(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113687913 B
(45)授权公告日 2024.08.23

(21)申请号 202110894630.8
(22)申请日 2021.08.05
(65)同一申请的已公布的文献号
申请公布号 CN 113687913 A
(43)申请公布日 2021.11.23
(73)专利权人 浪潮云信息技术股份公司
地址 250100 山东省济南市高新区浪潮路1036号浪潮科技园S01号楼
(72)发明人 范志海 罗天 孙兴艳
(74)专利代理机构 济南信达专利事务所有限公司 37100
专利代理师 郗艳荣
(51)Int.Cl.
G06F 9/455(2006.01)
G06F 8/76(2018.01)

(56)对比文件
CN 105242962 A,2016.01.13
CN 106339257 A,2017.01.18

审查员 赵冰

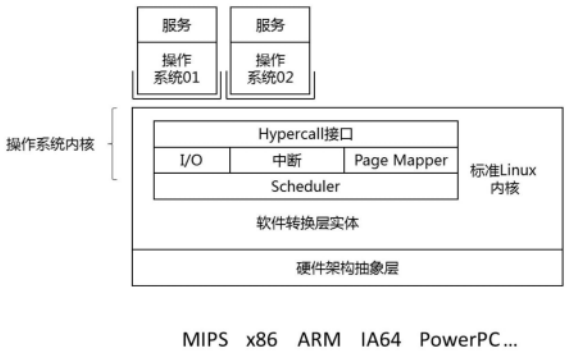
权利要求书1页 说明书6页 附图2页

(54)发明名称

一种面向边缘计算异构环境的轻量级应用适配方法

(57)摘要

本发明特别涉及一种面向边缘计算异构环境的轻量级应用适配方法。该面向边缘计算异构环境的轻量级应用适配方法,结合交叉编译技术和虚拟化技术构建自动化流水线驱动的集成开发环境,封装不同硬件环境的编译器,使用容器化交叉编译环境将编译过程中的不同步骤和任务自动串联起来,为异构环境提供统一的工作流框架,将需编译的代码与编译工具使用容器持久化的手段结合到一起,容器自动运行编译流水线最终生成各异构环境的可执行程序。该面向边缘计算异构环境的轻量级应用适配方法,能够方便的构建跨异构环境的边缘计算应用,适配各种边缘计算操作系统环境,并支持异构CPU环境。解决了在异构环境下需要重复编译应用的问题。



CN 113687913 B

1.一种面向边缘计算异构环境的轻量级应用适配方法,其特征在于:结合交叉编译技术和虚拟化技术构建自动化流水线驱动的集成开发环境,封装不同的硬件环境的编译器,使用容器化交叉编译环境将编译过程中的不同步骤和任务自动的串联起来,为异构环境提供统一的工作流框架,将需编译的代码与编译工具使用容器持久化的手段结合到一起,容器自动运行编译流水线最终生成各异构环境的可执行程序;

在系统初始化阶段,为边缘计算操作系统构建硬件抽象层,用于直接与硬件通讯,为应用层提供抽象支持,异构环境的内核层通过API的形式向应用层提供服务,使应用层的各个软件模块各自独立,相互不产生影响;

采用面向对象设计方法将数据与数据上的操作封装在对象的模块实体中,外界不能直接对对象内部进行访问和操作,只能通过消息的方式间接访问;

采用层次结构设计方法将硬件抽象层进一步细化为多个子层次,各子层之间定义统一的接口调用,从而将边缘操作系统中硬件相关和硬件无关的两部分程序代码隔离,使得硬件抽象层为应用层的软件模块提供一个已屏蔽硬件差异的接口;

在系统初始化阶段结束以后,边缘计算操作系统获得系统控制权,硬件抽象层转而负责向边缘计算操作系统提供服务,将应用层的调用转化为对硬件的直接访问和控制,建立与硬件相关的驱动程序;

应用层将驱动程序函数映射到硬件抽象层的API,改造后的驱动程序不再直接与硬件交互,而是通过硬件抽象层的API进行硬件资源的访问和控制;驱动程序表通过指向内部的驱动函数UART1Create()的指针建立通用的Create函数和设备指定的Create之间的连接;

使用I/O子系统提供的工具函数将硬件抽象层的API安装到驱动程序表中或从表中删除;

