CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 43 卷第 2 期 2021 年 2 月
Vol. 43, No. 2, Feb. 2021

文章编号:1007-130X(2021)02-0266-08

基于 RISC-V 架构的强化学习容器化方法研究 *

徐子晨,崔　傲,王玉皞,刘　韬

(南昌大学信息工程学院,江西 南昌 330031)

摘　要:RISC-V 作为近年来最热门的开源指令集架构,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、运行和测试机器学习框架是一个亟待解决的技术问题。使用虚拟化技术可以解决跨平台的模型部署和运行问题。但是,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。讨论在资源受限的 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,提高强化学习效率;最后,实现整体优化和容器化方法系统原型,并通过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,仅付出相对较小的额外性能代价,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 Hypervisor 虚拟机方法相比,基于 RISC-V 的模型具有近似的部署时间,并大量减少了性能损失。初步实验结果表明,容器化及其上的优化方法是实现基于 RISC-V 架构的软件和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

中图分类号:TP391.92　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2021.02.010

A containerization method for reinforcement learning based on RISC-V architecture

XU Zi-chen, CUI Ao, WANG Yu-hao, LIU Tao

(School of Information Engineering, Nanchang University, Nanchang 330031, China)

Abstract: As the hottest open-source instruction set architecture in recent years, RISC-V is widely used in a variety of domain-specific microprocessors, especially for modular customization in the field of machine learning. However, existing RISC-V applications require recompilation or optimization of legacy software or models on the RISC-V instruction set. Therefore, how to rapidly deploy, run, and test machine learning frameworks on RISC-V architectures is a pressing technology challenges. The use of virtualization technology can solve the problem of deploying and running models across platforms. However, traditional virtualization techniques, such as virtual machines, are often not applicable to RISC-V architecture scenarios due to their high performance requirements for native systems, high resource footprint, and slow operational response. Discussion of reinforcement learning virtualization on resource-constrained RISC-V architectures. Firstly, by adopting containerization technology, reducing the cost of virtualization for upper-level software builds, removing redundant middleware, and customizing

* **收稿日期**:2020-05-03;**修回日期**:2020-07-06
基金项目:国家自然科学基金(61702250);国家重点研发计划(2018YFB14043033);国家核高基(2018ZX01035-101);中科院计算机体系结构国家重点实验室开放课题(CARCHB202017)
通信作者:刘韬(tliu@ncu.edu.cn)
通信地址:330031 江西省南昌市南昌大学信息工程学院
Address: School of Information Engineering, Nanchang University, Nanchang 330031, Jiangxi, P. R. China

namespaces to isolate specific processes, we effectively improve the resource utilization for learning tasks and achieve the rapid execution of model training. Secondly, the features of the RISC-V instruction set are used to further optimize the upper neural network model and optimize the reinforcement learning efficiency. Finally, a system prototype of the overall optimization and containerization method is implemented and the performance evaluation of the prototype is completed by testing multiple benchmark test sets. Containerization techniques enable the rapid deployment and operation of more complex and deep learning software frameworks at a relatively small additional performance cost, compared to traditional methods of cross-compiling deep neural network models in RISC-V architectures. RISC-V based models have approximate deployment time and reduce substantial performance losses compared to the hypervisor VM method. Preliminary experimental results demonstrate that containerization and the optimization method on it are an effective way to achieve the rapid deployment of software and learning models based on RISC-V architecture.

Key words: virtualization; neural network; 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[1]。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,和前代相比它最大的优势在于开源和模块化,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,推出了一系列支持乱序执行的微处理器,如伯克利弹性乱序处理器 BROOM (Berkeley Resilient Out-of-Order Machine)等[2-4],将会应用在可穿戴设备、智能家居、机器人、自动驾驶和工业装置等领域的计算设备中[5],在边缘微设备的应用中具有广阔的前景。

