ShadowMonitor: 一种虚拟机高效安全监控框架

Bin Shi, Lei Cui, Bo Li, Xudong Liu, Zhiyu Hao, and Haiying Shen

in RAID 2018

报告人：沃天宇
北京航空航天大学

报告框架

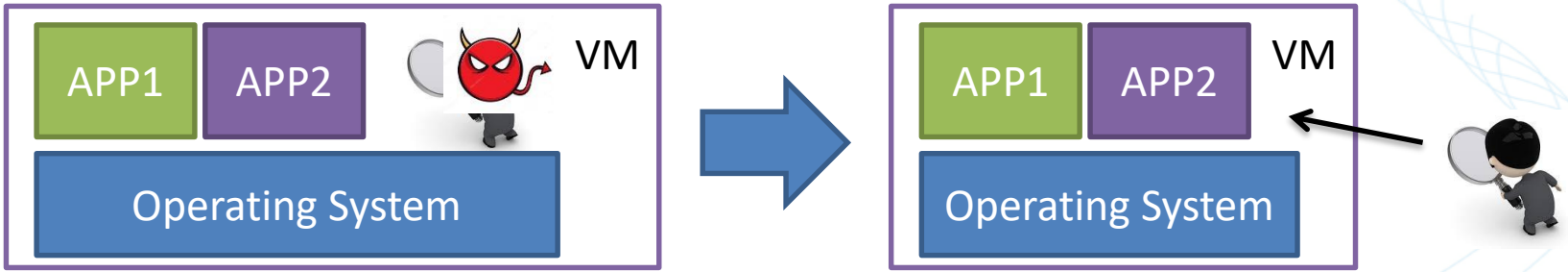
- 背景
- 问题
- 系统设计
- 系统实现
- 实验结果

云际中的监控需求

- **计算与数据存储资源云际之间共享**
  - 运行在多个云中
  - 相似抽象（VM？）
- **如何构建“可信”的资源分配与使用“痕迹”**
  - 可信：上链？
  - 痕迹（ Provenance ）来源？
  - 即便VM OS不可控，也不受“污染”
  - 还要高效（低overhead）

背景

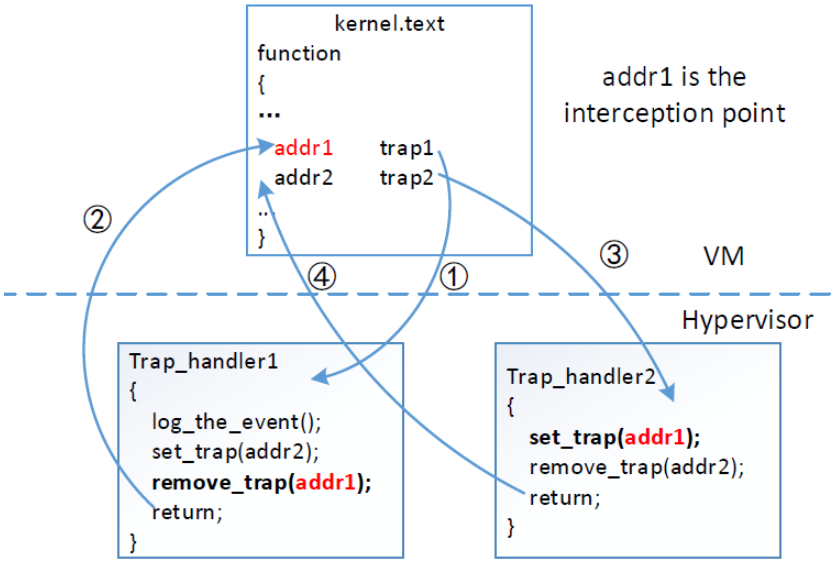
- 虚拟机实时事件监控技术
  - 广泛用于入侵检测，安全监控，入侵取证
  - 被监控虚拟机不可信，传统检测工具不法保证安全



- 虚拟机监控工具被放置在虚拟机外部(如LibVMI, Drakvuf)
  - **需要截获大量系统事件**
    - 包括进程切换, 系统调用, 中断, etc.
  - **需要在观测另一个地址空间内的对象**

