基于RISC-V架构的强化学习优化及容器化方法研究

Xx, xx, xx

（南昌大学信息工程学院，江西 南昌 330031）

摘 要：RISC-V作为近年来最火热的开源指令集架构，被广泛应用于各个特定领域的微处理器，特别是机器学习领域的模块化定制。但是，现有的RISC-V应用需要将传统软件或模型在RISC-V指令集上重新编译或优化，故如何能快速地在RISC-V体系结构上部署、运行及测试机器学习框架是一个亟待解决的技术挑战。使用虚拟化技术可以实现跨平台的模型部署及运行问题。但传统的虚拟化技术，例如虚拟机，对原生系统性能要求高、资源占用多，运行响应慢，往往不适用于RISC-V架构的应用场景。尝试解决在资源受限的RISC-V架构上的强化学习虚拟化问题。首先，通过采用容器化技术，减少上层软件构建虚拟化代价，去除冗余中间件，定制命名空间隔离特定进程，有效提升学习任务资源利用率，实现模型训练快速执行；其次，利用RISC-V指令集的特征，进一步优化上层神经网络模型，优化强化学习效率；最后，实现整体优化及容器化方法系统原型，并通过测试多种基准测试集，完成系统原型性能评估。容器化技术和传统RISC-V架构下交叉编译深度神经网络模型的方法相比，仅付出较小的额外80%性能代价，能快速实现更多、更复杂的深度学习软件框架的部署及运行；与硬件虚拟化方法相比，基于RISC-V的模型具有近似的部署时间，并减少大量的性能损失。初步实验结果证明，容器化及其上的优化方法是解决基于RISC-V架构的软件及学习模型快速部署的一种有效方法。

关键词：虚拟化，神经网络，RISC-V

1. 引言

随着登纳德缩放定律和摩尔定律的终结，标准微处理器性能提升的减速已成为了既定事实，体系结构在新的黄金时代需要寻求新的前进方向[3]。加州大学伯克利分校提出了RISC-V（RISC Five），即第五代RISC架构。RISC-V并非是精简指令集简单的版本迭代，相比之下它最大的优势在于开源，并且允许用户基于特定需求添加定制化拓展指令集。RISC-V由于其高度的灵活性在工业界和学术界均受到广泛关注，且将会应用在可穿戴设备、智能家居、机器人、自动驾驶及工业装置等领域的计算设备中[1]。

于此同时，深度学习在计算机视觉、语音识别、自然语言处理等领域有了越来越密集的应用。构建一个深度神经网络的工作流通常包括以下几步：

**步骤1：**预处理输入以使它们可被神经网络使用；

**步骤2：**使用这些输入执行神经网络推理过程；

**步骤3：** 对网络生成的预测进行后处理。

由于用户在移动和IoT（Internet of Thing）设备上使用深度神经网络模型的场景越来越多，在低功耗边缘设备上执行深度神经网络推理任务已经成为确切的需求。大部分低功耗边缘微处理器使用 ARM 指令架构，同时，由于RISC-V的开源和灵活的拓展性，使用RISC-V指令架构的低功耗边缘微处理器具有广阔的前景。

随之而来的一个问题是，目前主流的深度学习框架（如TensorFlow、PyTorch等）并不能很好的支持RISC-V指令架构的微处理器，研究人员往往需要通过构建复杂的交叉编译工具链，或者自定义指令集拓展的方式来部署神经网络模型[4]。在IoT设备上，虚拟化技术在快速部署的过程中起了重要作用。平台虚拟化技术允许在宿主机设备中运行多个异构的体系结构应用，通过虚拟机监视器（Virtual Machine Monitor，VMM，或称为Hypervisor）为用户提供抽象、虚拟的硬件环境。传统的指令级和硬件抽象级虚拟化技术，例如VMware ® Workstation和Virtual PC，提供了良好的跨平台兼容性，但对原生系统性能要求高、资源占用多，运行响应慢，不适用硬件资源有限的边缘设备的应用场景。为了节省边缘设备上有限地硬件资源，用户需要在操作系统层面使用更加轻量级的虚拟化技术，在提供高质量的虚拟环境的同时，降低对系统性能的影响。容器技术通过名字空间Namespace为每个容器提供特定的命名空间，对进程实现隔离，相对于传统的虚拟机，容器技术具有更少的系统占用，更快的启动速度和更高的资源利用率。Docker是目前最为常用的容器技术，但Docker尚不支持RISC-V架构，使用容器技术在RISC-V上快速部署应用是一个亟待解决的问题。

