你应该知道的关于IntelSGX性能的一切

DINHNGOC大学，IRIT，法国
法国IRIT的宝布伊
斯特拉·比特贝，IRIT，法国
alainTCHANA，I3S，法国
瑞士纽沙特尔大学的瓦莱里奥·夏基亚沃尼
pascalFELBER，瑞士纽卡特尔大学
丹尼尔·哈吉蒙特，IRIT，法国

IntelSGX已经吸引了学术界的广泛关注，并且已经为商业应用提供了动力。云提供商也已经开始在他们的云产品中实施SGX。到目前为止，对英特尔SGX的研究工作主要集中在其安全性和可编程性上。然而，还没有详细的研究
由虚拟化系统中的SGX造成的性能下降。考虑到虚拟化事实上是云基础设施的构建部分，但往往会对性能产生影响，这些设置尤其重要。本文首次对IntelSGX与裸金属系统相比进行了详细的性能分析。基于我们的发现，我们确定了几种优化策略，以提高IntelSGX在此类系统上的性能。

中国化学会概念：安全和隐私→可信计算；虚拟化和安全；其他关键词和短语：IntelSGX、性

能、虚拟化、受信任的硬件

ACM参考格式：
你丁，鲍布、比特贝、坎纳、希亚沃尼、费尔伯和哈吉蒙特。 2019.关于虚拟化系统上您应该知道的关于IntelSGX性能的一切。项目。ACM的平均值。肛门检查。投入工作。系统节。第3、1、第05条（2019年3月），共21页。https://doi.org/10.1145/3311076

05

1、产品介绍

随着业界对云计算技术和平台的兴趣日益浓厚，人们对这些平台上存储的私有数据的保密性和完整性产生了许多担忧。云平台的可信度取决于许多元素，其中即使是一个弱点也会导致其安全模型的崩溃。首先，客户需要信任云提供商使用的虚拟机管理程序，它通常不是开源的，也不是可审计的，并且可能存在漏洞或后门，这会危及其隔离保证。此外，几乎所有系统的硬件、软件和配置组件都存在漏洞和配置错误，例如，来宾操作系统、网络配置、访问控制等。此外，通常情况下，人类因素是最薄弱的环节：渗透测试人员考虑社会工程

作者地址：TuDinhngoc，IRIT，法国，dinhngtu@gmail.com；保，IRIT，法国，bao.bui@enseeiht.fr；Stella比切贝，IRIT，法国，bstellaceleste@gmail.com；阿兰·坎纳，I3S，Alain.Tchana@enseeiht.fr；瓦莱里奥·夏沃尼，瑞士，valerio.schiavoni@unine.ch；帕斯卡·费尔伯，瑞士，瑞士，pascal.felber@；丹尼尔·哈吉蒙特，IRIT，法国，daniel.hagimont@enseeiht.fr.

2019年9月3日-art05$15.00
https://doi.org/10.1145/3311076

05：2丁等人。

作为渗透安全环境的最有效方法之一，内部威胁可能滥用其特权进行数据盗窃、欺诈或破坏[37,46]。

可信计算系统[29,41]试图通过保证底层平台的完整性和/或保密来解决这些问题，以保护用户数据，并防止平台所有者或恶意攻击者未经授权的访问。为了确保这些保证，安全环境通常还提供了一些共同的功能。首先，平台保密可以防止攻击者学习系统的运行状态（CPU、内存、存储、网络等）。)。其次，完整性保证可以进一步保护环境，从而通过代码修改、注入或恶意软件攻击来阻止对平台预期行为的恶意改变。在安全环境中运行的软件可以通过认证机制向外部参与者证明其完整性及其环境的完整性，并使用平台提供的数据保护设施在静止和运输中保护其数据。

一些值得信赖的计算解决方案侧重于保护硬件平台。例如，可信平台模块(TPM)[15、19]主要通过测量关键系统软件（固件、引导加载程序、内核等）来用于平台完整性。通过使用平台配置寄存器(PCR)，并将特定的加密秘密绑定到一组PCR，例如，用于磁盘加密。然而，仅TPM并不是可靠计算的完整解决方案，因为用户仍然需要信任许多其他组件，如操作系统、存储设备或通信总线[32、42]。

为了解决这些限制，Intel引入了软件保护扩展(SGX)[1]，它提供了一个可信的计算系统，可以在处理器包内强制执行安全边界。使用了SGX，用户不再需要信任系统软件或任何其他硬件组件。此外，SGX是一个独立的解决方案，它不依赖于外部设备或配置，也不干扰虚拟化。

除了IntelSGX之外，其他供应商还创建了对虚拟化友好的可信计算解决方案。AMD提出了安全加密虚拟化(SEV)[30]，这是一种基于内存加密的技术，承诺保护虚拟机(VM)免受虚拟机监控程序攻击和硬件窥探。微软引入了屏蔽虚拟机[36]的概念，这是运行在特殊虚拟机的严格控制环境上，称为保护结构，不受主机管理员的窥探和更改。

本文专门关注IntelSGX，它引起了学术界的广泛关注[2,44]，并已经支持商业应用[27,40]。云提供商已经开始在其云产品中实现SGX，从微软的Azure机密计算或IBM云[22]开始，而新的云编程框架已经支持SGX[17]。

到目前为止，对IntelSGX的研究工作主要集中在其安全性和可编程性[8,34]上，特别是为了简化了其API的使用，并支持在受SGX保护的飞地内执行遗留代码。HotCalls[45]遵循不同的路径，重点关注SGX性能对本地系统的影响。然而，还没有任何工作详细研究过在虚拟化系统中，SGX造成的性能退化。考虑到虚拟化事实上是云基础架构的构建部分，并且经常会对性能产生影响，这些设置尤其重要[33]。

在本文中，我们首次详细分析了IntelSGX在虚拟化系统中与裸金属系统相比的性能。基于我们的发现，我们确定了几种优化策略，以提高IntelSGX在此类系统上的性能。

综上所述，我们的论文给出了以下发现：

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

你应该知道的关于IntelSGX性能的一切05：3

（1）虚拟机不需要拦截SGX指令才能启用虚拟机的SGX；当前唯一的异常是必须拦截才能虚拟化SGX启动控制[10]。
（2）在运行内存密集的基准测试时，虚拟机上的SGX开销主要来自于使用嵌套分页时的地址转换成本。使用阴影分页，虚拟化SGX在裸机上与SGX的性能几乎相同（飞地呼叫的平均开销低于1.3%；加密率为1%-3%）。相比之下，嵌套寻呼页面在飞地呼叫时比裸机页面慢了7.4%，在加密时慢了10%。
（3）相反，当运行涉及许多上下文切换的基准测试时(例如，HTTP基准测试)时，阴影分页的性能低于嵌套分页(13.8%相比。嵌套分页时的低请求速率为4.1%）。
（4）我们提出了一种动态检测给定工作负载特性的方法，以确定它是否适合嵌套分页或阴影分页，因此允许自动选择适当的内存虚拟化技术。
（5）SGX在应用程序和飞地之间切换时会施加严重的性能惩罚，根据所使用的呼叫机制，每次呼叫的周期从10000到18000次不等。如[3,45]所述，此惩罚会影响使用SGX的服务器应用程序。
（6）交换飞地页面缓存(EPC)页面非常昂贵，每次交换操作要花费多达数十万个周期。当SGX在初始化时测量飞地的内容，包括任何静态声明的数组时，如果飞地的内存大小大于可用的EPC大小，这种初始化将触发飞地交换。此外，在执行EPC驱逐操作时，虚拟化会导致28%到65%的开销，这取决于在飞地内部运行的线程的数量而增加。

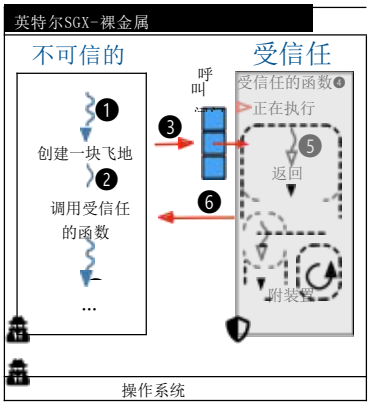
本文的组织结构如下：第2节简要介绍了IntelSGX是如何在裸金属和虚拟化环境上工作的。我们在第3节中概述了我们的评估方法，然后是从第4节和第5节中的实验中获得的评估结果和经验教训。第6节提出了改进虚拟化SGX环境中应用程序性能的优化。我们在第7节中介绍了对这些优化的评估。最后，我们将在第8节中讨论一些相关的工作。第9节介绍了我们工作的结论。