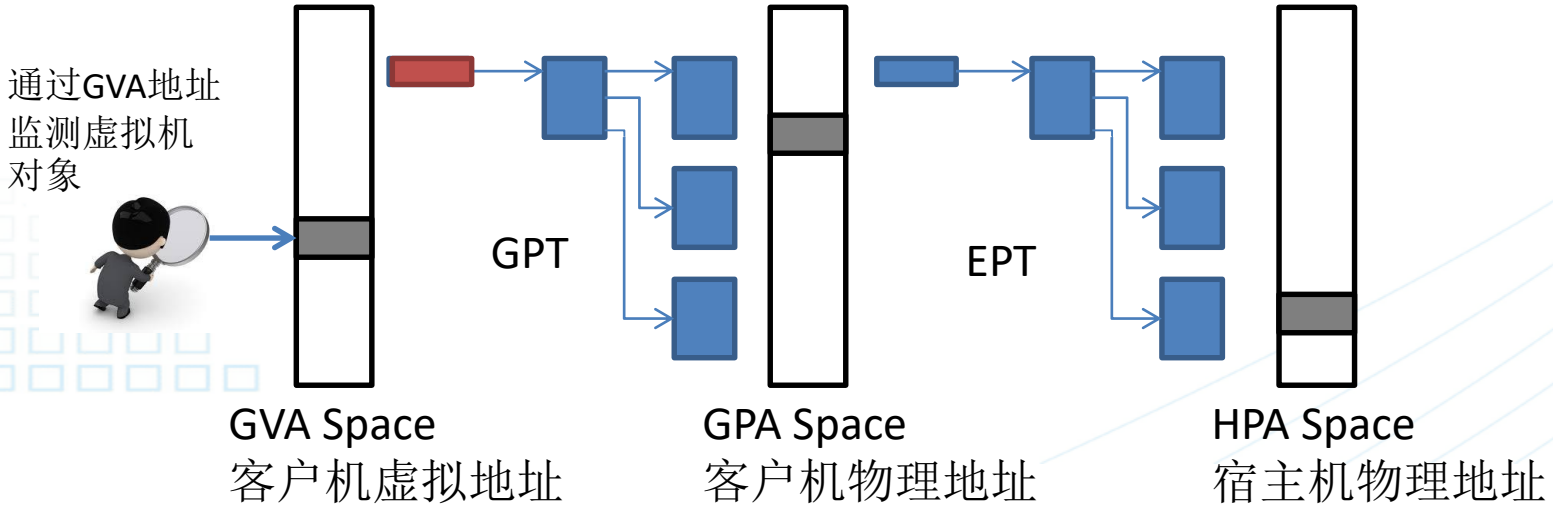
问题一

- **频繁的上下文切换**
  - 事件截获功能导致
  - 1 事件= 2 VM-exit + 2 VM-enter
  - VM-exit和VM-enter开销很大
- **上下文切换引起的性能开销问题限制了事件监控规模**



问题二

- 软件模拟内存地址转换
  - 监控工具每次监测(introspection)虚拟机中的对象都需要模拟一系列操作
  - 模拟内存转换速度远低于普通MMU或EPT
  - 1次introspection = 多次内存读操作（取决于页表深度）