在物理服务器和边缘计算操作系统之间构建软件转换层,用于协调访问服务器上的所有物理设备和轻量级虚拟机,非中断的支持多工作负载迁移;当服务器启动并执行软件转换层时,为每一台轻量级虚拟机容器分配适量的内存、CPU、网络和磁盘资源,并加载所有虚拟机容器的客户操作系统。

2.根据权利要求1所述的面向边缘计算异构环境的轻量级应用适配方法,其特征在于:采用Linux作为软件转换层,所述交叉编译技术包括在Linux PC上,利用arm-linux-gcc编译器,编译出针对Linux ARM平台的可执行代码。

3.根据权利要求1所述的面向边缘计算异构环境的轻量级应用适配方法,其特征在于:所述虚拟化技术采用Docker容器虚拟化技术,VMware虚拟机软件或Citrix虚拟化应用程序。

4.根据权利要求1所述的面向边缘计算异构环境的轻量级应用适配方法,其特征在于:所述自动化流水线驱动采用Jenkins工具,GitLab CI工具或Circle CI技术。

5.根据权利要求1所述的面向边缘计算异构环境的轻量级应用适配方法,其特征在于:所述编译器采用交叉编译器arm-linux-gcc,armv7-rpi2-linux-gnueabihf或armv6-rpi-linux-gnueabi。

6.根据权利要求1所述的面向边缘计算异构环境的轻量级应用适配方法,其特征在于:所述边缘计算操作系统环境采用微内核操作系统,容器内核操作系统或实时内核操作系统。

一种面向边缘计算异构环境的轻量级应用适配方法

技术领域

[0001] 本发明涉及边缘计算操作系统技术领域,特别涉及一种面向边缘计算异构环境的轻量级应用适配方法。

背景技术

[0002] 随着边缘计算大规模的运用,边缘操作系统面临边缘环境越来越复杂的挑战,其中异构平台的适配是重要的一环。一方面边缘计算操作系统需要适用在不同体系架构,另一方面要支持轻量级虚拟系统适配异构平台。

[0003] 边缘计算涉及到海量的终端设备、边缘节点,是数据采集、数据汇聚、数据集成、数据处理的前端,而这些设备往往存在异构性,来自于不同的生产厂商、不同的数据接口、不同的数据结构、不同的传输协议、不同的底层平台等。边缘计算操作系统采用架构抽象方式隔离底层的异构硬件,同时使用操作系统和虚拟环境间的转换层帮助虚拟环境本身进行适配。

[0004] 鉴于边缘计算在不同的应用场景时的环境数据存在很大差异,有许多针对场景设计的边缘操作系统。在智能家居场景下,仅通过wifi模块连接到云计算中心的做法,远远不能满足智能家居的需求。EdgeOSH是一种针对智能家居设计的边缘操作系统,其部署于家庭的边缘网关中,通过3层功能抽象连接上层应用和下层智能家居硬件。利用该操作系统,在家庭内部较易里连接和管理智能家居设备,并在本地处理这些设备所产生的数据,降低数据传输带宽的负载,同时基于EdgeOSH的应用服务程序可向用户提供更好的资源管理和分配。

[0005] 边缘计算操作系统面向多样的边缘计算任务,其服务管理层应具有差异性、可扩展性、隔离性和可靠性的需求。PhiOS是一种面向智能家居设备的边缘操作系统,其引入了轻量级的REST引擎和LUA解释器,帮助用户在家庭边缘设备上部署计算任务。OpenVDAP是针对汽车场景设计的数据分析平台,提出了面向网联车场景的边缘操作系统EdgeOSv,该操作系统能提供任务弹性管理、数据共享以及安全和隐私保护等功能。

[0006] 复杂的边缘计算环境,要求边缘计算操作系统采用架构抽象方式隔离底层的异构硬件,同时使用操作系统和虚拟环境间的转换层帮助虚拟环境本身进行适配。

[0007] 目前的现状问题是,边缘计算应用要独立的构建以适应不同的边缘计算异构环境,开发边缘计算应用要花费大量时间精力搭建各种边缘计算编译与运行环境,准备不同的异构环境的组件,费时费力。