2个背景信息
在本节中，我们介绍了一个SGX飞地的结构和操作，并提供了在虚拟化平台上的SGX飞地操作的详细信息。

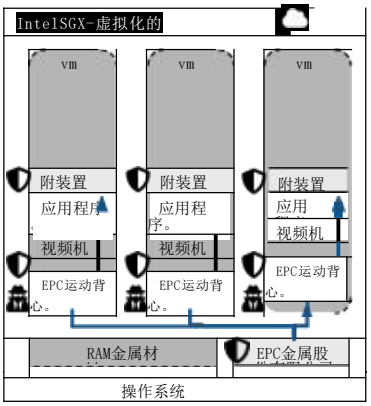
2.1裸金属系统上的IntelSGX
Intel软件保护扩展(SGX)[13]为应用程序开发人员提供了保护其代码和数据的手段。自Skylake微架构[39]以来，它就可以在英特尔的CPU中使用。使用IntelSGX，应用程序免受窥探和修改，包括来自其他进程、底层操作系统、管理程序甚至BIOS的攻击。为了受益SGX保护，应用程序创建称为飞地的安全隔离区域。飞地包含它们自己的内存和数据，这些数据由CPU加密和验证，并且可以通过飞地二进制文件中预定义的入口点与外部程序交互。应用程序开发人员可以依赖Intel提供的软件开发人员工具包(SDK)来轻松地跨不同操作系统创建启用SGX的应用程序[28]。

SGX的显著特性之一是一个小型的可信计算基础(TCB)：SGX飞地的TCB只包括CPU包本身，以及在飞地内部运行的任何应用程序代码。此外，还很容易识别和隔离项目中应该在飞地内运行的部分。

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。



图中示。 1.IntelSGX工作流。



图中示。 2.在虚拟化环境上的IntelSGX。

为了保护飞地免受未经授权的访问和修改，SGX使用死内存加密引擎(Mee)[20]和CPU内逻辑。因此，攻击，如读取和
对飞地内存区域的写入或监听CPU外部的内存总线会被SGX阻塞。

英特尔SGX包括本地和远程认证以及秘密密封功能。认证让飞地向其他项目证明他们的身份：当地认证在同一台机器上与飞地一起工作，而远程认证与外部服务一起工作。秘密密封使SGX飞地安全地存储受保护的数据，以在必要时重新加载。

为了使SGX飞地正常运行，除了为SGX提供必要的CPU和BIOS支持外，用户必须安装专用的IntelSGX驱动程序，管理飞地的创建和内存管理；以及IntelSGX平台软件，它为飞地启动管理、远程认证和其他平台服务提供建筑飞地。

在启动时，BIOS会验证是否启用了SGX。然后，它为CPU保留一个物理内存区域，指定为飞地页面缓存(EPC)。在SGX的当前迭代中，EPC的最大大小被限制在128MB以内，其中只有93MB可被应用程序使用；其余的被分配给元数据，不能被飞地使用。

图1说明了SGX飞地的典型操作，从其初始化开始到
执行所需的受信任的函数，并将控制权返回到不受信任的代码中。对于
为了清楚起见，我们省略了飞地的测量和认证等细节，以及详细的飞地的初始化过程。

一块飞地的初始化包括四个主要步骤：

（1）通过电子指令创建SGX飞地控制结构(SECS)。
（2）使用EADD指令将代码和数据页复制到飞地中。
（3）使用esepende指令创建飞地内存内容的加密测量。
（4）验证应用程序开发人员的签名，并使用EINIT指令激活飞地。

初始化完成后，应用程序可以使用EENTER指令进入飞地。一旦进入飞地，有两种方法可以离开飞地：要么使用EEXIT指令从飞地返回，允许应用程序再次进入飞地

EENTER；或者通过异步出口(AEX)机制退出飞地，该指令可以用于恢复飞地的执行。

SGX飞地必须托管在加密的、保留的EPC内存上。SGX通过将页面从EPC交换到正常内存，允许飞地使用超出可用容量的EPC内存。在从EPC中写出来之前，该页面由CPU加密和密封，并在被称为版本数组的特殊EPC页面中跟踪将来的恢复。它相应的TLB项也作为驱逐过程的一部分被刷新。当需要交换页面时，由CPU验证，解密并复制回EPC。这种交换支持通过为飞地提供更多的内存，缓解了SGX飞地内部的内存管理问题。

2.2虚拟化系统中的IntelSGX

SGX的主要用例之一是云计算，客户可以启动飞地并执行安全计算，而不向不受信任的平台提供商透露敏感数据。

SGX被设计为一个单独的执行模式，可以使用特殊指令从非特权模式（环>0）输入。虚拟机监控程序可以通过解除与SGX操作相关的CPUID标志来向虚拟机公开SGX支持。对于以SGX支持启动的每个虚拟机，管理程序分配EPC的一部分用作虚拟EPC；此EPC区域可以专门分配给一个虚拟机(在SGXv1上)，或超额订阅多个虚拟机(SGXv2)，并且与裸机机处理其EPC区域的处理方式相同。然后为每个虚拟机提供它自己的SGX运行时软件（驱动程序和平台软件），以提供飞地功能。飞地启动时，飞地内存从虚拟EPC分配，飞地正常工作。图2显示了启用SGX的虚拟机的可能部署，其中硬件EPC被分区并分发到几个虚拟机。

Intel提供了两个基于KVM和Xen管理程序的具有SGX支持的虚拟机管理程序实现[26]。在撰写这篇文章的时候，KVM对SGX的支持是更为最新的；因此，我们只在KVM上进行实验评估。

下一节介绍了我们在虚拟化系统中系统地评估SGX所采用的方法。

3、评价方法

为了方便SGX应用程序编程，Intel在Windows[24]和Linux[25]上提供IntelSGXSDK。SDK允许开发人员使用相同的开发工具链编写SGX应用程序的两个部分，即不受信任的应用程序和受信任的飞地。在应用程序中，开发人员可以使用SDK提供的两种不同的调用机制：ecall，这是不受信任应用程序对飞地的调用；ocall是飞地对不受信任应用程序的调用。

瀑布和瀑布都支持在应用程序和飞地之间传输缓冲区，其中SDK将自动处理必要的缓冲区传输逻辑。飞地调用是通过飞地定义语言(EDL)文件来定义的，该文件列出了飞地所允许的所有可能的飞地调用和缓冲区转移。SGXSDK提供了一个名为edger8r（发音作“编辑器”）的工具，用于自动生成代码以支持应用程序端和飞地端的这些调用。

为了研究IntelSGX的性能，我们的目标是测量裸金属虚拟机和虚拟机上的主要SDK函数。这包括：

普通经济和经济的表现。

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

05：6丁等人。

使用SDK中包含的缓冲区封送功能将缓冲区传输到/出/进出飞地的性能。
从飞地内部读写加密内存的性能。
从飞地内外对未加密内存的读写性能。
从EPC中驱逐页面的性能。
初始化和摧毁飞地的性能。

我们是第一个衡量在IntelSGX驱动程序中交换EPC页面的影响，以及飞地初始化/销毁时间。除了这些具体的评估之外，我们还实验了两个涉及到飞地中常用的函数的基准测试：加密基准测试和HTTP服务器基准测试。

对于调用延迟、页面驱逐和初始化/破坏实验，我们使用RDTSCP指令用CPU周期来衡量性能。为了确保一致性，我们多次运行并记录每个实验，在收集实验结果前进行一段热身。例如，普通八进制实验分10次运行，每一次包括10000次热身迭代和10万次实验迭代。在我们的结果中记录了运行的确切数量。我们还通过运行实验来考虑CPU缓存对运行时的影响，在这些实验中，缓存在每次执行之前都会被清除。这些实验完成后，通过计算平均值、标准差、最小值、最大值和中值来收集和分析其性能结果，并在相关结果进行讨论。

为了使我们的工作具有可重复性，所有的微和宏观基准都在[14]（匿名）提供。这些程序也可以用于执行SGX的其他评估，以避免需要重写这些棘手的基准测试。我们将在下文中讨论评估结果。

4、评价结果

在这一节中，我们首先介绍我们的实验设置，然后深入讨论我们在比较研究中获得的各种结果。

4.1实验性设置

我们用两台机器进行了实验。第一个是戴尔纬度5280，配备了英特尔酷睿i5-7200U，有2核和16GB内存。第二款是戴尔PowerEdgeR330，配备XeonE3-1270v6，有4核和64GB内存。除非另有说明，否则我们在第一台机器上进行了实验。所有的实验都在Ubuntu16.04上运行，使用的是kvm-sgx内核版本sgx-v4.14.28-r1和qemu-sgx版本sgx-v2.10.1-r1，保留的EPC大小为128MB。对于虚拟机上的实验，除非另有说明，我们使用2个虚拟CPU、6GB的RAM和32MB的EPC容量。在所有的实验中，我们禁用了自动CPU频率缩放、涡轮增压和超线程，以避免不一致的性能行为。基准测试软件是使用GNUGCC5.4.0构建的，并且使用-OO标志禁用了编译器优化。

