计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):3−14 [doi: 10.15888/j.cnki.csa.008844]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Kubernetes 的 RISC-V 异构集群云任务调度系统①

蒋筱斌[1,2], 熊轶翔[1,2], 张　珩[1], 侯朋朋[1], 武延军[1], 赵　琛[1]

[1](中国科学院 软件研究所, 北京 100190)
[2](中国科学院大学, 北京 100049)
通信作者: 张　珩, E-mail: zhangheng17@iscas.ac.cn

摘　要: 随着在云计算领域得到广泛的应用和关注, 集群容器编排管理平台 Kubernetes 已广泛应用于容器化应用服务的自动部署和发布、应用弹性扩展和回滚更新、故障检测和自我修复等服务场景. 第 5 代精简指令集计算机 (fifth-generation reduced instruction-set computer, RISC-V) 具有精简化、模块化、可扩展和开源 4 大技术特点和优势, 已经得到学术界和工业界的广泛关注. 本文立足于 Kubernetes 生态和 RISC-V 生态的协同研究点, 为 Kubernetes 调度器提供异构指令集架构 (instruction set architecture, ISA) 的云服务任务调度支持. 本文通过对生产环境中 RISC-V 指令集架构的各类计算任务需求进行了量化分析, 发现现有的集群容器编排平台 Kubernetes 不具备调度 RISC-V 指令集架构的计算任务的能力, 尤其是其调度算法无法利用 RISC-V 用户自定义的可扩展指令集架构特性提供高性能的可靠服务. 为解决上述问题, 本文提出了一种创建时调度的 ISAMatch 模型, 综合考虑指令集亲和性、同种指令集架构节点数量和节点资源利用率等多个方面, 实现任务的最佳分配. 本文以现有的集群调度器为基础, 完善其针对多种指令集架构任务的调度需求, 相对比默认调度器正确率 62% (调度 RISC-V 基础指令集任务)、41% (调度 RISC-V 扩展指令集任务)、67% (调度 RISC-V 扩展指令集任务且有“RISC-V”节点匹配标签), 在不考虑资源限制的条件下, ISAMatch 模型可以达到 100% 的任务调度正确率.

关键词: Kubernetes 调度器; RISC-V; ISAMatch

引用格式: 蒋筱斌,熊轶翔,张珩,侯朋朋,武延军,赵琛.基于 Kubernetes 的 RISC-V 异构集群云任务调度系统.计算机系统应用,2022,31(9):3–14. http://www.c-s-a.org.cn/1003-3254/8844.html

Cloud Task Scheduling System on RISC-V Heterogeneous Cluster Based on Kubernetes

JIANG Xiao-Bin[1,2], XIONG Yi-Xiang[1,2], ZHANG Heng[1], HOU Peng-Peng[1], WU Yan-Jun[1], ZHAO Chen[1]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: With the wide attention in the field of cloud computing, the cluster container orchestration management platform Kubernetes has been widely used in service scenarios such as automatic deployment and release of container application service, elastic expansion of application and rollback update, and fault detection and self-repairing. Fifth-generation reduced instruction-set computer (RISC-V) includes four technical characteristics and advantages: fine simplification, modularization, scalability, and open source, and it has attracted extensive attention from academia and industry. Based on the collaborative research of Kubernetes ecology and RISC-V ecology, this study supports scheduling tasks with heterogeneous instruction set architecture (ISA) for the Kubernetes scheduler. Through the quantitative analysis of various computing task requirements of RISC-V instruction set architecture in the production environment, it is found that the existing Kubernetes cannot schedule the computing tasks of RISC-V instruction set architecture, and in particular, it fails to employ the extended instruction set architecture characteristics defined by RISC-V developers to provide high-

① 本文由“RISC-V 技术及生态”专题特约编辑武延军研究员、宋威副研究员、张科高级工程师以及邢明杰高级工程师推荐.
收稿时间: 2022-03-30; 修改时间: 2022-04-25; 采用时间: 2022-05-16; csa 在线出版时间: 2022-07-22

performance and reliable services. In order to solve these problems, this study proposes an ISAMatch model which comprehensively considers the affinity of the instruction set, the number of nodes in the same instruction set architecture, and the utilization of node resources, so as to realize the optimal allocation of tasks. Based on the existing cluster scheduler, this study improves its scheduling requirements for multiple instruction set architecture tasks. In addition, compared with the default scheduler whose accuracy rate is 62% (scheduling RISC-V basic instruction set tasks), 41% (scheduling RISC-V extended instruction set tasks), and 67% (scheduling RISC-V instruction set tasks with "RISC-V" node matching label), ISAMatch model can achieve a task scheduling accuracy of 100% without considering resource constraints.

Key words: Kubernetes scheduler; RISC-V; ISAMatch

1 引言

大数据时代主体表现为数据的整体体量、发展速度和种类都呈现井喷式增长, 随着计算任务所需要处理的数据规模急剧增加, 计算模式从单机多处理器的并行计算发展到多机的分布式计算, 网格计算和云计算. 如今, 云计算以其最少的资源支持最大的用户体量和提供弹性化服务的优势成为最为热门的技术之一[1]. 云计算是基于 TCP/IP 通信协议的将多种计算机技术高度集成的产物, 提供对高性能处理器单元、大容量的分布式内存、高速网络等硬件资源的统一化服务抽象和虚拟系统架构平台[2]. 云计算的核心是虚拟化技术, 将硬件资源抽象成一个巨大的资源池, 对资源进行统一的管理和调度、按需分配, 实现动态的、可弹性伸缩性的、安全的云服务.

作为一种轻量级的虚拟化技术, Docker 具有高速、轻量、扩展性强、快速交付等特点[3]. 与传统虚拟化技术不同, 传统虚拟化技术需要虚拟出硬件并在硬件上运行一套完整的操作系统; 而 Docker 类似于一个应用进程直接运行在宿主机的内核系统上, Docker 容器本身没有硬件虚拟和内核系统, 通过 Linux Cgroups 机制确定允许容器消耗的资源 (如 CPU、内存和网络) 限制[4,5]. Docker 容器通过 Docker Image 镜像进行创建, 容器中包含了应用程序及其所需的整个工具链, 并通过类似于 Linux Namespace 机制进行空间隔离, 以保证多个应用程序之间隔离运行, 因此多个容器可以复用宿主机的硬件和内核系统, 极大的简化了容器的创建、维护和迁移.

容器通过容器编排工具部署到集群的节点上, 目前较为流行的容器编排工具有 Docker Swarm、Mesos、Kubernetes 等. 考虑到 Kubernetes 是目前应用最为广泛的开源容器编排工具之一, 本文选择以 Kubernetes[6] 为基础开展 RISC-V 开源指令集架构的适配和云任务调度器优化研究工作. Kubernetes 是一个可移植的、可扩展的开源平台, 用于管理容器化的工作负载和服务. Kubernetes 工作过程由 Master 节点内部的 kube-controller-manager 组件对集群中 Node (节点)、Pod、Namespace (命名空间) 等对象资源进行统计和管理, 并将待调度的 Pod 交由 kube-scheduler 进行调度, 最终实现 Pod 与工作节点之间的绑定. 本文主要聚焦 Kubernetes 的任务调度器开展性能优化和功能扩展方案研究.