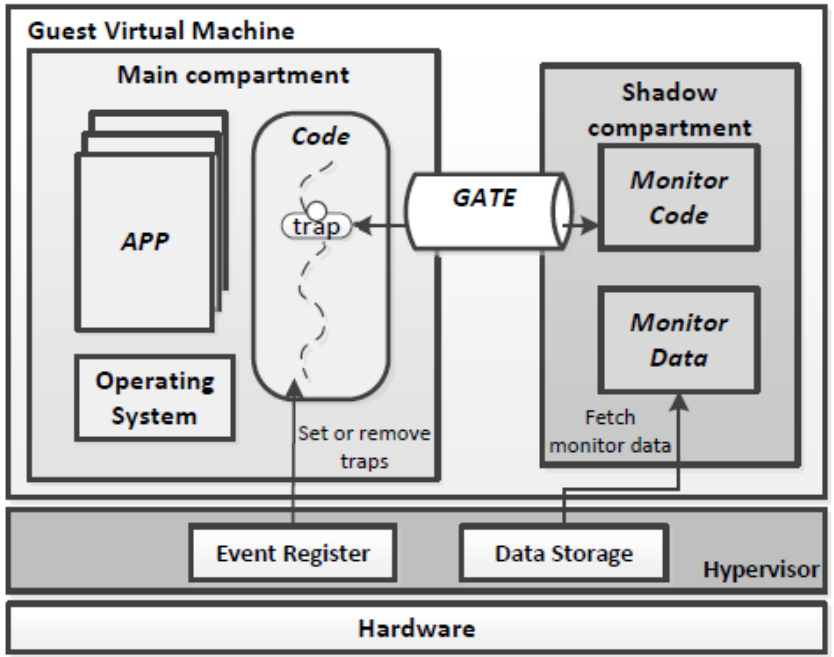


问题三

- 因为性能开销原因，许多工作开始将虚拟机事件监控工具放置回虚拟机中
  - 牺牲部分安全性
- SIM 提出用单独的影子页表为虚拟机监控工具提供一个独立的地址空间
  - 无法防御地址重定向攻击address translation redirection attack (Mapping attack)
  - 缺乏运行时动态配置的灵活性
  - 限定使用影子页表，无法适配硬件MMU (e.g., EPT or RVI)

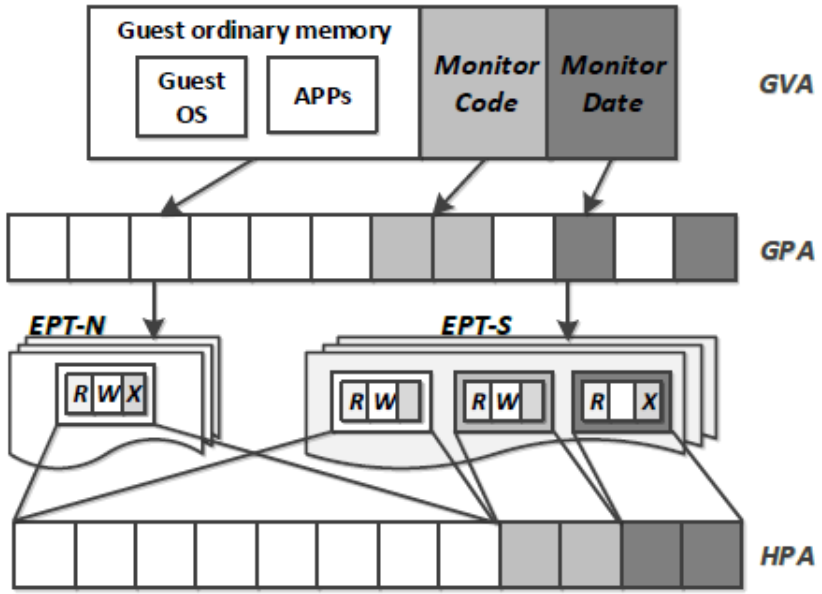
ShadowMonitor结构

- 硬件支持
  - Multi-EPT特性, VMFUNC指令
- ShadowMonitor 将虚拟机分解为两个独立的地址空间
- 主地址空间（Main Compartment）是虚拟机操作系统和上层应用的运行环境
- 影子空间（Shadow Compartment）是虚拟机监控工具的运行环境
  - 包含一小段内存：代码段和数据
- 两个空间可以通过Gate自由切换
- 该系统的安装和配置由hypervisor完成



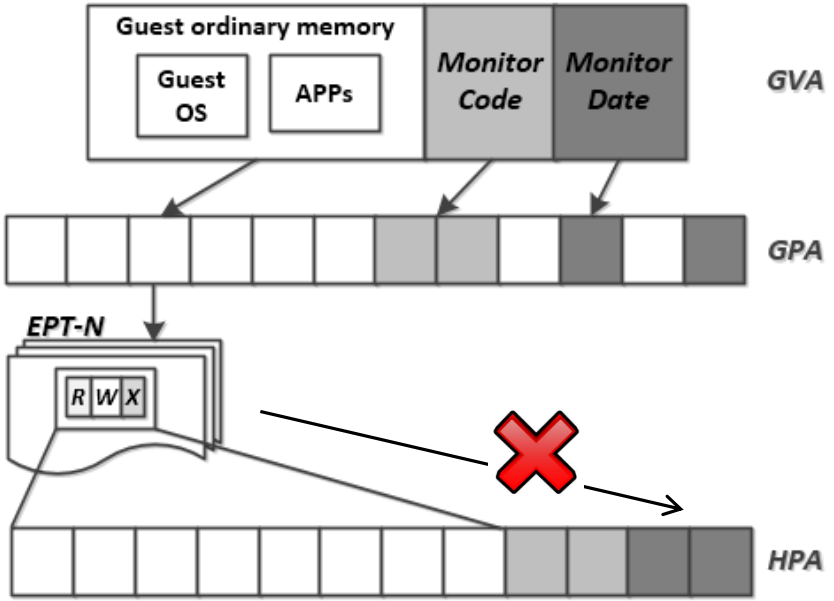
基本思路

- **1个系统，2个的EPT**
  - 使用硬件Multi-EPT特性实现两个地址空间的隔离
  - 将监控工具放置在另一个独立的地址空间内
  - 严格的访问控制
- **结果:**
  - 轻量的上下文切换（EPT switch）
  - 以本地速度的访问虚拟机内存对象
  - 硬件支持的隔离
  - 防止地址重定向攻击（address translation redirection attack）