机器学习的研究在近些年来的被关注度越来越高,尤其是在深度学习领域,各种深层网络模型层出不穷,在计算机视觉、语音识别和自然语言处理等领域的应用也越来越密集。构建一个深度神经网络的工作流程通常包括以下几步:(1)收集和准备训练数据;(2) 选择并优化深度学习算法;(3)训练并调整模型;(4)在生产环境中部署模型。机器学习可分为监督学习、无监督学习和强化学习,近年来,深度强化学习 Deep RL(Deep Reinforcement Learning)在自动驾驶、连续控制等领域的表现优异[6],但深度强化学习中的智能体训练时间长、计算力需求大,已成为限制深度强化学习进一步发展的瓶颈之一。强化学习的训练过程中带有大量的循环,适合在支持乱序执行的 RISC-V 处理器上进行加速。因此,在基于 RISC-V 指令架构的平台上构建深度强化学习模型,探索潜在的计算加速具有十分重要的意义。

深度学习模型的快速部署和推理应用是 RISC-V 架构面临的一个新挑战。目前传统的基于 Python 的深度学习框架(TensorFlow、PyTorch 和 MXNet 等)尚不支持 RISC-V 指令架构。欲在 RISC-V 平台上执行一个深度神经网络模型的推理过程,研究人员往往需要构建复杂的交叉编译工具链,修改深度学习库中特定的机器源代码,自定义指令集拓展[7],不利于模型快速部署和优化。我们希望在 RISC-V 平台上探索一种快速部署模型方式,为深度强化学习应用的加速提供新思路。

快速部署的解决方案首推虚拟化技术。传统的虚拟化技术通过虚拟机监视器 VMM(Virtual Machine Monitor 或称为 Hypervisor)实现,允许在宿主机设备上运行多个异构的体系结构应用,为用户提供抽象、虚拟的硬件环境。使用 Hypervisor 实现的虚拟化产品有 VMware Workstation 和 Virtual PC 等。Hypervisor 提供了良好的跨平台兼容性,但在 RISC-V 架构上直接使用虚拟化技术有以下几个问题:(1)每个虚拟机都需要运行一个完整的操作系统和大量的应用程序;(2)资源占用多;(3)运行响应慢[8]。在实际开发环境中,我们更关注自己部署的应用程序。其次,端设备上的硬件资源可能有限,用户需要使用在操作系统层面实现的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。在云计算框架中,面向轻量级软件虚拟化,Pahl[9]提出了容器化的解决方案。容器化技术通过命名空间(Namespace,表示一个标识符的可见范围)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,容器化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,在容器的基础上从文件系统、网络互联到进程隔离等进行了进一步的封装,极大简化了容器的创建和维护,使得Docker技术比虚拟机技术更为轻便、快捷。但是,Docker尚不支持RISC-V架构,使用容器技术在RISC-V平台上实现模型的快速部署是一个亟待解决的问题。

本文针对强化学习这一领域,对基于RISC-V架构的端设备上的强化学习容器化方法进行研究。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,实现整体优化和容器化方法系统原型,并通过多种基准测试集完成系统原型性能评估。在原型系统实现里,本文使用QEMU(Quick Emulator)模拟器仿真的RISC-V指令架构作为实验平台,叠进式设计、实现和测试多种强化学习优化算法,讨论模型在RISC-V平台上的可移植性和性能表现。

本文的结构如下:第2节讨论RISC-V上虚拟化相关的工作;第3节提出了一种RISC-V架构下的容器化方法;第4节对初步的实验结果进行评估;最后一节总结目前得到的结论以及工作的不足之处。