对于ecall和ocall基准测试，我们还使用kvm-sgx内核4.14.63进行了测试，其中包含了对L1TF漏洞的缓解措施[12]。后者是一个硬件漏洞，可以允许恶意进程访问其计划的IntelCPU核心的1级数据缓存(L1D)中可用的数据。提出了各种禁用对称多线程(SMT)、禁用EPT和熔断L1D等问题。我们怀疑L1TF的缓解措施可能会影响在SGX上的VM性能。请注意，在测试期间使用了L1TF缓解措施的默认（冲洗）选项。

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

4.2总体性能

我们首先要测量生态系统在各种条件下的性能。附件是SDK函数，允许应用程序通过一个简单的函数调用接口调用到飞地代码。它们还允许飞地开发人员指定在应用程序和飞地之间编组的缓冲区。有四种主要的缓冲区传递模式：

（1）从应用程序转移到飞地，由在EDL关键字指定。在此模式下，会将缓冲区从应用程序复制到飞地内存中。

（2）从飞地转移到应用程序中，由outEDL关键字指定。在此模式下，输出缓冲区被分配在飞地内，并在函数返回后复制回应用程序内存。

（3）转移在飞地，然后返回，使用进出的关键字。缓冲区首先被复制到飞地中，由生态函数处理，然后在函数返回时被复制回。这样可以防止在缓冲区内的数据处理过程中向内存窥探攻击者泄露秘密（例如，加密密钥）。

（4）未经验证的指针传递。在这种情况下，飞地接收到任意指针，并且不使用SDK提供的缓冲区封送功能。飞地在读取或写入它之前，必须手动验证该指针的有效性。此指针可以指向未加密内存中的地址，允许零复制数据从飞地/传输到飞地。

在我们的实验中，我们考虑了以下不同的场景：

执行空飞地功能（热/冷CPU缓存）；

执行飞地功能，传递2KB缓冲区（热/冷缓存）；

执行飞地功能，将2KB缓冲区“输出”（热/冷缓存）；以及

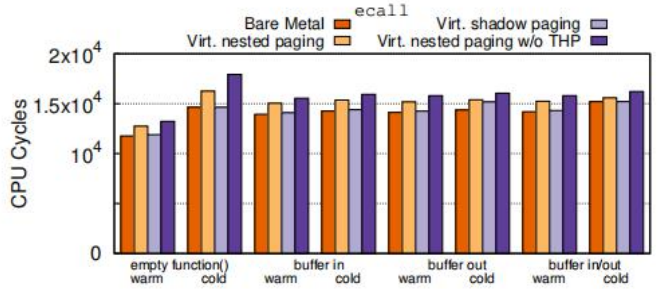
执行飞地功能，传递2KB缓冲区“进出”（热/冷缓存）。

为了评估来自冷CPU缓存的eclal函数调用，我们在每次基准测试运行之前通过读写大小大于CPU缓存的缓冲区来清除缓存，以确保防止编译器优化消除读取和写。清除缓存后，插入内存围栏（使用MFENCE指令）以确保缓存完全

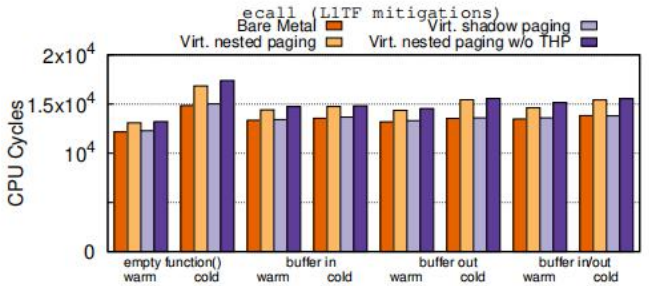
已刷新到RAM。

SGX必须在每个地址转换期间进行访问检查，就在提交转换之前，TLB[13]。这可能是导致性能开销的原因之一。因此，我们希望使用不同的内存虚拟化技术来评估SGX。大多数虚拟机管理程序支持一种基于软件的技术，称为阴影分页，并支持一种基于硬件的技术，称为嵌套的技术分页（在IntelCPU中称为EPT）[38]。虚拟化环境中的地址转换是关于将来宾虚拟地址(gVA)转换为主机物理地址(hPA)。使用阴影分页，对于VM中的每个进程，管理程序维护一个直接将GVA映射到HPA的阴影页表。然后，阴影页表（而不是来宾页表）被本地页面表行走机制用来进行地址转换。每次客户机VM更新其页面表时，管理程序都必须进行干预，以便相应地更新阴影页面表。这可能会导致上下文交换机的重大开销，并大大降低VM性能。

在每个现代x86CPU核心上，都有指向来宾页面表的页面表指针和指向嵌套的页面表的页面表指针。在嵌套分页下，使用两个页面表：首先来宾页面表将gVA映射到来宾物理地址（gPA)，然后嵌套页面表将gPA映射到hPA（这通常称为二维翻译）。因此，嵌套分页比阴影分页更昂贵，因为它需要更多的内存引用。例如，对于4级页面表，阴影分页只需要4个引用，而嵌套分页需要对2个页表的24个引用。在触发许多TLB丢失的应用程序中（例如，内存不足的应用程序）中，这可能会导致性能下降[5]。



图中示。 3.一个人花费的时间（较值更好）

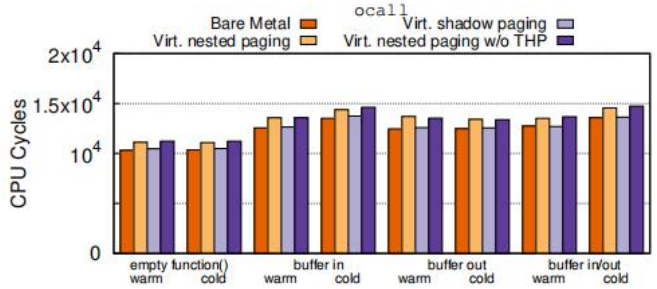


图中示。 4.一个人花费的时间（较值更好）

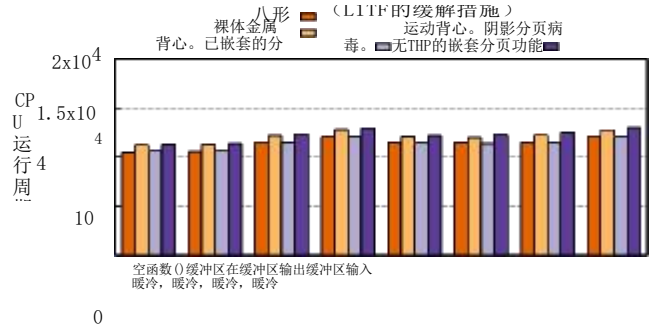
Linux内核支持透明的大页面(THP)，它允许在应用程序请求大分配时使用大的2MB的内存页面，而不必使用hugetlbfs接口。内核支持三种THP模式：始终、提出建议和从不。始终选项使THP可用于所有分配；m建议将THP限制为使用MADV_HUGEPAGE标志标记的分配，并且永远不会完全禁用该功能。默认情况下，KVM来宾使用THP消除主机上的一级别地址转换，从而减少对TLB丢失的2D地址转换造成的性能影响。

为了解释这些特性，我们在四个条件下评估了SGX飞地呼叫：在一个裸-金属系统；在启用嵌套分页和THP的VM中；禁用嵌套分页并使用阴影分页的VM中；在启用嵌套分页但禁用THP的VM中（由Virt表示。嵌套分页，无THP）。我们还使用热和冷缓存运行这些函数，在每次运行之前要传输的缓冲区用clflush指令刷新，除了空函数基准测试，我们读写一个大缓冲区来完全刷新CPU缓存。每个基准都被评估了100万次，周期计数的中位数如下。

图3显示了一个人在各种条件下所花费的时间。我们可以看到，具有阴影分页的虚拟机的性能更接近裸金属系统，比所有基准测试只有1.3%的平均开销，而启用嵌套分页的虚拟机的执行速度明显较慢，在相同场景的情况下，平均开销为7.4%。此外，我们可以看到在关闭THP时有一个小的性能损失：禁用THP的相同实验平均比裸金属设备慢12.5%，比启用THP的虚拟机慢4.7%。这种现象可以用大页纸来解释
在地址转换过程中添加一级页面表。从这些项目中获得的。ACM的平均值。肛门检查。投入工作



