Record-Replay Architecture as a General Security Framework

Yasser Shalabi UIUC

Mengjia Yan

Nima Honarmand

About

- The i-acoma group at UIUC
- 2018 HPCA
- 由美国国家科学基金委(NSF)部分赞助

Record and Replay

- 录音
- IBM：出于审计或问题确定的目的，可以将数据记录到数据库，然后查看并重放。
- 安全领域：
  - VM级：将分析过程和恶意软件执行进行隔离；保证透明度
  - V2E KVM+TEMU（QEMU）

背景

- 随着安全攻击变得越来越频繁和变化，人们越来越关注增强具有安全功能的处理器和系统硬件。
- 因此，处理器制造商开发了新的硬件架构，例如Intel的MPX ，AMD的安全处理器和ARM TrustZone技术。

Problem

- 硬件安全功能需要在 设计侵入性 和 方法完整性 之间取得平衡。此外，随着安全威胁的不断发展，它们需要具有 灵活性。

Engineering Record And Replay For Deployability Extended Technical Report

Robert O’Callahan

Chris Jones Albert Noll

Nathan Froyd

Whole-System Replay

- ReVirt（20002）：是一个早期项目，记录并重放整个虚拟机的执行。
- ReTrace（2007）：VMware曾使用相同的方法在VMware Workstation中支持记录和重放调试，但暂停了该功能。
- （2010）：全系统仿真器Simics通过确定性重新执行支持反向执行调试。

Whole-System Replay

- 这些方法需要记录和重放整个虚拟机，这是重量级的——用户必须设置和管理虚拟机，并且当用户经常只关心某些特定的过程时，必须记录和重放整个操作系统的状态。
- 这在诸如反向执行调试的记录和重放应用程序中尤其成问题，其中经常创建和恢复重放状态的检查点；在VM映像创建检查点通常比在一个或几个进程中更昂贵。

Replaying User-Space With Kernel Support

- Scribe (2010)，dOS (2010)和Arnold (2014)通过扩展具有记录和重放功能的OS内核来重放一个进程或一组进程。
- 内核更改使维护和部署更加困难——除非将记录和重放集成到基本操作系统中。 但是向内核添加侵入式新功能存在风险，这些方法需要运行修改后的操作系统内核，阻碍部署并为系统增加安全性和稳定性风险；因此如果可以在内核之外很好地实现记录和重放，那么将其移入内核可能并不理想。
- RnR-Safe

Problem

- 硬件安全功能需要在设计侵入性和方法完整性之间取得平衡。此外，随着安全威胁的不断发展，它们需要具有灵活性。

Contribution

- 提出了一种新的框架，其中记录和确定性重放（RnR）用于补充硬件安全性功能。我们称框架为RnR-Safe。
- RnR-Safe通过允许在检测攻击时不那么精确来降低安全硬件的成本，可能会报告误报（假阳性）。这是因为它依赖于即时重放，透明地验证警报是真实攻击还是误报。
- RnR-Safe使用两个重放器：一个永远在线的快速检查点重放器，定期创建检查点，以及在有威胁警报时触发的详细分析警报重放器。

An RNR SECURITY FRAMEWORK

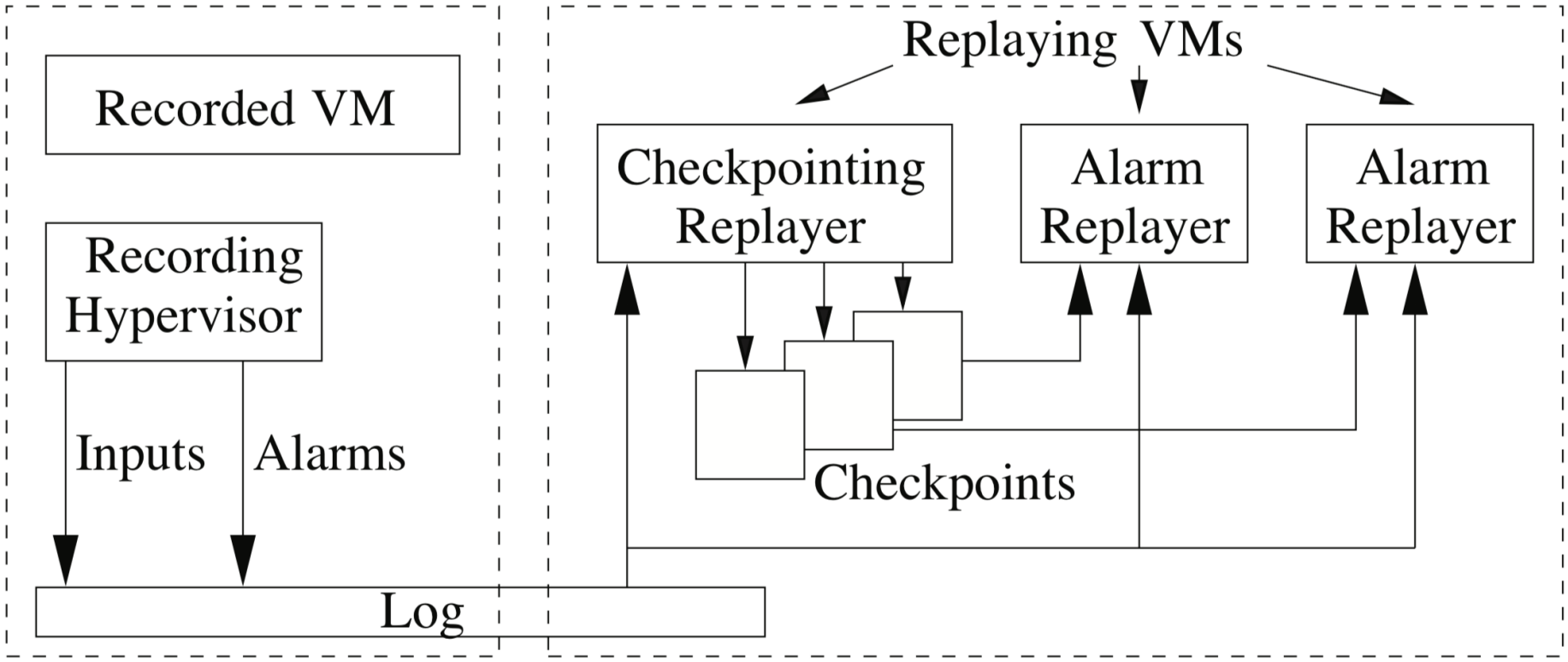


Figure 1: RnR-Safe organization.

