(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117807587 A
(43)申请公布日 2024.04.02

(21)申请号 202311738871.9

(22)申请日 2023.12.15

(71)申请人 南方科技大学
地址 518055 广东省深圳市南山区学苑大道1088号
申请人 支付宝(杭州)信息技术有限公司

(72)发明人 张锋巍 王晨旭 邓韵杰 闫守孟 何征宇

(74)专利代理机构 北京亿腾知识产权代理事务所(普通合伙) 11309
专利代理师 陈霁 周良玉

(51)Int.Cl.
G06F 21/53(2013.01)
G06F 21/64(2013.01)
G06F 21/71(2013.01)

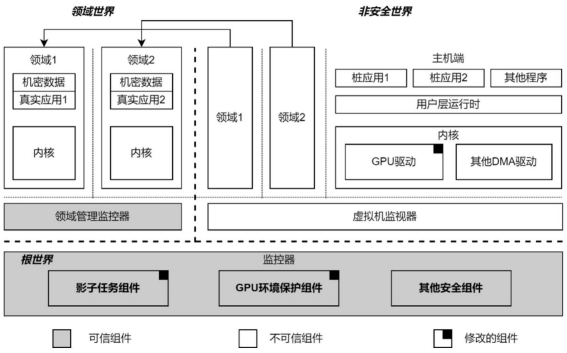
权利要求书2页 说明书11页 附图4页

(54)发明名称

在机密计算架构中执行GPU任务的方法和装置

(57)摘要

本说明书实施例提供一种在机密计算架构中执行GPU任务的方法和装置。机密计算架构包括,安全世界,领域世界,非安全世界,根世界;相应方法包括:非安全世界的GPU软件根据GPU任务的任务代码和缓存描述,在非安全世界内存中配置桩数据结构,其中包括,根据缓存描述分配的缓存区,以及指示各缓存区的元数据。然后,根世界中的根监视器在内存的领域区段中,配置与桩数据结构对应的真实数据结构,将待处理的机密数据存储于其中。根监视器更新GPT表,使得根据更新后的GPT表,存储元数据和任务代码的目标区段对于GPU可访问,对于其他对象均具有领域世界权限。此外,根监视器修改目标映射关系,使得GPU使用目标区段和真实数据结构执行上述GPU任务。



CN 117807587 A

1.一种在机密计算架构中执行GPU任务的方法,所述机密计算架构包括,安全世界,领域世界,非安全世界,根世界;所述方法包括:

非安全世界的GPU软件根据用户提供的第一GPU任务的任务代码和缓存描述,在内存的非安全世界区段中,配置第一桩任务的桩数据结构,所述桩数据结构包括,根据所述缓存描述分配的多个缓存区,以及指示各缓存区的元数据;

根世界中的根监视器在所述内存中对应于第一领域的区段中,配置与所述桩数据结构对应的真实数据结构,将待处理的机密数据存储于其中;

所述根监视器更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界的权限,其中所述目标区段存储所述元数据和任务代码;

所述根监视器修改目标映射关系,使得GPU使用所述目标区段和所述真实数据结构执行所述第一GPU任务。

2.根据权利要求1所述的方法,其中,所述GPU软件包括,GPU驱动和相关函数库。

3.根据权利要求1所述的方法,还包括,在配置第一桩任务的桩数据结构之前:

所述第一领域通过安全信道,接收用户提供的所述机密数据;

所述非安全世界的主机接收所述任务代码和缓存描述。

4.根据权利要求3所述的方法,还包括:

所述第一领域与所述用户通过密钥协商协议,协商出密钥;

基于所述密钥,构建所述安全信道。

5.根据权利要求1所述的方法,其中,配置第一桩任务的桩数据结构,包括:

分配代码缓存区,将所述任务代码存储于其中;

根据所述缓存描述,分配若干桩数据缓存区,将所述缓存描述存储于对应的桩数据缓存区;

生成所述元数据,其包括指向所述代码缓存区和所述若干桩数据缓存区的指针。

6.根据权利要求5所述的方法,其中,配置与所述桩数据结构对应的真实数据结构,包括:

分配与所述若干桩数据缓存区对应的若干真实数据缓存区,将所述机密数据存储于所述真实数据缓存区。

7.根据权利要求6所述的方法,其中,所述若干真实数据缓存区包括,输入数据缓存区和结果数据缓存区,所述输入数据缓存区存储所述机密数据,所述结果数据缓存区用于存储所述第一GPU任务的执行结果。

8.根据权利要求6所述的方法,其中,

配置第一桩任务的桩数据结构,还包括:根据所述桩数据缓存区,生成桩GPU页表;

配置与所述桩缓存结构对应的真实缓存区段,还包括:根据所述桩GPU页表,和所述若干真实数据缓存区,生成真实GPU页表。

9.根据权利要求6所述的方法,其中,所述根监视器修改目标映射关系,包括:

修改所述元数据中的指针,使其指向所述若干真实数据缓存区和所述代码缓存区。

10.根据权利要求8所述的方法,其中,所述根监视器修改目标映射关系,包括:

修改GPU内存映射,使其指向所述真实GPU页表。

11.根据权利要求5所述的方法,其中,所述用户提供的第一GPU任务的任务代码和缓存

描述附带有签名信息;在修改目标映射关系之前,所述方法还包括:

根监视器根据所述签名信息,校验所述代码缓存区中存储的任务代码。

12.根据权利要求1所述的方法,其中,更新颗粒度保护表GPT,包括:

更新第一版本GPT表,使得更新后的第一版本GPT表中,所述目标区段属于领域世界;所述第一版本GPT表用于CPU和若干外设;

更新第二版本GPT表,使得更新后的第二版本GPT表中,所述第一领域的区段和所述目标区段被设置为可访问的非安全世界内存,其他区段均不可访问;其中所述第二版本GPT表用于GPU执行第一领域的任务。

13.根据权利要求12所述的方法,其中,在更新后的第一版本GPT表中,GPU内存映射区段被设置为属于根世界。

14.根据权利要求12所述的方法,其中,所述第一版本GPT表包括,用于CPU的第一GPT表,和用于若干外设的若干第二GPT表;所述第一GPT表和若干第二GPT表通过表描述符指向同一次层表;所述次层表针对预设内存区段构建,所述目标区段属于该预设内存区段。

15.根据权利要求1所述的方法,还包括:

在执行所述第一GPU任务之后,将所述目标区段恢复为属于非安全世界。

16.一种机密计算架构中的根监视器,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根监视器位于所述根世界中,并包括影子任务组件和GPU环境保护组件,其中:

影子任务组件配置为,响应于非安全世界的GPU软件在内存的非安全世界区段中配置第一桩任务的桩数据结构,而在所述内存中对应于第一领域的区段中,配置与所述桩数据结构对应的真实数据结构,并将待处理的机密数据存储于其中,其中所述桩数据结构包括,根据用户提供的缓存描述分配的多个缓存区,以及指示各缓存区的元数据;

GPU环境保护组件配置为,更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界的权限,其中所述目标区段存储所述元数据和用户提供的第一GPU任务的任务代码;