2 相关工作

2.1 虚拟化技术

虚拟化技术是在一台主机上运行多个进程,将硬件资源(包括计算机的硬件资源、存储设备和网络资源等),抽象为虚拟逻辑对象的技术。虚拟化技术包括平台虚拟化、硬件虚拟化和应用程序虚拟化等,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VMM为用户提供抽象、虚拟的硬件环境。Popek等人[10]为将系统软件视为VMM定义了3个基本特征:(1)保真。VMM上的软件的执行与硬件上的执行相同,除非定时影响。(2)性能。绝大多数虚拟机指令由硬件执行,而无需VMM干预。(3)安全。VMM管理所有硬件资源。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,捕获文件执行时所需的系统调用。VMware Workstation、Virtual PC和QEMU等均采用这种方法实现硬件的虚拟化。Adams等人[11]对x86架构下的软硬件虚拟化技术进行了比较,得出硬件VMM的性能通常比纯软件VMM的性能低的结论。硬件虚拟化技术不具备性能优势的原因主要有2个:(1)它不支持内存管理单元MMU(Memory Management Unit)虚拟化。(2)它无法与用于MMU虚拟化的现有软件技术共存。Shuja等人[12]根据ARM架构下移动虚拟化的硬件支持的最新进展,调查了基于软件和硬件的移动虚拟化技术,并介绍了CPU、内存、I/O、中断和网络接口等在移动设备中虚拟化面临的挑战和问题。他们的研究最后提出,在资源受限的移动设备上实施基于CPU的虚拟化解决方案会占用大量CPU周期和内存空间,而使用静态二进制转换实现虚拟化的解决方案开销更低。针对资源有限的边缘设备必须使用资源有效的技术来解决上述问题。Bernstein[13]介绍了Docker和Kubernetes,前者是一个开源项目,可以实现Linux应用程序的自动化快速部署,后者是一个用于Docker容器的开源集群管理器。

2.2 基于RISC-V架构的加速和优化

在RISC-V平台上有关深度学习的工作大多是将深度学习的计算负载(卷积、激活和池化等)从RISC-V处理器转移到专用的硬件加速器上[7,14,15],采用软硬件协同设计的方法加速深度神经网络模型推理计算。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。

在RISC-V平台上部署模型最简单的方法是直接在RISC-V上编译深度神经网络模型,但由于硬件资源和模型性能存在限制,实际开发中通常采用交叉编译的方式来部署模型。Kong[16]提出了一个可以在RISC-V硬件平台(FPGA、QEMU模拟器等)上部署深度神经网络模型的计算框架RISC-V人工智能框架AIRV(AI on RISC-V),允许在RISC-V平台而不是硬件加速器上运行深度神经网络模型的推理过程。此外还证明了相比于直接在RISC-V平台上编译网络模型,在x86平台上交叉编译RISC-V目标架构的深度神经网络模型具有更高的资源利用率。Louis等人[7]在RISC-V指令集中的V矢量拓展模块基础上增加一层软件结构,修改了TensorFlow Lite C/C++库函数,此外还为RISC-V指令集交叉编译了TensorFlow

Lite 源码,并在 RISC-V 模拟器 Spike 上进行了验证。这种方法支持在 RISC-V 平台上部署多种网络模型,实验结果表明,使用这种方法可以将向处理器提交的指令数减少至直接编译方法的 1/8。Vega 等人[17]提出一种 RISC-V I/O 虚拟化 RV-IOV (RISC-V I/O Virtualization)的硬件,解决了部署 Rocket 内核时的资源限制问题。Rocket Chip 是一个开源的 System-on-Chip 设计生成器,可生成支持 RISC-V 指令集的通用处理器核心,并提供有序核心生成器(Rocket)和无序核心生成器(BOOM)[18]。RV-IOV 使用 I/O 虚拟化技术将 Rocket 内核与主机解耦,从而可以利用 ASIC 或更大规模的 FPGA 实现内核。

2.3 深度强化学习

强化学习 RL (Reinforce Learning)和监督学习、无监督学习都是机器学习任务的一种,与监督学习和无监督学习任务相比,强化学习强调智能体(Agent)在与环境(Environment)的交互中学习,利用评价性的反馈信号实现决策的优化[19]。强化学习的主要过程为:智能体在环境中学习,根据环境的状态(State),执行动作(Action),并根据环境的反馈(Reward)做出更好的决策。强化学习的马尔可夫决策过程可以表示为:

$$M=\langle S,A,P_{s,a},R\rangle \tag{1}$$

其中,S 表示有限状态集合,s 表示某个特定状态,$s\in S$;A 表示有限动作集合,a 表示某个特定动作,$a\in A$;转移模型:$T(S,a,S')\sim P_r(s'|s,a)$:根据当前状态 s 和动作 a 预测下一个状态 s',这里的 P_r 表示从 s 采取行动 a 转移到 s' 的概率;奖励 $R(s,a)=E[R_{t+1}|s,a]$ 表示 Agent 采取某个动作后的即时奖励,它还有 $R(s,a,s')$ 和 $R(s)$ 等表现形式,采用不同的形式,其意义略有不同。