RISC-V 是 2010 年美国加州大学伯克利分校研发并推出的 RISC (精简指令集) 第 5 代架构. 因为其开源性、简洁性、模块性和可扩展性, 吸引了无数产业界和工业界的研究人员的关注. 近几年里国内外涌现了许多突破性的成果, 如阿里巴巴平头哥研发的基于 RISC-V 的高性能处理器"玄铁-910"[7], 中国科学院计算技术研究所在 RISC-V 中国峰会上发布的开源高性能 RISC-V 处理器核"香山"[8], 佐治亚理工大学在 ACM 微体系结构国际会议上开源基于 RISC-V 的 GPU 实现 Vortex GPU[9] 等. 随着云计算、边缘计算等领域的不断发展, 带动了大量碎片化的 IoT 芯片需求, 这些微体系结构且具有特定功能的 IoT 芯片与 RISC-V 开源性、简洁性、模块性和可扩展性的特点完美契合. 随着 RISC-V 指令集架构的芯片设备日益增多, 亟需将 RISC-V 架构计算节点加入基于容器的云服务环境, 充分地融合这类 RISC-V 架构设备算力, 以提供合理高效的资源分配和计算任务协作, 这类需求已成为目前分布式容器编排平台的一大技术难点和挑战. 与 ARM/X86 架构不同, RISC-V 指令集并不是用一套统一的架构来满足各种不同的需求, 它采用模块化的方

式进行组织, 用以提供根据用户计算需求的可扩展性指令集; 其中, 每一个模块都使用一个英文来表示, 共包括 4 个基础指令集和 6 个已获批的扩展指令集. 除此之外, RISC-V 还有多个正在不断完善的扩展指令集. 如最为广泛应用的 V (向量) 扩展[10], 比传统的单指令多数据 (SIMD) 指令更具优势, 解决了 SIMD 指令集冗余和上层软件适配的问题. H (Hypervisor) 扩展[11] 支持监控程序, 提高了在单机上虚拟化部署多个操作系统的效率, 为搭载 RISC-V 芯片的虚拟化优化提供了可能.

目前的容器编排工具都不具备调度 RISC-V 指令集架构需求的任务, 尤其是 RISC-V 提供了用户自定义的扩展指令集的加入, 更是增加了调度的难度. 以最新版本 Kubernetes 为例, 将容器部署到具有异构节点的集群中将会导致容器镜像的拉取失败或是应用程序的运行失败. 标签式的调度方案可以解决粗粒度的架构选择, 但是无法实现 RISC-V 多种扩展指令集 ISA 匹配. 以 V (Vector) 扩展指令集为例, 与当前 ARM/X86 架构不同, RISC-V 的 V 扩展指令提供了向量化计算, 能够为向量计算和标量计算提供单指令多数据 (SIMD) 运算单元, 且不限于特定的向量长度, 能够为矩阵运算、机器学习、多媒体分析等提供指令集加速. 然而, 当前的 Kubernetes 无法根据计算任务的特征来匹配具有该指令集特征的计算单元, 即无法做到节点内的指令集感知来调度至具备 V 扩展指令集的 RISC-V 计算节点, 因此将会导致 Kubernates 中集群的 RISC-V 计算资源浪费.

为解决上述问题, 本文提出了一种基于 Kubernetes 异构 ISA 指令集感知的集群调度系统, 它主要由指令集感知 (ISAMatch) 模型来处理任务在异构 ISA 指令集架构节点中的调度. 具体地, 本文所设计的 Kubernetes 集群调度系统通过对集群内的计算节点的所有指令集提供探测核实, 并根据指令集亲和性和任务优先级来自动识别任务所需指令集架构和节点架构的亲和程度, 进而找到集群中匹配程度最高的节点. 本文整体系统设计基于开源项目 Volcano 调度器进行实现, ISAMatch 模型与 Volcano 调度器的其余插件完全兼容, 可由集群管理员动态选择是否开启 ISAMatch 模型. 在 ISAMatch 模型的支持下, Volcano 调度器可以很准确的将任务部署到对应架构的节点上, 并提高了调度性能和集群资源利用率.

本文的主要贡献总结如下:

(1) 调研 Kubernetes 的多种同构和异构资源调度策略和多种 ISA 指令集架构特点.

(2) 提出 ISAMatch 模型, 它能够准确的找到匹配任务 ISA 的节点列表并选择最佳匹配节点.

(3) 基于 Volcano 调度器实现了一个具有 ISA 感知能力的分布式异构云平台调度系统.

(4) 通过详细实验验证, 本文对所实现的云计算调度平台可以显著地避免任务在异构节点中的部署失败, 相对比默认调度器正确率 62% (调度 RISC-V 基础指令集任务)、41% (调度 RISC-V 扩展指令集任务)、67% (调度 RISC-V 扩展指令集任务且有“RISC-V”节点匹配标签), 在不考虑资源限制的条件下, ISAMatch 模型可以达到 100% 的任务调度正确率, 以此达到提高了集群资源利用率的效果.

2 相关工作及挑战

2.1 同构资源调度机制研究现状

同构资源的资源调度策略更注重资源调度的结果评估, 如资源利用率和集群负载均衡等. 目前, 云计算在同构资源调度领域的研究都集中在两个方向上.

(1) 根据各种调度算法及算法改进对资源进行合理调度; 同时, 还要对云平台的实时状态进行评估, 判断部分任务是否需要重新调度. 例如, Ge 等人[12] 将系统工作负载、任务队列和预测的执行时间作为输入, 调度器运用遗传算法生成最优调度方案; Zuo 等人[13] 提出了一种改进的蚁群算法解决最优调度问题, 使用任务完成时间和预算作为约束函数来评估成本以防止蚁群算法陷入局部最优解困境, 最终对调度方案提供反馈.

(2) 根据服务器负载的周期性, 通常采用机器学习算法, 对集群收集到的时序和历史负载信息进行分析, 寻找合适的调度方案. 何龙等人[14] 通过记录各个 Pod 对资源的使用情况, 在后续的调度过程中, 通过历史数据对调度的打分策略提供指导; Berral 等人[15] 采用机器学习算法, 对过往系统运行状态进行分析, 通过模型预测得出的功耗、CPU 负载和 SLA 时间调整调度决策.

2.2 异构资源调度机制研究现状

随着数据中心的计算节点技术架构和用户服务需求的不断变革, 为用户特定任务提供特定计算单元的异构资源系统已逐渐成为当前的主流计算模式, 例如

GPU (graphic processing unit) 服务器用于深度学习训练、NPU (neural processing unit) 节点用于神经网络推测以及 DPU (data processing unit) 用于数据资源的存储和计算服务. 面向这类融合 XPU 算力资源的资源调度策略更注重异构资源本身的算力特殊性. 近年的系统领域和计算机体系结构领域的研究成果大量集中在 GPU、FPGA 等异构计算单元的芯片设备上.

AntMan[16] 是阿里巴巴提出的一款具有动态缩放机制的集群异构资源调度器, 旨在解决在生产环境中 GPU 集群资源利用率低下的问题. AntMan 从动态显存管理和动态计算管理两个维度进行设计. 动态内存管理为了适应深度学习作业中不断变化的内存需求, 将部分显存内容转移到 CPU 内存中计算, 以实现高优先级任务的优先计算. 动态计算管理采用穿插运行模式, 高优先级任务运行时, GpuOpManager 组件实时监控任务在 GPU 上运行的时间和空闲间隙, 低优先级任务穿插间隙执行.

