块的测量结果均能良好地在内存相应内存配比下工作,验证了本审计系统对不同物理内存大小具备敏感性;从另一方面也表明,本系统能良好地适应 Xen 的“balloon

driver”动态内存管理情况下的内存测量工作。另外,本文采用“ps”的 vsize 和 rss 输出值取和为当前测试用例进程的内存开销,当 $N=10\ 000$ 和 $N=15\ 000$ 时内存最大,开销分别为 904 MB 和 1 048 MB。于是,当 $N=10\ 000$ 时,3个内存配比系统下的测量误差分别是 6.4%、3.8%和 2.4%;当 $N=15\ 000$ 时,3个内存配比系统下的测量误差分别是 5.1%、4.1%和 7.7%。

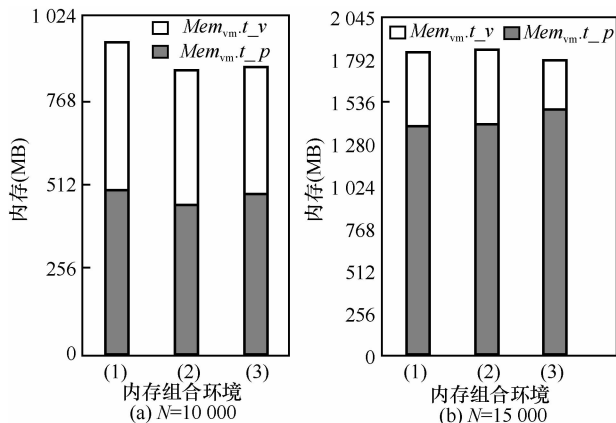


图7 固定负载情况下,不同内存下的测量结果

针对以上3个场景的实验过程及其结果,要指出的是,现代操作系统优先运用物理内存,当物理内存还有空闲时,宿主操作系统不会立即释放内存,即使占用内存的应用程序已经关闭(此时,该部分内存用作缓存)。同时内存写时复制(copy-on-write)技术, Guest OS 分配 2 GB 的内存,测试用例轻型负载情况下,随着系统运行一段时间后,内存空间也会被占满,这种设计思想的主要目的是使操作系统中进程的重新创建和内存的再读取变得轻量 and 快速^[9]。但是现代操作系统的这种设计思想给本文的实验结果带来一定的扰动。图5~图7均表明审计系统对不同的负载应用、不同的物理内存环境具有敏感性,同时验证了审计系

统的有效性。

4.2 性能开销

审计模块的守护进程对虚拟机访存范围的监控不可避免地给被审计系统带来性能扰动性,主要是由于虚拟内存和物理内存的测量伴随着不可避免的性能损失,本小节侧重于对性能开销在一定精度范围内的量化。性能开销包括存储开销和计算开销,其中,存储开销包括对 P2M 表的镜像开销,对于系统字长 32 位的宿主系统, GCC 编译环境下, P2M 表的每条记录位数为 20 byte, 则对于 Xen 支持 100 个满特征 Guest OS 来说, 同时审计的最大存储开销为 $100 \times 20 \text{ byte} = 2\,000 \text{ byte}$ 。但是由于系统中引入了 P2M 镜像页表释放机制, 随着审计的结束, P2M 镜像所占用的存储空间会被释放。

针对计算开销, 引入审计模块后, 对 Guest OS 计算性能的影响是不可避免的。通过比较部署审计模块前后虚拟机的性能, 下面通过对比实验对审计模型的额外计算性能开销进行量化评测。本文采用微内核基准测试程序 SPEC CPU2000, SPEC CPU2000 是标准性能评价组织 (standard performance evaluation corp, SPEC) 开发的用于评测通用型 CPU 性能的基准程序测试组^[18]。本文试验中, SPEC CPU2000 提供了对部署模块前后系统性能进行对比的测试标准。