强化学习可以分为基于值函数的强化学习和基于策略的强化学习。基于值函数的强化学习算法包括 Q-learning 和 Deep Q-learning 等,基于策略的强化学习算法包括策略梯度等[20]。

3 RISC-V 容器化方法

本文提出了一种 RISC-V 容器化方法,在其原型系统设计中,使用 QEMU 作为容器化引擎,定制命名空间隔离特定进程,在操作系统级别实现容器化。图 1 展示了原型系统的整体架构,左边是传统的使用 Hypervisor 的虚拟机架构,右边是容器化架构。原型系统使用 QEMU 模拟 RISC-V 处理器内核作为实验硬件平台,在 RISC-V 处理器上运行 Linux 系统,并安装 QEMU。

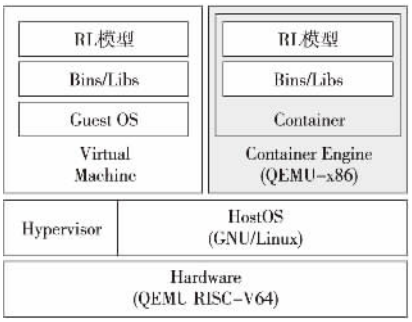

Figure 1 Architecture of the prototype system

图 1 原型系统整体架构

QEMU 是一个具有跨平台的特性、可执行硬件虚拟化的开源托管虚拟机,可通过纯软件方式实现硬件的虚拟化,模拟外部硬件,为用户提供抽象、虚拟的硬件环境。QEMU 既可实现全系统硬件虚拟化,也可在 User Mode 下通过为每个容器提供特定的命名空间实现容器化设计。在 User Mode 下,QEMU 不会仿真所有硬件,而是通过内核代码的 TCG (Tiny Code Generator)模块对异构应用的二进制代码进行翻译和转换。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。binfmt_misc 是 Linux 内核的一种功能,它允许识别任意可执行文件格式,并将其传递给特定的用户空间应用程序,如仿真器和虚拟机。QEMU 将注册的异构二进制程序拦截、转换成本地指令架构代码,同时按需从目标架构系统调用转换成宿主机架构系统调用,并将其转发至宿主机内核。TCG 定义了一系列 IR (Intermediate Representation),将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和目标处理器的指令集链接在一起。当 Hypervisor 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,目标二进制代码可不断调用已翻译代码块来运行,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,执行会暂停并跳回到 Hypervisor,Hypervisor 使用和协调 TCG 对需要进行二进制翻译的源处理器指令集进行转换和翻译并存储到转换缓存中。图 2 为 TCG 的工作示意图。