所述影子任务组件还配置为,修改目标映射关系,使得GPU使用所述目标区段和所述真实数据结构执行所述第一GPU任务。

17.一种计算设备,包括存储器和若干处理器,所述计算设备形成机密计算架构,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根世界包括权利要求16所述的根监视器。

在机密计算架构中执行GPU任务的方法和装置

技术领域

[0001] 本说明书一个或多个实施例涉及机密计算框架,尤其涉及一种在机密计算框架中执行GPU任务的方法及装置。

背景技术

[0002] 随着各行业计算技术的发展,以及云端和终端用户的增加,人们将大量数据存储在各种计算机设备中。在行业发展的同时,人们对于设备和数据安全的关注也在日益增加。为了确保设备和数据的安全性,各个架构厂商也分别提出了各自的解决方案,如ARM提出了可信区技术(TrustZone),AMD提出了安全虚拟机加密技术(SEV),英特尔提出了软件防护扩展(SGX)技术,等等。这些解决方案为用户提供一个安全的可信执行环境,用于机密地保存和处理数据,使其免受不可信的内核与传统应用程序的损害。以Arm可信区技术为例,它将传统内核和应用程序的运行环境视作为非安全世界,并创建了一个隔离的安全世界,以及定义了具有最高权限的安全层用于世界切换。非安全世界将无法直接访问安全世界,需要经过安全层的固件验证才能访问特定的资源。

[0003] 虽然ARM机密计算架构有效地确保了用户的数据安全,然而,其仍然存在一些不足,其中之一是无法提供对GPU等加速器上机密计算的支持。这使得在该技术框架下,采用GPU进行任务加速具有很大的安全方面的挑战,存在对此进行改进的需求。

发明内容

[0004] 本说明书一个或多个实施例描述了一种在机密计算架构中执行GPU任务的方法及装置,能够基于已有的机密计算架构的硬件特性,为GPU任务的执行提供机密计算环境,支持GPU机密计算。

[0005] 根据第一方面,提供一种在机密计算架构中执行GPU任务的方法,所述机密计算架构包括,安全世界,领域世界,非安全世界,根世界;所述方法包括:

[0006] 非安全世界的GPU软件根据用户提供的第一GPU任务的任务代码和缓存描述,在内存的非安全世界区段中,配置第一桩任务的桩数据结构,所述桩数据结构包括,根据所述缓存描述分配的多个缓存区,以及指示各缓存区的元数据;

[0007] 根世界中的根监视器在所述内存中对应于第一领域的区段中,配置与所述桩数据结构对应的真实数据结构,将待处理的机密数据存储于其中;

[0008] 所述根监视器更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界的权限,其中所述目标区段存储所述元数据和任务代码;

[0009] 所述根监视器修改目标映射关系,使得GPU使用所述目标区段和所述真实数据结构执行所述第一GPU任务。

[0010] 根据第二方面,提供了一种机密计算架构中的根监视器,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根监视器位于所述根世界中,并包括影子

任务组件和GPU环境保护组件,其中:

[0011] 影子任务组件配置为,响应于非安全世界的GPU软件在内存的非安全世界区段中配置第一桩任务的桩数据结构,而在所述内存中对应于第一领域的区段中,配置与所述桩数据结构对应的真实数据结构,并将待处理的机密数据存储于其中,其中所述桩数据结构包括,根据用户提供的缓存描述分配的多个缓存区,以及指示各缓存区的元数据;

[0012] GPU环境保护组件配置为,更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界的权限,其中所述目标区段存储所述元数据和用户提供的第一GPU任务的任务代码;

[0013] 所述影子任务组件还配置为,修改目标映射关系,使得GPU使用所述目标区段和所述真实数据结构执行所述第一GPU任务。

[0014] 根据第三方面,提供了一种计算设备,包括存储器和若干处理器,所述计算设备形成机密计算架构,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根世界包括如第二方面所述的根监视器。

[0015] 在本说明书实施例提供的方案中,通过影子任务机制实现兼容Arm机密计算架构CCA的GPU机密计算。根据影子任务机制,由非安全世界的GPU软件创建不包含真实数据的桩任务,并如常规流程调度和管理桩任务。在桩任务提交后,根监视器创建包含真实数据的真实GPU任务,并为其提供受保护的执行环境。然后,根监视器用真实GPU任务替换桩任务,从而使得,GPU硬件在受保护的执行环境中执行真实GPU任务。如此,在Arm机密计算架构CCA中实现GPU机密计算。

附图说明

[0016] 为了更清楚地说明本发明实施例的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其它的附图。

[0017] 图1示出Arm机密计算架构的示意图;

[0018] 图2示出机密计算架构中各个世界对物理地址空间的访问权限控制;

[0019] 图3示出根据一个实施例在机密计算架构中运行GPU任务的示意图;

[0020] 图4示出根据一个实施例在机密计算架构中执行GPU任务的方法;

[0021] 图5示出影子任务机制下各个阶段数据结构的示意图;

[0022] 图6示出在一个示例场景中,根监视器维护的GPT表;

[0023] 图7示出根据一个实施例的次层表示意图。

具体实施方式

[0024] 下面结合附图,对本说明书提供的方案进行描述。

[0025] 为了保证数据的安全性,ARM提供了TrustZone可信区技术。在该技术中,将传统内核和应用程序的运行环境视为非安全世界(Normal World),在此之外创建了一个隔离的安全世界(Secure World),并定义了具有最高权限的安全层用于世界切换。

[0026] 具体的,在Armv8-A架构中,CPU核基于特权划分将异常划分为4个等级,EL0至EL3,

其中,EL0表示应用级,EL1用于系统内核(kernal),EL2表示虚拟机管理器(hypervisor),EL3表示安全层监视器。这四个等级也可用于表示运行环境的权限等级。在TrustZone可信区技术中,CPU安全状态被划分为非安全(Normal)状态和安全状态。EL0和EL1可以运行于任意状态,例如,可以在非安全世界的EL1中执行非可信的操作系统OS(untrusted OS),在安全世界的EL1中执行可信OS。EL2可用于安全状态。EL3即安全层监视器,永远存在于安全世界,用于执行安全状态的切换。

[0027] 在该架构下,非安全世界无法直接访问安全世界,需要经过安全层监视器的验证才能访问特定的资源。敏感或机密的数据,以及高权限的软件应用运行于安全世界,从而为这些机密数据提供一种可信执行环境TEE。

[0028] 在以上的TrustZone基础架构基础上,ARM近来又发布了改进的Arm机密计算架构CCA(Confidential Compute Architecture)。Arm机密计算架构是Armv9-A架构的一部分,其在原本的TrustZone架构基础上引入了领域管理扩展,该扩展在可信区技术中已经存在的非安全世界与安全世界之外,额外引入领域(Realm)世界和根(Root)世界。为了支持不同世界的隔离,CCA架构在硬件层提供领域管理扩展RME(Realm Management Extension)组件,来扩展隔离模式。