[0008] 针对上述情况,本发明提出了一种面向边缘计算异构环境的轻量级应用适配方法。

发明内容

[0009] 本发明为了弥补现有技术的缺陷,提供了一种简单高效的面向边缘计算异构环境的轻量级应用适配方法。

[0010] 本发明是通过如下技术方案实现的：

[0011] 一种面向边缘计算异构环境的轻量级应用适配方法，其特征在于：结合交叉编译技术和虚拟化技术构建自动化流水线驱动的集成开发环境，封装不同的硬件环境的编译器，使用容器化交叉编译环境将编译过程中的不同步骤和任务自动的串联起来，为异构环境提供统一的工作流框架，将需编译的代码与编译工具使用容器持久化的手段结合到一起，容器自动运行编译流水线最终生成各异构环境的可执行程序，从而方便的构建跨异构环境的边缘计算应用，适配各种边缘计算操作系统环境，并支持异构CPU环境。

[0012] 在系统初始化阶段，为边缘计算操作系统构建硬件抽象层，用于直接与硬件通讯，为应用层提供抽象支持，异构环境的内核层通过API(应用程序编程接口)的形式向应用层提供服务，使应用层的各个软件模块各自独立，相互不产生影响；

[0013] 采用面向对象设计方法将数据与数据上的操作封装在对象的模块实体中，外界不能直接对对象内部进行访问和操作，只能通过消息的方式间接访问；

[0014] 采用层次结构设计方法将硬件抽象层进一步细化为几个子层次，各子层之间定义统一的接口调用，从而将边缘操作系统中硬件相关和硬件无关的两部分程序代码隔离，使得硬件抽象层为应用层的软件模块提供一个已屏蔽硬件差异的接口；

[0015] 在系统初始化阶段结束以后，边缘计算操作系统获得系统控制权，硬件抽象层转而负责向边缘计算操作系统提供服务，将应用层的调用转化为对硬件的直接访问和控制，建立与硬件相关的驱动程序。

[0016] 应用层将驱动程序函数映射到硬件抽象层的API，改造后的驱动程序不再直接与硬件交互，而是通过硬件抽象层的API进行硬件资源的访问和控制；驱动程序表通过指向内部的驱动函数UART1Create()的指针建立通用的Create函数和设备指定的Create之间的连接；

[0017] 使用I/O子系统提供的工具函数将硬件抽象层的API安装到驱动程序表中或从表中删除。

[0018] 在物理服务器和边缘计算操作系统之间构建软件转换层，用于协调访问服务器上的所有物理设备和轻量级虚拟机，非中断的支持多工作负载迁移；当服务器启动并执行软件转换层时，为每一台轻量级虚拟机容器分配适量的内存、CPU、网络和磁盘资源，并加载所有虚拟机容器的客户操作系统。

[0019] 采用Linux作为软件转换层，所述交叉编译技术包括在Linux PC上，利用arm-linux-gcc编译器，编译出针对Linux ARM平台的可执行代码。

[0020] 所述虚拟化技术采用Docker容器虚拟化技术，VMware虚拟机软件或Citrix虚拟化应用程序。

[0021] 所述自动化流水线驱动可以是Jenkins工具，GitLab CI(Gitlab Continuous Integration，持续集成)工具或Circle CI技术。

[0022] 所述编译器采用交叉编译器arm-linux-gcc，armv7-rpi2-linux-gnueabihf或armv6-rpi-linux-gnueabi。

[0023] 所述边缘计算操作系统环境采用微内核操作系统，容器内核操作系统或实时内核操作系统。

[0024] 本发明的有益效果是：该面向边缘计算异构环境的轻量级应用适配方法，为应用

提供了统一的调用方法,解决了在异构环境下需要重复编译应用的问题。

附图说明

[0025] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术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他的附图。

[0026] 附图1为本发明标准I/O函数与API集合映射关系示意图。

[0027] 附图2为本发明I/O驱动程序表、设备表与硬件架构抽象API的关联示意图。

[0028] 附图3为本发明面向边缘计算异构环境的轻量级应用适配方法示意图。

[0029] 附图4为本发明面向边缘计算异构环境的轻量级应用适配方法层级架构示意图。

具体实施方式

[0030] 为了使本技术领域的人员更好的理解本发明中的技术方案,下面将结合本发明实施例,对本发明实施例中的技术方案进行清楚,完整的描述。显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都应当属于本发明保护的范围。