4 性能评估和分析

4.1 实验设计

实验设计使用 QEMU 模拟的 RISC-V64 位

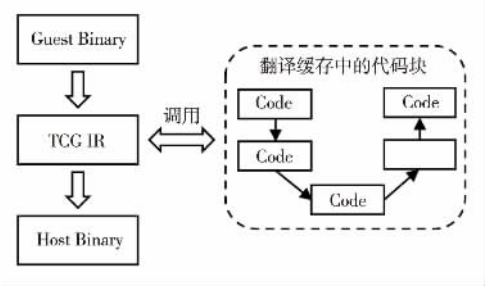

Figure 2 Tiny Code Generator translation process

图 2 Tiny Code Generator 翻译过程

指令架构 4 核处理器,内存为 2 GB,主频为 1.7 GHz,安装 Linux 系统和 QEMU。图 3 展示了在 RISC-V 处理器上部署深度神经网络模型的流程。本节设置了 3 组对照实验:(1)交叉编译;(2)Hypervisor 虚拟机;(3)容器化方法。首先在 x86 平台上构建强化学习 RL 模型,实验(1)使用交叉编译的方式[7,16]部署深度学习网络模型。部署过程中需要对每个模型单独配置环境,修改强化学习库函数,构建交叉编译工具链。实验(2)使用 Hypervisor 虚拟机的方式,使用 QEMU 在 RISC-V 平台上进行全系统模拟,在虚拟机上安装基于 x86 架构的操作系统,配置强化学习模型训练环境。实验(3)采用本文提出的容器化方法,使用 QEMU User Mode 在容器化的进程中执行基于 x86 架构的模型,模型文件需将运行所需的依赖库封装至二进制可执行文件,实现模型训练快速执行。

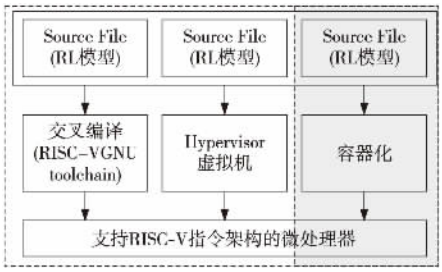

Figure 3 Three processes for deploying a deep neural network model on a RISC-V platform

图 3 在 RISC-V 平台上部署深度神经网络模型的 3 种流程

4.2 强化学习算法

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[6]。图 4 展示了在实验过程中使用强化学习算法解决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,通过学习决定小车的位置,使木棍在小车上竖立的时间尽量长。本节在 x86 平台上使用 PyTorch[21]构建深度神经网络模型,在 Gym[22]环境中模拟 Cart-Pole 模型的训练,并部署在 RISC-V 实验平台上。基准测试集中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)[20],叠进式设计、实现、测试了多种强化学习优化算法,完成原型系统的性能评估。

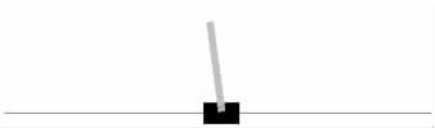

Figure 4 Cart-Pole model

图 4 Cart-Pole 模型

使用随机代理训练的模型不会收敛,只是记录执行一定步数后的奖励(Reward)总和,木棍会在小车上作随机方向运动,无法在小车上保持竖直。交叉熵算法可以实现一个稳定收敛的模型,基本思想是:使用当前策略(从一些随机的初始策略开始)对事件进行采样,并使用本文策略将最成功的样本的负面对数可能性最小化,恰好等于最小化交叉熵。具体做法是:首先,在环境和模型上运行 N 个轮次(Episode),每个轮次都是从开始到结束执行一次算法,计算每个轮次的奖励总和并确定一个奖励边界(如 70%);其次,舍弃所有低于边界值的轮次;最后,在剩余的轮次中进行训练;重复上述步骤直到对结果满意为止。基于策略梯度的训练和基于交叉熵的训练相比,在轮次的分割中具有更细的粒度。策略梯度的基本原理是通过反馈调整策略:在得到正向奖励时,提高相应动作的概率;得到负向奖励时,降低相应动作的概率。策略梯度具有以下优点:(1)更好的收敛性;(2)适合高维度或连续状态空间;(3)不必计算复杂的价值函数。

4.3 实验结果

本节首先进行了交叉编译、Hypervisor 虚拟机和容器化方法的性能比较实验。图 5 展示了随机代理、交叉熵和策略梯度算法在 3 种部署方式下的不同性能表现,比较了强化学习模型在 3 种方式下的模型训练时间。其中,交叉编译方式运行的模型训练时间最短;Hypervisor 虚拟机方式运行模型的时间最长,和交叉编译方式相比大约增加了 100 倍以上的模型训练时间;容器化方法的模型训练时间居于二者之间,比交叉编译方式约增加了 10 倍以上的模型训练时间。