[0029] 图1示出Arm机密计算架构的示意图。如图1所示,在Arm机密计算机构CCA中,运行环境被划分为四个世界:安全世界,非安全世界(Normal),领域世界和根世界。根世界中运行着拥有最高权限的根世界监视器,负责世界之间的隔离和通信。领域世界用于为虚拟机提供名为机密领域的受保护的虚拟机机密计算环境。领域世界中运行着领域管理监视器RMM,负责管理领域虚拟机的执行以及与非安全世界的交互。用户可以将虚拟机作为领域虚拟机放入机密领域中,隔离来自外界软件的非法访问。具体的,用户可以通过非安全世界中的虚拟机管理器创建虚拟机,通过领域管理监视器RMM将其转入到领域世界,成为领域虚拟机。领域管理监视器RMM会负责与机密领域安全相关的检查和保护。领域虚拟机之间使用虚拟化技术相互隔离,领域管理监视器会负责管理不同领域虚拟机的可访问地址空间。领域虚拟机不需要相信非安全世界和安全世界,只需要相信领域管理监视器和根世界监视器。

[0030] 相应的,Arm机密计算架构CCA将内存的物理地址空间PAS(Physical address spaces)也划分为四个世界。图2示出机密计算架构中各个世界的安全状态对物理地址空间的访问权限控制。如图2所示,根世界具有最高的访问权限,可以访问所有四个世界的地址空间。非安全世界具有最低的访问权限,仅可以访问非安全世界的地址空间。安全世界和领域世界,则可以访问非安全世界的地址空间,以及属于自己世界的地址空间。

[0031] 在Arm机密计算架构中,不同世界的地址空间访问控制,通过构建颗粒度保护表GPT(Granule Protection Table),并基于该GPT表进行颗粒度保护检查GPC(Granule Protection Check)而实现。具体的,机密计算架构CCA在内存中维护颗粒度保护表GPT,其中记录细粒度的每段物理内存的安全状态。典型的,记录的粒度是以内存页(4KB大小的区段)为单位。如此,GPT表中记录每个内存页的安全状态和访问权限。当内存页的分配在不同世界发生迁移和变更,则可以动态更新GPT中的条目。

[0032] 当处理器访问内存时,硬件层中的前述RME组件执行颗粒度保护检查GPC。在检查中,获取当前CPU的安全状态,并通过读取GPT表获取请求访问的内存页的安全状态,检查二者是否匹配。如果没有通过GPC检查(例如,如果非安全世界的主机OS请求访问领域世界的

内存),则会发出颗粒度保护异常信号,从而拒绝此次内存访问,由此保证世界之间的隔离。通过以上的隔离机制,Arm机密计算架构进一步为领域世界的领域虚拟机提供了隔离的机密计算环境。

[0033] 另一方面,越来越多的任务希望能够通过GPU来进行加速执行。然而,现有的Arm计算框架难以为GPU计算任务提供有效的机密保护。这一方面是因为,大多数Arm设备的GPU属于嵌入式GPU,它不含有独立的内存,而Arm计算框架将GPU视为非可信的普通外设,因此GPU需要与CPU和诸多不可信外设共享内存,更易受到攻击。另一方面,根据目前大多数ARM设备GPU的工作流程,GPU的任务执行和调度受到GPU软件(如GPU驱动和相关的编程软件库)的管理。而该GPU软件处于非安全世界中,容易受到攻击。

[0034] 具体的,GPU软件用于管理GPU的计算环境,并与GPU硬件进行交互。一般来说,为了准备执行环境,GPU软件根据GPU任务的要求,分配物理内存并创建GPU缓存。接下来,将GPU任务的核心组件加载到内存中,这些核心组件包括,GPU任务代码,待处理的数据,元数据(例如,指示GPU缓存地址的指针或作业描述符)。GPU软件还会创建GPU页表,并配置相应的GPU寄存器,从而允许GPU经由直接内存访问DMA访问上述核心组件。GPU软件还会规划GPU任务的执行顺序,并通过内存映射MMIO提交GPU任务。

[0035] 假设一个很强的敌手,他控制了非安全世界和安全世界的整个软件栈,包括,GPU软件,非可信OS,虚拟机管理器hypervisor,以及安全世界中同样层次的软件。该敌手想要窥探甚至篡改GPU任务的机密数据,包括任务的输入数据,中间数据或执行结果。那么,敌手有可能访问上述统一内存,读取其中存储的机密数据,或控制能够进行DMA的外设来读取上述内存数据,从而进行攻击。此外,敌手还有可能利用内核中的漏洞获取内核权限,攻陷或控制GPU软件,从而获取任务的核心组件信息,或通过改变机密任务的执行顺序,修改GPU寄存器状态等发起篡改攻击。

[0036] 为解决GPU安全运算的问题,有研究者提出使用GPU可信计算环境TEE,为GPU运行创建隔离的环境。目前,大多数GPU可信计算环境的工作部署在Intel平台上,然而,考虑到硬件架构特性的差异,难以将这些工作直接移植到Arm平台。而针对Arm平台的设计,有研究者提出了StrongBox方案和CRONUS方案,但它们均依赖于传统的Arm可信区(TrustZone)技术和安全世界的组件,无法抵御Arm机密计算架构所设想的攻击者(即安全世界也会被攻击者破坏并利用)。此外,这些方案与Arm机密计算架构的工作流程存在一定的冲突。例如StrongBox方案借助虚拟化,导致无法直接兼容于虚拟机管理器,而CRONUS方案将GPU软件完全部署于可信执行环境内部,与CCA机密计算架构的框架设计不符。

[0037] 有鉴于此,在本说明书的实施例中提出一种方案,基于Arm机密计算架构的硬件特性,在不影响Arm机密计算架构原有功能设计的基础上,提供对GPU计算任务的安全保护,从而支持GPU上的机密计算。

[0038] 图3示出根据一个实施例在机密计算架构中运行GPU任务的示意图。图3所示的系统架构符合Arm机密计算架构CCA,其中,非安全世界中运行有主机端,主机端包含GPU驱动和其他DMA外设驱动。虚拟机管理器hypervisor创建若干机密计算“领域”,并对其进行管理和调度。而在机密计算架构CCA新引入的领域世界中,部署领域管理监控器RMM来实现不同“领域”的内存隔离。根世界部署具有最高权限的根监视器,用于管理世界之间的隔离和切换,以及提供秘钥管理、远程验证等安全认证机制。根监视器可以实现为安全固件的形式。

[0039] 在以上CCA架构中,认为领域世界的领域管理监控器RMM和根世界为完全可信,这是因为这些组件所需的内存和代码量非常小,因此暴露的攻击面较小,不易受到攻击。此外,其他组件则被视为不可信,包括安全世界的软件。

[0040] 在本说明书实施例的方案中,为了实现GPU机密计算,在根世界的根监视器中引入两个组件:影子任务组件和GPU环境保护组件,其中,影子任务组件用于实现各实施例中的影子任务机制,GPU环境保护组件则用于保护GPU运行环境不受攻击。此外,本实施例的方案中,GPU软件(包括GPU驱动和相关函数库)仍然运行在非安全世界的主机端,但是需要进行少量修改,以协助完成影子任务机制。