本文尝试解决在资源受限的RISC-V架构上的强化学习虚拟化问题，结合容器化技术的思想，通过命名空间隔离系统进程，在操作系统中实现轻量级的虚拟化技术。为了对结果，我们选择在QEMU上模拟RISC-V指令架构的微处理器作为实验平台，比较多种不同强化学习算法的异构模型在实验平台上的性能表现，最终完成的工作如下：

（1） 首先，通过采用容器化技术，减少上层软件构建虚拟化代价，去除冗余中间件，定制命名空间隔离特定进程，有效提升学习任务资源利用率，实现模型的快速执行；

（2） 其次，利用RISC-V指令集的特征，进一步优化上层神经网络模型，优化强化学习效率；

（3）最后，实现整体优化及容器化方法系统原型，并通过测试多种基准测试集，初步完成系统原型性能评估。

本文的结构如下：第2节讨论了与本文相关的工作；第3节介绍了轻量级虚拟化系统整体设计；第4节展示了初步的实验结果；最后一节总结全文。

2 相关工作

之前的工作大多是将深度学习的计算负载（卷积、激活、池化等）从RISC-V处理器转移到专用的硬件加速器中[2,6,7]，采用软硬件协同设计的方法实现深度神经网络模型推理的高效部署。这种方法通常需要根据特定用途设计专用的硬件加速器，同时需要相应的自定义函数库、编译器等工具链。

Kong等人[4]提出了AIRV，一个可以RISC-V上部署深度神经网络模型的计算框架，直接在RISC-V平台而不是硬件加速器中运行深度神经网络模型的推理过程。此外还显示在x86平台上交叉编译RISC-V目标架构的深度学习推理应用具有更短的开发周期。

