WP474 (v1.0) 2016 年 3 月 31 日

用 Zynq UltraScale+ MPSoC 上的 Xen 管理程序实现虚拟化

运行在 UltraScale+ ™ MPSoC 上的 Xen 管理程序，可提供稳健可靠的硬件加速虚拟化和易用性，有助于嵌入式系统设计人员最大化硬件投资回报。

内容摘要

虚拟化对桌面系统已是司空见惯，但对嵌入式系统设计人员来说，却一直是一个棘手的问题，因为嵌入式系统设计人员需要优化 SoC 系统的利用率和性能。

传统上，虚拟化难以用于嵌入式领域，是因为缺乏既可简化解决方案实现工作，又能提供满意性能的合适硬件资源。因此，需要在同一处理器上运行异构软件协议栈的系统不得不依赖于人工管理各种软件协议栈，或不得不接受未加速虚拟化所带来的更大时延和性能特性下降等缺陷。

位于每个 Zynq UltraScale+ MPSoC 核心的 ARM®v8 架构能确保真正的硬件加速虚拟化，克服这些实现障碍。此外，Xen 管理程序提供的易用性为在嵌入式系统中实现桌面级性能和生产力奠定了基础。运行在 Zynq UltraScale+ MPSoC 上的 Xen 管理程序为系统设计人员提供完整的解决方案，能充分释放嵌入式系统设计的全部潜力。

简介

虚拟化通过让多个软件协议栈同时运行在同一处理器上，已经征服了桌面系统，现在将进军嵌入式系统。VMWare 和 VirtualBox 这样的常用套件已经让虚拟化成为桌面用户司空见惯的做法。这类软件被视为台式电脑生产力增强器，不过赛灵思 Zynq UltraScale+ MPSoC 等嵌入式片上系统 (SoC) 也可运用同一原理充分发挥最大潜能。

虚拟化的作用随系统发生变化。对某些设计人员来说，虚拟化能够让处理器始终保持满负载状态，从而节省电力，最大化性能。对其他系统，虚拟化可对各个软件协议栈进行分区，实现隔离或冗余。

传统上，虚拟化难以用于嵌入式领域，是因为缺乏既可简化解决方案实现工作，又能提供满意性能的合适硬件资源。因此，需要在同一处理器上运行异构软件协议栈的系统不得不依赖于人工管理各种软件协议栈，或不得不接受未加速虚拟化所带来的更大时延和性能特性下降等缺陷。

位于每个 Zynq UltraScale+ MPSoC 核心的 ARM®v8 架构能确保真正的硬件加速虚拟化，克服这些实现障碍。此外，Xen 管理程序提供的易用性为在嵌入式系统中实现桌面级性能和生产力奠定了基础。运行在 Zynq UltraScale+ MPSoC 上的 Xen 管理程序为系统设计人员提供完整的解决方案，能充分释放嵌入式系统设计的全部潜力。

为什么要虚拟化嵌入式系统?

是否需要虚拟化一般由下列三大系统设计特征之一决定的：

- 处理器必须能根据性能规范要求尽量保持满载。
- 应安全、可扩展性和 / 或可靠性的要求需要进行软件隔离或分区。
- 为满足可靠性要求，需要进行缩放或提供冗余。

硬件加速的虚拟化位于 Zynq UltraScale+ MPSoC 的 ARMv8 架构的核心位置。这种协同架构不仅能满足上述每一项要求，实际上它还能简化实现工作。之所以能这样，是因为它为每个客户软件协议栈提供一个隔离的沙盒。没有这样的硬件加速，这些系统会变得复杂得多，给实现和管理带来切实的问题。

经优化的系统加载

对系统加载进行精心管理是嵌入式系统的共通难题，在没有管理程序辅助时还会造成一系列严峻的运行问题。Linux 等传统操作系统非常擅长于处理对称多处理 (SMP) 任务，此时所有处理器核都处在它的控制之下。但是如果不是每一个处理器都有任务会怎么样？ Linux 会让没有使用的处理器处于待机状态。Linux 的多个实例可按原样虚拟化（即对 Linux 不做修改），每个实例都运行自己的一组任务。可以根据需要将 Linux 的新实例投入使用。相反如果需求较低，这些 Linux 实例可以关闭。因此处理器可以根据需要激活或待机，但系统总体上能保持近乎持续的繁忙。另外，管理多个不同的软件协议栈也可采用类似的方法。这样无需要求所有的虚拟操作系统相同，从而提高最终系统的灵活性和模块化水平。