[0041] 影子任务机制是为了使得GPU工作流程兼容于Arm机密计算架构所引入的新机制,其核心思想是,允许主机端中GPU软件创建和管理一些桩应用程序,例如分配内存、创建GPU缓冲区、调度和提交任务等。这些桩应用程序,例如图3中的桩应用1,桩应用2,其数据结构与普通GPU任务相似,例如包含GPU页表,元数据和GPU缓冲区,但这些桩应用程序的GPU缓冲区内并不包含真实的待处理数据,仅提供对GPU缓冲区的描述。GPU软件可以照常提交这些桩应用程序,与常规不同的是,在其提交时,位于根世界的根监视器将桩应用程序替换为对应的真实GPU应用。真实GPU应用具有与桩应用类似的数据结构,但是其中填入有真实待处理的机密数据。根监视器将真实GPU应用最终提交给GPU进行运算。由此,本实施例的方案允许非安全世界在不访问真实机密数据的前提下,调度和管理来自不同领域的GPU任务,符合Arm机密计算架构的设想。

[0042] 下面结合单个GPU应用,或称为GPU任务,描述通过影子任务机制,调度和执行GPU计算任务并为其提供机密隔离环境的过程。

[0043] 图4示出根据一个实施例在机密计算架构中执行GPU任务的方法;图5示出影子任务机制下各个阶段数据结构的示意图。可以理解,图4和图5中的方法和影子任务机制,基于图3所示的机密计算架构执行。

[0044] 首先在初始化阶段或预备阶段,用户可以申请一个领域,并通过加密信道将需要GPU任务处理的真实数据传入其中。具体的,非安全世界的hypervisor可以根据用户的请求,创建一个虚拟机,通过与领域管理监控器RMM交互,将其部署到领域世界中作为一个机密领域。为了描述的方便(以及必要时区分于其他具体领域),下文中,将该用户申请的领域称为第一领域。在创建第一领域之后,用户可以与该第一领域进行密钥协商,以建立安全信道。具体的,用户可以通过DH(Diffie-Hellman)协议,基于椭圆曲线的DH协议,或其他各种协议,与第一领域交换密钥,从而协商出加密密钥。于是,基于协商出的密钥,二者可以建立一个安全的加密信道。通过该安全加密信道,第一领域可以接收用户传输的机密数据,并将其存储于领域世界。

[0045] 另一方面,用户将待执行的GPU任务(下文称为第一GPU任务)的另外两项核心组件:任务代码和缓存描述,提供给非安全世界的主机端。其中,缓存描述用于示出对GPU数据缓存的要求和描述,其中包括期望GPU数据缓存具有的属性,例如,缓存区的数量和大小,各缓存区应该存储的数据属性和类型(例如,输入数据或结果数据,数据的类型和大小等)。为了防止GPU软件被攻击后篡改任务代码,在一个实施例中,用户还提供签名信息,即对任务代码和缓存描述进行签名,将签名附加在传输的任务代码和缓存描述之后。

[0046] 如图5所示,根据上述初始化阶段,第一领域获得用户提供的机密数据/真实数据,

将其存储于受保护的领域世界(灰色示出)。主机端获得用户提供的两项核心组件(任务代码和缓存描述),将其存储于非安全世界对应的未保护区域(白色示出)。

[0047] 在主机端获得上述两项核心组件的基础上,非安全世界的主机端中的GPU软件,就可以基于这两项核心组件创建桩任务。如前所述,GPU软件主要包括GPU驱动软件,也包括与之相关的一些函数库,例如用户层运行时函数库(比如OpenCL库)。该GPU软件经过修改,基于影子任务机制,在非安全世界创建桩应用。

[0048] 具体的,如图4中步骤S41所示,非安全世界的GPU软件根据用户提供的第一GPU任务的任务代码和缓存描述,在内存的非安全世界区段中,配置第一桩任务的桩数据结构,其中包括,根据所述缓存描述创建的多个缓存区,以及指示缓存区的元数据。

[0049] 具体的,GPU创建第一桩任务,配置桩数据结构可以包括以下步骤。如前所述,缓存描述指示了期望GPU数据缓存的数量、大小、填充数据等特点。根据这样的缓存描述,GPU软件可以在内存的非安全世界区段中,分配相应的内存空间,在其中创建代码缓存区,和符合缓存描述要求的若干桩数据缓存区。在一些实施例中,缓存描述会要求创建多个数据缓存区,例如其中之一用于存储输入数据,另一个用于存储执行结果。可选的,有时缓存描述还会指示创建存储中间结果的数据缓存区。GPU软件根据缓存描述的要求,对应分配这些数据缓存区作为桩数据缓存区。

[0050] 与常规处理不同的,GPU软件仅将任务代码存储于代码缓存区,但并不在桩数据缓存区填充真实数据。也就是说,桩数据缓存区中至多存储该缓存区应存入数据的描述信息,并不存储真实的待处理数据。基于如此分配的多个缓存区,GPU软件生成元数据,用于指示该多个缓存区。具体的,该元数据中可以包括一些指针,指向各个缓存区的地址。GPU软件将该元数据也存储于内存的非安全区段中。

[0051] 此外,GPU软件还根据分配的内存,生成用于执行该第一桩任务的GPU页表,可以称为桩GPU页表。该页表中记录有GPU任务执行过程中虚拟地址与内存物理地址的映射。初始地,GPU内存映射MMIO(Memory-Mapped Input/Output)被配置为指向该桩GPU页表。具体地,GPU中具有页表基地址寄存器TTBR,用于存储页表的基地址。通过GPU内存映射MMIO,将TTBR映射为内存中的地址。初始地,通过GPU MMIO,页表基地址指向存储所述桩GPU页表的内存地址,即指向该桩GPU页表。

[0052] 如此,GPU软件创建了第一桩任务,为其配置了桩数据结构。如图5所示例的,在非安全世界的内存区段中,GPU软件分配了数据缓冲区1,数据缓冲区2和代码缓冲区,数据缓冲区1和2中仅存储对应的数据描述,代码缓冲区中存储之前获取的任务代码。根据分配的缓冲区,GPU软件生成元数据,该元数据指向上述数据缓冲区1和2,以及代码缓冲区。此外,GPU软件生成了桩GPU页表,并使得GPU内存映射指向该桩GPU页表。

[0053] 可以看到,创建桩任务的过程与创建常规任务的过程是相似的,只是并不向其中的数据缓存中填充真实数据。因此,创建的桩任务是一个不具有真实数据的“空壳”任务,但是具有与真实任务完全相同的数据结构,可以进行任务的管理和调度。

[0054] 因此,GPU软件在创建上述第一桩任务后,如常地将其插入到GPU任务队列中,安排任务的执行顺序,并经由根监视器向GPU硬件提交该第一桩任务。

[0055] 根监视器接收到第一桩任务的提交,就会在领域世界创建真实任务,即执行图4的步骤S42。在该步骤中,根监视器在内存中对应于第一领域的第一领域区段中,配置与桩数

据结构对应的真实数据结构,将用户提供的机密数据存储于其中。

[0056] 具体的,根监视器根据桩数据结构中的桩数据缓冲区创建对应大小和属性的真实数据缓冲区,并根据其中的描述填入真实数据或暂时留空。在一个实施例中,桩数据结构中具有多个数据缓冲区,其中包括用于存储输入数据的输入数据缓存区,和用于存储结果数据的结果数据缓存区。于是,根监视器在第一领域区段中对应创建真实的输入数据缓存区和结果数据缓存区,将之前用户提供的机密数据存储于创建的输入数据缓存区,将结果数据缓存区暂时留空。