图 6 展示了随机代理、交叉熵和策略梯度算法在 3 种方式下部署模型需要修改的代码行数。在所有的算法中,交叉编译方式除了对模型本身文件进行修改外,还需要修改深度学习模型依赖库(如 Numpy,Scipy 和 Gym 等)中的函数,修改代码数

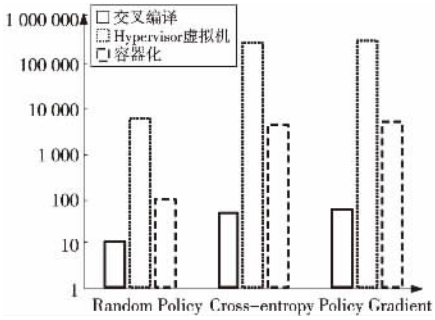

Figure 5 Performance comparison of different algorithms under three processes

图 5 3 种部署流程下不同算法性能比较

量最多;使用 Hypervisor 虚拟机方式只需对模型本身文件进行修改,所需修改的代码量最少;容器化技术需要将深度学习模型文件和所需依赖库进行封装,无需对库函数进行修改,和 Hypervisor 虚拟机相比增加了约 40%的代码。

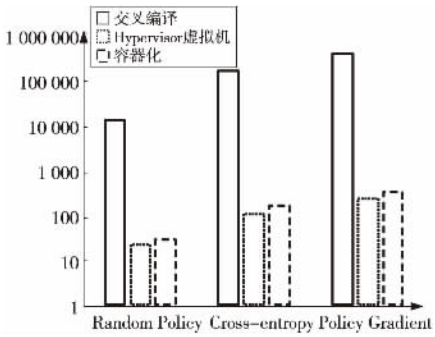

Figure 6 Number of code lines to be modified when deploying model

图 6 部署模型需要修改的代码行数

本节进一步对端到端的模型训练和运行时延进行了分析,如图 7 所示。3 种部署流程下建立模型的时间相同,部署到 RISC-V 指令架构的实验平台的过程中,交叉编译的方式耗费时间最多,Hypervisor 虚拟机方式耗费的时间最少。和虚拟机方式相比,容器化方式在模型部署过程中多耗费了约 30%的时间开销;和交叉编译方式相比,减少了约 85%的时间开销。

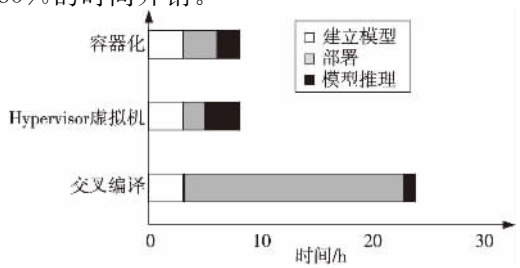

Figure 7 Comparing the total time to build the model, deploy the model in RISC-V, and train the model under three workflows

图 7 比较 3 种部署流程下建立模型、在 RISC-V 部署模型和训练模型的总时间

4.4 评估分析

图 8 综合了图 5～图 7 的实验结果,其中,X 轴表示模型部署时间,Y 轴表示模型训练时间,气泡大小表示模型部署包括的代码数量。从图 8 中可以看出,容器化方法在 3 个指标上均处于中间位置。和交叉编译的方式相比,容器化方法付出了约 10 倍的性能代价,减少了 85%的模型部署时间和 95%以上的代码数量;和虚拟机的方式相比,仅增加了约 30%的模型部署时间和约 40%的代码数量,减少了 100 倍的模型训练时间,大幅提升了模型性能。