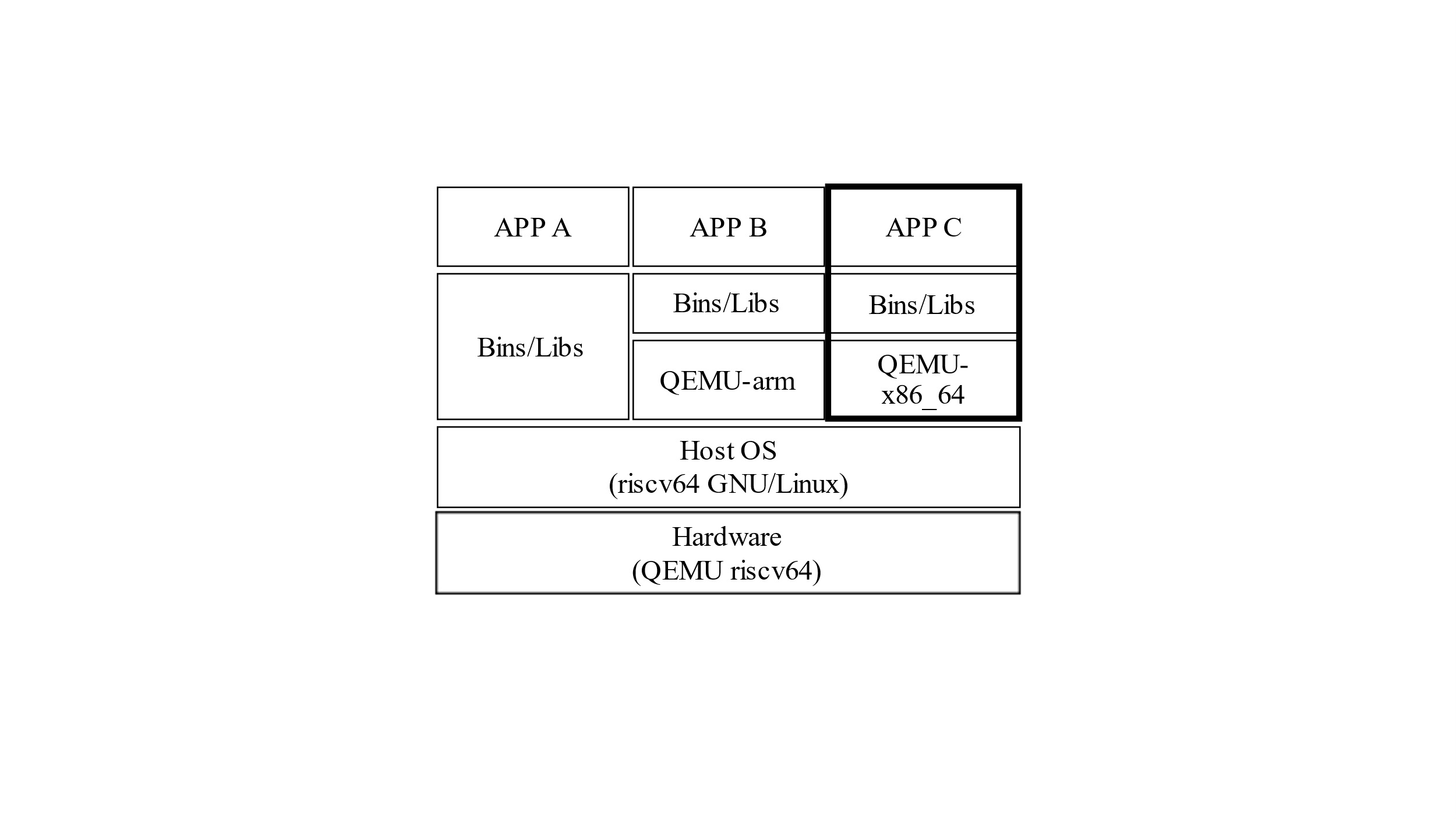
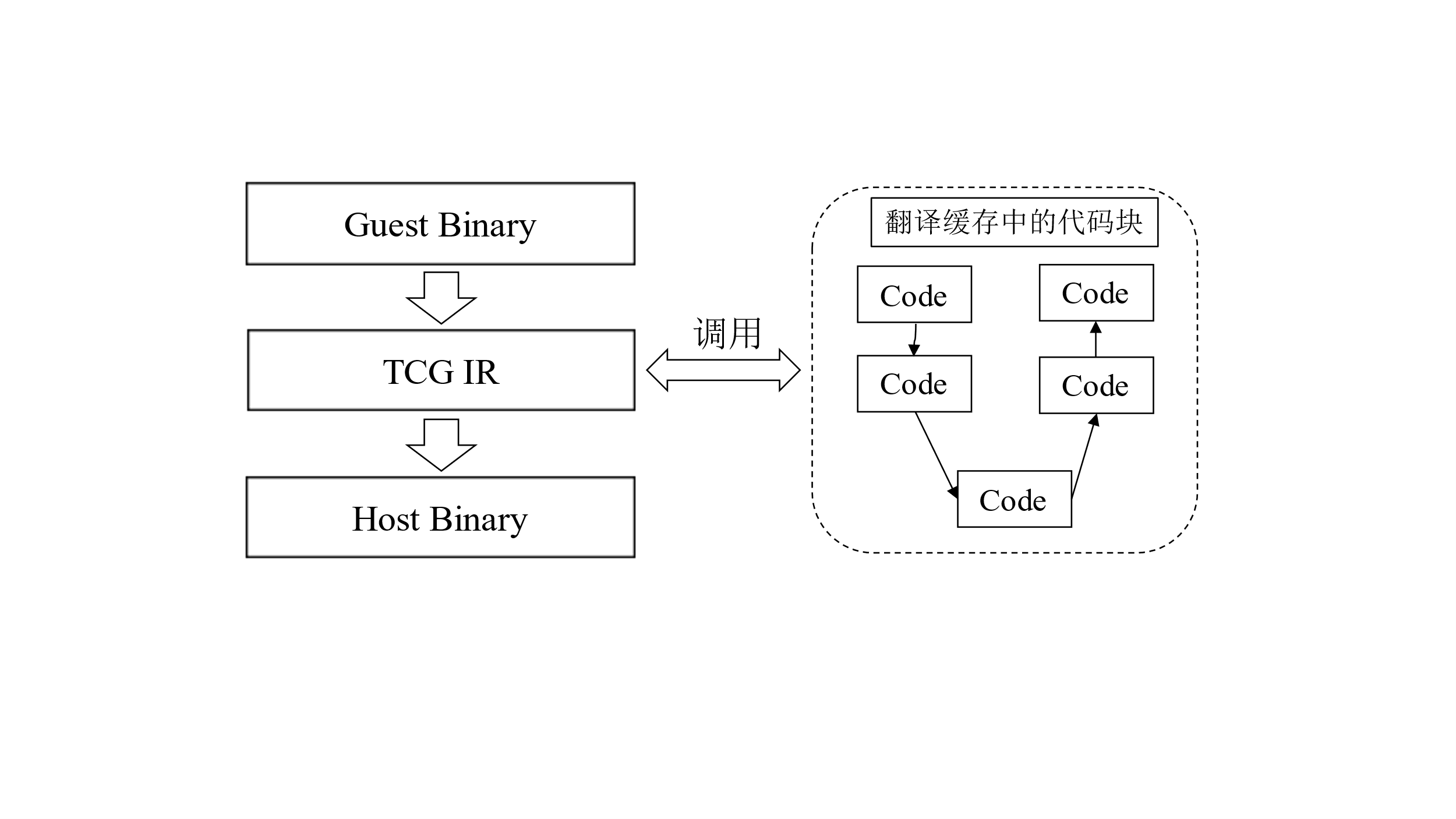
Louis等人[8]使用RISC-V V矢量拓展模块，并在此基础上增加一层软件结构，修改了TensorFlow Lite C/C++ 库中的部分函数。为RISC-V指令集交叉编译了TensorFlow Lite源码，并在RISC-V模拟器Spike上进行验证。这种方法支持在RISC-V上部署多种网络模型，同时实验结果表明，使用这种实现可以将向处理器提交的指令数减少8倍。