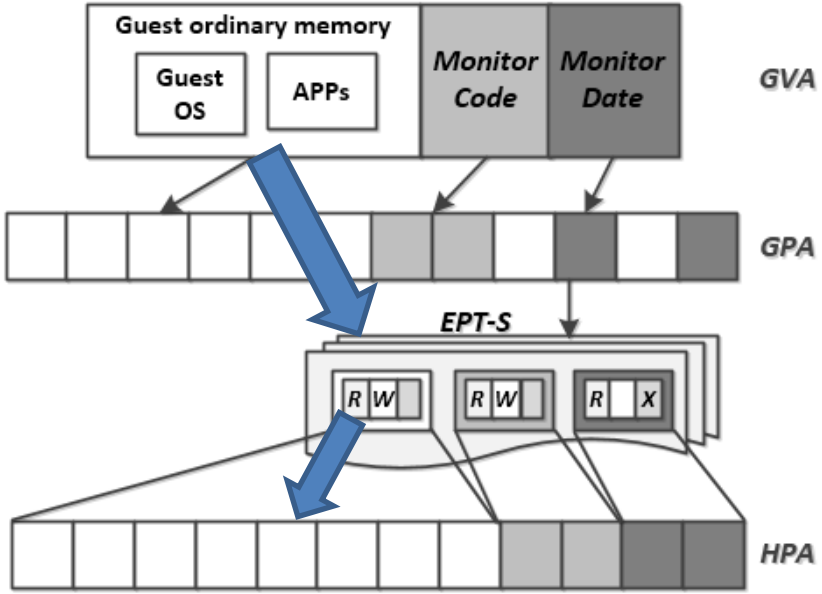
主空间

- 使用EPT-N
  - 在该EPT中，我们令对应监控代码和监控数据的客户机虚拟机地址（GVA失去正确的映射
  - 不可信的客户机无法查看更无法修改监控代码和监控数据