[0031] 编译是从源代码到能直接被计算机或虚拟机执行的目标代码的翻译过程。编译器可以生成用在与编译器本身所在的计算机和操作系统(平台)相同的环境下运行的目标代码,也可以生成用在其它平台上运行的目标代码,这种编译器叫做交叉编译器。交叉编译器在生成新的硬件平台时非常有用。有时是因为目的平台上不允许或不能够安装我们所需要的编译器;有时是因为目的平台上的资源贫乏,无法运行我们所需要编译器;有时又是因为目的平台还没有建立。在本发明中,交叉编译器可以解决在异构环境下需要重复编译的问题。通过交叉编译工具,我们就可以在CPU能力很强、存储空间足够的主机平台上(比如PC上)编译出针对其他平台的可执行程序。

[0032] 关于容器虚拟化技术,虚拟化技术是云计算的重要技术,主要用于物理资源的池化,从而可以弹性地分配给用户。容器也是虚拟化,但是属于“轻量级”的虚拟化,目的和虚拟机一样,都是为了创造“隔离环境”。但是,它又和虚拟机有很大的不同——虚拟机是操作系统级别的资源隔离,而容器本质上是进程级的资源隔离。容器是为应用程序提供了隔离的运行空间,每个容器内都包含一个独享的完整用户环境空间,包含需要部署的应用和它依赖的系统环境,容器之间共享同一个系统内核。Docker是解决运行环境和配置问题的软件容器,方便做持续集中并有助于整体发布的容器虚拟化技术。在本发明中,容器化技术主要为实现轻量级异构适配技术提供运行环境,方便部署和运行。

[0033] 流水线是将源代码转换为可发布产品的多个不同的任务和作业,通常串联成一个软件“管道”,一个自动流程成功完成后会启动管道中的下一个流程。就是运行在一个的工作流框架,将原来独立运行于单个或者多个节点的任务连接起来,实现单个任务难以完成的复杂流程编排和可视化的工作。

[0034] 该面向边缘计算异构环境的轻量级应用适配方法,结合交叉编译技术和虚拟化技术构建自动化流水线驱动的集成开发环境,封装不同的硬件环境的编译器,使用容器化交

叉编译环境将编译过程中的不同步骤和任务自动的串联起来,为异构环境提供统一的工作流框架,将需编译的代码与编译工具使用容器持久化的手段结合到一起,容器自动运行编译流水线最终生成各异构环境的可执行程序,从而方便的构建跨异构环境的边缘计算应用,适配各种边缘计算操作系统环境,并支持异构CPU环境。

[0035] 在系统初始化阶段,为边缘计算操作系统构建硬件抽象层,用于直接与硬件通讯,为应用层提供抽象支持,异构环境的内核层通过API(应用程序编程接口)的形式向应用层提供服务,使应用层的各个软件模块各自独立,相互不产生影响;

[0036] 应用层在进行硬件操作时,不需要了解设备的具体细节,从而大大减少系统理解和开发的复杂度。将接口和实现分离开来,将具体的功能块隐藏在抽象的接口背后,以保证每个模块可以在不影响其他模块的情况下进行改变,将模块之间的依赖关系仅仅限定于接口。软件模块之间是相互独立的关系,而不是层次之间相互依赖的关系。

[0037] 对象是结构化使用模块的方法。为了提高软件的扩展性、维护性和重用性,采用面向对象设计方法将数据与数据上的操作封装在对象的模块实体中,外界不能直接对对象内部进行访问和操作,只能通过消息的方式间接访问;

[0038] 采用层次结构设计方法将硬件抽象层进一步细化为几个子层次,各子层之间定义统一的接口调用,从而将边缘操作系统中硬件相关和硬件无关的两部分程序代码隔离,使得硬件抽象层为应用层的软件模块提供一个已屏蔽硬件差异的接口;

[0039] 为了保证开发语言的移植性,还引入了面向对象的思想。C语言虽不能直接支持面向对象的数据结构,如数据类型的动态绑定、多态函数或类继承。但在设计时可以借鉴面向对象语言的特点实现基于C语言的对象,却又不依赖于它,从而使系统开发难度大为降低。