3 系统设计

3.1 虚拟化技术

虚拟化技术的目的是在一台主机上运行多个进程，将硬件资源抽象为虚拟逻辑对象的技术，包括计算机的硬件资源、存储设备和网络资源的虚拟等。虚拟化技术包括平台虚拟化、硬件虚拟化、应用程序虚拟化等，平台虚拟化技术允许在宿主机设备中运行多个异构的体系结构应用，通过虚拟机监视器（Virtual Machine Monitor，VMM，或称为Hypervisor）为用户提供抽象、虚拟的硬件环境。通过在软件和硬件层面引进虚拟化层，将软件和硬件分离。真实的硬件环境称之为宿主机，虚拟化实现的环境称之为虚拟机。在虚拟化的操作系统中执行应用程序，可以提高应用的安全性。

VMM通过内核代码的二进制翻译实现虚拟化，在宿主机和虚拟机之间添加一层中间层，将宿主机处理器的指令代码转换、翻译成目标处理器的指令集，捕获文件执行时所需的系统调用。VMware ® Workstation、Virtual PC、 QEMU等均是采用的这种方法实现硬件的虚拟化。

QEMU是通过纯软件方式实现硬件的虚拟化过程，在RISC-V指令架构的硬件平台上可使用QEMU模拟外部硬件，通过这种方式可执行异构文件。 QEMU既可实现全系统硬件虚拟化，也可通过Namespace在操作系统级别实现容器级别的虚拟化。

**图2 TCG的翻译过程**

**图 1 系统整体架构设计，使用QEMU模拟RISC-V微处理器作为实验平台，实现操作系统级虚拟化，虚拟化的进程之间互不干扰**

3.2 软件环境

QEMU是一个可执行硬件虚拟化的开源托管虚拟机。本文使用QEMU模拟RISC-V处理器内核作为实验平台。在RISC-V处理器上运行Linux系统，实现操作系统级虚拟化。图1展示了系统整体架构。