示例应用程序

- 使用RnR-Safe来阻止面向返回的编程（ROP）攻击。
- 设计扩展了相对低开销的硬件返回地址堆栈（RAS）。
- 使用运行Linux的虚拟机上的各种工作负载评估RnR-Safe；发现RnR-Safe非常有效。
- 由于RAS硬件扩展和管理程序更改，检查点重放器的执行速度与记录的执行速度相当。此外，警报重播器需要处理很少的误报。

RnR-Safe应用

Attack	Alarm Trigger	Possible First Detection Technique	Role of Replay
ROP (this paper)	RAS mis-prediction	Manage a multi-threaded RAS, use a whitelist	Execute a kernel-compatible shadow stack algorithm
Jump Oriented Program-ming (JOP)	Stray indirect branch or call	Table of begin and end addresses of the most common functions	Verify if the target is one of the less common functions
Denial of Service (DOS)	Kernel scheduler inactivity	Counter of number of context switches	Identify reason for low switching frequency

Table 1: Examples of potential RnR-Safe uses.

返回地址堆栈 RAS

- 现代处理器使用硬件返回地址堆栈（RAS）来预测返回指令的目标。
- 当执行过程调用时，硬件将其后面的指令的地址推送到RAS的顶部。 当解码返回指令时，硬件弹出RAS顶部的条目并使用其值作为返回的预测目标。
- ROP攻击会导致RAS误预测，因为攻击者强制返回意外指令。
- 但是，RAS错误预测本身不能用作ROP攻击的指标，因为RAS有时会在良性程序执行过程中误预测。

RAS

- ROP攻击导致RAS误预测（不存在假阴性—受到攻击没有触发警报）
- RAS误预测不一定有ROP攻击（存在假阳性—没有受到攻击就触发警报）
- 充分不必要条件

Problem

- 问题：解决RAS假阳性误报；
- 挑战：需要在 硬件设计侵入性 和 方法完整性 之间取得平衡；
- 体现在对RAS误预测处理上—RAS扩展度和误报率的平衡。

RAS误预测

- 首先，多线程的影响。在多线程环境中，在从线程i到线程j的上下文切换中，RAS仍然可以保留属于线程i的一些条目。在线程j中执行代码时，这些条目可能会被错误地弹出并用于预测。如果是这样，不仅Thread j会遇到错误预测，而且当我再次执行Thread i时，这些条目不再可用于i。因此，线程i也将错误预测。
- 第二个影响是内核中的非过程返回。例如，有时在上下文切换期间内核将地址插入到软件堆栈中，稍后将用作返回指令的目标。由于匹配地址没有事先调用，因此RAS将不包含相应的条目并将进行错误预测。

RAS误预测

- RAS下溢是不精确的第三个来源。 如果代码执行许多嵌套过程调用，则RAS可能会退出一些早期的返回地址。 之后，当执行从内部调用返回并尝试弹出与外部调用相对应的入口时，RAS将为空（下溢）并将进行错误预测。
- 最后，过程调用的不完美嵌套是RAS错误预测的另一个原因，一种程序被调用但从未返回的情况。

支持多线程环境

- 为解决此问题，RnR-Safe增强了微编码虚拟化硬件，以便它还可以在上下文切换上执行以下操作：
- 首先，它将当前RAS备份到内核可达范围之外的安全内存区域。
- 然后，它根据即将到来的运行线程的需要恢复RAS状态。
- 为了知道正确的存储区域以移动数据，硬件使用管理程序设置的新硬件指针。