[0057] 此外,根监视器还根据桩GPU页表创建真实GPU页表,存储于第一领域区段。为此,根监视器可以首先对桩GPU页表中记录的页表条目进行校验,例如检查其中是否存在重复映射或非法映射。如果校验通过,根监视器通过复制或重放其中的页表条目,构建真实GPU页表。但是,需要注意的是,由于桩缓存结构中的数据缓存区并不存储真实数据,也不参与真实GPU计算,因此,在真实GPU页表中,将与数据缓存区相关的条目修改为指向真实数据结构中的真实数据缓存区。

[0058] 延续之前的示例,如图5所示,在真实应用创建阶段,在受保护的领域区段中,创建与桩任务中的2个数据缓冲区对应的真实的数据缓冲区1和数据缓冲区2。将真实的机密数据存储于数据缓冲区1,而将数据缓冲区2暂时留空,用于存储结果数据。此外,根监视器还生成真实GPU页表,存储于该第一领域区段。

[0059] 在需要执行真实GPU任务时,根监视器首先为真实GPU任务的执行提供受保护的执行环境。具体的,根监视器通过Arm机密计算架构中提供的基于颗粒度保护检查(GPC)的内存保护机制,来保护GPU任务中核心组件的内存访问。由于真实数据缓冲区和真实GPU页表已经处于受保护的领域世界中,需要对存储于非安全世界且存储重要敏感数据的目标区段进行额外保护,该目标区段包括存储元数据和任务代码的区段。此外,GPU内存映射MMIO区段由于包含GPU页表基地址,也属于应受保护的区段。

[0060] 为此,根据GPC机制,根监视器执行步骤S43,根监视器更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界权限。

[0061] 如前所述,机密计算架构CCA在内存中维护颗粒度保护表GPT,其中记录细粒度的每段物理内存的安全状态,用于进行GPC检查,从而实施内存隔离。根据本实施例的方案,根监视器可以维护多个版本的GPT表,使得目标区段对于不同对象具有不同访问权限。

[0062] 具体地,根监视器至少维护第一版本GPT表和第二版本GPT表。第一版本GPT表用于CPU和其他外设对内存的访问,在更新后的第一版本GPT表中,上述存储元数据和任务代码的目标区段可以被设置为属于领域世界。此外,为了保护GPU内存映射MMIO区段,可以在上述第一版本GPT表中将其设置为属于根世界。根据图2所示的不同世界的权限,通过CPU或其他外设请求内存访问的各种应用,包括安全世界中的软件,均不能访问上述目标区段和GPU内存映射。

[0063] 第二版本GPT表是用于GPU的针对第一领域的GPT表,该GPT表可以在创建第一领域时生成并初始化。通过步骤S43的更新,在该GPT表中将目标区段设置为可访问区段。具体的,在一个例子中,在该更新后的第二版本GPT表中,第一领域区段和目标区段被设置为属于非安全世界,可以访问;其他区段均被设置为属于根世界,不能访问。

[0064] 如此,根监视器通过更新GPT表,确保真实GPU任务所需的内存空间隔离,不被其他

应用访问或窃取,从而为真实GPU任务的执行提供受保护的执行环境。

[0065] 较优地,在执行真实GPU任务之前,还对任务代码和缓存描述进行验证,防止被攻击的GPU软件篡改任务代码或缓存描述。具体的,如前所述,用户可以在提供任务代码和缓存描述时附上签名。于是,根监视器可以基于该签名对当前代码缓存区中存储的任务代码,和桩数据缓存区中存储的缓存描述进行验签。此外,根监视器还可以对GPU状态进行检查,确保其中没有隐藏存储恶意任务代码。

[0066] 在各项检查/验证通过之后,根监视器可以提交并启动真实GPU任务的执行。为此,在步骤S44,根监视器修改目标映射关系,使得GPU使用上述目标区段和真实数据结构执行第一GPU任务。

[0067] 具体的,根监视器修改GPU内存映射,将GPU页表基地址指向存储于第一领域中的真实GPU页表,如此,用真实GPU页表替换桩GPU页表。此外,根监视器还修改元数据的指针,使其指向真实的数据缓冲区,如此,用真实的数据缓冲区替换桩数据缓冲区。如此,GPU使用目标区段和真实数据结构执行第一GPU任务。具体的,GPU基于真实GPU页表和修改后的元数据,寻址到代码缓存区和真实数据缓存区,从而基于代码缓存区中的任务代码,对真实数据缓存区中的真实机密数据进行任务处理。

[0068] 延续之前的示例。如图5所示,为执行真实GPU任务,通过前述步骤S43,将存储元数据和任务代码的目标区段和GPU内存映射设置为受保护区域。从而,在执行真实GPU任务时,根据GPU内存映射,寻址到真实GPU页表;根据真实GPU页表和元数据,定位出代码缓存区和真实数据缓存区,于是可以基于其中存储的任务代码,处理真实机密数据。根据元数据的指示,GPU还可以将任务处理得到的运行结果,存储于真实的结果数据缓存区(即数据缓冲区2)。通过步骤S43的设置,上述过程中所使用的数据均处于受保护的内存区域中,从而确保了任务执行的机密性。

[0069] 在上述第一GPU任务执行完成后,根监视器可以恢复原有执行环境。具体的,根监视器可以首先恢复GPU内存映射MMIO中寄存器映射地址和元数据中的指针。然后,清除之前的GPU执行环境,包括刷除GPU页表对应的TLB条目等。在清除GPU执行环境后,可以通过再次更新GPT表,恢复前述目标区段的常规访问权限,即,将目标区段恢复为属于非安全世界。

[0070] 如图5所示,在GPU任务执行后的环境恢复阶段,可以清除执行过程中的数据,并将之前临时受保护的目标区段(GPU内存映射区段,元数据区段和代码缓存区段)恢复为未受保护的非安全世界。执行产生的结果数据存储于领域世界的第一领域区段,用户可以通过安全信道,从第一领域中读取该运行结果。由此,完成GPU任务的机密执行和机密计算。

[0071] 回顾以上过程,通过影子任务机制实现兼容Arm机密计算架构CCA的GPU机密计算。根据影子任务机制,由非安全世界的GPU软件创建不包含真实数据的桩任务,并如常规流程调度和管理桩任务。在桩任务提交后,根监视器创建包含真实数据的真实GPU任务,并为其提供受保护的执行环境。然后,根监视器用真实GPU任务替换桩任务,从而使得,GPU硬件在受保护的执行环境中执行真实GPU任务。如此,在Arm机密计算架构CCA中实现GPU机密计算。

[0072] 如前所述,GPU安全执行环境的提供通过维护和更新多个版本的GPT表来实现。维护多个版本的GPT表可能对整个架构的执行性能带来一些压力。在一些实施方式中,还提出针对GPT表维护方式的优化,以进一步提升系统性能。