图中示。 5.一个八占用的时间（低更好）



图中示。 6.一个八占用的时间（低更好）

结果，我们可以得出结论，虚拟机上的所有性能主要受到地址转换开销的影响。

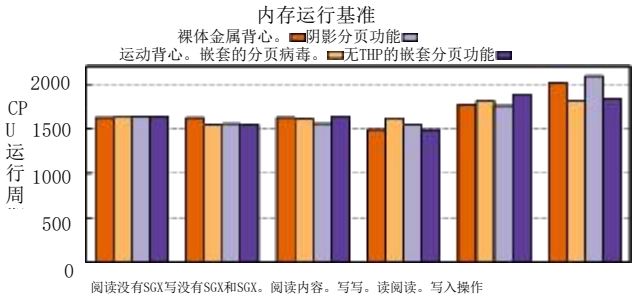
对于具有冷CPU缓存的空函数基准，我们观察到裸金属和使用阴影或嵌套分页虚拟化的开销为23%-28%，但是禁用THP的虚拟机在性能上落后于，开销明显高于35%。此外，当比较涉及缓冲区传输与热和冷缓存的函数调用时，我们注意到只有2.1%-2.4%（在传输缓冲区时）和1.4%-1.7%（在传输缓冲区“输出”时），除了在传输缓冲区时阴影分页，减速为6.5%。但是，在裸机和阴影分页上“进出缓冲区”时，这种放缓会放大（启用/禁用THP的嵌套分页分别为7.0%和6.2%，分别为2.2%和2.6%）。

我们还重复了在linux内核4.14.63上的测量，并修复了L1TF漏洞的补丁。内核4.14.63中的L1TF缓解措施使用的默认选项是保持启用SMT（但在测试期间手动禁用SMT）并启用条件L1D刷新。图4显示了使用此新内核进行测试的结果。虽然在这种情况下，在性能上没有太大的变化，但启用了更积极的L1TF缓解措施(例如。无条件的L1D刷新)可能会导致更多的性能开销。

4.3整体性能

与瀑布类似，ocalls允许与SGX接口，但方向相反：飞地可以通过预定义的接口调用应用程序函数。这使得飞地可以轻松地执行I/O调用（例如，打印到标准输出），而不必依赖于调用它的应用程序。瀑布支持与瀑布相同的四种缓冲区通过模式，除了“in”现在意味着从

05：10图等人。



图中　读写2KB缓冲区所需的时间（下限更好）

示。 7.

这块飞地进入了应用程序中，反之亦然。我们尝试了相同的四种情况：到空函数和应用程序函数，传递一个2KB缓冲区“入”、“出”、“出”和“输入&出”。这四种情况下的实验都是用热缓存和冷缓存进行的。

图5显示了这些结果。我们可以看到，瀑布通常比瀑布快；使用温暖的缓存，瀑布比使用空功能的瀑布快12.4%，在传输2KB缓冲区时快10.6%。平均而言，性能故障与所有基准相同：裸金属，阴影分页（慢0.9%），启用THP的嵌套分页（慢7.5%），最后禁用THP的嵌套分页（慢8.2%）。

在比较热缓存和冷缓存的性能时，我们观察到空函数和缓冲“输出”基准的性能差异很小。但是，对于缓冲区“in”和“in&out”基准测试，根据缓存较冷时的虚拟化方法，我们发现性能开销为6.0%-8.6%。

图6显示了使用L1TF缓解措施的结果。我们可以看到在性能方面有一些改进（可能是由于新内核），但总体上行为是相似的。

4.4内存访问

为了衡量SGX内存加密的开销，我们比较了来自飞地的内存读写与来自应用程序外部的正常读写的性能。实验包括三种情况：

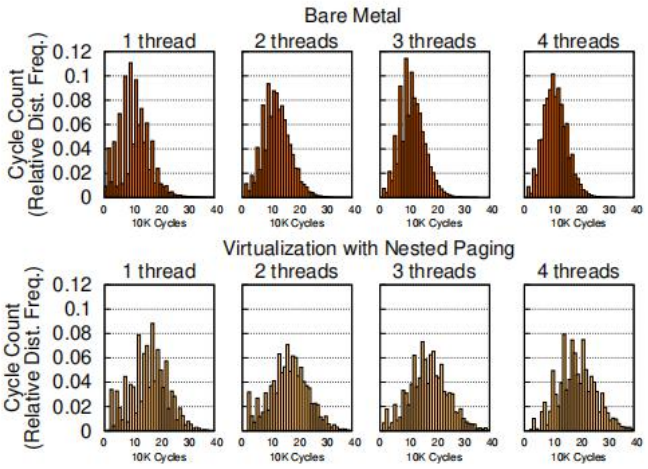
（一）从应用程序中读写未加密的内存（无需飞地参与）；

（二）从该飞地读写加密内存；

（3）从这块飞地读取和写入未加密的内存。

由于从飞地内阅读和写作的记忆需要一个飞地的召唤，我们
减去显示运行时之前实验获得的中值空呼叫开销（这是因为在基准测试中使用的SGX版本中，我们无法从飞地内部读取TSC）。图7给出了我们的研究结果。请注意，在某些情况下(例如。没有SGX)，裸金属中的性能似乎低于虚拟化环境中的性能。然而，性能上的差异很小。我们可以说，这些结果是在误差范围内得出的。虽然我们看到从飞地读写到加密内存的内存类似于没有SGX的性能（不到2%)，但从飞地到未加密内存的读写速度要稍微慢一些(约17%）。类似地，比较虚拟化的方法
平均而言，我们观察到所有Proc的性能差异都不到1%。ACM的平均值。肛门检查。投入工作。

系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。



图中示。 8.一次驱逐行动所需时间的频率分布（四舍五入到10,000个周期）

虚拟化方法；然而，单个基准测试显示出了更多的可变性
基准运行时间在11%到8%之间。

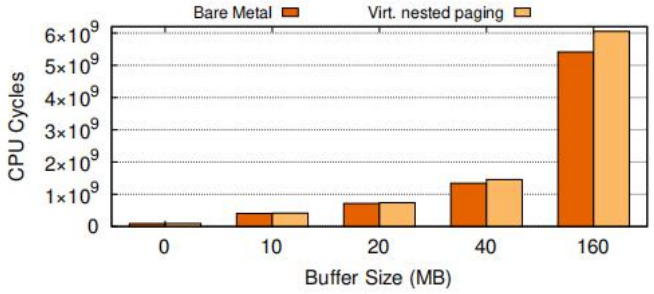
4.5相互交换

LinuxSGX驱动程序允许将EPC过度提交到飞地；也就是说，应用程序消耗的EPC可能比机器上实际使用的EPC还要多。EPU使用SGX的CPU上的EPC页面驱逐功能支持EPC交换；因此，交换会在页面写出和读回过程中带来开销。为了将飞地页面驱逐出EPC，首先使用EBLOCK指令将其标记为“阻止”；然后，无法创建指向该页面的新TLB条目。在EBLOCK之后，执行ETRACK以跟踪有关该页面的TLB条目。然后，操作系统必须使在飞地内执行的所有处理器使用内部处理器退出
中断以刷新这些TLB条目。最后，一旦所有的处理器都离开了这块飞地，操作系统就可以使用EWB来完成驱逐过程，并将密封的内存页面写入RAM。

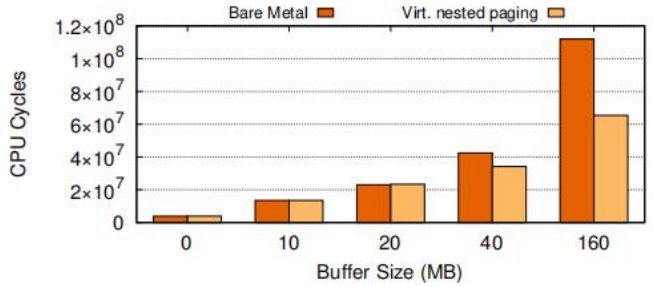
需要注意的是，页面驱逐过程需要中断在拥有要交换的页面的飞地内执行的所有CPU，以便在最终驱逐该页面之前刷新与该页面相关的TLB条目。这一过程可能需要很长时间才能完成，并影响到飞地的行动。