支持多线程环境

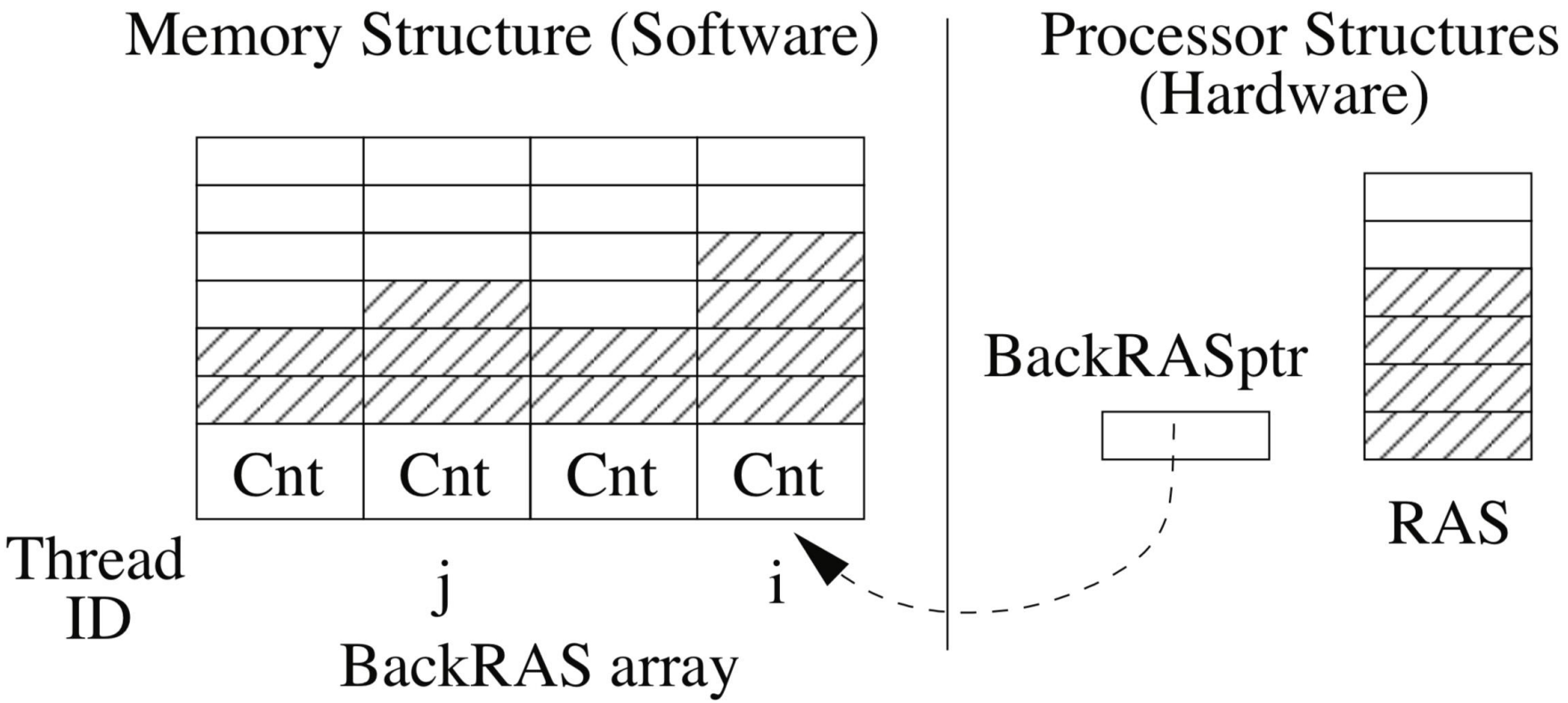


Figure 2: Structures used to support multiple threads.

支持多线程环境

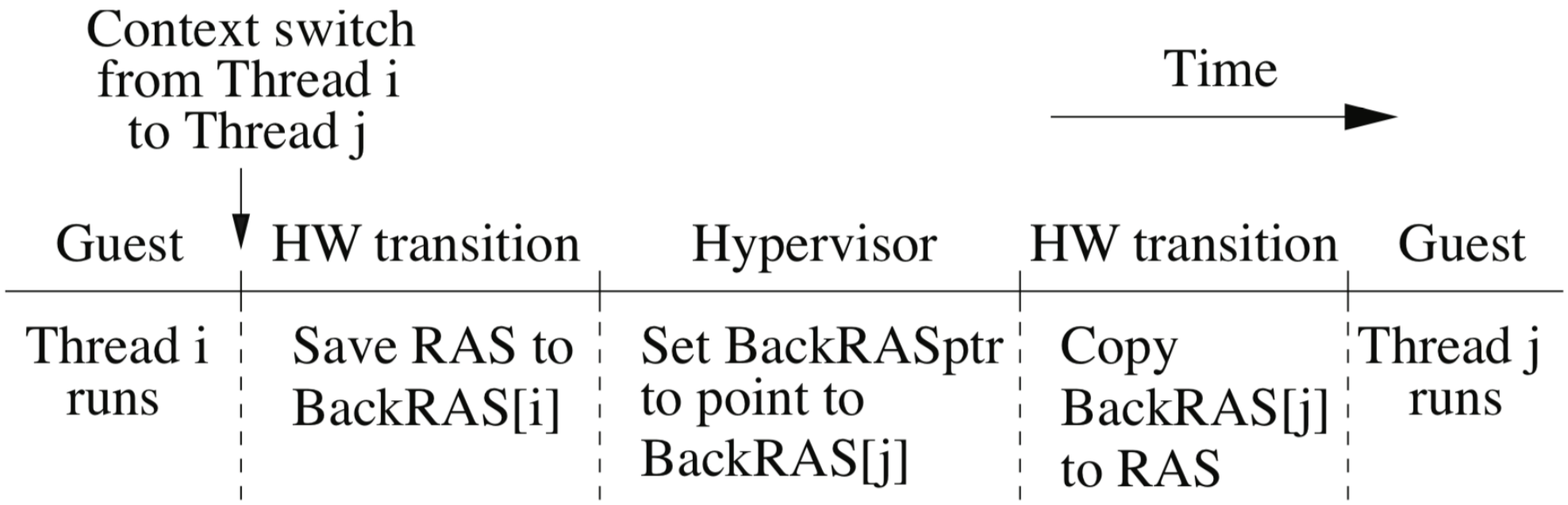


Figure 3: Algorithm and timeline to handle multiple threads.

支持非程序性返回

- 为解决此问题，RnR-Safe使用“白名单”地址表扩展了处理器硬件。
- 存在单条目返回白名单（RetWhitelist），其中指令的PC具有非程序性返回，并且目标白名单（TarWhitelist）具有可以作为该返回的目标的三条指令的PC。
- 在返回地址预测期间，如果返回PC及其目标PC与表中的条目匹配，则不会弹出RAS，也不会引发警报。
- 这些列表只能由管理程序写入。

RAS下溢和不完美嵌套

- 这两种情况很少发生，但需要非常复杂的硬件才能透明地处理。
- 对于很少发生的误报，RnR-Safe的方法是避免记录的VM中的复杂硬件，而是引发警报并依靠重放VM来处理它们。