[0040] 在系统初始化阶段结束以后,边缘计算操作系统获得系统控制权,硬件抽象层转而负责向边缘计算操作系统提供服务,将应用层的调用转化为对硬件的直接访问和控制,建立与硬件相关的驱动程序。

[0041] 应用层将驱动程序函数映射到硬件抽象层的API,改造后的驱动程序不再直接与硬件交互,而是通过硬件抽象层的API进行硬件资源的访问和控制;驱动程序表通过指向内部的驱动函数UART1Create()的指针建立通用的Create函数和设备指定的Create之间的连接;

[0042] 使用I/O子系统提供的工具函数将硬件抽象层的API安装到驱动程序表中或从表中删除。

[0043] 附图1中表示了硬件相关设备驱动程序的标准I/O函数与硬件架构抽象API集合的映射关系。从图中可看出,驱动程序表的第二行、第N元素是一个指向内部的驱动函数UART1Create()的指针。这个指针建立了通用的Create函数和设备指定的Create之间的连接。

[0044] 在物理服务器和边缘计算操作系统之间构建软件转换层,用于协调访问服务器上的所有物理设备和轻量级虚拟机,非中断的支持多工作负载迁移;当服务器启动并执行软件转换层时,为每一台轻量级虚拟机容器分配适量的内存、CPU、网络和磁盘资源,并加载所有虚拟机容器的客户操作系统。

[0045] 软件转换层之于操作系统类似于操作系统之于进程。它们为执行提供独立的虚拟容器硬件平台,而虚拟容器硬件平台反过来又提供对底层设备的虚拟空间进行完整的访

问。轻量级虚拟化就是通过软件转换层隐藏底层物理硬件的过程,从而让多个操作系统可以透明地使用和共享它。在典型的分层架构中,提供平台虚拟化的层就是软件转换层。客户操作系统支持异构容器镜像,因为对这些异构虚拟化容器而言,硬件是专门针对它们虚拟化的。对于进程来说,操作系统将对机器的底层资源的访问虚拟化为进程。软件转换层也做一样的事情,但其对象不是进程,而是整个客户操作系统。

[0046] 在较高级别上,软件转换层需要少量设施启动客户操作系统:一个需要驱动的内核映像、一个配置(比如IP地址和所需的内存量)、一个磁盘盒一个网络设备。磁盘和网络设备通常映射到机器的物理磁盘和网络设备。最后,需要使用一组客户操作系统工具启动和管理客户操作系统。

[0047] 采用Linux作为软件转换层,所述交叉编译技术包括在Linux PC上,利用arm-linux-gcc编译器,编译出针对Linux ARM平台的可执行代码。

[0048] 一个简化的软件转换层架构实现的关键功能是使客户操作系统可以和宿主操作系统同时运行。以Linux为基础开发软件转换层受益于稳步前进的Linux,以及为改进Linux投入的大量工作。从典型的优化、bug修复、调度和内存管理创新到支持不同处理器架构,Linux都是一个不断进步的平台。除了可以将Linux平台用作软件转换层之外,还可以将其用作操作系统。因此,除了可以在Linux软件转换层上运行多个客户操作系统之外,可以在该级别上运行其他传统的应用程序。

[0049] 所述虚拟化技术采用Docker容器虚拟化技术,VMware虚拟机软件或Citrix虚拟化应用程序。

[0050] 所述自动化流水线驱动可以是Jenkins工具,GitLab CI(Gitlab Continuous Integration,持续集成)工具或Circle CI技术。

[0051] 所述编译器采用交叉编译器arm-linux-gcc,armv7-rpi2-linux-gnueabihf或armv6-rpi-linux-gnueabi。

[0052] 构建跨异构环境的边缘计算应用,包括医疗保健,视频分析,智能家居,移动大数据分析等。

[0053] 所述边缘计算操作系统环境采用微内核操作系统,容器内核操作系统或实时内核操作系统。

[0054] 所述异构CPU环境中,CPU架构包括X86、ARM、RS64、Power PC等。

[0055] 与现有技术相比,该面向边缘计算异构环境的轻量级应用适配方法,具有以下特点:

[0056] 第一、支持不同类型指令集和不同体系架构的异构计算架构下的虚拟化,充分发挥各种计算单元的优势,实现性能、成本、功耗、可移植性等方面的均衡,能够满足边缘计算业务场景的需求。