我们测试了LinuxSGX驱动程序的代码来测量从EPC中驱逐出一页所花费的时间，从EWB完成后调用EBLOCK之前计数。我们试验了我们的Xeon机器。对于在这台机器上运行的虚拟机，我们分配给VM8个vCPU和8GB的RAM，同时将飞地线程的数量从1个变化到4个。每个飞地线程不断地访问飞地内的一个大内存缓冲区中的随机地址，因此迫使内存页面不断被从EPC驱逐出EPC。在这个实验中，我们分配了虚拟机可用的最大EPC容量，并重复了每次执行的1亿次访问。

图8显示了裸金属机和虚拟机上的每个驱逐操作的执行时间的分布，这取决于飞地线程的数量。我们看到，在裸金属上，无论使用1、2、3或4条飞地线，驱逐行动几乎需要同一时间。然而，虽然我们在直方图中观察到一个峰值有4个飞地线，但情况并非如此
只有一个飞地线程：周期计数往往集中在几个值范围，而不是过程。ACM的平均值。肛门检查。



图中示。 9.飞地启动时间中位数（较低则更好）



图中示。 10.飞地关闭时间中位数（较低则更好）

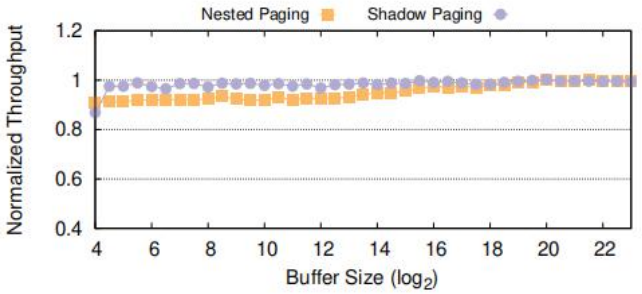
而不是在一个峰值中积累。这种行为可能源于对硬件计时器的处理，即CPU在收到周期性中断时就必须停止飞地线程。

然而，当在虚拟机上运行相同的实验时，我们观察到随着飞地线程数量的增加，驱逐操作所采取的周期数量会增加。特别是，我们看到驱逐时间从1个飞地线程到4个飞地线程增加了12%。因此，与裸金属机相比，我们在虚拟机上的驱逐实验中看到的虚拟化开销为28%到65%，这取决于飞地线程的数量。这可能是由于处理器间中断(IPI)，虚拟机比裸机要慢。此外，我们还看到了a
与我们在裸机实验中看到的类似，在虚拟机上运行4个飞地线程时集中频率的行为达到峰值。

4.6飞地初始化和销毁

在飞地准备使用之前，CPU会测量其内存内容，以生成加密哈希。我们希望测量飞地启动和关闭时间造成的开销。为此，我们在飞地代码中添加了一个大型的静态数组，并测量了飞地的启动时间和s。阵组大小（从0MB到160MB不同）。每个实验
运行了1万次，飞地启动和关闭的总时间是使用RDTSCP指令测量。

图9和10分别显示了裸金属启动时间和停机时间
和虚拟化的飞地。请注意，裸金属机采用了128MB的EPC大小，其中约有93MB可供应用程序使用，而VM使用了32MB的虚拟EPC。同时
我们看到了缓冲区大小为20MB或更低的Proc的飞地的启动和关闭时间。ACM的平均值。肛门检查。投入工作。系统节



图中示。 11.SGX-SSL加密吞吐量标准化为裸金属飞地（较高更好）

大约相等，从40MB的缓冲区大小和以上，我们看到裸金属机器需要更少的时间来初始化飞地，但更多的时间来关闭。这可以用裸金属主机的较大的EPC大小来解释：在启动期间，在其加密测量期间必须触摸分配给飞地的所有页面。更大的EPC意味着这块飞地需要的交换量更少
触摸所有页面，从而加快启动；然而，在飞地关闭时，更大的EPC也意味着必须释放飞地工作集中的更多页面，从而导致关闭较慢的速度。

在对SGX操作的单独评估之后，我们进一步使用涉及密码学的宏观基准测试来评估了SGX。

4.7SGX-SSL基准测试

我们在裸机和虚拟机上运行了[21]中提供的SGX-SSL库的基准测试。该基准测试测量了使用AES-128-GCM对从16字节到16MB不等的不同大小的内存缓冲区进行加密的性能。以平均吞吐量（MB/s）测量加密性能。我们进行了15次实验，将结果与裸金属飞地上的SGX-SSL进行了标准化，并将性能结果记录在图11中。

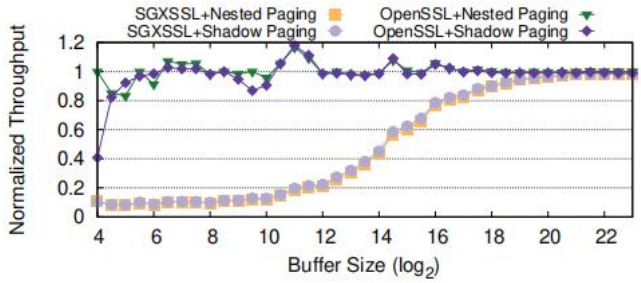
一般来说，虚拟化飞地比裸金属飞地的性能损失很小（不到10%）。在小缓冲区大小（低于256KB）时，启用嵌套分页的虚拟化飞地比启用阴影分页的飞地性能更差（最慢7.6%）。随着缓冲区大小的增大，此放缓将消失，其中嵌套分页和阴影分页的性能接近裸机性能（在1%内）。

为了完成我们的实验，我们还记录了没有SGX的加密性能，看看SGX是否引入了任何差异。图12显示了标准OpenSSL库的结果，它是SGX-SSL库的基础库。我们可以看到，在这种情况下，整体嵌套分页的性能略好于阴影分页。换句话说，在SGX-SSL基准测试中，SGX为嵌套寻页造成了更多的性能开销。

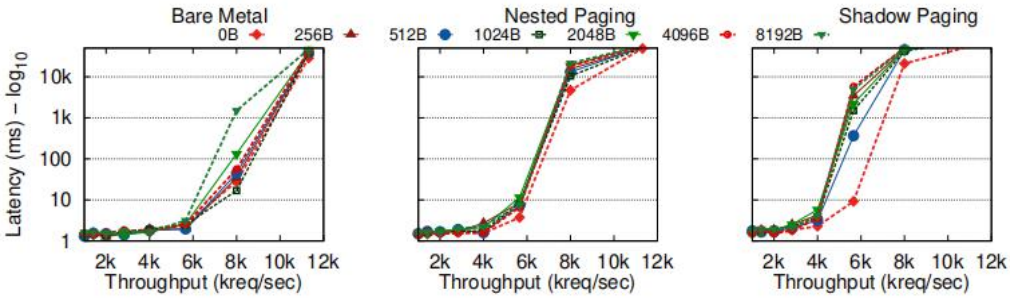
4.8HTTP基准测试

TaLoS库[3]提供了一个基于OpenSSL的API，它可以处理SGX内部的TLS连接
飞地，以保护连接秘密（私钥、会话秘密等）。从窥探中。在这个实验中，我们测量了在使用TaLoS进行TLS连接终止时，Nginxweb服务器的HTTP请求吞吐量和延迟。web服务器被配置为使用TLS进行加密，并服务于从0到8KB的各种有效负载大小。我们使用了一台通过1Gbit以太网连接连接的单独机器和wrk2实用程序[43]来进行基准测试。实验结果
在裸金属机器上执行，在有嵌套分页的VM和在有阴影分页的VM上执行。项目。ACM的平均值。肛门检查。投入工作。

05：14丁等人。



图中示。 12.裸机上的SSL加密吞吐量标准化为OpenSSL（较高更好）



图中示。 13.用来增加每秒和不同响应大小的请求的Nginx吞吐量/延迟

虚拟机上的基准测试是以虚拟网卡设置为桥接模式运行虚拟机的。每个基准测试连续运行5分钟，吞吐量和中值延迟由wrk2记录。 图13显示了裸金属系统以及具有嵌套和阴影分页的虚拟机的性能。基准测试包括以越来越高的恒定速率（x轴）发出请求，直到响应延迟达到峰值（y轴、日志尺度）。与我们的微基准测试和SSL基准测试不同，HTTP分页相比，HTTP基准测试比嵌套分页的性能更差：在每秒1000个请求时，阴影分页的平均延迟影响为13.8%，而相比之下，嵌套分页的影响只有4.1%。差异从每秒4000个请求的请求速率开始扩大，其中阴影分页将请求延迟翻倍，而嵌套分页只显示8.9%的影响。这可以解释为基准测试是i/o重的，因此涉及到更多的上下文开关，这在使用阴影分页时有很高的惩罚。尽管如此，在我们的评估中，嵌套分页仍然显示吞吐量从4.1%减少到56×以上，这可能是由KVM的网络接口虚拟化造成的。