重放

检查点重放

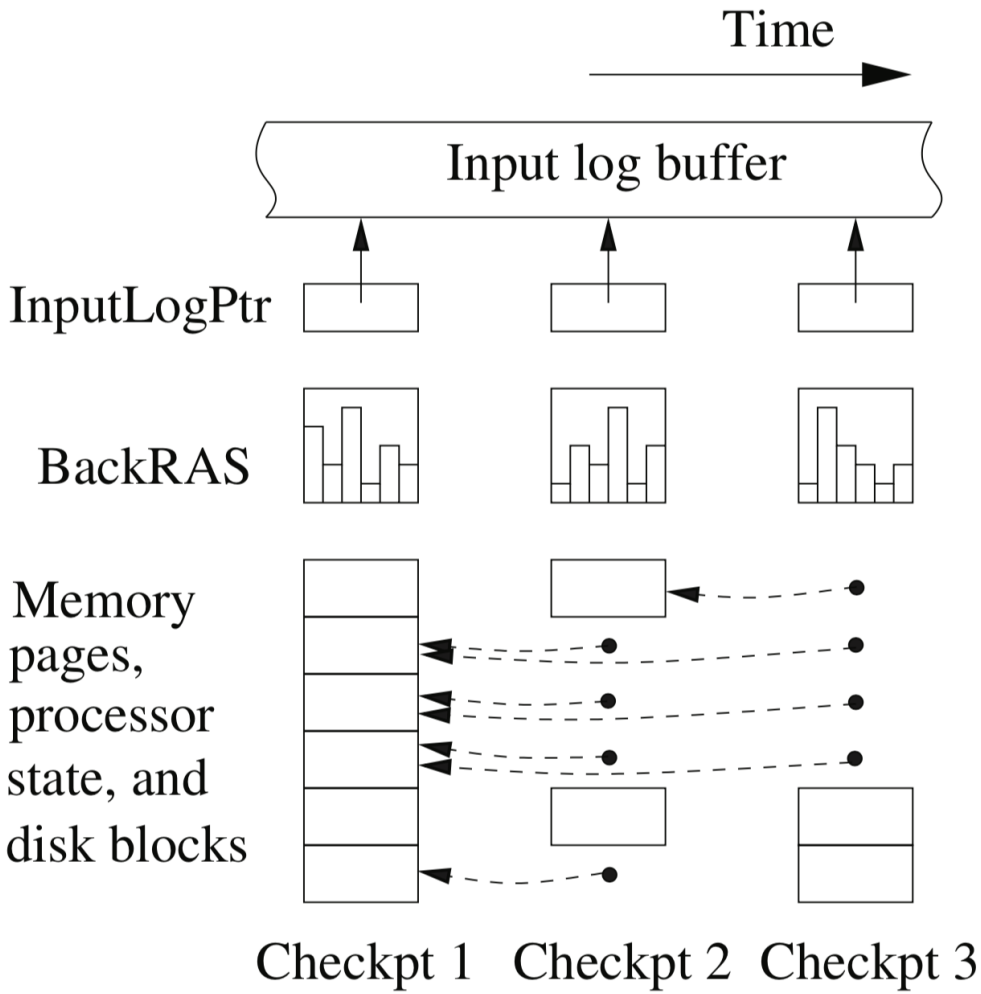


Figure 4: Checkpoints created by the checkpointing replayer.

重放

检查点重放CR

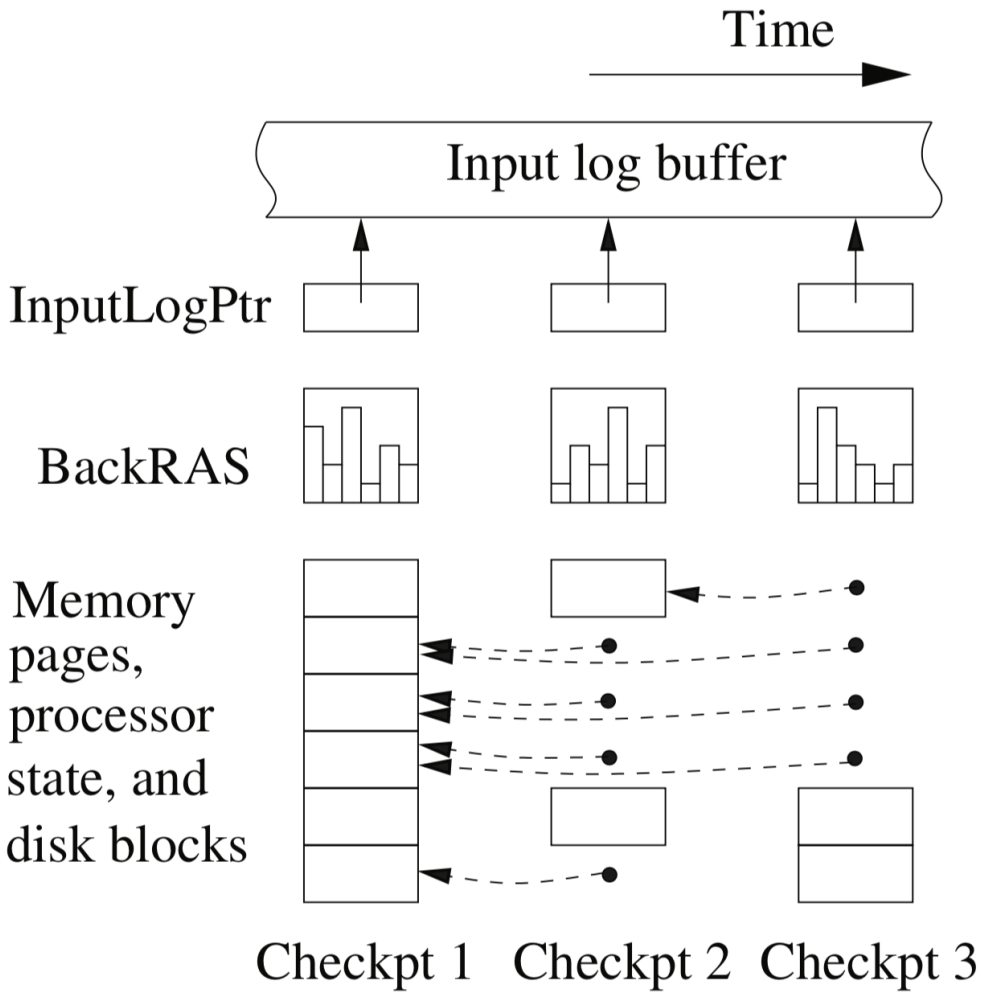


Figure 4: Checkpoints created by the checkpointing replayer.

重放—警报重放AR

- 当CR遇到警报时，它会从紧接警报之前的检查点开始启动警报重播器（AR）VM。AR将确定警报是由ROP引起还是误报。
- AR首先使用检查点初始化VM状态。 它将检查点中的所有页面和块标记为写入时复制，以避免修改初始状态。
- 然后，它将检查点的BackRAS读入一个软件数据结构，用于模拟RAS。 接下来，它将保存的处理器状态从存储器加载到处理器寄存器中。 最后，它开始执行，从检查点InputLogPtr开始读取日志。
- 一旦AR在日志中遇到警报，它就会检查RAS不匹配是否只能被解释为ROP攻击。如果是，则AR在ROP攻击点提供系统状态。专家程序员可以研究状态以收集有关攻击的信息。