GaiaGPU[17] 是腾讯公司设计的一款细粒度异构资源调度系统. 一改传统异构设备的完整调度 (1 GPU), 将异构设备进一步细粒度化的切分调度 (0.x GPU). 在细粒度调度的基础上, 充分考虑多异构设备通信延迟, 构建 GPU 集群拓扑结构, 针对申请小于 1 GPU、申请等于 1 GPU 和申请大于 1 GPU 三种状态分别讨论, 实现 GPU 资源的最大利用率和通信的最小延迟.

KubeHICE[18] 是 Yang 等人提出的一款基于 ARM/X86 的异构 ISA 指令集架构的边缘计算资源调度平台. KubeHICE 通过请求的 YAML 文件中镜像的名称以及镜像详细信息包括镜像编号、架构等是否带有架构相关信息来选择合适的 ISA 指令集设备. 在 ISA 指令集架构感知的基础上, KubeHICE 还讨论不同 ISA 指令集架构芯片的算力, 根据单核算力重新定义资源数量以实现集群计算资源的负载均衡. 然而, KubeHICE 策略未考虑到同名异构的镜像调度, 同时, 指令集只针对了通用指令集, 未充分考虑到多种扩展指令集.

除了对具体异构设备资源调度的研究, 还有在异构环境下单任务切分的资源调度研究. Boran 等人 [19] 通过将任务进行细粒度分割, 针对任务在不同阶段的运行特性, 对任务迁移至不同的异构设备中运行, 实现任务的最佳运行效率.

2.3 RISC-V 概述

RISC-V 作为 RISC (精简指令集) 第 5 代架构, 于 2012 年由 Berkeley 团队[20] 提出, 旨在解决当前 ISA 指令集架构的过于庞杂、商业化闭源、灵活性缺失等一系列技术问题. 与 ARM/X86 等主流指令集架构相比, RISC-V 指令集架构作为新兴指令集架构具有精简化、模块化、开源化[21] 3 大突出特点.

(1) RISC-V 遵循“大道至简”的设计哲学[22]. 主流指令集架构在不断发展中为了可以向后兼容, 在原有的指令集中添加新的指令, 使得指令集文档过于冗余. 据统计, X86 指令集文档约 5 000 多页, 而同样基于 RISC 架构的 ARM 指令集文档也拥有 2 700 多页. RISC-V 摒弃了向后兼容的历史包袱, 采用“基本指令集+扩展指令集”的模式, 基础的基本指令集仅 40 余条, RISC-V 指令集手册也只有 200 多页[21].

(2) RISC-V 采用模块化的指令集设计模式, 采用“基本指令集+扩展指令集”的设计思想, 不仅精简了指令集数量, 还可以自由定制所需的指令集. RISC-V 架构的处理器设计人员可以根据需要选择是否需要添加这些扩展指令, 尤其适用于具有特定功能的物联网芯片, 降低了芯片的开发周期和成本.

(3) RISC-V 遵守开放开源原则, 与 ARM 和 X86 收取高额的专利费相比, 允许芯片开发人员根据自身需求自由修改开源代码, 并可直接用于商业活动, 降低了 RISC-V 的开发成本和门槛.

2.4 挑战

图 1 展示了 Kubernetes 在异构 ISA 指令集架构环境下的任务创建时调度过程, 开发者将带有应用程序和开发环境的镜像打包上传至镜像仓库中, 通过编写 YAML 文件生成 Pod 等待集群调度, controller 整合 Pod 信息通过调度算法与具体工作节点进行绑定, 由绑定节点的 kubelet 从镜像仓库中拉取对应架构的镜像生成执行 Pod, 通常镜像仓库针对某一基础镜像都有多种架构选择, kubelet 会选择与节点自身架构相同的架构进行拉取. 然而, 在调度过程中, 现有的调度器无法判断应用程序所需 ISA 指令集架构, 因此存在镜像拉取失败或者运行环境错误的调度失败的可能性.

图 2 统计了在 3 种异构 ISA 指令集架构 (RISC-V、ARM、X86) 环境下部署 100 个 RISC-V 任务时的调度正确频率, 实验平台见表 1 所示, 在调度 RISC-V 基础指令集任务时, 由于 3 个 RISC-V 节点都可以对其进行正常部署, 调度正确率为 62%; 而在调度没有添加任何 ISA 相关标签的 RISC-V 扩展指令集任务时, 任

务不仅需要 RISC-V 指令集架构环境, 还需要 RISC-V 节点具有特定扩展指令集架构, 调度正确率只有 41%; 即使添加了“RISC-V”的 ISA 标签, 由于特定扩展指令集架构的影响, 调度正确率也只有 67%.

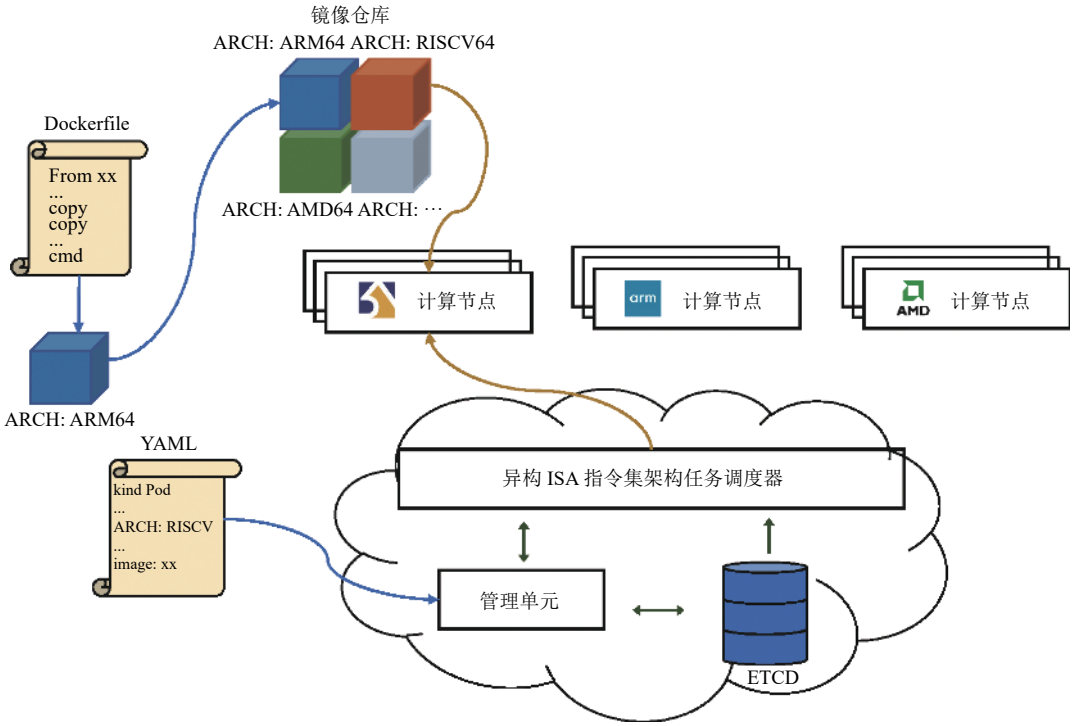


图 1 基于 Kubernetes 异构 ISA 指令集架构调度

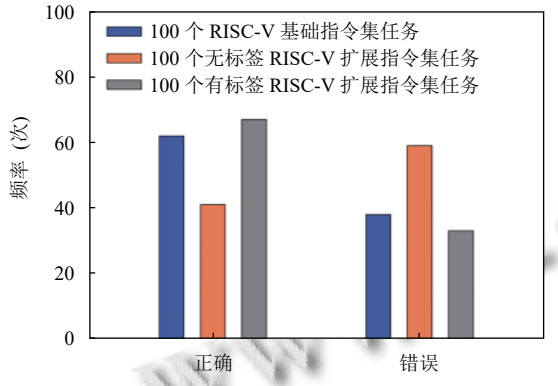


图 2 RISC-V 任务在默认调度器下的调度正确率

表 1 实验集群设备参数

主机名	角色	指令集	扩展指令集	CPU	内存 (GB)	QEMU
节点1	Master	X86	—	8	32	IMFDV
节点2	Worker	ARM64	—	8	32	—
节点3	Worker	RISCV64	IMFD	8	32	—
节点4	Worker	RISCV64	IMFV	8	32	—
节点5	Worker	RISCV64	IMFDV	8	32	—