[0057] 第二、采用轻量级虚拟化适配技术,极大的复用了主机系统的资源,实现了对主机资源进行更加细粒度的隔离管理,成本及性能损耗也比较小,具有更好的资源利用率。在提供隔离机制的同时,实现资源共享,满足了边缘计算可扩展,多租户,安全性,隐私性和灵活性的关键要求。

[0058] 第三、基于虚拟化容器技术来构建边缘计算异构环境的交叉编译流水线,可以很方便构建适合各种异构环境的边缘计算应用;使用流水线能管理整个项目生命周期,明确

阶段,方便处理问题,并且当前异构环境可以放到项目代码中进行版本管理,易维护。

[0059] 以上所述的实施例,只是本发明具体实施方式的一种,本领域的技术人员在本发明技术方案范围内进行的通常变化和替换都应包含在本发明的保护范围内。

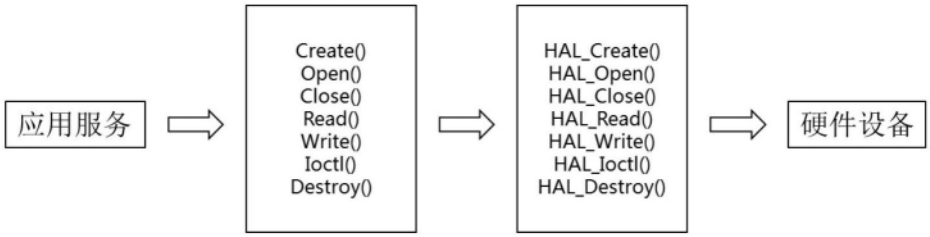

应用服务
Create()
Open()
Close()
Read()
Write()
Ioctl()
Destroy()
HAL_Create()
HAL_Open()
HAL_Close()
HAL_Read()
HAL_Write()
HAL_Ioctl()
HAL_Destroy()
硬件设备

图1

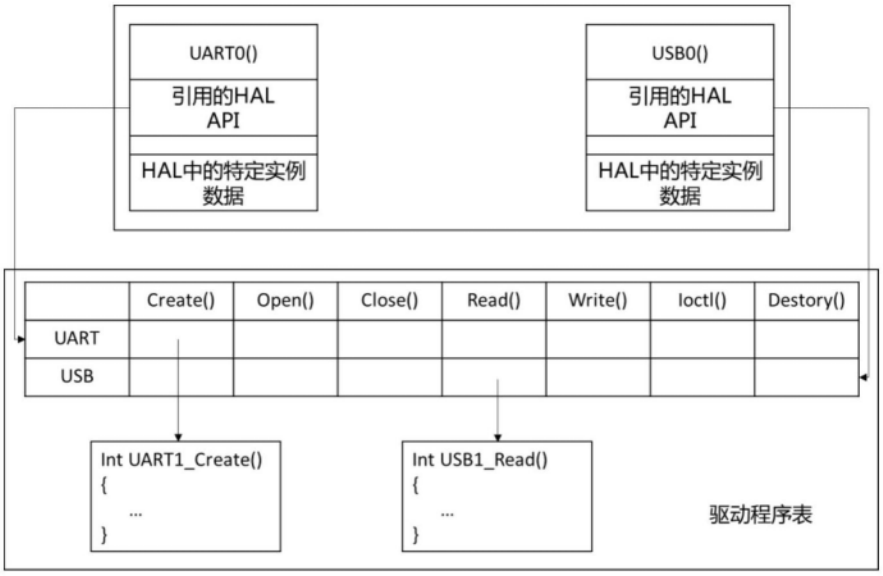

UART0()
引用的HAL API
HAL中的特定实例数据
USB0()
引用的HAL API
HAL中的特定实例数据
Create()
Open()
Close()
Read()
Write()
Ioctl()
Destory()
UART
USB
Int UART1_Create()
{
...
}
Int USB1_Read()
{
...
}
驱动程序表