实现

- VMCS （VM Control Structure）
- VMEnte
- VMExit

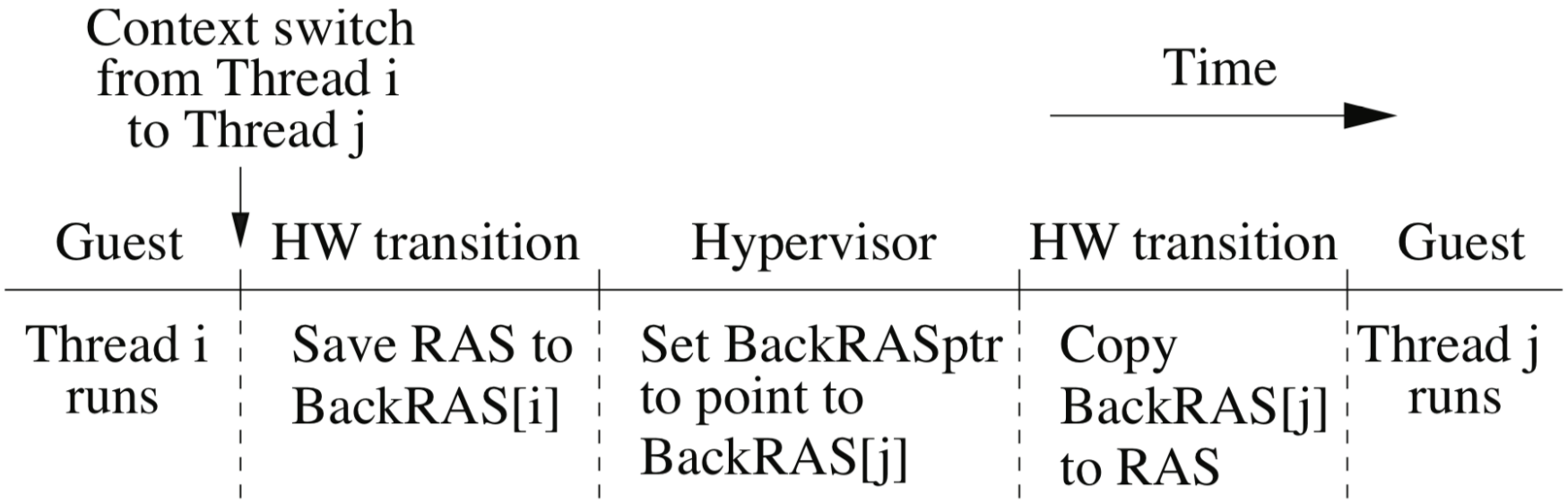


Figure 3: Algorithm and timeline to handle multiple threads.

实现

- 硬件支持（设计侵入性）：
- 重放平台（检查点），
- RAS硬件的小扩展，
- BackRASptr寄存器，
- 两个白名单表。

评估—目标

- 记录和重放（CR+AR）的开销；
- 日志生成的速率；
- 保存/恢复RAS所消耗的带宽；
- 警报的频率。

评估环境

- 1. 评估录制和重放模式的性能。
- Insight ，一种基于修改后的Linux KVM管理程序和QEMU设备的VM RnR工具。

- 2. 评估技术的正确性和我们硬件的功能特性。
- 在仿真模式下使用QEMU， 此模式可以轻松模拟硬件并评估其功能。

系统配置

Host machine	
CPU: Xeon E3-64bit,4-cores,3.1GHz	Memory: 8 Gbytes
OS: Ubuntu, Linux kernel 2.6.38-rc8	
Guest machine	
CPU: uniprocessor	Memory: 1 Gbyte
OS: Debian, Linux kernel 3.19.0	Disk: 32 Gbytes

Table 2: System configuration for performance evaluation.

Benchmark

Benchmark	Parameters
apache	-n100000 -c20
fileio	–file-total-size=6G –file-test-mode=rndrw –file-extra-flags=direct –max-requests=10000
make	linux-4.0 config with all-no
mysql	–test=oltp –oltp-test-mode=simple –max-requests=500000 –table-size=4000000
radiosity	-p1 -bf 0.005 -batch -largeroom

Table 3: Benchmarks executed.