不同的测试用例具有不同的测试重点, 根据测评不同方面的性能损失, 采用表 1 所示的 6 组的测试用例。

在表 1 中, asctime-1000.sh 是循环执行 1 000 次 asctime 操作, 在 Linux 中 asctime 仅仅是单一的系统调用操作, 内存的负载很小, 所以本测试用例旨在测试审计模块对大量重复 Kernel 系统调用操作时开销的测量。文件解压和文件压缩侧重于计算密集型应用的, 有着较大的内存负载。本地文件复制测试用例是 I/O 密集型应用操作, 其中包含大量的读/写离散型内存负载操作。远程

表 1 基准测试应用程序

测试用例	命令
asctime	asctime-1000.sh
文件解压 (bz2)	tar xzf linux-source.tar.bz2
文件压缩 (gz)	tar zcf linux-source-dir
本地文件复制	cp-r linux-source-dir target directory
远程文件复制	scp-r linux-source-dir target directory
Telnet	telnet remote host

文件复制和 Telnet 则分别是重量级和轻量级网络 I/O 负载测试用例。

本实验中, 初始化客户虚拟机 VM_{src} 和 VM_{dst} 各分配 1 024 MB 的物理内存, 在部署审计模块的前后, 在 VM_{src} 虚拟机上分别执行各测试用例。对于单机测试用例 asctime、文件解压、文件压缩、本地文件复制在 VM_{src} 客户虚拟机中执行。另外, 远程文件复制和 Telnet 测试用例的源主机和目的主机分别为 VM_{src} 、 VM_{dst} , 并且实验结果统计的是源主机 VM_{src} 的额外计算开销。6 组测试用例的实验结果如图 8 所示。

从图 8 中看出, 针对 6 种不同的测试用例负载, 相对于未部署审计模块的情况下, 各测试用例的执行时间均有一定程度的增加, 这说明虚拟机性能有一定的损失, 同时也不难看出, 审计模块对各个测试程序的性能均有不同程度的影响。其中, 对于 asctime 影响最小为 2.6%, 对远程文件复制测试用例的性能惩罚性开销为 9.8%。需要指出的是, 其中 acstime 性能损失较小, 其主要原因是频繁的系统调用后, 内核执行系统调用的时候处于内核上下文。根据现代操作系统局部性的思想, 在后续的 hypercall 调用过程中, acstime 直接读入内存中的已有数据, 引起较小的 P2M 表更新, 由此带来较小性能损失。作为内存

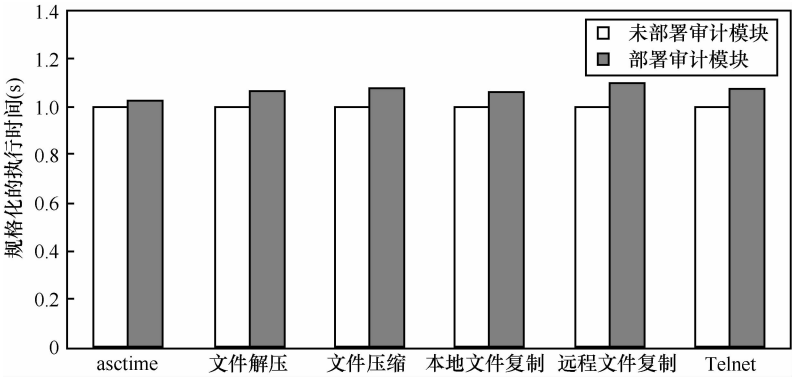


图 8 部署审计模块和未部署审计模块各基准程序执行时间



敏感型以及颠簸型文件解压和文件压缩测试用例,导致在 P2M 表镜像过程中较大的性能损失。远程文件复制面临着内存密集型和网络 I/O 密集型操作,其性能损失最大,其主要原因除了 P2M 表镜像过程中的性能损失,也包含密集型 I/O 操作引起的内存页表映射快速密级更新所导致的性能开销。

5 相关工作

在云计算基础设施环境服务中,如何审计 CSP 提供服务质量的 SLA 合约性,目前研究较少,现有的研究成果主要集中在两个方面:基于硬件访问时间特性的审计方案、基于信誉和历史服务记录评价系统的策略。