[0073] 如之前结合步骤S43所述,为了针对第一GPU任务提供安全的执行环境,根监视器

至少需要维护两个版本的GPT,其中第一版本GPT表用于CPU和其他外设对内存的访问控制,第二版本GPT表用于GPU执行第一领域任务。实际上,虽然其他外设和CPU关于第一GPU任务对应的目标区段具有相同的访问权限设置,但是不同外设对于内存的其他区段仍具有其独有的、不同于CPU的权限设置。此外,领域世界中可能具有多个互相隔离的领域,支持多个GPU任务的机密执行。此时,还需要针对其他领域维护其对应的GPT表。因此实际上,根监视器可能需要维护显著多于2个版本的GPT表。

[0074] 图6示出在一个示例场景中,根监视器维护的GPT表。如图6所示,在该示例场景中,领域世界中至少包括领域R1和领域R2。假定两个领域各自的用户分别要求,基于领域R1执行GPU任务1,基于领域2执行GPU任务2。于是,GPU软件在原本属于非安全世界的区段中创建R1桩任务,和R2桩任务。将R1桩任务中存储其对应的元数据和任务代码的区段称为目标区段1,将R2桩任务中存储其对应的元数据和任务代码的区段称为目标区段2。

[0075] 为了给GPU任务1和GPU任务2分别提供隔离的执行环境,根监视器至少要维护图6所示的4个GPT表。

[0076] 在针对CPU的GPT表中,领域R1和领域R2对应的内存区段照常属于领域世界区段。此外,还将原本属于非安全世界的上述目标区段1和目标区段2设置为领域世界区段。此外,GPU内存映射MMIO区段被设置为属于根世界。

[0077] 针对不可信外设的GPT表与针对CPU的GPT表总体相似,其中关于领域R1,领域R2,目标区段1和目标区段2的设置完全相同。不同的是,外设具有其对应的内存访问限制,CPU可以访问的部分内存区段(例如图中最前面的一小段),在外设的GPT表被设置为属于根世界,外设无权访问。

[0078] 针对领域1的GPU GPT表,是GPU执行领域1对应的GPU任务1时适用的GPT表。在该表中,领域R1和目标区段1被设置为属于非安全世界,可以访问;所有其他区段均被设置为属于根世界,不能访问。这意味着,在GPU执行领域1对应的GPU任务1时,仅仅可以访问领域R1和目标区段1的内存数据,无权访问任何其他区段数据。

[0079] 针对领域2的GPU GPT表,是GPU执行领域2对应的GPU任务2时适用的GPT表。在该表中,领域R2和目标区段2被设置为属于非安全世界,可以访问;所有其他区段均被设置为属于根世界,不能访问。这意味着,在GPU执行领域2对应的GPU任务2时,仅仅可以访问领域R2和目标区段2的内存数据,无权访问任何其他区段数据。

[0080] 当硬件(CPU,GPU或外设)请求访问内存时,硬件层中的RME根据对应适用的GPT表进行GPC检查,从而进行内存访问控制。

[0081] 通过针对领域1和领域2的两个GPU GPT表可见,不同领域的GPU任务之间,也进行了内存隔离,确保执行环境的安全。可以理解,如果需要基于更多领域执行更多GPU任务,则需要维护更多的GPT表。

[0082] 为了减轻维护多个GPT表的负担,在一个实施例中,采用分层表的方式维护GPT表。具体的,GPT表支持由顶层表(top-level)和次层表(sub-level)构成的分层表架构,其中,次层表中的描述符仅用于描述内存区段的安全属性,而不必描述地址,读写权限和其他属性。基于这样的特点,可以构建CPU和各个外设共享的次层表。具体的,预先划定一段内存区域,构建一个次层表来管理这段内存区域的访问控制。GPU软件被设置为,在这段预定内存区域中创建桩任务。进一步的,配置CPU和各个外设的GPT表,使其表描述符均指向该统一的

次层表。在使用时,只需要在不同阶段灵活地修改次层表中的安全属性,就可以设定是否保护存储元数据和任务代码的目标区段。而GPU和各个外设的GPT表,由于均指向该次层表,就可以共享该次层表中的安全设置。

[0083] 图7示出根据一个实施例的次层表示意图。如图7所示,将预定内存区段划定为GPU内存,针对这段内存建立次层表,其中的页描述符指示对应页的安全属性(属于哪个世界)。CPU和各个外设的GPT表,均通过表描述符指向该次层表。每当GPU软件创建桩应用时,在该段GPU内存区段中创建桩数据结构,并在需要保护元数据和任务代码时,修改次层表中对应部分的安全属性。由于该次层表被多个GPT表共享,上述修改即同步到各个GPT表,从而简化多个GPT表的维护和相同内容的同步。

[0084] 关于针对多个领域的多个GPT表,例如图6中针对领域1的GPU GPT表和针对领域2的GPU GPT表,其具有相似的模式:仅包含非安全世界(可访问)和根世界(不可访问)两种世界属性。因此,可以基于同一模板生成不同领域的GPT表,根据该模板,首先将整个内存全都设置为根世界,然后将对应领域区段设置为非安全世界(normal)。在进行GPU机密计算时,附加地将本次GPU任务对应的目标区段(存储元数据和任务代码的区段)设置为非安全世界,从而允许GPU对其进行访问。

[0085] 通过以上方式,可以优化GPT表的维护和更新,进一步提升GPU任务的执行性能。

[0086] 另一方面,与上述方法过程相对应的,本说明书实施例还披露一种机密计算架构中的根监视器,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根监视器位于所述根世界中。根监视器可以包括影子任务组件和GPU环境保护组件。

[0087] 影子任务组件配置为,响应于非安全世界的GPU软件在内存的非安全世界区段中配置第一桩任务的桩数据结构,而在所述内存中对应于第一领域的区段中,配置与所述桩数据结构对应的真实数据结构,并将待处理的机密数据存储于其中,其中所述桩数据结构包括,根据用户提供的缓存描述分配的多个缓存区,以及指示各缓存区的元数据。

[0088] GPU环境保护组件配置为,更新颗粒度保护表GPT,使得根据更新后的GPT表,目标区段对于GPU可访问,对于其他对象均具有领域世界的权限,其中所述目标区段存储所述元数据和用户提供的第一GPU任务的任务代码。

[0089] 影子任务组件还配置为,修改目标映射关系,使得GPU使用所述目标区段和所述真实数据结构执行所述第一GPU任务。

[0090] 影子任务组件和GPU环境保护组件的具体执行过程示例,可以参照之前结合图4和图5的描述,不复赘述。

[0091] 在典型实施例中,所述根监视器实现为安全固件。

[0092] 根据再一方面的实施例,还提供一种计算设备,包括存储器和若干处理器,所述计算设备形成机密计算架构,所述机密计算架构包括,安全世界,领域世界,非安全世界和根世界;所述根世界包括前述的根监视器。

[0093] 本领域技术人员应该可以意识到,在上述一个或多个示例中,本发明所描述的功能可以用硬件、软件、固件或它们的任意组合来实现。当使用软件实现时,可以将这些功能存储在计算机可读介质中或者作为计算机可读介质上的一个或多个指令或代码进行传输。

[0094] 以上所述的具体实施方式,对本发明的目的、技术方案和有益效果进行了进一步详细说明,所应理解的是,以上所述仅为本发明的具体实施方式而已,并不用于限定本发明