影子空间

- 使用EPT-S
  - 在该EPT中，我们令所有客户机内存映射都和主空间保持一致，但取消其运行权限
  - 客户机对象可以被监控工具以本地速度直接读取访问

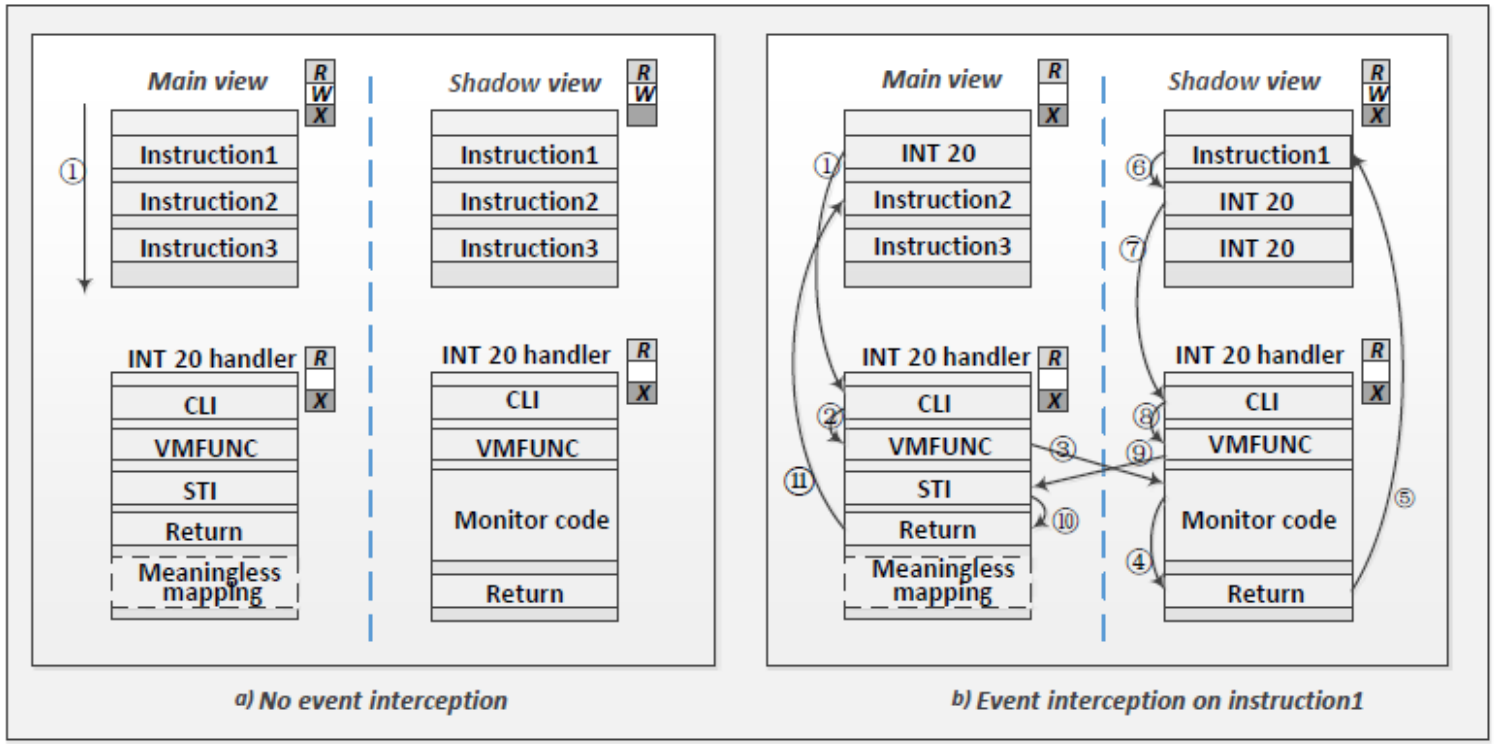


细节需求

- 监控事件运行时需要能触发虚拟机监控工具
  - 1. 当系统运行到监控点时，系统需要切换到影子空间（Shadow compartment）
  - 2. 此时记录事件和相应上下文(e.g. 寄存器信息，栈信息)
  - 3. 切换回主空间
  - 4. 越过监控点

解决方案

- 使用 INT 20中断



特殊情况

- 1. 如果被替换指令长度小于INT20?
  - 让影子空间执行2条指令
- 2. 如果被替换指令是一个跳转指令？
  - 跳转指令跳转到监控工具所在地址
    - 阻止并警报(监控工具所在的虚拟地址在主空间内无意义，指令正常情况下不应该跳转至这里)
  - 让系统在跳转终点处触发INT20，并切换回主空间
    - 使用#VE特性, 让EPT page fault触发#VE，从而产生INT 20
    - 将被替换指令附近的可执行指令替换为INT20

- **我们不鼓励用户在这两种情况下插入监控点**

安全分析

- **中断响应表（IDT）重定向攻击**
  - 如何保证INT 20和INT 20响应机制的安全
- 恶意攻击可以通过**攻击IDT和中断响应机制**的方式使我们的地址空间切换功能失效。
- 解决方案
  - 写保护INT20处理函数段
  - 写保护中断响应表（IDT）
  - 通过将LIDT指令设置为敏感指令以写保护IDTR寄存器
  - 写保护上述内存的对应客户机页表项

安全分析

- **INT20注入攻击**
- 恶意攻击可以故意执行INT20指令从而发起地址空间切换
  - 导致隔离性出现问题
- 采用相关工作提出的解决方案
  - 采用LBR信息(last branch recording)[1]
    - 检查将分支切换至INT20处理函数的代码位置，若不是特定的监控点，则判断为恶意攻击

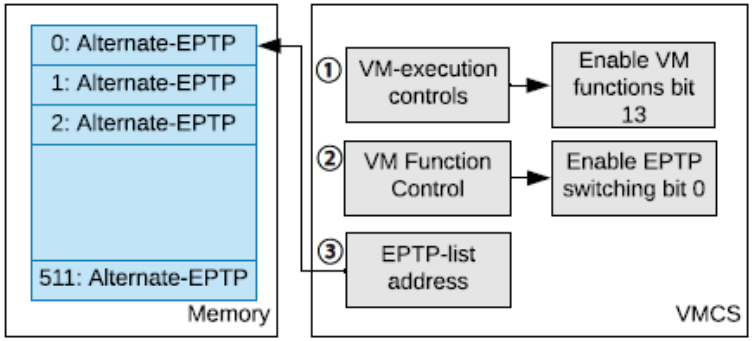
[1] Monirul I. Sharif et al. Secure in-vm monitoring using hardware virtualization. In the Conference on Computer and Communications Security, CCS 2009.