因此, 面对异构 ISA 指令集架构的云平台, 当同时部署 ARM、X86、RISC-V 架构节点, 现有 Kubernetes 调度方案都无法正常调度. 尤其是面对具有多种 RISC-V 扩展指令集, 当前的 Kubernetes 云服务调度平台需要满足如下的 3 点需求:

需求① (指令可编译性): 在现有场景下, 支持对应扩展的计算任务的交叉编译模式.

需求② (执行正确性): 合理分配满足指令集架构的节点.

需求③ (架构兼容性): 适配多种指令集, 满足指令集在云平台节点上的架构兼容性.

为了解决创建时多种架构节点的分配问题, 本文设计了指令集感知与匹配 (ISAMatch) 模型来对节点 ISA 指令集进行匹配. 依次通过架构过滤、指令集过滤和节点打分找出最佳匹配节点.

3 基于 Kubernetes 的 ISA 架构匹配策略的设计与实现

ISAMatch 模型是一种针对异构 ISA 指令集架构节点环境下的集群调度方案. 如图 3 左侧所示, controller-manager 作为集群内部的管理中心, 负责管理集

群的节点、作业和资源等信息, 并整合用户申请的 YAML 信息交由调度器进行调度; scheduler 获取待调度 Pod 信息后由 ISAMatch 模型读取 Pod 和候选 Node 节点列表的 ISA 指令集架构数据, 并选择合适的 Node 节点进行绑定, 根据绑定信息将 Pod 交由对应节点 kubelet, 从镜像库中拉取与宿主机架构匹配的镜像.

图 3 右侧是 ISAMatch 模型的整体工作流程, ISAMatch 模型可以分为两个阶段, 两个阶段以是否需要交叉编译作为分隔.

第 1 步 (交叉编译), 判断当前任务是否具有交叉编译需求.

第 2 步 (ISA 指令集), 判断当前任务是否有具体 ISA 指令集架构需求.

第 3 步 (架构过滤), 粗粒度的过滤不匹配的顶层指令集架构节点.

第 4 步 (指令集过滤), 细粒度的过滤 RISC-V 特有的扩展指令集架构.

第 5 步 (节点打分), 综合指令集亲和性、同种指令集架构节点数量、资源利用率对节点打分, 选取得分最高的节点作为候选绑定节点.

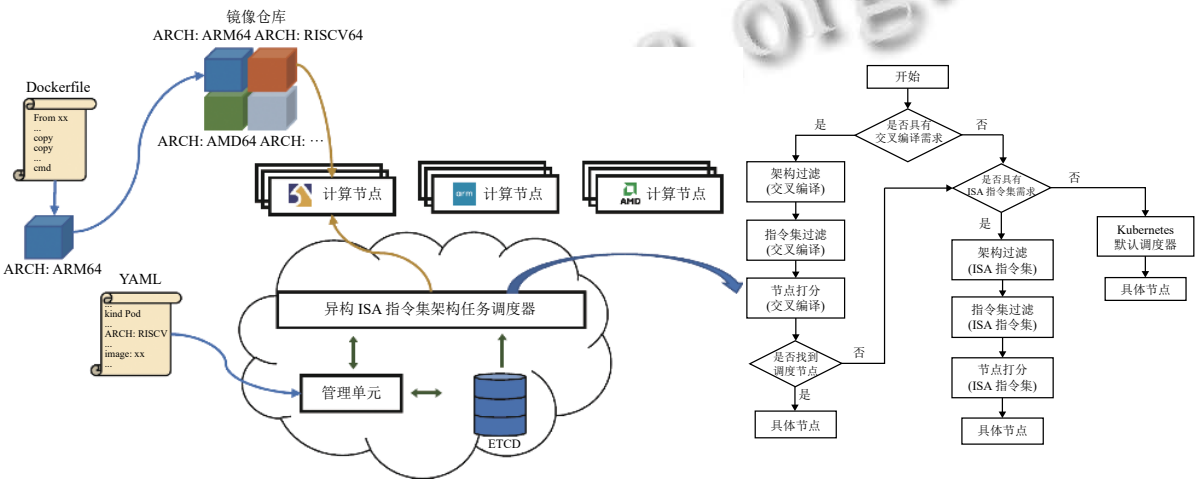


图 3 基于 Kubernetes 的 ISA 架构匹配策略架构和流程图

任务进入集群后首先判断其是否具有交叉编译需求, 如果有则直接跳过第 2 步, 根据其交叉编译所需要的 ISA 指令集依次执行架构过滤、指令集过滤和节点打分操作. 当任务不具有交叉编译需求或通过交叉编译判断未能找到合适的调度节点, 则继续执行第 2 步判断其是否有具体 ISA 指令集需求, 如果有, 则 ISAMatch 会根据任务所需 ISA 指令集依次执行架构过滤、指令集过滤和节点打分操作, 最终选出合适 Node 节点. 如果同时不具备交叉编译和 ISA 指令集需求时, 任务将交由 Kubernetes 默认调度器进行调度.

综合上述流程, 本小节将对交叉编译和 ISAMatch 模型进行详细描述.

3.1 交叉编译

截至目前, RISC-V 扩展指令集中有 12 个指令集已获批, 12 个指令集处于草案状态, 另有 5 个指令集已冻结. 因此, 应对多种还未完成的扩展指令集, 市场上尚未有对应指令集扩展的芯片, 应对复杂指令集需求的 Pod 请求, 云端集群也不可能完备地集成具有所有指令集模块组合芯片的节点.