我们将在下一节中讨论从这些结果中吸取的主要经验教训。

我们吸取了5个经验教训

在评估了IntelSGX的性能和源代码之后，我们吸取了以下关于SGX虚拟化的教训：

(i)虚拟机不需要拦截SGX指令来启用虚拟机的SGX；当前唯一的例外是拦截创建以虚拟化SGX启动控制[10]。因此，与裸金属平台上的SGX相比，虚拟机上的SGX具有可接受的开销。

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

(ii)运行内存密集的基准测试时虚拟机上的SGX开销主要包括使用嵌套分页时的地址转换开销。通过阴影分页，虚拟化的SGX与裸金属机上的SGX具有几乎相同的性能。

(iii)相反，当运行涉及许多上下文切换的基准测试（例如，HTTP基准测试）时，阴影分页的性能比嵌套分页更差。

其次，以下课程一般适用于IntelSGX（即在裸金属平台和虚拟化平台上）：

(iv)SGX对在应用程序和飞地之间的切换施加了严重的性能处罚。此惩罚会影响使用SGX的服务器应用程序，如[3,45]所述。[45]此外，还建议改进SGX调用机制。

(v)交换EPC页面费用昂贵，每次交换操作要花费数十万个周期。当SGX在开始时测量飞地的内容，包括任何统计初始化数列时，如果飞地内存大小大于可用的EPC大小，这将触发飞地交换。虚拟化会导致额外的开销，这会根据在飞地内部运行的线程数量而增加。

课程(ii)表明，嵌套寻页会导致了虚拟机上的飞地放缓，如我们的飞地微基准中所示。可以通过使用EPC的阴影分页来优化SGX飞地的地址转换，以减少翻译开销和嵌套分页，从而实现[16]中所述的敏捷地址转换器。

除了此优化之外，我们还提出了在所有系统中应用于SGX的进一步优化：

（1）课程(iv)可以通过使用热调用[45]等机制来解决，这些机制可以在应用程序和飞地代码之间提供一个快速的调用接口。高效的呼叫接口有助于减少安装和安装的显著开销，并有助于轻松将应用程序移植到IntelSGX。

（2）第(v)课将EPC页面交换到RAM确定为开销的主要来源，因此最小化飞地的规模有助于减少飞地启动和飞地运营期间的交换，从而提高飞地的性能。

（3）除了最小化飞地之外，在EPC页面驱逐期间，更明智的页面选择也有帮助在免费EPC较低时减少EPC交换开销。

接下来，我们详细介绍了我们对虚拟机上的SGX的优化。

6个优化方案

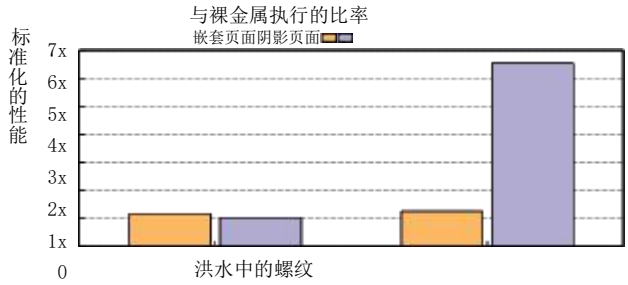
如上一节所示，阴影分页和嵌套分页都对虚拟机上运行的应用程序造成开销。使用阴影分页时，TLB丢失并不比TLB丢失慢，但是，页面表的每次修改都需要捕获到管理程序中来更新阴影页表。相反，对于嵌套分页，页面表的修改不需要捕获到管理程序中，但是每次TLB错过都需要更昂贵的二维页面行走[4]。因此，生成大量TLB未命中的VM应该使用阴影分页来进行内存虚拟化，而在其他情况下，嵌套分页可以用于方便更快的页表编辑。或者，[16]提出的敏捷解决方案可以用来同时利用这两个世界。在哪一种情况下，管理程序都应该能够动态识别VM的工作负载类型，以便在这两种方法之间做出适当的选择。

在本节中，我们提出了一种非侵入性的方法来测量给定工作量的特征，以确定它是否适合嵌套分页或阴影分页，因此允许自动选择适当的内存虚拟化技术。

首先，我们确定了影响阴影和嵌套分页性能的因素。

正如在前面的结果中所看到的，阴影分页在SGX调用和密码学Proc中表现良好。ACM的平均值。肛门检查。投入工作

05：16丁等人。



图中示。 14.标准化的基准测试运行时间（较低的值更好）

基准测试是CPU和内存密集的基准测试，但对于像HTTP服务器这样的重I/O的应用程序来说很差。相反，嵌套分页在I/O重的应用程序中速度更快，但在内存不足的基准测试中执行得较慢。根据上述虚拟化机制的不同，我们选择了两个标准作为应用程序特性的指标：TLB错过的总数和上下文切换的数量。虽然上下文切换并不是阴影分页造成性能影响的唯一原因(不像TLB错过，它直接减缓二维页面行走导致的嵌套分页)，我们选择上下文切换的数量作为指标有两个原因：首先，用户模式过程的页面表更新涉及上下文切换；其次，使用性能监控系统可以很容易地配置上下文切换。

为了描述阴影和嵌套寻页的成本，我们在裸机和虚拟机上试验了两个工作负载，它们代表了它们各自的“最坏情况”性能。第一个工作负载涉及对大缓冲区中随机地址的内存操作，以强制发生许多TLB未命中，这将显著降低嵌套寻页的速度。对于第二个工作负载，我们使用了系统台[31]线程基准测试，其中20个线程不断相互让步，旨在通过对调用的每个上下文切换要求管理程序陷阱来减缓阴影分页。

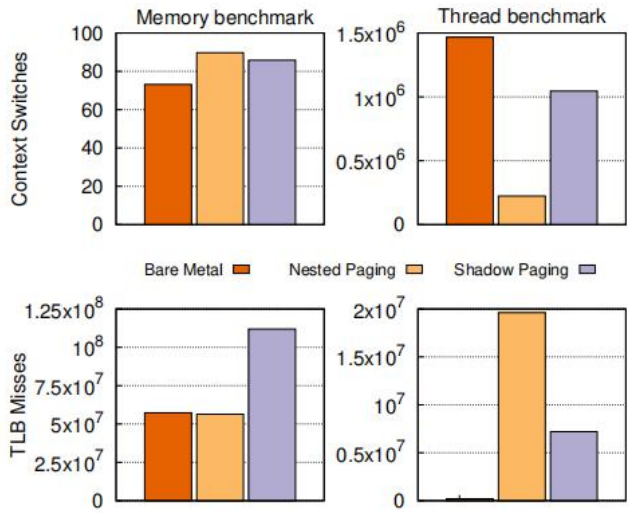
图14显示了两种内存虚拟化方法相比的相对基准性能。虽然阴影分页在内存基准中获胜，而嵌套分页的17%只有3%，但相反，线程基准中明显下降了6.5×，而嵌套分页的速度只有25%。一般来说，这些结果与我们上述基准的目标一致。

接下来，我们执行了基准测试，以根据上述标准来确定这些工作负载的特征。我们使用perf实用程序（在主机中执行）来测量由基准测试执行的TLB未命中和上下文切换的数量。

图15显示了在执行基准测试过程中每秒发生的上下文切换和TLB丢失的平均次数。在内存基准测试中，虽然我们看到裸金属和阴影分页中类似的TLB丢失数量，但嵌套分页导致的TLB丢失数量是裸金属的两倍，这表明嵌套分页本身是内存基准测试中这些额外TLB丢失的来源。相反，虽然在裸金属上的线程基准测试每秒只导致少量TLB丢失，但两种内存虚拟化方法的事件计数都要高得多（分别为37×-100×，具有嵌套和阴影分页）。最后，对于线程基准测试，我们看到性能和每秒上下文切换的数量之间存在强烈的反相关性，这表明基准测试将大部分时间花在了上下文切换上。

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

你应该知道的关于IntelSGX性能的一切05：17



图中示。 15.上下文切换和TLB每秒错过事件

通过上面提出的观察结果，我们现在可以建立阴影和嵌套寻页的基线性能。我们假设，在内存基准测试中，上下文开关的运行时贡献是最小的。设为c0 是线程基准测试中上下文开关的总数，