在Linux系统中，使用 QEMU 的User Mode模拟命名空间 Namespace，隔离Linux系统进程。在 User Mode 下，QEMU不会仿真所有硬件，而是通过内核代码的二进制翻译运行异构应用。异构文件在执行时，通过binfmt\_misc识别可执行文件格式并传递至QEMU的用户空间中，QEMU将注册的二进制格式处理程序异构拦截、转换成本地指令架构代码，同时将系统调用按需从目标系统转换成当前系统，并将其转发至本地主机内核。其中的转换主要是通过QEMU中TCG ( Tiny Code Generator )模块实现。

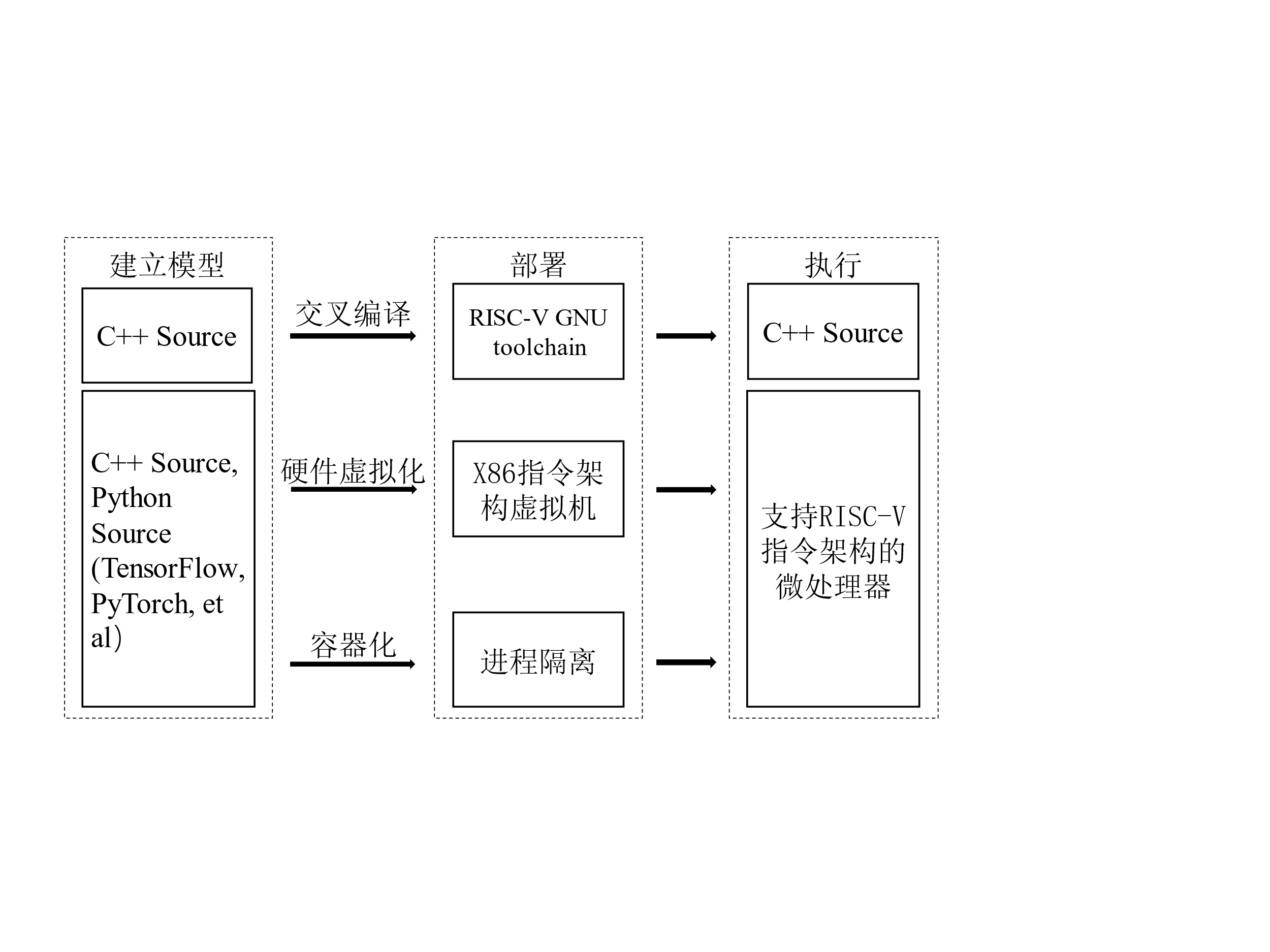
TCG 定义了一系列IR (Intermediate Representation )，已经翻译的代码块放在转换缓存中，并通过跳转指令将源处理器的指令集（ISA）和目标处理器的指令集（ISA）链接在一起。当`Hypervisor`（虚拟机管理程序）在执行代码时，存放于转换缓存中的链接指令可以跳转到指定的代码块，并且执行可以在不同的已翻译代码块上运行，直到需要翻译新块为止。在执行的过程中，如果遇到了需要翻译的代码块，执行动作就会暂停并回会跳回到Hypervisor，Hypervisor就会使用和协调TCG对需要进行二进制翻译的源处理器指令集（ISA）进行转换和翻译并存储到转换缓存中。图二展示了TCG的工作示意图。

