中国科学院软件研究所

课程文档

|  |  |  |
| --- | --- | --- |
| 课程名称 | ： | 《openEuler应用编程技术》 |
| 教师姓名 | ： |  |
| 提交时间 | ： |  |

中国科学院软件研究所智能软件研究中心制

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程  名称 | 《Linux应用编程技术》 | | 章 | | 第九章 | | | 课程类型 |
| 节 | | 第一节 | | | 授课(√ )  实训( ) |
| 名称 | Kunpeng及Arm架构加速器及指令介绍 | | | | | | |
| 教师 |  | 课时 | | | | 课时 90分钟 | | |
| 参考  资料 | 《鲲鹏加速引擎 技术白皮书》  《Armv8.4 MPAM介绍及实践》 | | | | | | | |
| 教学  目的  要求 |  | | | | | | | |
| 教学  重点  难点 |  | | | PPT页面 | | | 时间分配 | |
| 教学重点 | | | | | | | |
| 1. 鲲鹏加速引擎 | | |  | | |  | |
| 1. Arm架构加速器 | | |  | | |  | |
| 教学难点 | | | | | | | |
| 1. 虚拟化场景逻辑架构 | | |  | | |  | |
| 1. 基于OpenSSL的加速引擎 | | |  | | |  | |
| 1. 多条数据流共享同一个DDR访存通道 | | |  | | |  | |

|  |  |  |
| --- | --- | --- |
| 教学方法 | | |
| 以课堂讲授为主，图片与电子教案相结合 | | |
| 讲稿 | 操作演示 | 知识点 |
| 第1页  本讲介绍Kunpeng及Arm架构加速器及指令 |  |  |
| 第2页  内容包括鲲鹏加速引擎和Arm架构加速器两个部分，首先介绍鲲鹏加速引擎。 |  | 目录 |
| 第3页  鲲鹏加速引擎是基于鲲鹏920处理器的硬件加速模块，包含了对称加密、非对称加密和摘要，压缩解压缩等算法，用于加速SSL/TLS应用和数据压缩，可以显著提升所支持算法的运算效率，节省CPU资源。此外，鲲鹏加速引擎对应用层屏蔽了其内部实现细节，用户通过OpenSSL、Zlib标准接口即可以实现对鲲鹏加速引擎的调用。目前鲲鹏加速引擎主要支持以下算法：  1. 摘要算法SM3/MD5，支持同步异步模式。  2.对称加密算法，包括：SM4，支持同步异步模式，支持CTR/XTS/CBC/OFB/ECB分组模式（XTS模式仅支持内核态应用）；  3.AES，支持同步异步模式，支持ECB/CBC/CTR/XTS分组模式。  4.非对称加密算法RSA，支持同步异步模式，支持 Key Sizes 1024/2048/3072/4096。  5.密钥协商算法DH, 支持同步异步模式，支持 Key Sizes 768/1024/1536/2048/3072/4096。  6.压缩解压算法，支持GZIP/ZLIB两种数据格式。 非对称加密算法RSA，支持同步异步模式，支持 Key Sizes 1024/2048/3072/4096。 密钥协商算法DH, 支持同步异步模式，支持 Key Sizes 768/1024/1536/2048/3072/4096。 压缩解压算法，支持GZIP/ZLIB两种数据格式。。 |  | 鲲鹏加速引擎简介 |
| 第4页  鲲鹏加速引擎为开发者提供一套可编程的接口，用户可通过此套接口调用硬件加速引擎的能力，从而帮助上层集成应用提升业务性能。主要价值点如下：  1.接口标准化  适配OpenSSL、Zlib等开源标准库，采用标准开源库接口，应用层可以不需要做代码级适配修改，如：针对OpenSSL的应用只需要在初始化时指定KAE引擎，其它接口不用修改。  2.卓越性能  使用KAE引擎后，性能相比软件运算会有大幅提升，如：OpenSSL场景下SM4加解密处理最高可获得4倍以上的性能提升（提升幅度跟包长大小有关系，一般包长越长性能收益越明显，包长小的场景不建议用KAE引擎，具体建议详见4 产品规格章节描述）。  3.源码开放  鲲鹏加速引擎源码包括中间层适配库和驱动两部分，所有源码都可公开获得，开发者用户可以根据自身业务需要对代码进行修改或裁剪，其中驱动源码遵循GPL 2.0，应用加速库遵循Apache 2.0开源许可协议。 |  | 鲲鹏加速引擎主要价值点 |