为了解决需求① (指令可编译性), 本文设计了采用交叉编译的模式来解决复杂指令集请求, 以满足上述复杂指令集动态组合的条件. 本系统主要分两种编译模式: 1) 混合编译模式 QEMU; 2) 节点指令集架构属性的同构编译 ARCH, 如图 4 所示, 节点 ARCH 标签代表节点具有的指令集架构属性; 任务 ARCH 标签代表任务所请求的指令集架构属性; 节点 QEMU 代表节点具有的交叉编译指令集架构属性; 任务 QEMU 标签代表任务所请求的指令集架构属性. 本系统通过对用于匹配交叉编译模式下的架构属性进行标记, 标记为混合编译模式 QEMU (以图 4 为例, 节点具有 RV64IMFDV 的交叉编译属性, 即具有该指令集架构需求的任务可以在该节点上交叉编译运行). 如图 3 右侧流程图, 在架构过滤阶段判断 Pod 是否具有 QEMU 标签请求. 当一个具有 QEMU 标签请求的 Pod 进入调度器时, 调度

器会根据节点 QEMU 标签进行粗粒度架构过滤, 进入指令集过滤阶段后根据 QEMU 标签的扩展指令集部分进一步细粒度的过滤不匹配架构节点, 最后由节点打分模型分别对指令集亲和性、节点数目、节点资源利用率进行评估选出合适绑定节点. 若 QUME 标签的 ISAMatch 模型没有找到候选节点, 即所有候选节点 QEMU 标签中没有匹配的节点, 那么会进行常规的通过 ARCH 标签的调度.

```
01.  1.  # Pod YAML Label
02.  2.  metadata:
03.  3.      mame: test
04.  4.      labels:
05.  5.          ARCH: AMD64
06.  6.  # NODE YAML Label
07.  7.  Name:          Nodel
08.  8.  Roles:         Master
09.  9.  Labels:        ARCH=AMD64
10.  10.                QEMU=RV64IMFDV
```

图 4 Pod 和 Node 标签设置

3.2 ISAMatch 模型

为了解决需求② (执行正确性) 和③ (架构兼容性) 的云平台任务调度技术挑战, 本文创新地定义了 RISC-V 指令集亲和性, 用以描述任务请求指令集模块和节点指令集模块匹配程度的量化指标. 与 ARM/X86 指令集向后兼容不同, RISC-V 具有模块化和可扩展的指令集架构, 除了 4 组基本指令集外, 还具有多个扩展指令集, 这也使得 RISC-V 架构的处理器设计人员可以根据自身需求选择是否添加这些扩展指令. 以指令集架构 RV64IMFDV 为例, 其表示该架构支持 RV64I 的基础指令集与 M、F、D、V 四种扩展指令集, 那么类似于 RV64I、RV64IM、RV64IMFD 等 12 种架构组合需求的 Pod 都能在该架构的处理器节点上运行. 指令集亲和性值最大的节点即表示 Pod 指令集架构需求层面匹配最优的 Node 节点.

定义 (指令集亲和性): 用于描述任务请求指令集模块和节点指令集模块的相似程度:

$$instructionAffinity = \text{Pod模块数}/\text{Node模块数}$$

其中, 模块数是除去 RV 和处理器位数后的字母模块的数目即 IMFDV, Pod 指令集模块表示 Pod 请求的指令集架构, Node 指令集模块表示 Node 具有的指令集架构. 当两者数目越接近, 则指令集亲和性越高, 指令集匹配程度越高.

在具有多种混合架构的 Kubernetes 集群中, 指令集架构匹配是 Pod 正常运行的硬性指标. 由于 Kubernetes 的节点选择 (nodeSelector) 约束和亲和性约束都属于完全约束, 且不支持模糊匹配, 所以本文设计的 ISAMatch 模型采用 Kubernetes 的标签属性, 旨在识别具有共同特征或属性的资源对象. ISAMatch 模型具有两类标签属性: ARCH 标签和 QEMU 标签. 当 Pod 进入调度器时, 如图 3 右侧流程图所示 ISAMatch 模型会首先判断待调度 Pod 是否具有 QEMU 即是否具有交叉编译需求, 如若没有 QEMU 标签则继续判断是否具有 ARCH 标签即是否具有特定指令集架构需求, 如若没有 QEMU 标签和 ARCH 标签则转入 Kubernetes 默认调度器进行调度. QEMU 标签过滤和 ARCH 标签过滤的实现过程相似, 均采用架构过滤、指令集过滤和节点打分依次执行.

对所有候选 Node 节点执行架构过滤, 粗粒度的选出同族 ISA 指令集架构节点, 而对于具有特有扩展指令集架构的 RISC-V 节点, 还需要执行指令集过滤, 对同族 ISA 指令集架构节点进一步对扩展指令集的模块名称进行过滤, 必须具备 Pod 请求模块是 Node 支持模块子集的特点:

$$instruction_{\text{Pod}} \subseteq instruction_{\text{Node}}$$

选出的候选节点为细粒度的同族 ISA 指令集节点. 候选节点进一步进行节点打分. 节点打分会选出具体绑定的节点, 打分机制将综合考虑指令集亲和性、同种指令集架构节点的数量和节点资源利用率. 其优先级为:

$$\text{指令集亲和性} > \text{同种指令集架构节点数量} > \text{节点资源利用率}$$

具体示例如图 5 所示, 在 14 个异构 ISA 指令集架构节点的集群中任务申请为 RV64IMF. 架构过滤会根据任务 YAML 文件中的 ARCH 标签字段过滤粗粒度指令集架构为 9 个 RISC-V 节点. 指令集过滤根据细粒度的扩展指令集架构进一步过滤节点为 7 个支持 IMF 模块的 RISC-V 节点. 节点打分策略依据优先级顺序首先计算候选节点的指令集亲和性, 亲和性结果会选出一组或多组指令集亲和性相等的指令集架构节点. 由于指令集亲和性存在一定几率选出多组指令集亲和性相等的节点族, 因此, 本文设计了统计满足条件的同种指令集架构节点数量作为打分标准, 选择拥有节点数量更多的指令集架构节点将会为集群剩下更多的其余指令集架构节点, 以方便之后对于其余指令集架构需

求任务的调度. 当具体选定了某一细粒度扩展指令集架构时, 从候选节点中选择资源利用率最小的节点进行绑定, 以提高在满足调度申请条件的前提下的资源负载均衡程度和任务计算效率.