记录开销

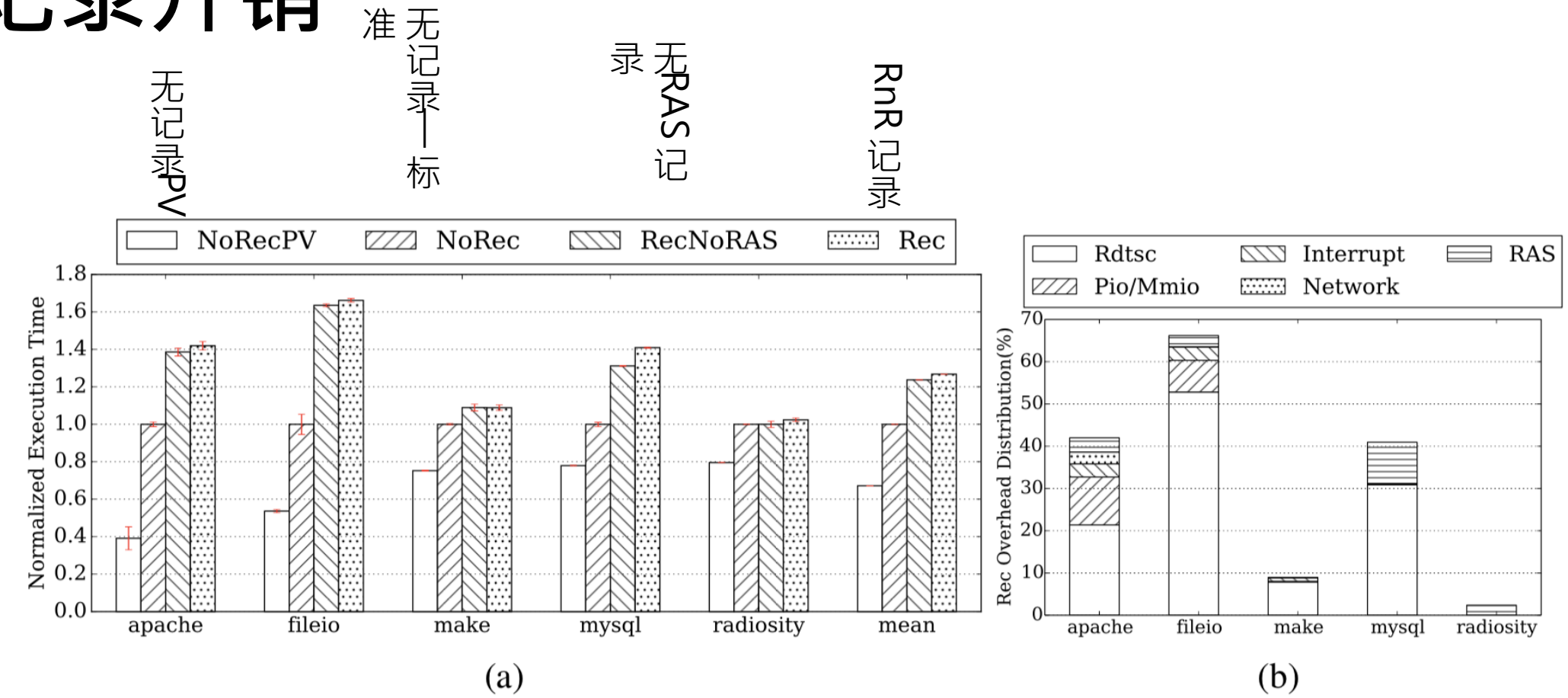


Figure 5: Execution time of recording setups (a) and breakdown of the *Rec* overhead over *NoRec* (b).

记录—开销来源

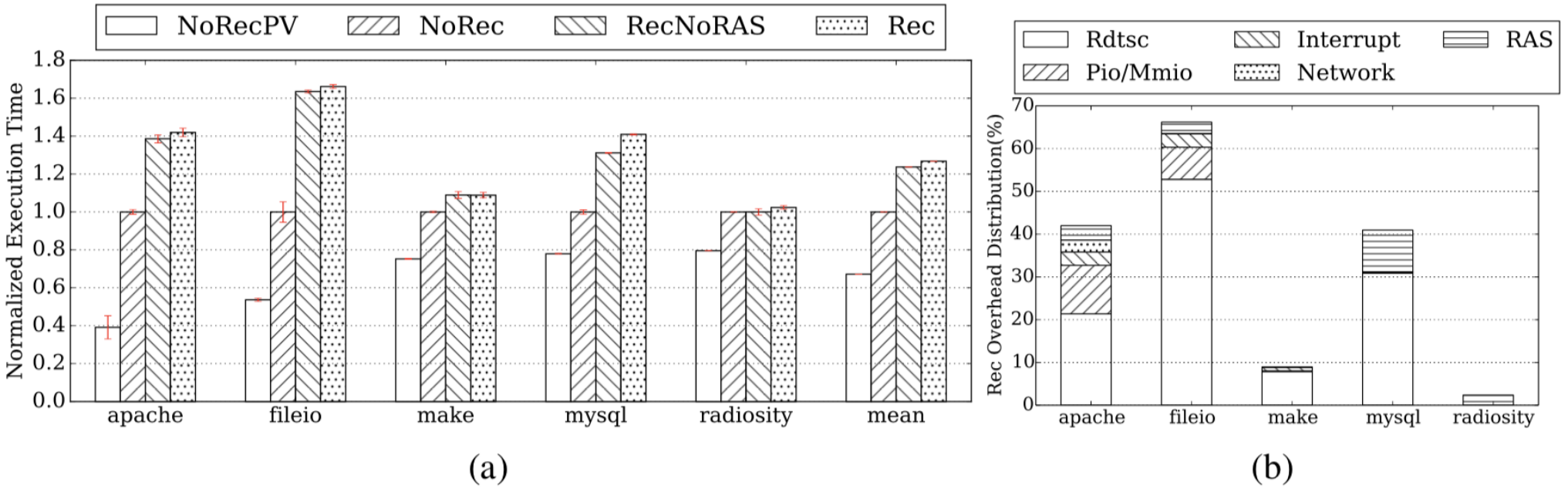
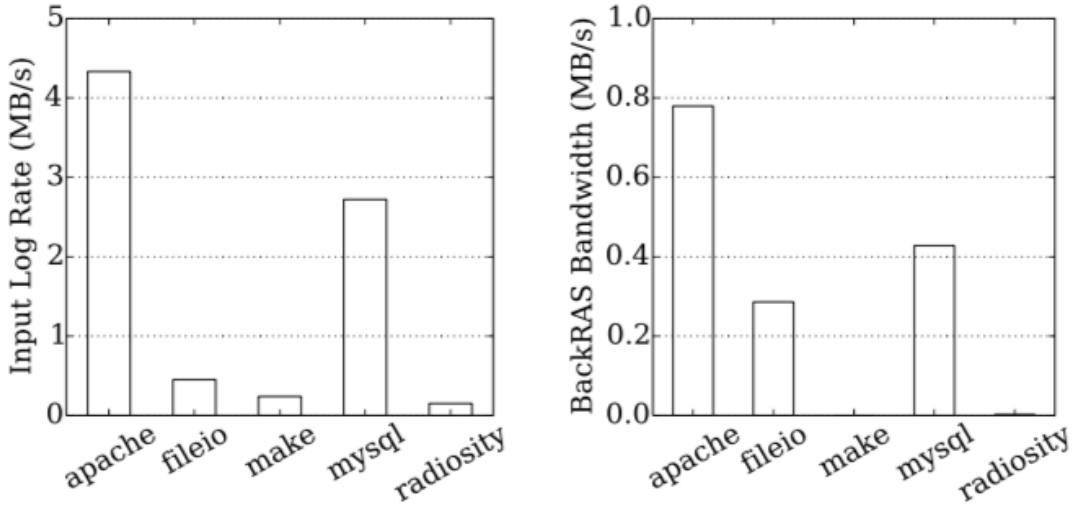


Figure 5: Execution time of recording setups (a) and breakdown of the *Rec* overhead over *NoRec* (b).

日志生成速率&RAS保存和恢复的带宽



(a) (b)

Figure 6: Input log generation rate (a) and bandwidth to save and restore the RAS at context switches (b).