| 第5页  上下文架构  加速引擎是基于Kunpeng 920芯片的硬件加速解决方案。其上下文架构如下图所示。  1.芯片加速器子系统、BIOS子系统和BMC子系统为服务器整机产品自带子系统。  2.加速器驱动子系统，向上层提供各子加速器模块统一的驱动接口，是本系统的核心子系统。  3.应用库子系统（openssl/zlib），向上层应用提供标准接口。  4.应用系统（APPs），指上层应用系统，包括大数据应用、Web应用等，属于用户层面系统。 芯片加速器子系统、BIOS子系统和BMC子系统为服务器整机产品自带子系统。 加速器驱动子系统，向上层提供各子加速器模块统一的驱动接口，是本系统的核心子系统。 应用库子系统（openssl/zlib），向上层应用提供标准接口 应用系统（APPs），指上层应用 |  | 上下文架构 |
| 第4页  鲲鹏加速引擎上下文架构组件说明   | 子系统名称 | 功能描述 | | --- | --- | | 芯片加速器子系统 | 集成在芯片中，提供硬件加速器功能。该子系统是加速器的硬件实现，不直接开放给客户。 | | BIOS子系统 | 单板BIOS软件系统，负责解析License信息，并根据解析结果对加速器模块进行使能控制，并上报加速器ACPI表到内核（加速器驱动子系统处理）。 | | BMC子系统 | 服务器BMC软件系统，支持加速器子系统的License管理。 | | 加速器驱动子系统 | 本系统的核心，向上层提供各子加速器模块统一的驱动接口。 | | 应用库子系统 | 应用库子系统包括OpenSSL加速器引擎、Zlib替代库等，向上层提供标准接口。 | | 应用系统 | 用户系统，通过调用应用库子系统或驱动子系统实现加速器的调用。 | |  |  |
| 第6页  软件逻辑架构  1.灰色背景的模块，为开源模块或用户模块  2.蓝色背景部分，为加速器驱动子系统，包括各加速器内核态驱动模块和用户态Warpdrive模块。  3.绿色背景部分，为应用库子系统，包括OpenSSL加速引擎、Zlib替代库。 |  | 软件逻辑架构 |
| 第7页  鲲鹏加速器软件各模块职责   | 功能模块名称 | 职责 | | --- | --- | | HPRE | 对应HPRE加速器模块内核态驱动 | | ZIP | 对应ZIP加速器模块内核态驱动 | | SEC | 对应SEC加速器模块内核态驱动 | | RDE | 对应RDE加速器模块内核态驱动 | | UACCE | User space Accelerator，内核驱动框架模块，为用户态提供统一驱动接口 | | Warpdrive | 用户态驱动统一接口 | | Engine of Openssl | OpenSSL的加速引擎，将SEC/HPRE模块的加速器适配到OpenSSL | | Zlib Adapter | Zlib适配模块，将zip加速器适配Zlib库 | | UIO | Linux系统上的一种运行在用户空间的IO技术 | |  | 鲲鹏加速器软件各模块 |
| 第8页  虚拟化场景逻辑架构  针对虚拟化场景，硬件加速器支持SR-IOV，可通过VF直通方式在虚拟机中使用加速器硬件，支持SR-IOV的加速模块包括：  - SEC：Security Engine，安全加速引擎。  - HPRE：High Performance RSA Engine，高性能RSA加速引擎。  - ZIP：压缩加速引擎。  以上每个加速器支持1个PF，63个VF。  每个设备有1024个硬件队列，虚拟化场景下队列划分规则：  1.PF默认256个队列，在加载驱动时用户可以设置调整。  2.单VF队列数 =（设备总队列数 - PF队列数）/ VF数量。其中余数队列放最后一个VF。  3.VF的数量通过/sys/devices/下的sriov\_numvfs设置。 |  | 虚拟化场景逻辑架构 |
| 第9页  VM场景逻辑架构  此场景为用户只在VM上使用加速器能力，配置约束如下：  1.需打开SMMU（通过BIOS设置），以支持虚拟化能力。  2.HostOS为每个设备创建指定个数VF，VF个数由业务需求指定。  3.VM关联直通VF，每个VM上可以关联1个VF或多个VF，具体多少个由业务决定。  4.VM关联时最好选择与VM在同一个CPU的VF进行关联。 |  | VM场景逻辑架构 |
| 第10页  HostOS需加载的软件包括：  1.驱动包，只加载需要的模块驱动（如只需要用RSA运算，则只加载HPRE驱动及公共驱动模块即可）。  VM需加载的软件包括：  1.驱动包，是HostOS加载的驱动包子集，与关联的VF配套。  2.Warpdriver用户态驱动包。  3.Openssl/Zlib上层应用库。 |  |  |