并让t0 ， 电话以及吨分别是它在裸金属、嵌套分页和阴影分页上的运行时间。 同样地，让m0 是内存基准测试中TLB错过的次数，和u0 ， 星期二还有我们，它的运行时间。注意，由于基准运行时间u0 ， 星期二，美国公司包括指令和内存访问所花费的时间，我们在裸机上运行相同的内存基准，透明的大页面可以建立基准的运行时间，同时最小化所花费的时间在TLB上错过了。我们表示这个运行时u。′0因此，花在TLB缺失上的总时间可以通过减去这两个时间来计算：

$$v0 = u0 - u_0'$$

$$= = ue - u_0'$$

$$vs = us - u_0'$$

一旦建立了基线，我们将使用性能计数器来测量工作负载的特性，以确定它是重TLB还是重上下文切换。假设在一段时间内，程序生成c上下文开关和mTLB丢失，则可以使用以下算法选择最佳内存虚拟化方法：

（1）推断嵌套分页的性能降低：

$$\text{回复} = \frac{c\ m}{c0} \text{ 电话} + \quad \text{平均值}$$

（2）推断阴影分页的性能降低：

$$\text{小时} = \frac{c\ m}{c0} \text{ 吨} + \quad \text{与}$$

（3）计算阴影与阴影的性能比。具有嵌套的分页功能：

$$k = \frac{\text{回复}}{\text{小}}$$

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

05：18图丁克等人。

基准测试	k	设计方法
文件读写，mmap（2个线程）	1.116	阴影
文件读写文件（2个线程）	0.628	嵌套的
文件读写（2个线程，周期f同步）	0.220	嵌套的
随机内存读取	1.138	阴影
MySQLOLTP读取基准测试	0.509	嵌套的
Nginxweb服务器，空响应	0.350	嵌套的

表1。对各种工作负载上优化的评估结果

（4）使用k，选择首选内存虚拟化方法：如果为k≤1，则选择嵌套分页，如果是，则选择阴影分页。

7、最优化评价

在前一节中，我们提出了一种选择各种应用程序的内存虚拟化最优方法的算法。为了验证我们的算法，我们将其应用于各种基准测试上：

随机读写由内存映射而成的文件。
使用同步输入输出进行随机读写文件。
使用同步输入/O和周期帧同步的随机读取/写入文件。
随机内存读取。
总共有2GB。
MySQLOLTP读取基准测试，该表包含1000万行。
Nginxweb服务器提供了一个空的HTTP响应。

除了Nginxweb服务器基准测试之外，上述所有基准测试都是使用系统台工具执行的。我们的文件基准测试在一个包含128个文件的测试文件集中工作，每个文件的大小为16MB

表1显示了我们在这些基准测试中的算法的结果。k列显示了嵌套分页与之间的性能影响的预测比率。阴影分页功能。简而言之，k的比率意味着，当考虑上下文切换和TLB丢失所花费的组合时间时，阴影寻页将比嵌套寻页快k倍。因此，k>1表示重TLB的工作负载，其中首选阴影分页，对于k≤1，如“方法”列所示。

我们的结果预测，嵌套分页比阴影分页更好，除了两个基准测试：使用mmap进行文件读写和随机内存读取。然而，由于这些基准测试的预测k比接近1（分别为1.116和1.138），我们的算法只预测了这些重TLB基准测试的一个小的性能改进。相比之下，其余的基准测试在嵌套分页方面显示出了更大的优势。例如，Nginx基准测试预测的k值为0.350，或为阴影分页而减速为2.9×。这些预测在很大程度上与我们上面提出的基准测试的结果一致：涉及许多TLB丢失的工作负载倾向于支持阴影分页，而具有Io绑定或有多个上下文开关(例如，Nginx)的工作负载支持嵌套分页。

8项相关工作

我们注意到，以前没有任何工作可以研究SGX在虚拟化系统上的性能。话虽如此，我们将IntelSGX的相关工作分类如下四类。

正在审查SGX的安全性。科斯坦和德瓦达斯[13]非常深入地展示了英特尔的SGX架构。作者概述了它的设计和特点，并将其与以前的可信计算平台进行了比较。[13]还包括有关物理攻击者、特权软件、内存映射、外设、缓存时间和侧通道的SGX的安全分析[35]。作者演示了一些与SGX相关的安全属性：(i)SGX进行的TLB检查与其安全保证一致；(ii)分配页面的EPCM条目符合飞地的要求；(iii)在飞地执行期间，EPCM和TLB内容属于飞地独家页面是一致的；并且(iv)只有在不存在与该页面对应的TLB映射时才能修改页面的EPCM条目。最后，[13]提出了一种跟踪页面驱逐期间TLB刷新的方法，并指出在IntelSDM[23]中没有描述实际使用的过程。对SGX设计的评估也导致了新的攻击。AsyncShock[44]操作用来执行飞地代码的线程的调度，以利用多线程代码的同步错误。最近，SgxPectre[9]利用了已知的投机极限
执行x86架构，从飞地泄露秘密。

使用SGX用于安全关键型系统。我们将scone[2]纳入这一类，一个完全在SGX飞地内运行容器化应用程序的框架。该框架为安全部署基于容器的系统打开了可能性，这在基于云的应用程序中越来越常见。安全管理员[6]在IntelSGX环境的屏蔽边界内运行动物管理员协调服务，因此支持敏感信息的分发。如果存在与网络流量相关的隐私问题的分布式系统，您可以依靠TaLoS[3]直接在飞地内建立TLS端点。

性能分析。HotCalls[45]观察到，SGXSDK提供的瀑布和瀑布有一个高开销，根据缓存状态从大约8000到17000个循环不等。作者提出了一种新的异步设计，使用共享缓冲区，可以将此开销降低到620-1400个周期。与我们的工作类似，作者确定了SGX造成的三个潜在开销来源：(i)进出飞地，(ii)在应用程序和飞地之间传递缓冲区，以及(iii)读取和写加密内存的开销。最近对LinuxSGXSDK的升级引入了类似的功能[11]。

关于改进SGX的建议。最后，[47]和[7]展示了英特尔将集成到下次SGX版本中的未来特性，即动态内存分配(在SGXv2中)和超额订阅。

9个结论

本文首次详细介绍了IntelSGX在虚拟化系统(在我们的例子中是KVM)上的性能分析，后者是云平台的构建部分。由于知道云计算是SGX的主要用例之一，而且虚拟化带有开销，所以在虚拟机中对SGX进行性能分析很有用。为此，我们的性能建立在一套广泛的微观和宏观基准测试之上。

我们构建了一组微基准测试，以仔细捕获虚拟化（及其不同实现）对每个SGX功能(ecall、ocall、传输缓冲区到/输出/飞地创建/破坏)的影响，比较在裸机系统上实现的性能。

我们的宏基准(加密基准和HTTP服务器基准)捕获飞地中常用的函数。所有的基准都在[14]作为开源代码提供，以促进实验的重现性，并允许研究人员利用我们的结果。

我们对每一个获得的结果提供了全面的解释，并总结了我们的主要发现和经验教训。基于这些发现，我们确定了几个优化方案

这将是一种能够提高IntelSGX在虚拟化系统上的性能的策略。我们实现了Proc。ACM的平均值

05：20图等人。

并评估其中一个优化，而其他优化是我们未来工作的主题。我们相信，这项研究将为SGX的开发人员和用户，以及未来TEE微架构的设计者带来有价值的见解。

参考文献

[1]阿纳蒂，谢伊盖隆，西蒙约翰逊和文森特斯卡拉塔。 2013.基于CPU的认证的创新技术和密封件。在第二届关于安全和隐私的硬件和架构支持的国际研讨会的论文集中，卷。 13.纽约，纽约，美国。

[2]谢尔盖·阿诺托夫，波丹·特拉奇，弗朗茨·格雷格，托马斯·克诺斯，安德烈·马丁安，克里斯蒂安·普里比，约书亚·林德，迪维亚穆图库马兰，丹奥基夫，马克斯蒂尔威，等人。 2016.斯康内：使用IntelSGX保护安全的Linux容器。在OSDI中，卷。 16. 689 - 703.