警报频率

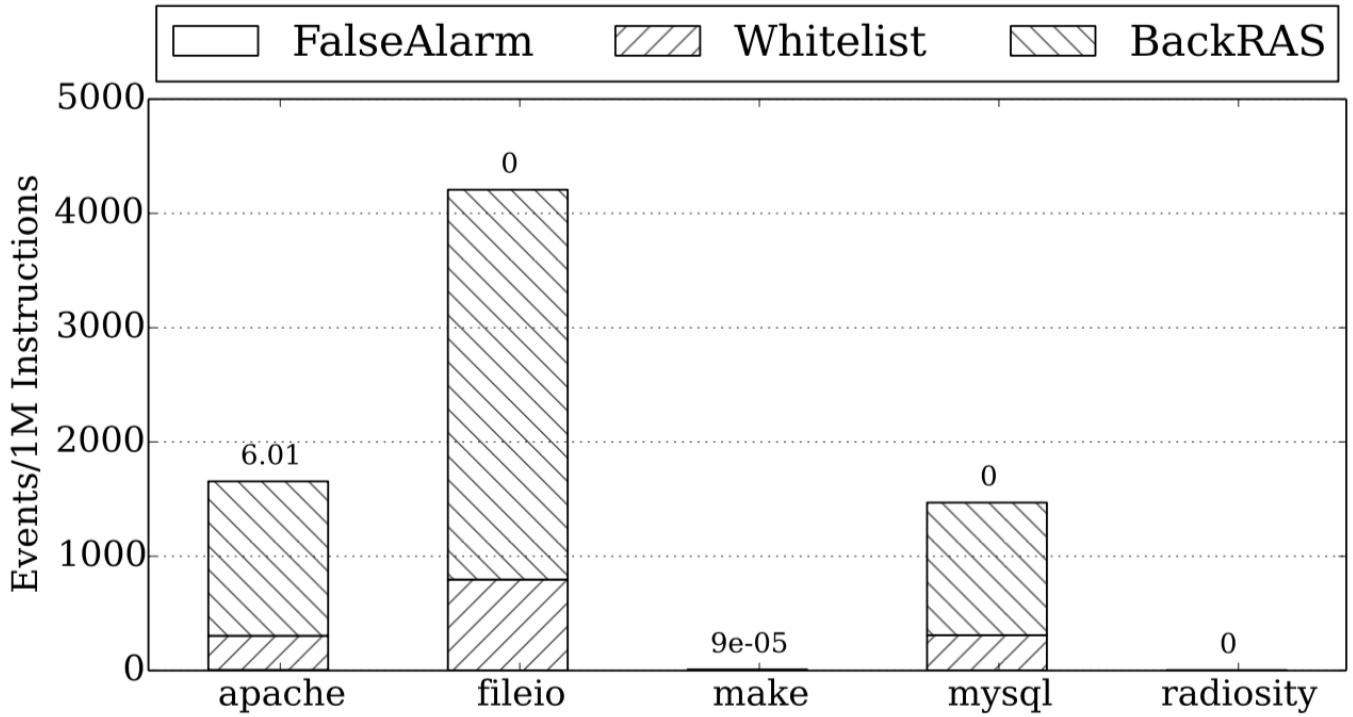


Figure 8: Kernel false alarms suppressed (*Whitelist* and *BackRAS*) and reported to the replayers (*FalseAlarm*).

重放CR

Rec:记录
RepNoChk: 无检查点的重放
RepChk5:每5S使用检查点的重放
...

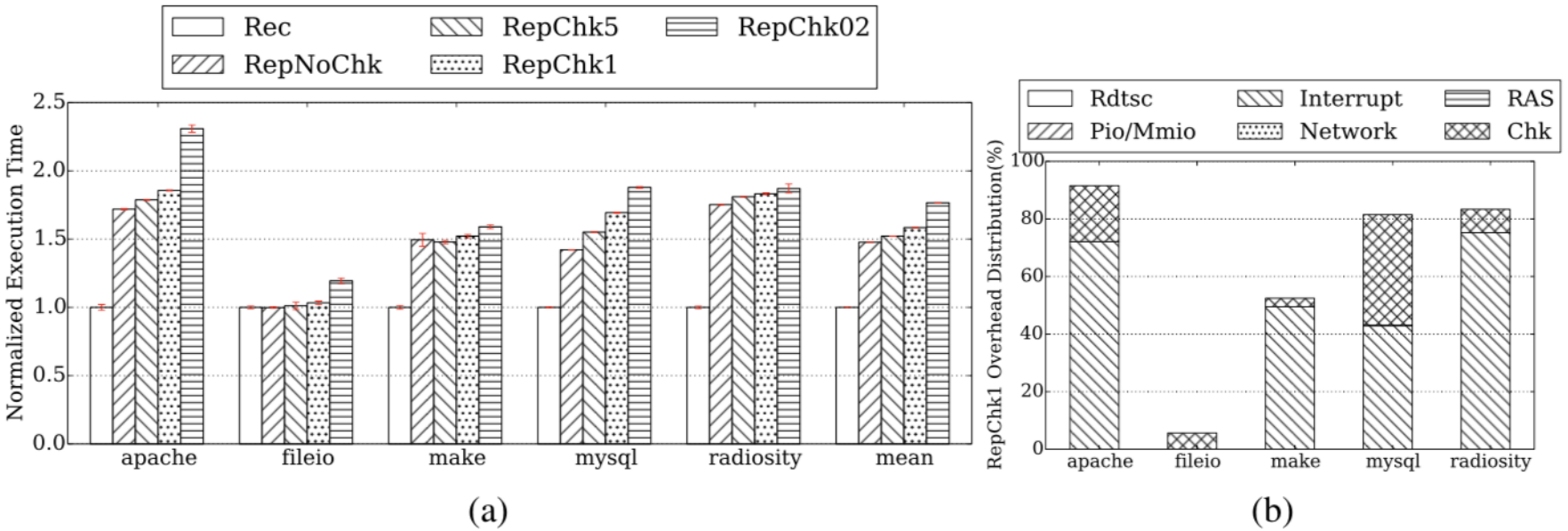


Figure 7: Execution time of checkpointing replay setups (a) and breakdown of the *RepChk1* overhead over *Rec* (b).