| 第11页  容器场景逻辑架构  此场景为用户只在容器上使用加速器能力，配置约束如下：  1.需打开SMMU（通过BIOS设置），以支持虚拟化能力。  2.HostOS为每个设备创建指定个数VF，VF个数由业务需求指定。  3.容器关联直通VF，每个容器上可以关联1个VF或多个VF，具体多少个由业务决定。  4.容器关联时最好选择与容器在同一个CPU的VF进行关联。 |  | 容器场景逻辑架构 |
| 第12页  HostOS需加载的软件包括：  1.驱动包，只加载需要的模块驱动（如只需要用RSA运算，则只加载HPRE驱动及公共驱动模块即可）。  容器需加载的软件包括：  1.驱动包，是HostOS加载的驱动包子集，与关联的VF配套。  2.Warpdriver用户态驱动包。  3.Openssl/Zlib上层应用库。 |  |  |
| 第13页  VM+容器场景逻辑架构  此场景为用户在VM上创建容器，在容器中使用加速器能力，配置约束如下：  1.需打开SMMU（通过BIOS设置），以支持虚拟化能力。  2.HostOS为每个设备创建指定个数VF，VF个数由业务需求指定。  3.VM关联直通VF，每个VM上可以关联1个VF或多个VF，具体多少个由业务决定。  4.VM关联时最好选择与VM在同一个CPU的VF进行关联。  5.容器中关联的VF需要在VM关联的范围内。  HostOS需加载的软件包括：  1.驱动包，只加载需要的模块驱动（如只需要用RSA运算，则只加载HPRE驱动及公共驱动模块即可）。  VM需加载的软件包括：  1.驱动包，是HostOS加载的驱动包子集，与关联的VF配套。  容器需加载的软件包括：  1.驱动包，是VM加载的驱动包子集，与关联的VF配套。  2.Warpdriver用户态驱动包。  3.Openssl/Zlib上层应用库。 |  | VM+容器场景逻辑架构 |
| 第14页  Host+VM场景逻辑架构  此场景为用户在Host和VM上同时使用加速器能力，配置约束如下：  1.需打开SMMU（通过BIOS设置），以支持虚拟化能力。  2.HostOS设置iommu.passthrough=1，当前在打开SMMU的情况下，只能设置为passthrough模式。  3.HostOS为每个设备创建指定个数VF，VF个数由业务需求指定。  4.VM关联直通VF，每个VM上可以关联1个VF或多个VF，具体多少个由业务决定。  5.VM关联时最好选择与VM在同一个CPU的VF进行关联。  HostOS需加载的软件包括：  1.驱动包，只加载需要的模块驱动（如只需要用RSA运算，则只加载HPRE驱动及公共驱动模块即可）。  2.Warpdrive用户态驱动包。  3.Openssl/Zlib上层应用库。  VM需加载的软件包括：  1.驱动包，是HostOS加载的驱动包子集，与关联的VF配套。  2.Warpdrive用户态驱动包。  3.Openssl/Zlib上层应用库。 |  | Host+VM场景逻辑架构 |
| 第15页  鲲鹏加速引擎技术规格 |  | 鲲鹏加速引擎技术规格 |
| 第16页 |  | 鲲鹏加速引擎技术规格 |
| 第17页 |  | 鲲鹏加速引擎技术规格 |
|  |  | 鲲鹏加速引擎技术规格 |
| 第18页  兼容性 |  | 兼容性 |
| 第19页  性能指标 |  | 性能指标 |
| 第20页  各种签名算法性能比较  1.RSA同步签名性能 |  | RSA同步签名性能 |
| 2.RSA异步签名性能 |  | RSA异步签名性能 |
| 3.RSA同步验签性能 |  | RSA同步验签性能 |
| 4.RSA异步验签性能 |  | RSA异步验签性能 |
| 第21页  特性描述  1. 基于OpenSSL的加速引擎  2. 基于Zlib的压缩、解压  3. 基于Linux内核态的应用 |  | Kunpeng特性描述 |
| 第22页  基于OpenSSL的加速引擎  通过OpenSSL的第三方引擎，对接Kunpeng加速器的用户态驱动，从而让上层应用可以操作加速硬件。具体的调用栈如图所示。  业务层通过OpenSSL的SSL API或者EVP API调用KAE Engine的Engine API，继而调用加速器的用户态驱动Warp Driver API，从而操控加速硬件。  KAE加速引擎实现了部分对称加密、非对称加密和摘要等算法，主要支持如下算法：  - 国密算法SM3、SM4  - 对称加密算法AES  - 非对称算法RSA  - 密钥协商算法DH  - 摘要算法MD5 |  | 基于OpenSSL的加速引擎 |
| 第23页  国密特性  1. SM3算法  2. SM4算法 |  | 国密特性 |