[3]皮埃尔-路易斯奥布林，弗洛里安凯尔伯特，丹奥基夫，穆图穆马兰，克里斯蒂安普里比，约书亚林德，罗伯特克拉恩，克里斯托夫费泽尔，大卫埃耶斯和彼得皮祖奇。 2017.标签：在SGX飞地内设置安全和透明的TLS终端。技术报告。伦敦帝国理工学院。

[4]拉维巴尔加瓦，本杰明塞勒布林，弗朗西斯科斯帕迪尼，和斯里拉塔曼内。 2008.加快二维页面的速度针对虚拟化系统的行走。在ACM签名计算机架构新闻，Vol。 36.acm，26-35岁。

[5]尼希尔·巴蒂亚。 2009.对IntelEPT硬件辅助系统的性能评估。VMware，股份有限公司（2009年）。

[6]斯蒂凡·布伦纳，科林·伍尔夫，大卫·戈尔茨舍，尼科·韦奇布罗特，马蒂亚斯·洛伦茨，克里斯托夫·费泽尔，彼得·皮祖策，和鲁迪格·卡皮扎。 2016.安全管理员：使用英特尔SGX的保密动物管理员。第17届国际中间件会议会议记录。宏量m

[7]萨姆纳·特查克拉巴蒂，丽贝卡莱斯利-赫德，蒙娜维吉，弗兰克麦肯，卡洛斯罗萨斯，卡斯皮，伊利亚亚历山大罗维奇，和还有一个蒂。 2017.用于在虚拟化环境中超额订阅安全内存的Intel软件保护扩展(IntelSGX)体系结构。在对安全和隐私的硬件和建筑支持的程序中。acm，7。

[8]蔡蔡，E。波特和莫娜·维吉。 2017.石墨烯-SGX：一个针对未修改的应用程序的实用库操作系统在SGX上。2017年USENIX年度技术会议（USENIXATC17）。USENIX协会，圣克拉拉，CA，645-658。https://www.usenix.org/conference/atc17/technical-sessions/presentation/tsai

[9]陈股息、陈三川、袁肖、张钱、林志强、十赖。 2018.SgxPectre攻击：通过投机执行从SGX飞地窃取英特尔的秘密。arXiv预印本arXiv：1802.09085（2018年）。

肖恩·克里斯托夫森。 2017.KVM：vmx：添加对SGX启动控制的支持。https://github.com/intel/kvm-sgx/commit/e9a065d3c1773ad72bfb28b6dad4c433f392eda8.

英特尔公司。 2018.IntelLinuxSGXSDKv2.2--无交换机呼叫。https://download.01.org/intel-sgx/linux-2.2/docs/Intel_SGX_Developer_Reference_Linux_2.2_Open_Source.pdf.

英特尔公司。 2018.L1端子故障。https://software.intel.com/security-software-guidance/software-guidance/l1终端故障。

维克多·科斯坦和德瓦达斯。 2016.IntelSGX已进行了解释。IACR密码学电子打印档案2016（2016），86。你不了。 2018.SGX基准测试源代码。https://github.com/sgxbench/sgxbench/releases.

爱德华·费尔滕。 2003.理解可信计算：它的好处是否大于缺点？IEEE安全保障&隐私权，99年、3年（2003年），60-62年。

杰尼尔甘地，马克D希尔，和迈克尔M斯威夫特。 2016.敏捷的分页：超过嵌套和阴影的最佳效果正在执行分页操作。在ACM签名计算机架构新闻，Vol。 44.IEEE出版社，707-718。

谷歌。 2018.Asylo：一个用于机密计算的开源框架。https://cloudplatform.googleblog.com/2018/05/Introducing-Asylo-an-open-source-framework-for-confidential-computing.html.

大卫·格拉洛克。 2009.可信平台的动力学：一种构建块的方法。英特尔出版社。

最受信任的计算组。 2007.设计原则规范1.2第2版103第1部分。

谢伊·盖伦。 2016.针对通用处理器的内存加密。IEEE安全与隐私6（2016），54-62。丹尼·哈尼克和艾利亚德·茨法迪亚。 2017.对英特尔SGX性能的印象。https://medium.com/@danny_harnik/22442093595a.

[22]ibm. 2018.使用IntelSGX的IBM云上的使用数据保护。https://www.ibm.com/blogs/bluemix/2018/05/data-use-protection-ibm-cloud-using-intel-sgx/。

英特尔。 2018.英特尔软件开发手册。https://software.intel.com/en-us/articles/intel-sdm.英特尔公司。[n.d.]。Intel软件保护扩展SDK。https://software.intel.com/en-us/sgx-sdk.

英特尔公司。[n.d.]。Intel软件保护扩展SDK为Linux。https://01.org/智能网络-软件保护程序的扩展。

英特尔公司。[n.d.]。SGX虚拟化技术。https://01.org/intel-software-guard-extensions/sgx-virtualization.

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期：2019年3月。

你应该知道的关于IntelSGX性能05：21

英特尔公司。 2017.英特尔和中性狮带来安全，4KUHD体育流到电脑。新闻编辑室。
intel.com/news/intel-neulion-bring-secure-4k-uhd-sports-streaming-computers/。
英特尔公司。 2017.Intel软件保护扩展SDKforLinux操作系统。https://download.01.org/intel-sgx/linux-2.
0/docs/Intel_SGX_Installation_Guide_Linux_2.0_Open_Source.pdf
国际标准化组织。 2015.iso/iec11889-1：2015。
大卫卡普兰，杰里米鲍威尔，和汤姆沃勒。 2016.AMD内存加密。白皮书（2016年）。科皮托夫。[n.d.]
。
克劳斯·库萨，德里斯·谢莱肯斯和巴特·普雷内尔。 2005.正在分析受信任的平台通信。在ECRYPT中
车间，崩溃图形在安全硬件。
郑丽、基尔、路清华和安德森。 2017.绩效管理费之间的比较
基于虚拟机管理程序和容器的虚拟化。在先进的信息网络和应用(AINA)中，2017年IEEE第31届国际会议上关于。
ieee，955-962年。
约书亚·林德，克里斯蒂安·普里比，穆图库马兰，丹·奥基夫，皮埃尔-路易斯·奥布林，弗洛里安·凯尔伯特，托拜
大卫戈尔茨切，大卫埃耶斯，卡皮扎，克里斯托夫费泽尔，和彼得皮祖奇。 2017.魅力：针对IntelSGX的自动应用
程序分区。在2017年USENIXUsenix年度技术会议记录(USENIXATC '17)。USENIX协会，伯克利，美国，285-298。
http://dl.acm.org/citation.cfm? id=
3154690.3154718
奥莱克森科、博丹·特拉奇、罗伯特·克拉恩、马克·西尔伯斯坦和克里斯托夫·费泽尔。 2018.变种化：保护SGX
从实际的侧通道攻击中获得的飞地。2018年USENIX年度技术会议(USENIXATC18)。美尼
协会，波士顿，马州，227-240。https://www.usenix.org/conference/atc18/presentation/oleksenko
瑞恩帕弗和丽莎波格梅耶。 2016.防护织物和屏蔽虚拟机概述。https://docs.microsoft.com/en-us/
windows-server/virtualization/guarded-fabric-shielded-vm/guarded-fabric-and-
shielded-vms.里瓦·里士满。[n.d.]。
马可。莱特尼。 2010.启用Intel虚拟化技术的功能和好处。英特尔公司的白皮书。检索月份检索
15 (2010)，2012.
罗特姆和高级首席工程师。 2015.英特尔架构，代号为Skylake深潜水：一个新的
用来管理电力性能和能源效率的架构。在英特尔开发者论坛上。
马克·鲁西诺维奇。 2018.Azure机密计算。https://azure.microsoft.com/en-us/blog/
天蓝色的机密计算。
三星电子股份有限公司。 2017.三星诺克斯安全解决方案。https://www.samsungknox.com/docs/
SamsungKnoxSecuritySolution.pdf.
埃文·斯帕克斯和斯帕克斯。 2007.可信平台模块安全评估的计算机科学
技术报告TR2007-597。 (2007)。
吉的。 2018.已进行WRK2Http基准测试。https://github.com/giltene/wrk2.
尼科韦希布罗特，阿尼尔库尔穆斯，彼得皮祖奇和鲁迪格卡皮扎。 2016.异步冲击：正在利用同步功能
英特尔SGX飞地中的漏洞。在欧洲计算机安全研究研讨会上。施普林格出版社，第440-457页。
安全的飞地。第44届计算机建筑国际研讨会论文集。acm，81-93年。
瓦莱里亚·贝尔塔科和托德·奥斯门。 2017.有效时间重新获得失去的周期：SGX的快速接口

[46]是有线的。 2009.谷歌的黑客攻击是非常复杂的。https://www.wired.com/2010/01/operation-aurora/。本塞德
里克星，马克沙纳罕和丽贝卡莱斯利赫德。 2016.Intel软件保护扩展(IntelSGX)软件
支持在程序包内的动态内存分配。2016年安全和隐私支持论文集。宏量m，11。

2018年11月收到；2018年12月修订；2019年1月接受

项目。ACM的平均值。肛门检查。投入工作。系统节，卷。3，No.第1条，第05条。出版日期
：2019年3月。