重放CR—开销来源

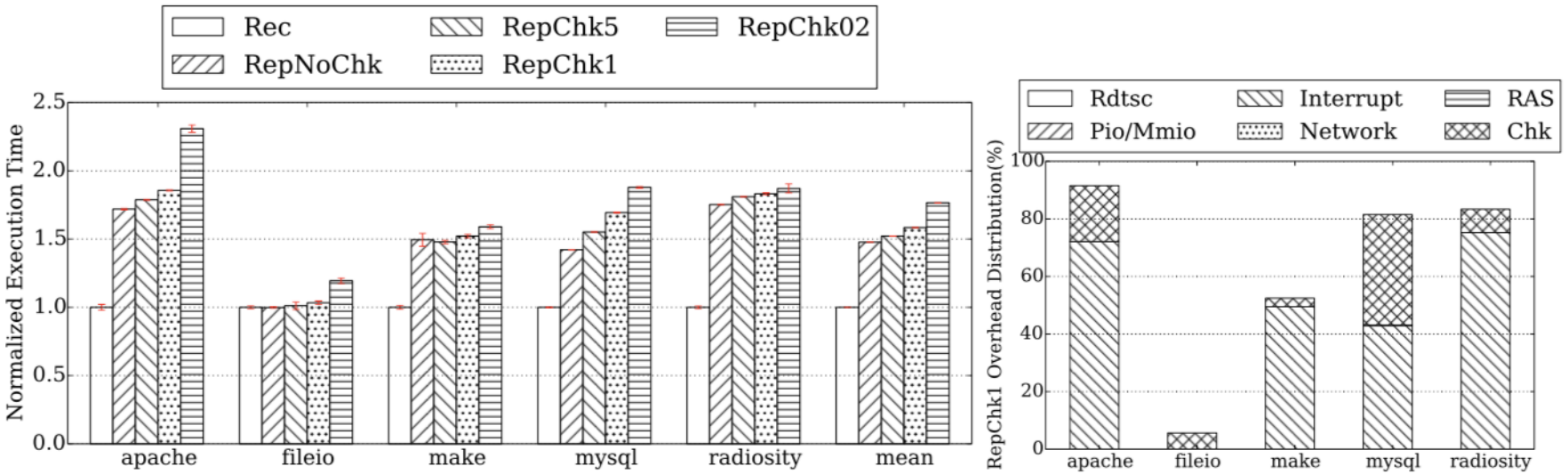


Figure 7: Execution time of checkpointing replay setups (a) and breakdown of the *RepChk1* overhead over *Rec* (b).

重放AR

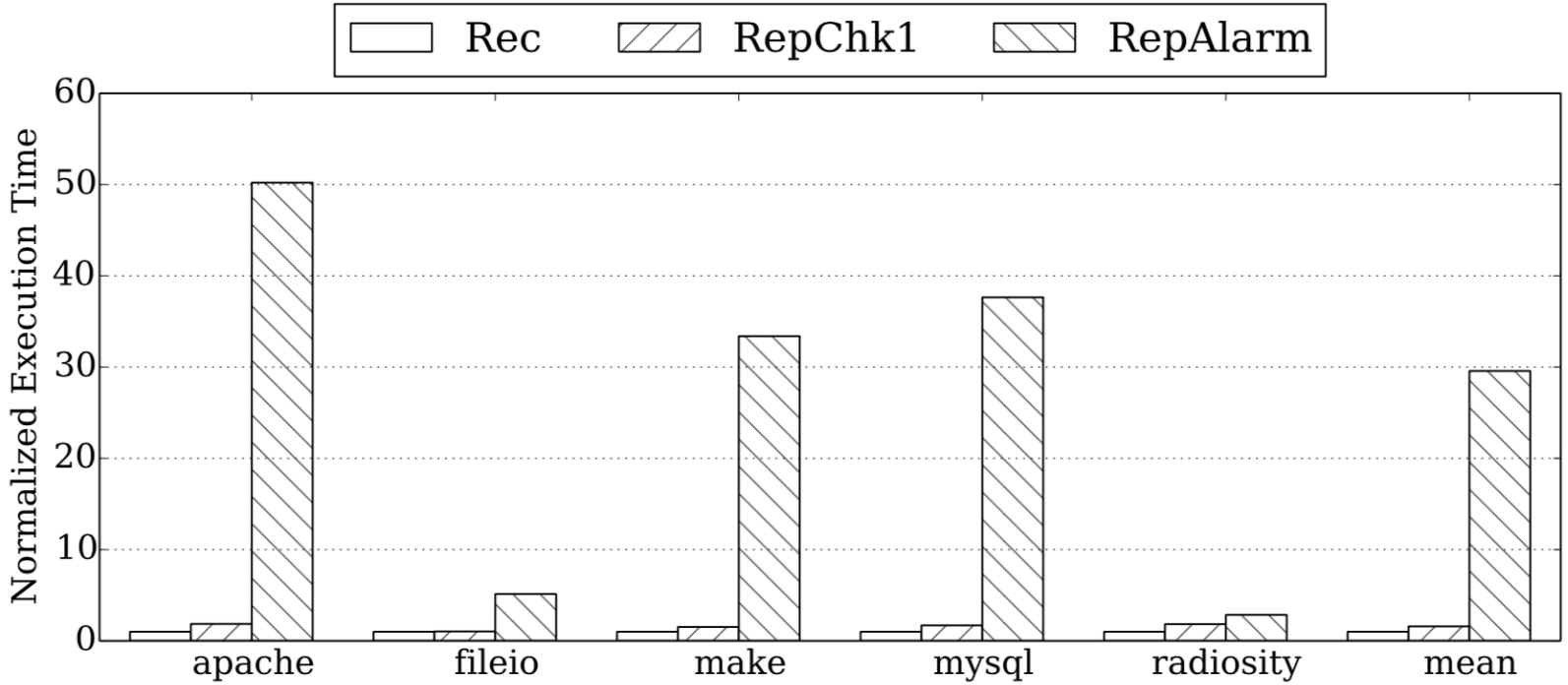


Figure 9: Execution time of alarm replay checking for ROP attacks in the kernel.

结论

- 本文提出了RnR-Safe框架，用于补充硬件安全功能，允许这些功能不精确并遭受误报。 此属性通常使功能更少侵入或复杂。
- RnR-Safe使用两个即时重播器：一个检查点和一个报警器。
- 通过应用RnR-Safe来阻止ROP攻击及其评估，我们发现RnR-Safe是一种非常有效的协同设计。 检查点重放器具有与记录器相当的执行速度，并且可以一直重放。 此外，警报重放器只需要处理很少的误报。