的保护范围,凡在本发明的技术方案的基础之上,所做的任何修改、等同替换、改进等,均应包括在本发明的保护范围之内。

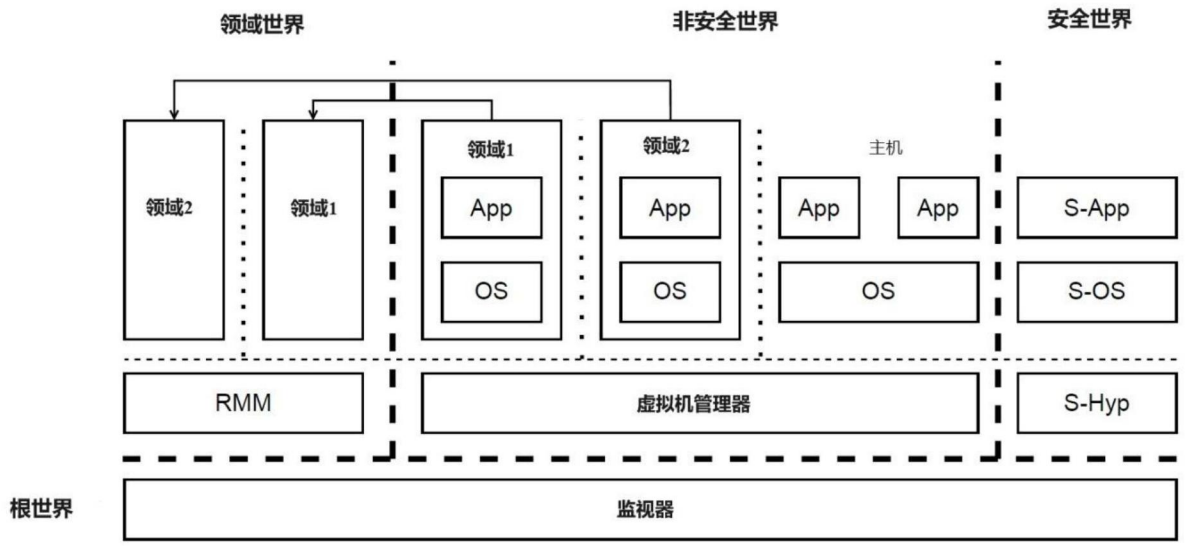


图1

安全状态	非安全 PAS	安全 PAS	领域 PAS	根 PAS
非安全	✓	×	×	×
安全	✓	✓	×	×
领域	✓	×	✓	×
根	✓	✓	✓	✓

图2

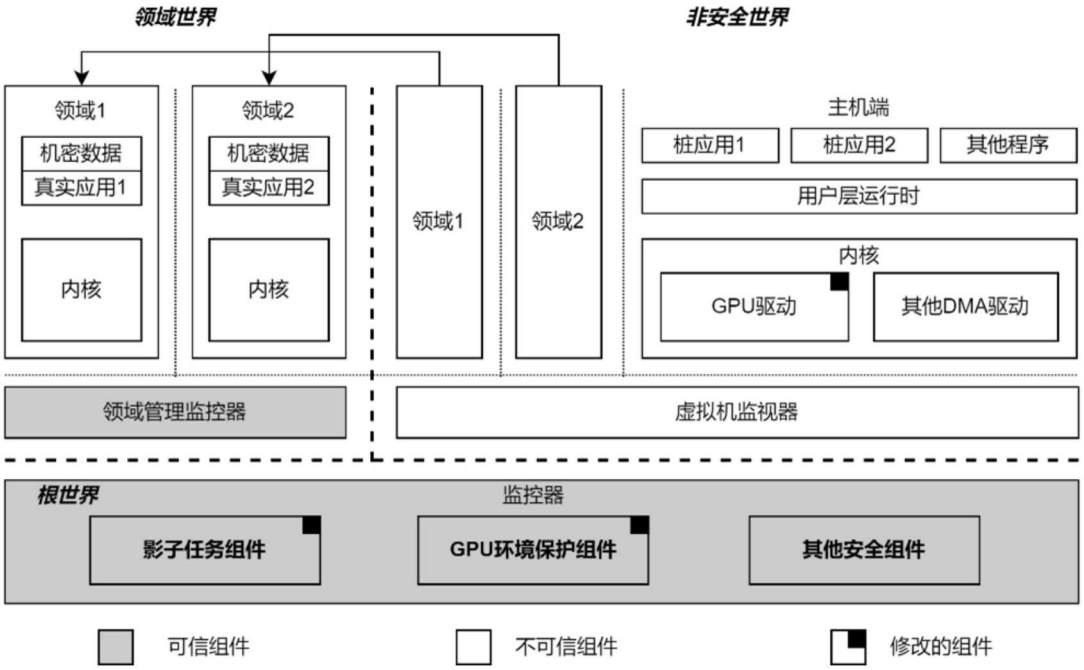

领域世界
非安全世界
领域1
机密数据
真实应用1
内核
领域2
机密数据
真实应用2
内核
领域1
领域2
主机端
桩应用1
桩应用2
其他程序
用户层运行时
内核
GPU驱动
其他DMA驱动
领域管理监控器
虚拟机监视器
根世界
监控器
影子任务组件
GPU环境保护组件
其他安全组件
可信组件
不可信组件
修改的组件

图3

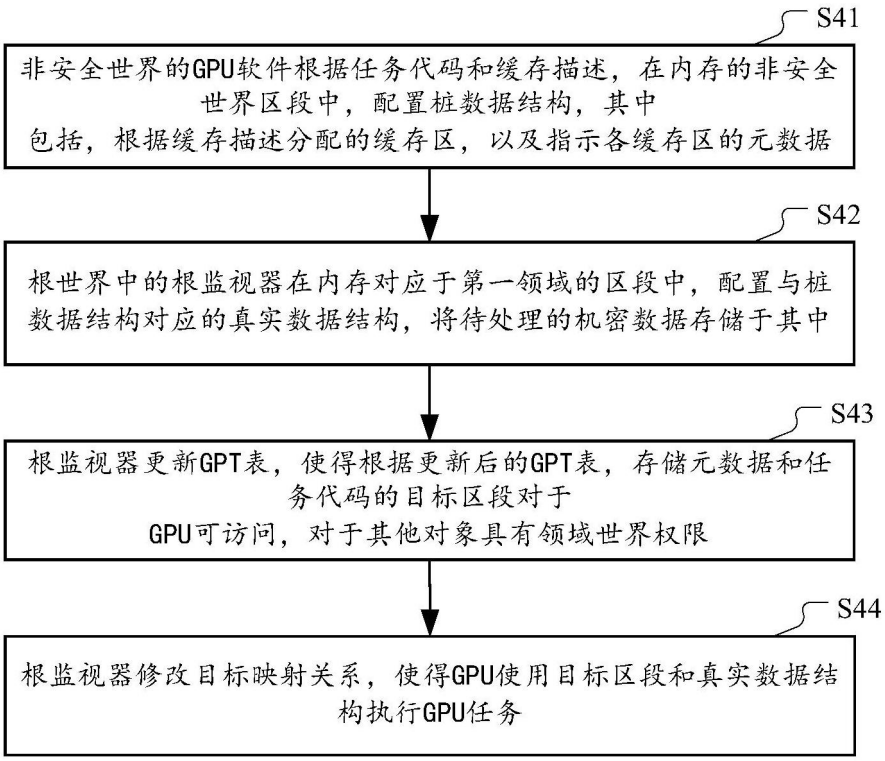

S41
非安全世界的GPU软件根据任务代码和缓存描述，在内存的非安全世界区段中，配置桩数据结构，其中
包括，根据缓存描述分配的缓存区，以及指示各缓存区的元数据
S42
根世界中的根监视器在内存对应于第一领域的区段中，配置与桩数据结构对应的真实数据结构，将待处理的机密数据存储于其中
S43
根监视器更新GPT表，使得根据更新后的GPT表，存储元数据和任务代码的目标区段对于
GPU可访问，对于其他对象具有领域世界权限
S44
根监视器修改目标映射关系，使得GPU使用目标区段和真实数据结构执行GPU任务