| 第24页  SM3算法  SM3是杂凑算法，适用于商用密码应用中的数字签名和验证消息认证码的生成与验证以及随机数的生成，可满足多种密码应用的安全需求。使用加速器引擎执行SM3算法业务流程如上图。 |  | SM3算法 |
| 第25页  OpenSSL摘要接口  应用层通过引擎的ID获取KAE引擎，然后通过OpenSSL EVP标准摘要算法接口进行运算，主要OpenSSL摘要接口如下： |  | SM3 OpenSSL摘要接口 |
| 第26页  SM4算法  SM4是分组密码算法，将明文数据按固定长度进行分组，然后在同一密钥控制下逐组进行加密，从而将各个明文分组变换成一个等长的密文分组的密码，适用数据表加密、大数据等场景。KAE引擎支持SM4三种分组模式：CBC/CTR/XTS。使用加速器引擎执行SM4算法业务流程如上图。 |  | SM4算法 |
| 第27页  应用层通过引擎的ID获取KAE引擎，然后通过OpenSSL EVP标准算法接口进行加解密，主要OpenSSL接口如下表 |  | SM4 OpenSSL摘要接口 |
| 第28页  RSA特性  RSA是一种非对称加密算法，普遍认为是目前最优秀的公钥方案之一。KAE引擎支持同步模式和异步模式的RSA算法。同步流程和SM4算法类似，不再赘述，异步的业务流程如下图。业务层使用使用开源的Asynch Nginx或者Tengine，支持OpenSSL层的异步功能。OpenSSL 1.1.x提供了ASYNC的功能用于异步，KAE Engine通过动态加载的方式注册到OpenSSL Engines里，同时开启一个后台线程。上层应用通过ASYNC机制发起一个异步任务，该任务具体执行一个RSA算法，比如签名，上层发起一个异步任务后立刻返回得到一个需要等待的异步结果的错误状态以及一个文件描述符，KAE后台轮询线程查看有没有异步任务有没有返回结果，然后告知RSA API任务完成，并通知上层来取结果。 |  | RSA特性 |
| 第29页  OpenSSL EVP标准算法接口进行RSA加解密，主要接口如下 |  | OpenSSL EVP标准算法接口 |
| 第30页  AES特性  高级加密标准（AES，Advanced Encryption Standard）为最常见的对称加密算法，KAE引擎支持ECB、CBC、CTR、XTS 4种工作模式，其业务流程和SM4类似如下 |  | AES特性 |
| 第31页  使用OpenSSL EVP标准算法接口进行AES加解密，主要OpenSSL接口如下 |  | AES OpenSSL接口 |
| 第32页  DH特性  DH（Diffie–Hellman key exchange） 是一种安全协议。它可以让双方在完全没有对方任何预先信息的条件下通过不安全信道创建起一个密钥。这个密钥可以在后续的通讯中作为对称密钥来加密通讯内容。其流程如下图 |  | DH特性 |
| 第33页  基于openSSL使用DH硬算接口 |  | DH基于openSSL使用DH硬算接口 |
| 第34页  MD5特性  MD5是一种消息摘要算法（Message-Digest Algorithm），可以生成128位散列值，常用于信息完整性校验等场景。但因为MD5算法无法防止碰撞攻击，因此不适用于安全性认证，如SSL公开密钥认证或是数字签名等用途。本版本支持MD5硬加速，主要应用于分布式对象存储的完整性校验场景，主要用于快速校验本地磁盘文件是否被破坏，非安全用途。 |  | MD5特性 |
| 第35页  基于openSSL使用MD5硬件的接口。 |  | 基于openSSL使用MD5硬算的接口 |
| 第36页  基于Zlib的压缩、解压  Zlib是提供数据压缩用的函式库  KAE解压缩特性选择通过Zlib第三方函式库的方式对外呈现，向上呈现统一的用户态接口，向下屏蔽芯片硬件驱动层。通过Zlib第三方函式库可以对接加速器的用户态驱动warpdrive，从而上层应用可以操作底层加速硬件，最终实现zlib、gzip两种模式下的硬件加速功能。具体的软件架构图如下所示。 |  | 基于Zlib的压缩、解压 |
| 第37页  Zlib库硬算压缩调用流程如下图，解压流程基本一致。 |  | Zlib库硬算压缩调用流程 |
| 第38页  Zlib关键接口 |  | Zlib关键接口 |
| 第39页  接下来我们介绍基于Linux内核态的应用  1.Linux内核Crypto框架  2.基于dm-crypt的透明磁盘加密 |  | Linux内核态的应用 |