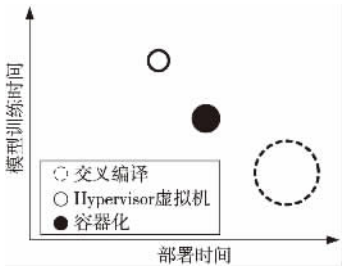

Figure 8 Consolidated experimental

图 8 综合实验结果

初步的实验结果表明,使用交叉编译方式部署模型虽然可以得到最高的模型性能,但部署周期最长,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,Hypervisor 虚拟机方式可以大幅减少模型部署时间的消耗,但每次模型训练都需要运行完整的虚拟机,加载工作环境,和交叉编译方式相比会带来额外的资源占用;容器化方式和传统 RISC-V 架构下交叉编译深度神经网络模型的方法和使用 Hypervisor 虚拟机的方式相比,仅付出相对较小的额外性能代价,在近似的部署时间内,可实现更多、更复杂的深度学习软件框架的快速部署和运行。因此,容器化方式及其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。

4.5 不足与展望

在容器化的流程中,需将模型文件和依赖的库函数一同封装成镜像文件,但目前的封装方式较为粗糙,复杂模型打包后的镜像文件占用了 GB 级别的磁盘空间,我们将逐步完善整体工作流程,对模型实现更细粒度的封装,进一步减少资源占用。

强化学习的计算包含大量的循环过程,通过指令的乱序执行可以大幅提高指令的并行性,提高计算效率。未来会使用支持乱序执行的 RISC-V 处理器核心对强化学习的计算过程进行优化。

5 结束语

使用虚拟化技术可以解决跨平台的模型快速部署和运行问题。但是,传统的虚拟化技术,例如虚拟机,对原型系统性能要求高,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。

本文首先通过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,提高强化学习效率;最后,实现整体优化和容器化方法系统原型,并通过多种基准测试集完成系统原型性能评估。容器化技术和在传统 RISC-V 架构上交叉编译深度神经网络模型的方法相比,仅付出相对较小的额外性能代价,就能快速实现更多、更复杂的深度学习软件框架的部署和运行;与 Hypervisor 虚拟机方式相比,基于 RISC-V 的模型具有近似的部署时间,并减少了大量的性能损失。初步实验结果表明,容器化方式及其优化方法是解决基于 RISC-V 架构的软件和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研究仍有待进一步探索,未来将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,形成基于 RISC-V 架构的深度神经网络模型构建、镜像打包、容器化技术部署的完整流程。

参考文献:

[1] Hennessy J L,Patterson D A. A new golden age for computer architecture [J]. Communications of the ACM,2019,62(2):48-60.

[2] Celio C,Chiu P F,Asanovic K,et al. BROOM: An open-source out-of-order processor with resilient low-voltage operation in 28-nm CMOS [J]. IEEE Micro,2019,39(2):52-60.

[3] Zhang Kun-ning, Zhao Shuo, He Hu, et al. Convolution-accelerated SoCs based on RISC-V processors [J/OL]. Computer Engineering[2020-04-25]. https://doi.org/10.19678/j.issn.1000-3428.0057835. (in Chinese)

[4] Yang Wei-ke. Research on design method of convolution neural network accelerator based on RISC-V open source processor[D]. Shanghai: Shanghai Jiao Tong University,2018. (in Chinese)

[5] Greengard S. Will RISC-V revolutionize computing? [J]. Communications of the ACM, 2020,63(5):30-32.

[6] Lillicrap T P,Hunt J J,Pritzel A,et al. Continuous control with deep reinforcement learning [C]// Proc of the ICLR (Poster),2016:1.

[7] Louis M S, Azad Z,Delshadtehrani L,et al. Towards deep learning using TensorFlow lite on RISC-V [C]// Proc of the ACM Workshop on Computer Architecture Research Using RISC-V,2019:51-56.

[8] Ramakrishnan J,Shabbir M S,Kassim N M,et al. A comprehensive and systematic review of the network virtualization techniques in the IoT [J]. International Journal of Communication Systems,2020,33(7):e4331.

[9] Pahl C. Containerization and the PaaS cloud [J]. IEEE Cloud Computing,2015,2(3):24-31.

[10] Popek G J,Goldberg R P. Formal requirements for virtualizable third generation architectures [J]. Communications of the ACM,1974,17(7):412-21.