图4

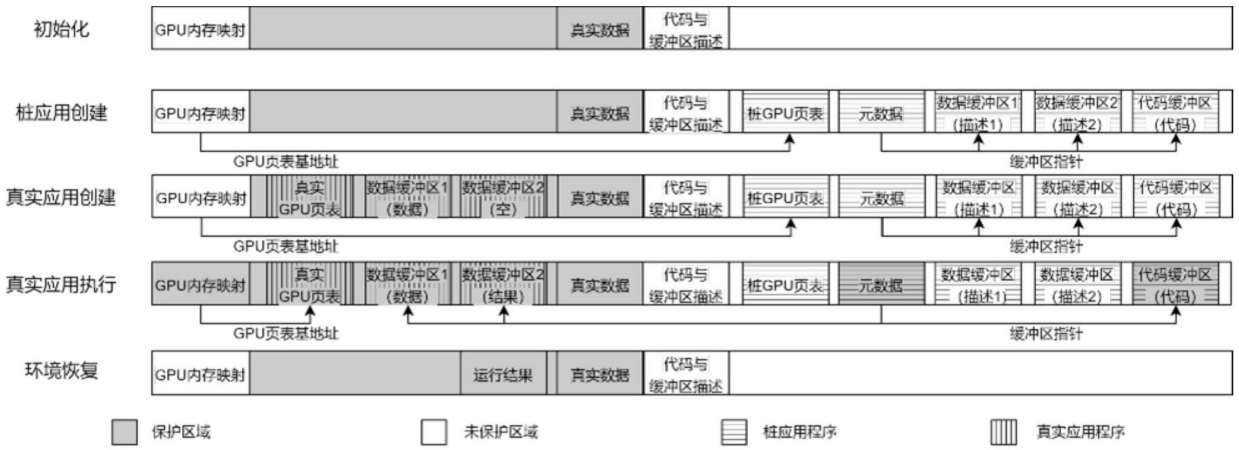

初始化
GPU内存映射
真实数据
代码与缓冲区描述
桩应用创建
桩GPU页表
元数据
数据缓冲区1（描述1）
数据缓冲区2（描述2）
代码缓冲区（代码）
GPU页表基地址
缓冲区指针
真实应用创建
真实GPU页表
数据缓冲区1（数据）
数据缓冲区2（空）
真实应用执行
数据缓冲区2（结果）
环境恢复
运行结果
保护区域
未保护区域
桩应用程序
真实应用程序

图5

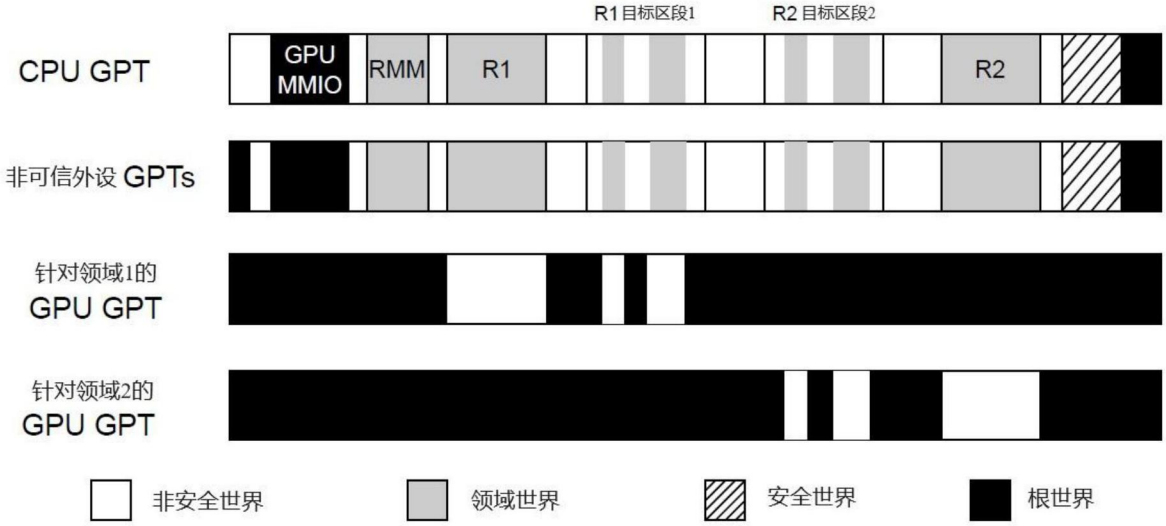

R1 目标区段1
R2 目标区段2
CPU GPT
GPU MMIO
RMM
R1
R2
非可信外设 GPTs
针对领域1的 GPU GPT
针对领域2的 GPU GPT
非安全世界
领域世界
安全世界
根世界

图6

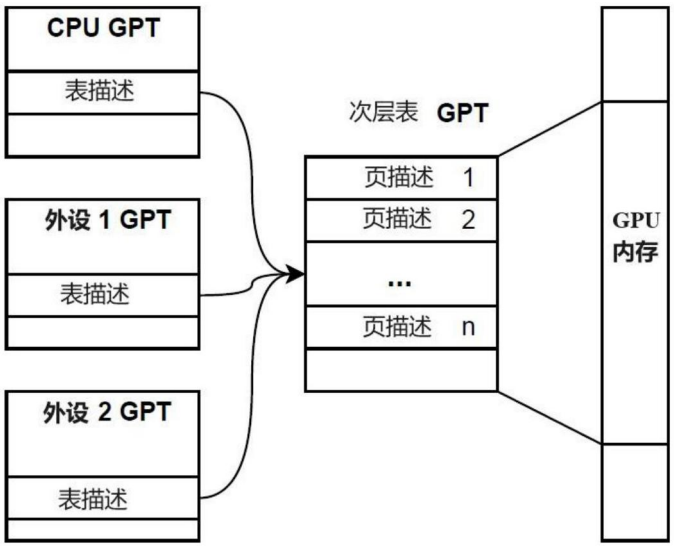

CPU GPT
表描述
外设 1 GPT
表描述
外设 2 GPT
表描述
次层表 GPT
页描述 1
页描述 2
...
页描述 n
GPU
内存

图7