经优化的软件隔离与分区

软件隔离和分区是另一个能给非托管环境带来严峻挑战的常见用例。这个问题与系统加载问题类似，但是包含一项额外的要求，即每个软件协议栈不能与任何同时运行的软件协议栈发生干扰。这种情况的最简单的例子是两个并行运行的实时操作系统。满足这些需求的传统方法会增大软件的复杂性，以确保每个 RTOS 只与分配给自己的有限资源交互。相比之下，虚拟化系统采用的沙盒模式能让每个 RTOS 完全控制其沙盒中可用的资源。它把这个沙盒视为完整系统，在使用时无需了解是否有其他软件可能在系统上运行。这种沙盒模式能极大地减少绑定到特定硬件平台的资源，提高代码的可移植性。此外，该 RTOS 无需感知系统中的任何其他软件，让它能大幅得到简化。这种脱离依赖性的方法，对开发人员来说是一个巨大的福音，因为他们只需编写一次代码，就能将它部署到众多的不同系统上。

经优化的缩放与冗余

对于嵌入式系统，缩放与冗余需求会随着 SoC 的性能和功能的增强而提升。

缩放要求加载在处理器上的软件量随系统需求增长而增加。例如高性能计算环境要求将更多 Linux 操作系统实例投入使用，满足用户提出的更多请求。通过运用虚拟化，能够根据需要将 Linux 的相同复本投入使用。随后，随着系统上的需求减少，这些 Linux 的实例可被关闭。

冗余要求特定服务保持可用，甚至是在系统处于非常时期的时候也一样。例如 RTOS 能提供对特定系统功能的关键监测。如果该 RTOS 因某种原因失效呢？使用虚拟化，系统监控器能检测到故障，还能重新启动该 RTOS，或者启动该 RTOS 的一个新实例，从而最大限度地降低或消除这些关键系统服务的停机时间。

为什么选择 Xen

在选择管理程序解决方案的时候，重要的是让它稳健和可靠。此外，它必须通过积极主动的开发来跟上它周围世界发生的变化的步伐。Xen 管理程序就是这样的一种解决方案。

Xen 最初是上个世纪 90 年代晚期在剑桥大学作为更大的 Xenoserver 项目的子项目开始的。它于 2000 年代早期发布给开源社区，并在 2013 年得到 Linux 基金会的支持。在 Linux 基金会的鼎力支持下，Xen 已经成为基于 Linux 的操作系统的事实管理程序解决方案。

虽然 Xen 的传统架构一直是兼容 x86，近期的主机开发使之也成为 ARM 架构上的稳健解决方案。Xen 能充分发挥包括系统存储器管理单元 (SMMU) 在内的 ARMv8 底层虚拟硬件的作用。

Xen 免费提供，并配套标准的 GPLv2 许可证，有活跃的用户社区开发新特性，以及丰富的技术支持资源。对需要商业化维护和支持的集成商而言，有 DornerWorks 这样的厂商为 Xen 提供专业支持和结构。

在选择管理程序解决方案时，软件支持也是一个关键的差异化因素。Xen 本身是一种 I 类管理程序，即它能直接在底层硬件上运行，而不像 II 类管理程序 VMWare 或 VirtualBox 运行在主机操作系统上。

Xen 管理程序把客户操作系统划分为域。它使用特殊的管理接口 Dom0 控制管理程序的运行时操作。该域为 Xen 管理程序提供专用软件基础架构。这种运行结构对 Xen 的正常工作是必须的。Dom0 使用其内核内的专用软件以及能够直接访问底层硬件的专用管理驱动程序。最常见的 Dom0 操作系统之一是 Linux，它能够有力地支持 Zynq UltraScale+ MPSoC。

所有存在于非专用域内的标准客户操作程序都被集中称之 DomU。包括 Linux 这样的高级操作系统、FreeRTOS 这样的实时操作系统乃至裸机代码等各种客户软件都在底层受 Xen 支持用于 DomU。任何主机与客户的组合都完全受系统设计人员的需求驱动。相比之下，大部分商业管理程序解决方案只支持有限数量的客户而且是在极为具体的配置中，一般是管理程序提供方自己开发的配置中。

将 Xen 管理程序与 Zynq UltraScale+ MPSoC 集成简便易行

即便管理程序解决方案使用最出色的技术组件，如果不能简单直观地实施在用户系统中，也基本上没有什么用处。通过极为贴近 Linux 内核，Xen 在这方面能提供很多便利。实现 Xen 管理程序功能与在 Linux 内核中实现任何其他特性没有区别。此外。管理 DomU 沙盒既可以手动处理，也可以通过称为 Xen 工具的工具套件处理。

对于将 Linux 用作 Dom0 的设计人员，通过命令行就能简单易行地安装 Xen 工具。Xen 工具可使用源代码构建和安装，也可借助 RPM 或 APT 等通用封装管理器构建和安装。

安装完成后，Xen 工具能从 Linux 用户空间用名为 xl 的工具创建、管理和损坏 DomU 环境。

创建新的虚拟机与创建描述虚拟化环境的明文 ASCII 配 置文件一样简单。该文件设定如下的详细内容：

- 为虚拟机分配的存储器数量
- 虚拟化的 CPU 的数量
- 联网详情
- 磁盘镜像文件

配置文件还可以借助文本解析工具和版本控制软件轻松地加以管理。请参阅如下的实例代码：

```
# This configures an HVM rather than PV guest
builder = "hvm"

# Guest name
name = "My Virtual Machine"

# Initial memory allocation (MB)
memory = 128

# Number of VCPUs
vcpus = 2

# Disk Devices
# A list of `diskspec' entries as described in
# docs/misc/xl-disk-configuration.txt
disk = [ '/dev/vg/guest-volume,raw,xvda,rw' ]
```