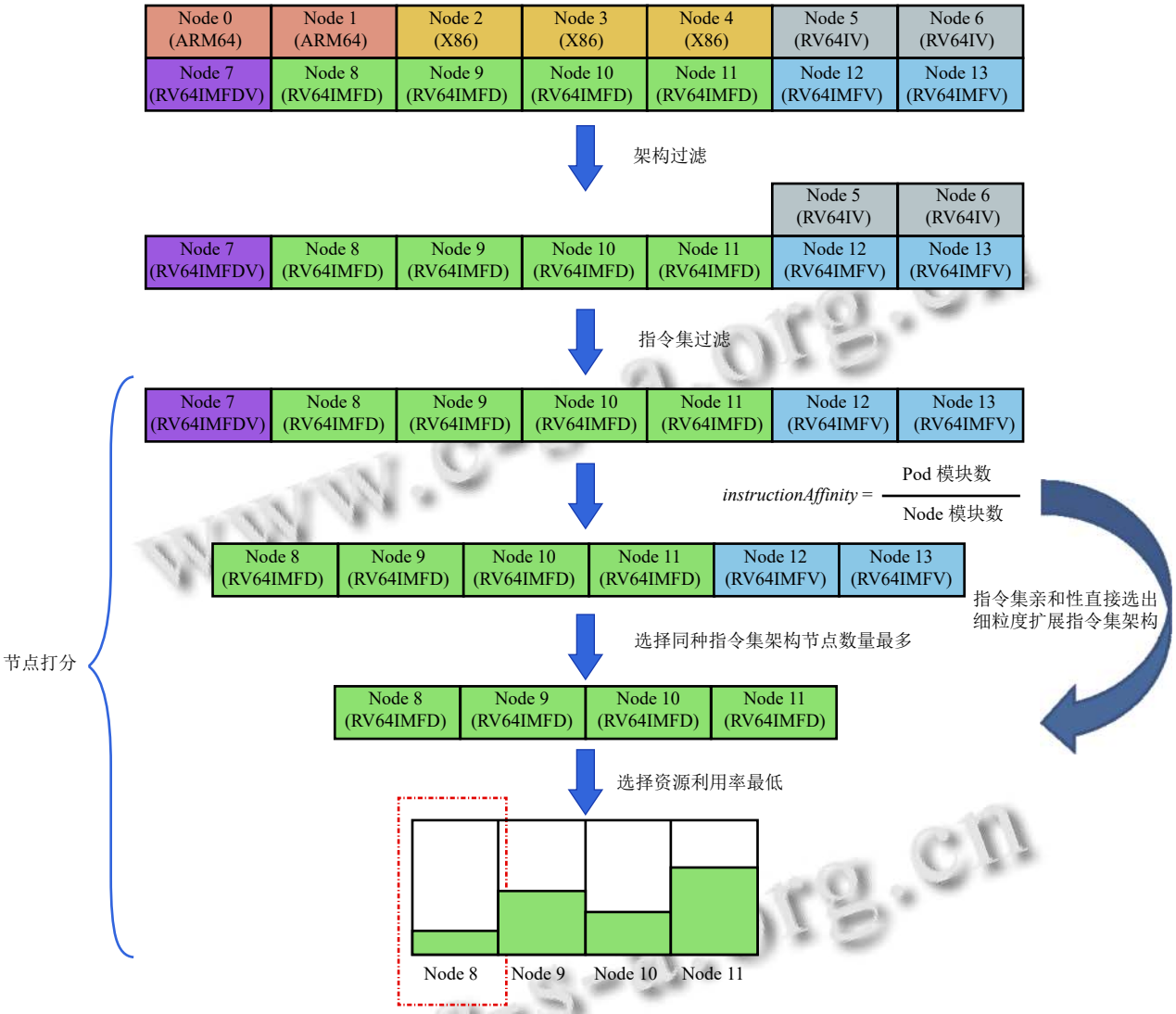


图 5 ISAMatch 模型工作流程图

4 实现与优化

本文通过基于 Volcano 调度器实现了支持多种 ISA 指令集架构任务的调度需求. 在不破坏 Volcano 源代码结构的前提下, 以插件的形式设计并实现 ISA-Match 模型.

4.1 标签自动机

由于 ISAMatch 模型依赖于 Pod 和 Node 的 ARCH 标签和 QEMU 标签, 其中 Pod 所具有的标签是由用户在提交 YAML 文件时主动设置, 代表用户提交任务所需要的指令集架构, 而集群中节点所支持的标签需要集群管理员手动添加, 需要在集群部署完成后为所有节点添加节点支持的 ISA 指令集架构和节点支持的 ISA 指令集交叉编译架构. 在规模较大的 Kubernetes 集群场景下, 这种由集群管理员手动添加标签的方式非常麻烦, 因此, 本文设计实现了自动标签设置脚本.

自动标签设置脚本运行在集群 Master 节点上, 通过 kubectl 命令工具从 ETCD 中获取所有节点信息, 并由正则表达式过滤出节点 IP. 同时, 在将节点部署进集群后, 节点会自动生成 kubernetes.io/arch 标签用于识别节点指令集架构, 脚本从节点详细信息中读取节点 IP 和粗粒度节点指令集架构. 而对于 RISC-V 指令集架构节点, 标签显示为 kubernetes.io/arch=riscv64, 因而

无法细粒度的识别 RISC-V 的具体扩展指令集架构, 因此需要从节点 gcc-v 中的--with-arch 标签获取更详细的 ISA 信息 (如图 6).

```
01.  1.  $ gcc -c -Q --help=target | grep march
02.  2.  -march=                    rv64imfdv
```

图 6　gcc 中的 ISA 信息

最后脚本模拟手动通过 kubectl 命令“kubectl label node $node ARCH=$ISA”添加 ARCH 标签方式.

交叉编译环境被默认安装在/opt/RISCV/bin/路径下, 同样的, 由/opt/RISCV/bin/riscv64-unknown-linux-gnu-gcc-v 获取详细扩展指令集架构, 最后通过 kubectl 命令工具由脚本模拟添加 QEMU 标签.

4.2　架构和指令集过滤以及节点打分流程优化

架构过滤和指令集过滤伪代码如算法 1 所示.

算法 1. 架构过滤和指令集过滤

```
candidateNodes={}
IF Contains(PodARCH, “RV”) THEN
    FOR node in nodeList do
        IF node→nodeARCH[:2]==“RV”&&
       ContainIns(node→nodeARCH[4:], PodARCH[4:]) THEN
            candidateNode=append(candidateNodes, node)
        END IF
    END FOR
END IF
ELSE
    FOR node in nodeList do
        IF Contains(node→nodeARCH, PodARCH) THEN
            candidateNodes=append(candidateNodes, node)
        END IF
    END FOR
END ELSE
return candidateNodes
```

架构过滤是对系统指令集架构进行的粗粒度过滤, 是对 Pod 和 Node 节点设置的 ARCH 标签或 QEMU 标签进行的模糊匹配, 过滤不符合 Pod 任务申请的架构的 Node 节点.

指令集过滤是针对 RISC-V 特有的扩展指令集的细粒度过滤, 由于扩展指令集兼容性的情况, 通过 containIns 函数判断 Pod 任务申请的指令集架构是否为 Node 支持的指令集架构的子集.

如图 7, 打分机制综合考虑指令集亲和性、同种指令集架构节点数量和节点资源利用率. 分别计算各个节点与请求 Pod 之间的指令集亲和性、同种 ISA 指令集架构数量和各个节点资源利用率, 并组合形成 NodeISA-Info 结构体. 针对 NodeISAInfo 依次对指令集亲和性、同种指令集架构节点数量和节点资源利用率进行排序, 依据优先级公式选择指令集亲和性最大的一组或多组 ISA 指令集架构节点, 在多组 ISA 指令集架构节点中比较同种指令集架构节点数量, 选择多者, 最后在单组中选取资源利用率最低的节点作为候选的绑定节点.

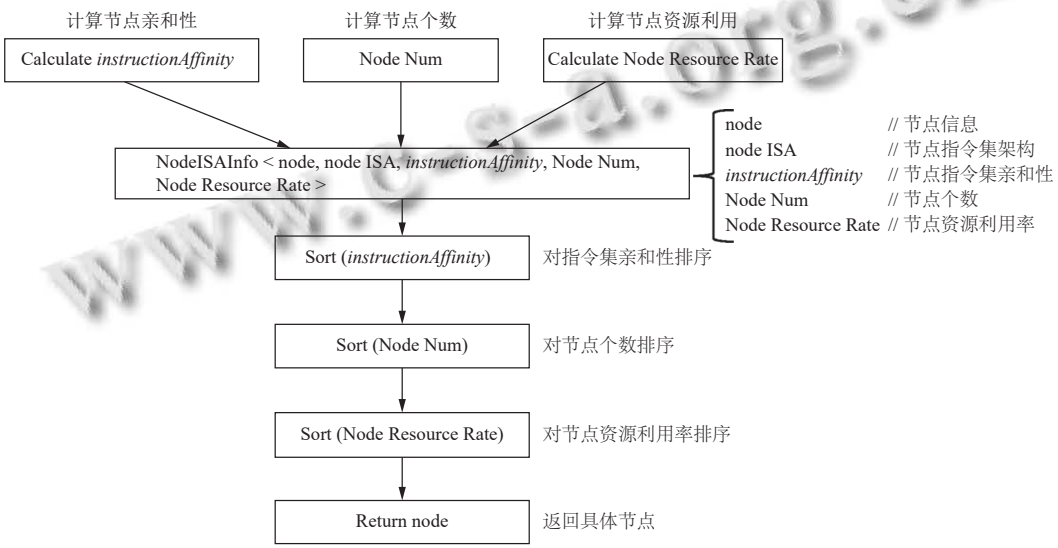


图 7　打分机制函数流图

5　评估实验和结果分析

本文以 Linux 平台最为常见的 3 种架构作为研究对象 (ARM、X86、RISC-V) 进行分析, 主要进行:

(1) 评估在异构 RISC-V 指令集架构的集群环境

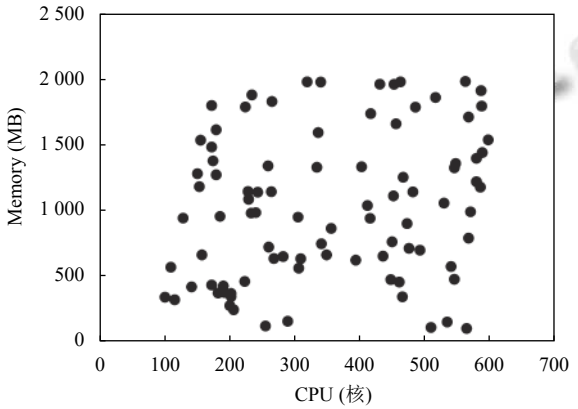
下, 部署不同架构 (含 X86、ARM 和 RISC-V 指令集架构芯片) 需求任务的调度正确性;

(2) 统计在调度中与传统调度器相比的性能延迟.

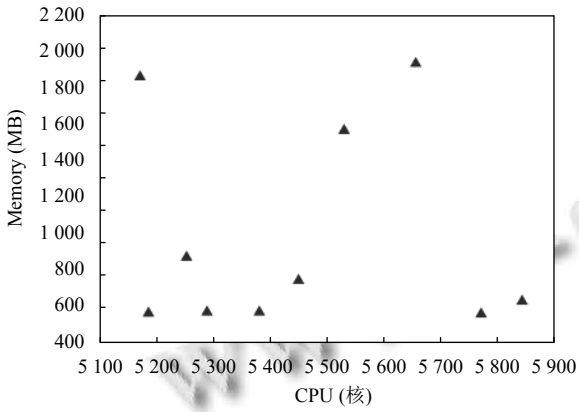
5.1 实验平台及数据

实验平台: 在 5 台部署有 Kubernetes 集群的虚拟环境上进行验证实验. 其中包括一台 X86 的虚拟主机、一台 ARM64 的虚拟主机和 3 台 RISC-V 的虚拟主机. 具体参数见表 1.

实验数据: 实验数据由随机函数生成了 90 个服务型任务和 10 个计算型任务, 100 个任务所申请的 CPU 和 Memory 资源如图 8 所示.



(a) 90 个服务型任务资源分布



(b) 10 个计算型任务资源分布

图 8 100 个任务的资源分布情况

5.2 ISAMatch 模型正确性测试

本文通过部署 100 个 RISC-V 任务对集群进行 ISA 架构匹配的测试. 分别测试验证: 1) 部署 100 个 RISC-V 基础指令集任务, 即只需要在 RISC-V 节点上就能正常运行; 2) 部署 100 个未带有指令集架构相关标签的 RISC-V 扩展指令集任务; 3) 部署 100 个带有“RISC-V”标签的 RISC-V 扩展指令集任务.

实验表明 (如图 9), RISC-V 基础指令集任务只需要部署到 RISC-V 节点上就能正常运行, 调度正确率为 62%; 而未带有指令集架构相关标签的 RISC-V 扩展指令集任务需要精确到具体的扩展指令, 如 RV64IMFD 只能运行在节点 3 和节点 5 上, 调度正确率为 41%; 带有“RISC-V”标签的 RISC-V 扩展指令集任务虽然排除了 ARM 和 X86 两种架构, 但部分调度未能精确找到匹配的扩展指令架构的节点, 调度正确率为 67%. 而在 ISAMatch 模型下, 调度正确率达到 100%, ISAMatch 模型可以准确的将 Pod 调度到合适的节点, 并能够更细粒度的通过指令集亲和性找到最佳匹配节点, 如图 10 所示, RV64IMFD 可以成功运行在节点 3 和节点 5 上, 而根据指令集亲和性, ISAMatch 模型会选取更为匹配的节点 3 作为最佳匹配节点, 而只有当节点 3 无法满足调度需求时才会选用节点 5 进行调度.

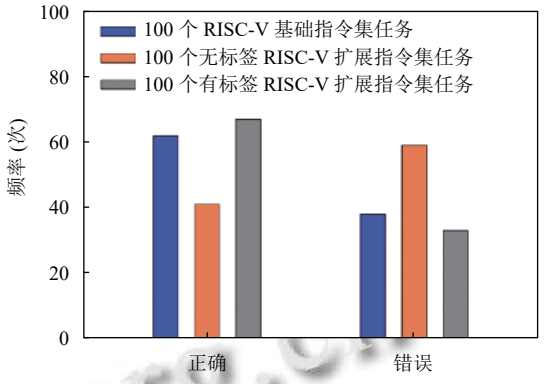


图 9 RISC-V 任务在默认调度器下的调度正确率

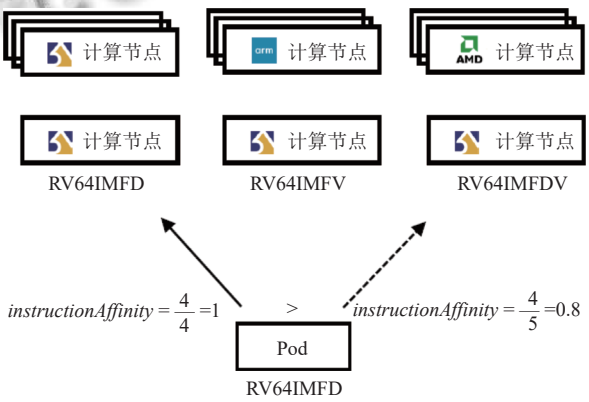


图 10 指令集亲和性下 RISC-V 扩展指令集任务调度选择

5.3 调度延迟分析

进一步, 为了验证具有 ISAMatch 模型调度器和传统调度器的调度延迟, 实验测试了调度 1–100 个 Pod.