| 第40页  Linux内核Crypto框架介绍  Linux内核提供了一套加解密框架Crypto Framework，通过这套框架可以注册用户自己的加解密算法实现，可以用系统默认的，如：aes/rsa等，对内核应用提供统一加解密接口。框架如下  加速器驱动可以通过Crypto Framework框架的注册接口，将硬件加速器的能力与内核Crypto框架对接，上层应用代码无须修改。  当前加速器驱动注册到Crypto的算法包括：SM4-XTS。 |  | Linux内核Crypto框架 |
| 第41页  下面介绍基于dm-crypt的透明磁盘加密  dm-crypt向上呈现为一个device mapper机制的target device，经过映射挂载后就可以作为透明加密磁盘使用。dm-crypt是一个框架，具体使用的算法都注册在Crypto模块中，若需要增加算法，只需要在内核态实现算法后，在Crypto模块中进行注册。例如：实现sm4-xts算法并注册到Crypto中，其中的成员函数setkey、encrypt，decrypt接口，通过调用SEC引擎驱动实现加解密硬件加速。  算法完成注册后，就可以通过LUKS（cryptsetup）进行配置并使用 |  | 基于dm-crypt的透明磁盘加密 |
| 第42页  Linux内核Crypto框架  加速器驱动可以通过Crypto Framework框架的注册接口，将硬件加速器的能力与内核Crypto框架对接，上层应用代码无须修改。  当前加速器驱动注册到Crypto的算法包括：SM4-XTS。 |  |  |
| 第43页  下面介绍Arm架构加速器 |  | 目录 |
| 第44页  Armv8.4 MPAM特性介绍  MPAM（Memory Partitioning and Monitoring）是Arm Architecture v8.4的 Extension特性。用于解决服务器系统中，混部不同类型业务时，由于共享资源的竞争 （Cache，DMC，Interconnect），而带来的某些关键应用性能下降或者系统整体性能 下降的问题。 |  | Armv8.4 MPAM特性介绍 |
| 第45页  MPAM应用的典型场景  例如云场景下，可针对不同业务同时作用于硬件访存路径上产生的竞争和冲突进行控制，从而帮助服务器提升利用率。 |  | MPAM应用的典型场景 |
| 第46页  MPAM控制访存关键路径上竞争的机理  1）任意一个进程的访存过程可称为一条业务流，使用MPAM可以标记业务流，从而使得数据流被硬件MPAM模块感知；  2）使用MPAM获取标记的数据流在共享资源（例如L3 Cache）中的实际使用情况；  3）结合业务需求和获取的时间使用情况配置对应数据流可用的共享资源。 |  | MPAM控制访存关键路径上竞争的机理 |
| 第47页  MPAM控制访存关键路径上竞争的机理  可以针对task或cpu标记业务流，MPAM使用closid（该概念来源于Intel RDT，可参考https://www.intel.cn/content/www/cn/zh/architecture-and-technology/resource-director-technology.html）标记业务流，系统默认task及cpu的closid为0。  MPAM隔离访存共享资源的主要对象是L3 Cache和DDR访存通道，如下图所示，被标记的业务流会使用共享的L3 Cache，从而在shareable的状态下各业务流之间会相互抢占同一块cache空间，使用MPAM可以在cache way的粒度下为对应的业务流划分cache way的使用。  多条数据流共享同一个L3 Cache介质  MPAM隔离访存共享资源的主要对象是L3 Cache和DDR访存通道，如图2所示，被标记的业务流会使用共享的L3 Cache，从而在shareable的状态下各业务流之间会相互抢占同一块cache空间，使用MPAM可以在cache way的粒度下为对应的业务流划分cache way的使用。 |  | MPAM控制访存关键路径上竞争的机理 |
| 第48页  不同机器配置的DDR通道可能不一样，但是都有一个最大可通过的流量约束，不同业务流在通过访存通道时会互相发生挤占，导致某些业务流的流量变小，从而影响对应业务的性能，同样可以使用MPAM约束业务流对DDR访存通道的使用  多条数据流共享同一个DDR访存通道 |  | 多条数据流共享同一个DDR访存通道 |
| 第49页  鲲鹏920能够提供的MPAM功能如表 |  | 鲲鹏920能够提供的MPAM功能如表 |
| 第50页  本节到此结束，谢谢大家。 |  |  |

|  |
| --- |
|  |
| 课堂练习、作业 |
| 无 |
| 课后总结 |
|  |