用于管理虚拟机的命令被称为 xl。该命令可让设计人员管理 Xen 虚拟机从启动到关断的整个生命周期。要根据已有的配置文件创建新的虚拟机实例，设计人员可以使用一个简单的单行命令创建 Xen 管理程序：

```
xl create <path_to_configuration_file>
```

在虚拟机的生命周期中，其他 xl 选项如 list、reboot 和 shutdown 都可以用于管理任务。

结论

多类系统都能从虚拟化获益。许多采用其他方法会变得复杂化和高劳动强度的系统需求，也能够使用 Xen 实现在多处理器平台上，如拥有 ARMv8 处理器的赛灵思 Zynq UltraScale+ MPSoC。使用虚拟化能够实现下面这些想实现而且往往相当关键的特性：

- 分区
- 隔离
- 可靠性
- 冗余

虚拟化不仅能够实现这些系统，还能让实现工作相当简单。系统设计人员应该在评估自己的设计时，考虑使用虚拟化解决方案来确保最优异、最具成本效益的结果。采用 ARMv8 多处理器的赛灵思 Zynq UltraScale+ MPSoC 对 Xen 管理程序和 Xen 工具集而言是理想的设计平台。

对软件开发人员而言，虚拟化能让软件协议栈的开发可移植、可靠，采用最少的胶合逻辑就能让它们按要求运行。此外，在 Linux 基金的鼎力支持下，Xen 相当稳健，能跨越多种类型的操作系统和软件协议栈提供稳健的集成和支持。最后，基于 Xen 的虚拟机的管理使用易于使用的命令行界面和适合版本控制的明文 ASCII 文本配置，相当简单。

赛灵思在 GitHub（网址：https://github.com/Xilinx/linux-xlnx）上提供 Linux 内核支持 Xen 功能。关于 Xen 程序管理器和 Xen 工具的源代码以及 Xen 项目的更详细介绍，请访问 http://www.xenproject.org/。

DornerWorks 是赛灵思的 Xen 管理程序合作伙伴。如需了解更多详情，敬请访问：http://dornerworks.com/services/xilinxxen

修订历史

下表列出了本文档的修订历史：

日期	版本	修订描述
2016 年 03 月 31 日	1.0	赛灵思初始版本。

免责声明

本文向贵司 / 您所提供的信息（下称“资料”）仅在对赛灵思产品进行选择和使用时参考。在适用法律允许的最大范围内：(1) 资料均按“现状”提供，且不保证不存在任何瑕疵，赛灵思在此声明对资料及其状况不作任何保证或担保，无论是明示、暗示还是法定的保证，包括但不限于对适销性、非侵权性或任何特定用途的适用性的保证；且 (2) 赛灵思对任何因资料发生的或与资料有关的（含对资料的使用）任何损失或赔偿（包括任何直接、间接、特殊、附带或连带损失或赔偿，如数据、利润、商誉的损失或任何因第三方行为造成的任何类型的损失或赔偿），均不承担责任，不论该等损失或者赔偿是何种类或性质，也不论是基于合同、侵权、过失或是其他责任认定原理，即便该损失或赔偿可以合理预见或赛灵思事前被告知有发生该损失或赔偿的可能。赛灵思无义务纠正资料中包含的任何错误，也无义务对资料或产品说明书发生的更新进行通知。未经赛灵思公司的事先书面许可，贵司 / 您不得复制、修改、分发或公开展示本资料。部分产品受赛灵思有限保证条款的约束，请参阅赛灵思销售条款：http://china.xilinx.com/legal.htm#tos; IP 核可能受赛灵思向贵 司 / 您签发的许可证中所包含的保证与支持条款的约束。安全保护功能，不能用于任何需要专门故障安全保护性能的用途。如果把赛灵思产品应用于此类特殊用途，贵司 / 您将自行承担风险和责任。请参阅赛灵思销售条款 :http://china.xilinx.com/legal.htm#tos。

关于与汽车相关用途的免责声明

赛灵思产品并非为故障安全保护目的而设计，也不具备此故障安全保护功能，不能用于任何需要专门故障安全保护性能的用途，比如与下列有关的用途：(1) 安全气囊设置；(2) 车辆控制，除非在该赛灵思产品中具备故障安全保护或者额外功能（但不包括对安装在赛灵思设备中用于执行该等额外功能的软件的使用）且会对操作人员操作失误发出警告信号；或者 (3) 可能会导致死亡或者人身损害的用途。客户应当自行承担因赛灵思产品被用于该等用途而产生的全部风险和责任。Zynq UltraScale+ MPSoC 处理系统的主要特性。