其中, 调度单个 Pod 所需的调度延迟是统计待调度 Pod 在 Volcano 调度器中从调度起始函数 OpenSession 开始到绑定具体节点后的 CloseSession 结束所用的时间. 从图 11 可以得出当部署 40 个以内的 Pod 数时, 传统调度器和具有 ISAMatch 模型的调度器调度所用延迟在误差范围内接近, 大约在 1 ms 左右. 而当部署 40 个以上 Pod 时, 具有 ISAMatch 模型的调度器所消耗的调度延迟略大于传统调度器, 并随着 Pod 数量的增加, 两者之间的差值也逐步增大, 在部署 100 个 Pod 时, 具有 ISAMatch 模型的调度器比传统调度器多耗时约 6 ms. 这是由于随着 Pod 数目的增加, 每一轮 Session 对单个 Pod 进行调度时都需要额外对其 ISA指令集架构进行匹配, 经测量, 对单个 Pod 进行 ISA 指令集架构匹配的平均耗时为 60.57 μs. 部署 100 个 Pod 有无 ISAMatch 模型耗时的差值与 Pod 个数不构成线性关系 (如图 12), 且与平均耗时不成倍数关系, 其原因是随着 Pod 数目增大后无法在同一个 Session中完成调度, 需要开启多个 Session 进行调度, 因此会有额外的调度损耗.

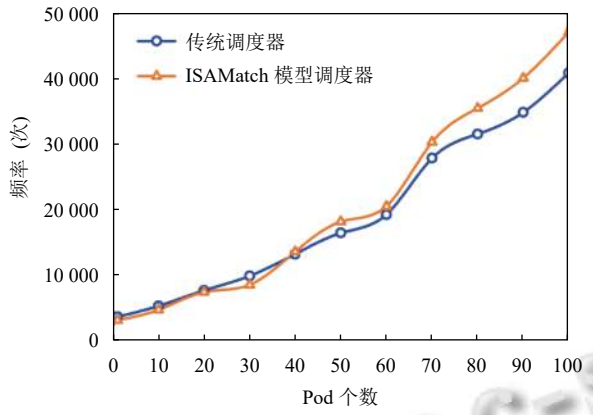


图 11 ISAMatch 模型调度延迟

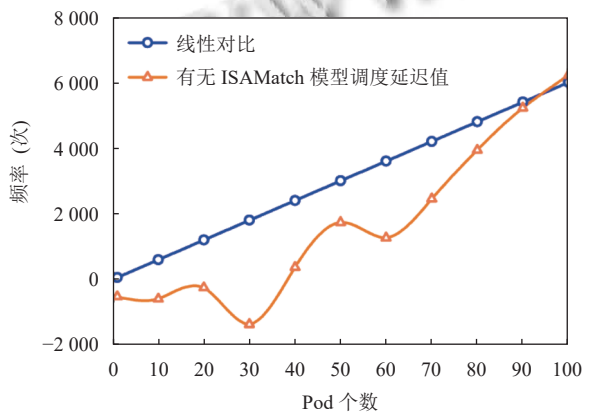


图 12 有无 ISAMatch 模型延迟差与单任务延迟线性对比

6 结语

针对目前 Kubernetes 集群无法支持异构 ISA 指令集架构任务调度的问题, 提出了一种具有 ISA 指令集感知能力的异构资源调度方案. 应对多种 RISC-V 扩展指令集架构需求任务, 提出了指令集亲和性标准, 并在此基础上通过架构过滤、指令集过滤和节点打分 3 个维度的综合评价, 在保证调度性能的前提下, 提高集群调度正确性和资源利用率, 使集群有能力应对更多复杂场景.

参考文献

1 Qian L, Luo ZG, Du YJ, *et al*. Cloud computing: An overview. Proceedings of the 1st International Conference on Cloud Computing. Beijing: Springer, 2009. 626–631.

2 Gong CY, Liu J, Zhang Q, *et al*. The characteristics of cloud computing. 2010 39th International Conference on Parallel Processing Workshops. San Diego: IEEE, 2010. 275–279.

3 Rad BB, Bhatti HJ, Ahmadi M. An introduction to Docker and analysis of its performance. International Journal of Computer Science and Network Security, 2017, 17(3): 228–235.

4 Higgins J, Holmes V, Venters C. Orchestrating Docker containers in the HPC environment. 30th International Conference on High Performance Computing. Frankfurt: Springer, 2015. 506–513.

5 Menage P. Cgroups. https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt. (2015-02-07).

6 Burns B, Grant B, Oppenheimer D, *et al*. Borg, omega, and Kubernetes. Communications of the ACM, 2016, 59(5): 50–57. [doi: 10.1145/2890784]

7 Chen C, Xiang XY, Liu C, *et al*. Xuantie-910: A commercial multi-core 12-stage pipeline out-of-order 64-bit high performance RISC-V processor with vector extension: Industrial product. 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). Valencia: IEEE, 2020. 52–64.

8 中科院发布国产 RISC-V 处理器“香山”. 信息系统工程, 2021, 7: 2.

9 Tine B, Yalamarthy KP, Elsabbagh F, *et al*. Vortex: Extending the RISC-V ISA for GPGPU and 3D-graphics. MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2021. 754–766.

10 刘畅, 武延军, 吴敬征, 等. RISC-V 指令集架构研究综述. 软件学报, 2021, 32(12): 3992–4024. [doi: 10.13328/j.cnki.jos.006490]

11 Sá B, Martins J, Pinto SES. A first look at RISC-V virtualization from an embedded systems perspective. IEEE Transactions on Computers, 2021. [doi: 10.1109/TC.2021.3124320]
12 Ge YJ, Wei GY. GA-based task scheduler for the cloud computing systems. 2010 International Conference on Web Information Systems and Mining. Sanya: IEEE, 2010. 181–186.
13 Zuo LY, Shu L, Dong SB, *et al*. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access, 2015, 3: 2687–2699. [doi: 10.1109/ACCESS.2015.2508940]
14 何龙, 刘晓洁. 一种基于应用历史记录的 Kubernetes 调度算法. 数据通信, 2019, 3: 33–36. [doi: 10.3969/j.issn.1002-5057.2019.03.009]
15 Berral JL, Goiri Í, Nou R, *et al*. Towards energy-aware scheduling in data centers using machine learning. Proceedings of the 1st International Conference on Energy-efficient Computing and Networking. Passau: ACM, 2010. 215–224.
16 Xiao WC, Ren SR, Li Y, *et al*. AntMan: Dynamic scaling on GPU clusters for deep learning. 14th USENIX Symposium on Operating Systems Design and Implementation. Online: USENIX Association. 2020. 533–548.
17 Gu J, Song SB, Li Y, *et al*. GaiaGPU: Sharing GPUs in container clouds. 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). Melbourne: IEEE, 2018. 469–476.
18 Yang SQ, Ren Y, Zhang JF, *et al*. KubeHICE: Performance-aware container orchestration on heterogeneous-ISA architectures in cloud-edge platforms. 2021 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). New York: IEEE, 2021. 81–91.
19 Boran NK, Rathore S, Udeshi M, *et al*. Fine-grained scheduling in heterogeneous-ISA architectures. IEEE Computer Architecture Letters, 2021, 20(1): 9–12. [doi: 10.1109/LCA.2020.3045056]
20 Waterman A, Lee Y, Patterson DA. The RISC-V instruction set manual, volume I: Base user-level ISA. Technical Report, Berkeley: University of California at Berkeley, 2011.
21 种丹丹. 基于 RISC-V 的开源芯片生态发展现状及未来机遇. 中国集成电路, 2021, 30(8): 25–30. [doi: 10.3969/j.issn.1681-5289.2021.08.005]
22 何杨阳. RISC-V 及后编译技术研究与实现 [硕士学位论文]. 北京: 北京邮电大学, 2021. [doi: 10.26969/d.cnki.gbydu.2021.000556]

(校对责编: 孙君艳)