安全分析

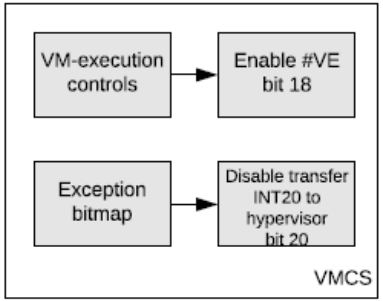
- **VMFUNC注入攻击**
- 恶意攻击可以故意执行VMFUNC指令从而发起地址空间切换
  - 导致隔离性出现问题
- 解决方案
  - 强制使地址空间转换回主空间
    - 使用#VE
    - 将监控代码页的第一条指令替换为VMFUNC

系统实现

- 原型系统
  - qemu-kvm-2.4.1Linux kernel 4.10.2 (Ubuntu 14.04 64-bit LTS)
  - Patch了KVM代码以支持多EPT，EPT切换，以及#VE(虚拟异常处理)
  - 预留新的用户层接口IOCTL
  - 客户机操作系统负责在启动后预留虚拟地址，并调用hypervisor接口
  - Hypervisor在收到接口指令后负责创建隔离地址空间并部署监控代码
  - 目前只支持linux客户机



a) Enable EPTP-switching



b) Enable #VE

实验

- **我们使用LibVMI 和SIM 作为基准**
  - LibVMI:
    - 一个最常用的开源虚拟机监控工具（安全工具在虚拟机外部）
    - 是许多高级安全监控工具（如Volatility, Drakvuf）的底层部分
  - SIM:
    - 将安全工具部署在虚拟机内部从而加快监控速度的工具
- **实验环境:**
  - 硬件: Intel Core i7-6700 3.4GHz处理器, 16GB DDR内存, 1000GB WD磁盘
  - 客户机配置： 2 vcpus, 4GB内存

实验

- 地址空间切换速度
  - 执行一次虚拟机地址空间切换的时间

Operations	Average Time(ns)
syscall	69.38
VMFUNC	75.26
VM-exit	653.71

- 监控代码调用速度
  - 截获事件，切换至虚拟机监控工具处，记录事件并返回系统正常执行状态的总时间

Approaches	Average Time(ns)	Standard Deviation(ns)
ShadowMonitor	471	63.8
SIM	488	61.4
LibVMI	5231	139.1

实验

- **客户机内存访存速度**
  - 我们测量了从虚拟机监控工具读取不同大小客户机内存所用的时间
  - 测试时清空了TLB 并禁用了LibVMI缓存

Bytes	ShadowMonitor(μs)	LibVMI(μs)	SIM(μs)
4	0.357	17.3	0.187
64	0.351	17.4	0.194

实验

- **整体系统性能开销**
- **实验环境:**
  - 监控所有的系统调用和进程切换
- **测试工具**
  - 内核编译(开发应用负载)
  - Apache bench (WEB负载)
  - UnixBench (系统多方面性能测量)
  - 文件压缩(日常应用负载)

实验

- 我们测试了不同的系统
  - 不开启监控, ShadowMonitor, LibVMI，SIM
  - 测量测试工具的吞吐量和工作完成时间
- 结果:
  - ShadowMonitor在各方面性能上都胜过LibVMI
  - ShadowMonitor在内存密集型负载上性能优于SIM

Benchmark	No monitor	Our Overhead	LibVMI Overhead	SIM Overhead
Kernel Compile	4106.87s	7.34%	**65.1%**	**54.8%**
Apachebench	4323lps	5.48%	74.2%	**41.9%**
File Compress	41.69s	1.12%	8.47%	1.55%
Whetstone	3339Mwips	0.09%	3.83%	0.1%
Process Creation	1785.8lps	0.71%	9.1%	**613%**
File Copy[4]	251.1MBps	10.1%	**93.9%**	11.3%
System Call	2.7Mlps	119%	**7134%**	123%
Average	-	20.55%	**1056%**	**120.8%**

结论

- 基于Multi-EPT等硬件支持，ShadowMonitor 在性能方面和现有虚拟机监控工具相比有很大提升
- 硬件提供的multi-EPT 隔离性为 ShadowMonitor提供了更安全的工作环境。

Q&A?