图2

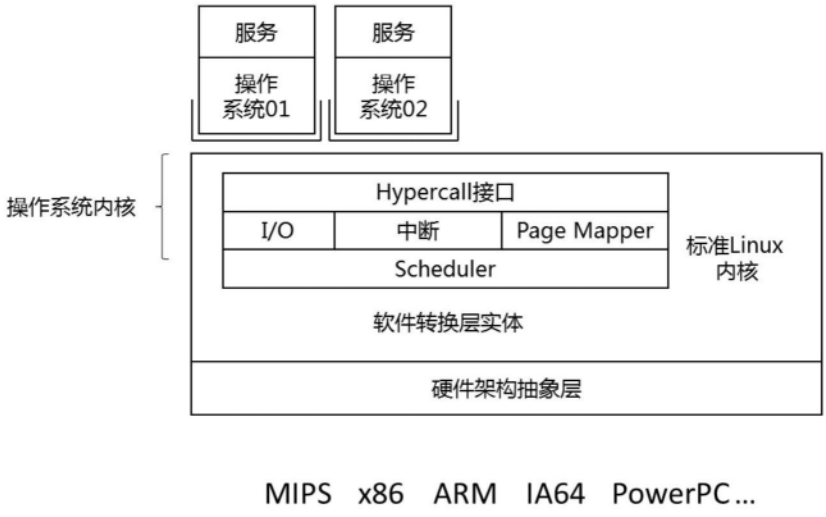

服务
操作系统01
服务
操作系统02
操作系统内核
Hypercall接口
I/O
中断
Page Mapper
Scheduler
标准Linux内核
软件转换层实体
硬件架构抽象层
MIPS x86 ARM IA64 PowerPC...

图3

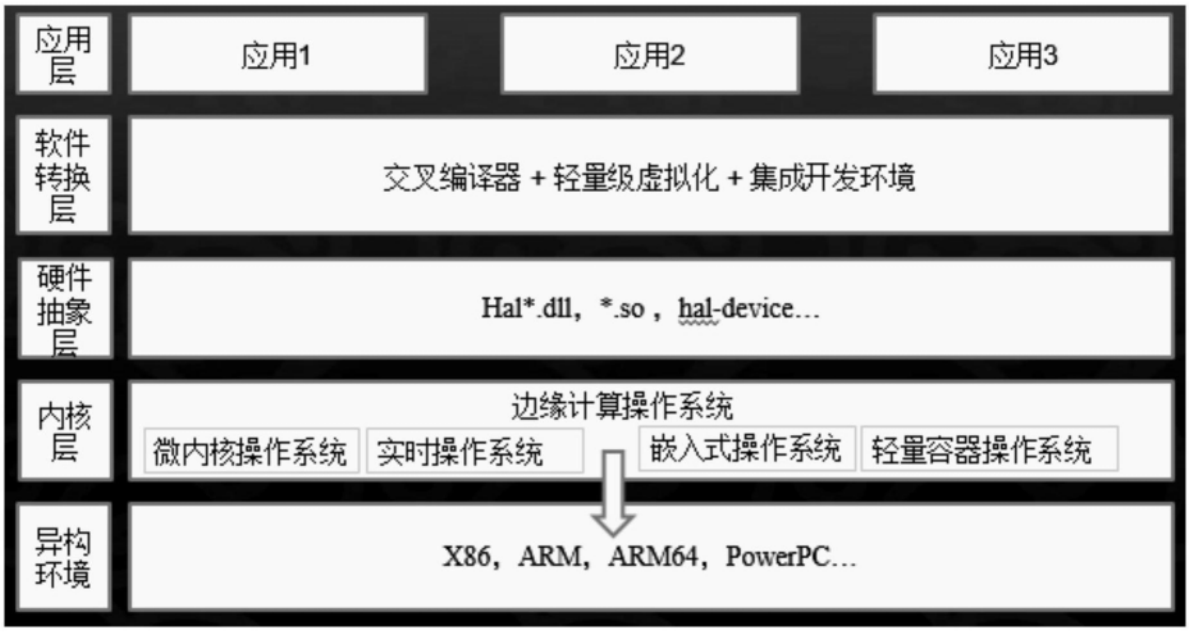

应用层
应用1
应用2
应用3
软件转换层
交叉编译器 + 轻量级虚拟化 + 集成开发环境
硬件抽象层
Hal*.dll，*.so，hal-device...
内核层
边缘计算操作系统
微内核操作系统
实时操作系统
嵌入式操作系统
轻量容器操作系统
异构环境
X86，ARM，ARM64，PowerPC...

图4