4 系统评估

实验平台为 QEMU 模拟的RISC-V指令架构的4核虚拟处理器，内存为2G，主频为1.7 GHz。

表2中列出了本文评估系统可行性及性能使用的强化学习模型。强化学习在自动驾驶、连续控制等领域的表现甚至可以和人类相媲美[5]。本文在实验评估过程中模拟连续控制领域的经典问题——Cart-Pole模型：木棍在一个可移动的小车上竖立，通过学习使木棍竖立的时间尽量长。

在x86平台使用PyTorch[8]在Gym[9]环境下构建深度神经网络模型，并部署在RISC-V实验平台上。测试集的强化学习算法包括：

1. 随机代理 (Random Policy )
2. 交叉熵 (Cross-entropy)
3. 策略梯度 (Policy Gradient )

**图3 在RISC-V平台上部署深度神经网络模型的三种工作流，分别为交叉编译、硬件虚拟化、容器化**

图3展示了三种在RISC-V处理器上部署深度神经网络模型的工作流。直接采用交叉编译的方式[7]部署深度学习网络模型可以得到较好的性能表现，同时易于针对特定应用对模型进行优化。但这种方式需要对每个模型单独配置环境，对开发环境要求高，不利于模型快速部署。硬件虚拟化的方式使用QEMU在RISC-V平台上模拟全系统硬件，具有良好的跨平台拓展性，但是会占用大量系统资源，启动速度慢，性能比宿主机性能略差。容器化方式在相互隔离的进程中运行模型文件，定制命名空间隔离特定进程，实现模型训练快速执行。

初步实验结果表明，硬件虚拟化技术占用大量磁盘资源；需要在宿主机上先运行虚拟机，再执行进程，启动速度较慢；受系统资源限制，可同时运行的虚拟机数量有限；性能比宿主机进程差；由于使用了多层虚拟化，资源利用率较低。而容器化技术磁盘占用空间取决于制作的镜像文件大小；可直接在宿主机进程中启动，启动速度快；可同时运行多个进程；性能接近宿主机本地进程；资源利用率高。表1列出了两种虚拟化的方式的优缺点比较。

**图4 硬件虚拟化、容器化方案和x86下交叉编译并部署在直接执行测试集中深度神经网络模型的情况相比较，****Y轴坐标为模型训练时间以10为底取对数。**

我们比较了测试集算法在硬件虚拟化和容器化方案下的运行时间，和在x86平台上直接运行的情况相比较，并在图4中展示了得到的结果。

**表1 容器技术与虚拟机技术对比**

|  |  |  |
| --- | --- | --- |
|  | 容器化 | 硬件虚拟化 |
| 磁盘占用空间 | 一般为MB | 一般为GB |
| 启动速度 | 秒级 | 分钟级 |
| 运行状态 | 操作系统上 | Hypervisor上 |
| 并发性 | 可运行多个容器 | 一般几十个 |
| 性能 | 接近原生 | 弱于原生 |