(1) 基于硬件特性访问时间的审计方案

HomeAlone^[7]关注单一物理机上是否存在多虚拟机并存的问题,基于这样一个场景:云租户所购买的云计算端资源是独立物理机,但是由于虚拟机技术,使得云租户无法判断自己所使用的是否是独立的物理机。Lin Y^[6]等研究了虚拟机物理内存合约性审计的问题,该文从虚拟化技术仍然保持了物理硬件的访问时间的特性出发,引入 TPA (trusted party auditor) 机制,通过在 Xen 中部署底层审计模块,由 TPA 触发审计进程,审计进程通过“限制性循环机制”生成用户合约物理内存的海量数据,达到尽可能占用完虚拟机物理内存的目的。若实际物理内存与用户合约物理内存不相符,则此时会触发虚拟内存,即 Guest OS 将用户数据写入硬盘空间,直接影响到后续对数据所消耗的时间。审计过程的核心思想是通过对物理内存大小的海量数据在内存中对其操作所消耗的时间的对比及对该数据全部或部分在硬盘中时该操作消耗时间的比较,判断 Guest OS 是否分配了合约性的物理内存,如果内存没有按照 SLA 约定,用户生成的数据要被存放到硬盘中。

(2) 基于信誉和历史服务记录评价系统的 SLA 合约性方案

面对众多的 CSP,如何根据 SLA 选择合适的云计算平台,参考文献[19]提出了支持 SLA 审计的云计算平台,其主要思想是通过信誉系统(reputation system)评估云平台的可靠性。同时结合 SLA 模板池,在云平台 and 云租户之间提供 SLA 协商机制,便于云租户选择云计算平台。Wang Z L^[20]等提出基于虚拟化技术,通过 SLA 感知的云基础设施供给方案,其主要目的是在不破坏 SLA 合约性的情况下,实现 CSP 和云租户利益的双赢。

目前大多数的合约性审计模型都是利用不同硬件在访问时间上的巨大差异,很少是通过部署 VMM 内核层的审计模块,大多时候只能获得一定范围精度上的审计结果。其中由于网络的反馈结果的延迟等诸多外部因素,云平台软硬件等综合性因素也直接影响了审计结果的可信性和可靠性。基于信誉系统的主观性较大,在较大程度上影响了结果的可信性。本文在现有成果的研究基础上,提出了通过部署 Xen 内核层的审计模块的高效解决方案,同时提出了基于用户数字证书的密钥传出方案,支持了可信策略和可信告警的双向传输,较好地弥补了已有研究成果的一些不足。

6 结束语

本文提出了一种面向云计算平台的虚拟机运行环境内存 SLA 合约性审计机制。作为云平台自身可信性度量的补充,设计了云平台内虚拟机内存 SLA 合约性度量机制,同时实现了由云平台内部度量单元向云租户报告自身的度量结果,通过带身份认证的密钥交换协议保证了审计系统和云租户交互的安全性。利用 Linpack 标准测试用例在不同内存负载情况下对本文原型系统进行了测试,验证了审计模块的有效性。接着通过不同的负载测试用例对原型系统的额外存储开销和计算开销进行了理论分析和实验验证,其中,系统最大额外计算性能开销为 9.8%,验证了审计模块所带来的性能惩罚在可接受的范围之内,满足轻量级应用的要求。

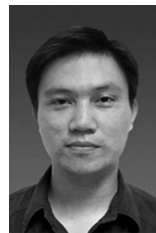
本文下一步的工作拟对影子内存映射模式进行进一步的研究,设计出针对影子工作模式 Guest OS 物理内存测量的功能模块,进一步丰富和完善本系统的应用场景,本模型的提出对云计算商业化发展有着重要的推动作用,同时,在云计算可信服务测评研究领域具有非常广阔的基础化应用前景。

参考文献

- 1 Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in thirdparty compute clouds. Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, USA, 2009: 199~212
- 2 Amazon relational database service (Amazon RDS). <http://aws.amazon.com/rds>, 2012
- 3 Xen, the powerful open source industry standard for virtualization. <http://www.xen.org>, 2012