[11] Adams K,Agesen O. A comparison of software and hardware techniques for x86 virtualization [J]. ACM SIGARCH Computer,2006,34(5):2-13.

[12] Shuja J,Gani A,Bilal K,et al. A survey of mobile device virtualization: Taxonomy and state of the art [J]. ACM Computing Surveys (CSUR),2016,49(1):Article 1.

[13] Bernstein D. Containers and cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing,2014,1(3):81-84.

[14] Davidson S,Xie S,Torng C,et al. The Celerity open-source 511-core RISC-V tiered accelerator fabric: Fast architectures and design methodologies for fast chips [J]. IEEE Micro,2018,38(2):30-41.

[15] Flamand E,Rossi D,Conti F,et al. GAP-8: A RISC-V SoC for AI at the edge of the IoT [C]// Proc of the IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP),2018:1-4.

[16] Kong Y Y. AIRV: Enabling deep learning inference on RISC-V [C]// Proc of International Symposium on Benchmarking, Measuring and Optimization,2019:91-98.

[17] Vega L,Taylor M. RV-IOV: Tethering RISC-V processors via scalable I/O virtualization [C]// Proc of the ACM Workshop on Computer Architecture Research Using RISC-V,2017:65-70.

[18] Asanovic K, Patterson D A, Celio C. The Berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-V processor: Technical Reort No. UCB/EECS-2015-167 [R]. Berkeley: University of California,2015.

[19] Zhao Xing-yu,Ding Shi-fei. Review of research on deep reinforcement learning[J]. Computer Science,2018,45(7):1-6. (in Chinese)

[20] Mirhoseini A,Pham H,Le Q V,et al. Device placement optimization with reinforcement learning [C]// Proc of the 34th International Conference on Machine Learning,2017:2430-2439.

[21] Paszke A,Gross S,Massa F,et al. PyTorch: An imperative style,high-performance deep learning library [C]// Proc of the NeurIPS,2019:8024-8035.

[22] Brockman G,Cheung V,Pettersson L,et al. OpenAI Gym [J]. arXiv:1606.01540,2016.

附中文参考文献:

[3] 张坤宁,赵烁,何虎,等. 基于 RISC-V 处理器的卷积加速 SoC

[J/OL]. 计算机工程[2020-04-25]. https://doi.org/10.19678/j.issn.1000-3428.0057835.

[4] 杨维科. 基于 RISC-V 开源处理器的卷积神经网络加速器设计方法研究[D]. 上海:上海交通大学,2018.

[19] 赵星宇,丁世飞. 深度强化学习研究综述[J]. 计算机科学,2018,45(7):1-6.

作者简介:

徐子晨(1985-),男,江西南昌人,博士,教授,CCF 会员(76038M),研究方向为复杂数据感知系统。**E-mail**:xuz@ncu.edu.cn

XU Zi-chen, born in 1985, PhD, professor, CCF member (76038M), his research interest includes complex data-conscious system.

崔傲(1996-),男,山东荷泽人,硕士生,CCF 会员(C3225G),研究方向为虚拟化和高性能计算。**E-mail**:aocui22@outlook.com

CUI Ao, born in 1996, MS candidate, CCF member(C3225G), his research interests include virtualization, and high-performance computing.

王玉皞(1977-),男,湖北汉川人,博士,教授,博士生导师,CCF 会员(78395M),研究方向为宽带无线通信与雷达传感融合系统、信道测量与建模、非线性信号处理、图像与视频处理和机器学习。**E-mail**:wangyuhao@ncu.edu.cn

WANG Yu-hao, born in 1977, PhD, professor, PhD supervisor, CCF member(78395M), his research interests include wideband wireless communication & radar sensing fusion system, channel measurement and modeling, nonlinear signal processing, image & video processing, and machine learning.

刘韬(1978-),男,江西南昌人,硕士,讲师,CCF 会员(C3671G),研究方向为人工智能。**E-mail**:tliu@ncu.edu.cn

LIU Tao, born in 1978, MS, lecturer, CCF member (C3671G), his research interest includes artificial intelligence.