图5展示了选择一种深度学习算法后，经过三种工作流在RISC-V平台上部署深度神经网络模型的时间周期。

**图5 以x86平台下的模型部署时间为对比，比较三种工作流下的模型部署时间Y轴坐标为模型训练时间以10为底取对数。**

**图6 综合实验结果，对三种工作流从模型移植的部署难易程度和移植后的资源占用两个维度定性分析**

实验结果表明，使用交叉编译方式部署模型虽然具有最少的性能损失，但具有最长的开发周期，这是因为在移植模型的过程中需要对库中所有特定的机器源代码进行修改；在设置完毕虚拟机环境后，硬件虚拟化的方式可以大幅减少模型部署时间的消耗，但和交叉编译方式相比会带来额外约620%资源占用；容器化技术和传统RISC-V架构下交叉编译深度神经网络模型的方法相比，仅付出较小的额外80%性能代价，能快速实现更多、更复杂的深度学习软件框架的部署及运行；与硬件虚拟化方法相比，基于RISC-V的模型具有近似的部署时间，并减少大量的性能损失。初步实验结果证明，容器化及其上的优化方法是解决基于RISC-V架构的软件及学习模型快速部署的一种有效方法。

5 结束语

本文在资源有限的RISC-V处理器平台上实现了一种轻量级的容器化实现。和传统的硬件虚拟化相比，与硬件虚拟化方法相比，基于RISC-V的模型具有近似的部署时间，并减少大量的性能损失，减少宿主机工作负载。本文使用强化学习模型作为测试集的实验证明，该方法的可行性。在目前流行的Docker等容器化技术不支持RISC-V平台的背景下，本文提出的容器化方法是解决基于RISC-V架构的软件及学习模型快速部署的一种有效方法。目前在RISC-V平台上对各种虚拟化方案性能方面的探索仍有待于进一步研究，未来我们会对深度神经网络模型进行量化、减枝等操作，针对特定领域对模型进行专门优化，构建模型制作、镜像打包、容器化技术部署在RISC-V硬件平台上的完整工作流。

参考文献：

[1] Greengard, S amuel. "Will RISC-V revolutionize computing?." (2020): 30-32.

[2]J. Ramakrishnan, M. S. Shabbir, N. M. Kassim, P. T. Nguyen和D. Mavaluru, 《A comprehensive and systematic review of the network virtualization techniques in the IoT》, International Journal of Communication Systems, 卷 33, 期 7, 页 e4331, 2020, doi: 10.1002/dac.4331.

[3]J. L. Hennessy和D. A. Patterson, 《A new golden age for computer architecture》, Commun. ACM, 卷 62, 期 2, 页 48–60, 1月 2019, doi: 10.1145/3282307.

[4]Y. Kong, 《AIRV: Enabling Deep Learning Inference on》, 页 9.

[5]T. P. Lillicrap等, 《Continuous control with deep reinforcement learning》, arXiv:1509.02971 [cs, stat], 7月 2019, 见于: 4月 28, 2020. [在线]. 载于: http://arxiv.org/abs/1509.02971.

[6]E. Flamand等, 《GAP-8: A RISC-V SoC for AI at the Edge of the IoT》, 收入 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 7月 2018, 页 1–4, doi: 10.1109/ASAP.2018.8445101.

[7]S. Davidson等, 《The Celerity Open-Source 511-Core RISC-V Tiered Accelerator Fabric: Fast Architectures and Design Methodologies for Fast Chips》, IEEE Micro, 卷 38, 期 2, 页 30–41, 3月 2018, doi: 10.1109/MM.2018.022071133.

[8]M. S. Louis等, 《Towards Deep Learning using TensorFlow Lite on RISC-V》, 页 6, 2019.