- 4 Azab A M, Ning P, Wang Z, *et al.* HyperSentry: enabling stealthy incontext measurement of hypervisor integrity. Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, USA, 2010:38~49
- 5 Zhang Y, Juels A, Oprea A, *et al.* Home alonecoresidency detection in the cloud via sidechannel analysis. Proceedings of the 23rd IEEE Symposium on Security and Privacy, Oakland, USA, 2011:313~328
- 6 Ye L, Zhang H, Shi J, *et al.* Verify cloud service level agreement. Proceedings of IEEE Global Communications Conference, Anaheim, USA, 2012:777~782
- 7 Wang X, Zang J, Wang Z, *et al.* Selective hardware/software memory virtualization. Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Newport Beach, USA, 2011: 217~226
- 8 Zhu J, Jiang Z, Xiao Z, *et al.* Optimizing the performance of virtual machine synchronization for fault tolerance. IEEE Transactions on Computers, 2011, 60(12): 1718~1729
- 9 Szefer J, Lee R B. Architectural support for hypervisorsecure virtualization. Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, London, UK, 2012: 437~450
- 10 Payne B D, Carbone M D P, Lee W. Secure and flexible monitoring of virtual machines. Proceedings of the 23rd Annual Computer Security Applications Conference, Miami Beach, USA, 2007: 385~397
- 11 Tian H, Zhan Y, Wang Y M. Analysis of host authentication mechanism in current POD copy protection system. IEEE Transactions on Consumer Electronics, 2005, 51(3): 922~924
- 12 Liu L, Chu R, Zhu Y, *et al.* DMSS: a dynamic memory scheduling system in server consolidation environments. Proceedings of the 15th IEEE International Symposium on Object/Component/ServiceOriented RealTime Distributed Computing Workshops, Shenzhen, China, 2012: 70~75
- 13 Baker Z K, Prasanna V K. A computationally efficient engine for flexible intrusion detection. IEEE Transactions on Very Large Scale Integration Systems, 2005, 13(10):1179~1189
- 14 Heo J, Zhu X, Padala P, *et al.* Memory overbooking and dynamic control of xen virtual machines in consolidated environments. Proceedings of 11th IFIP/IEEE International Symposium on Integrated Network Management, New York, USA, 2009: 630~637
- 15 Long N, Colin B, Nieto G, *et al.* Automated proofs for diffiehellmanbased key exchanges. Proceedings of the 24th IEEE Computer Security Foundations Symposium, France, 2011: 51~65
- 16 HPLA portable implementation of the highperformance linpack benchmark for distributedmemory computers. <http://netlib.org/benchmark/hpl>, 2012
- 17 Kistler M, Gunnels J, Brokenshire D, *et al.* Programming the linpack benchmark for roadrunner. IBM Journal of Research and Development, 2009, 53(5): 1~11
- 18 SPEC CPU2000. <http://www.spec.org/cpu2000>, 2012
- 19 Wang M, Wu X, Zhang W, *et al.* A conceptual platform of SLA in cloud computing. Proceedings of the IEEE 9th International Conference on Dependable, Autonomic and Secure Computing, Sydney, Australia, 2011:1131~1135
- 20 Wang Z, Tang X, Luo X. Policybased SLAaware cloud service provision framework. Proceedings of 7th International Conference on Semantics Knowledge and Grid, Beijing, China, 2011:114~121

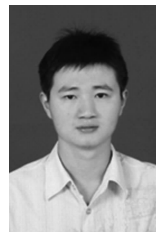
[作者简介]



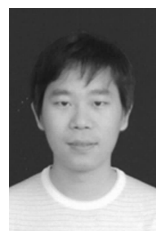
李攀攀,男,哈尔滨工业大学博士研究生,主要研究方向为云计算、网络安全等。



张宏莉,女,博士,哈尔滨工业大学教授,主要研究方向为云计算、网络安全、网络计算等。



邓会敏,男,哈尔滨工业大学硕士研究生,主要研究方向为云计算、网络安全等。



周志刚,男,哈尔滨工业大学博士研究生,主要研究方向为云计算、数据库安全等。

(收稿日期:2013-03-28)