计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(12):16−25 [doi: 10.15888/j.cnki.csa.010067] [CSTR: 32024.14.csa.010067]

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

基于 RISC-V 异构平台的大语言模型推理加速①

沈郑东[1,3], 刘雨冬[2], 于佳耕[2], 田　青[1]

[1](南京信息工程大学 软件学院, 南京 210044)
[2](中国科学院 软件研究所, 北京 100190)
[3](中国科学院大学南京学院, 南京 211135)
通信作者: 于佳耕, E-mail: jiageng08@iscas.ac.cn; 田　青, E-mail: tianqing@nuist.edu.cn

摘　要: 随着大语言模型在各类生成任务中的广泛应用, 其高计算负载对底层硬件平台提出了更高的性能要求. RISC-V 作为一种新兴的开源指令集架构, 凭借其良好的可定制性和扩展性, 展现出巨大的发展潜力. 然而在部署主流大模型方面, RISC-V 平台仍面临生态不完善、算力受限等诸多挑战. 本文提出一种基于 RISC-V 平台的大语言模型推理加速方法, 通过构建寒武纪 MLU370 加速卡的异构运行环境, 成功完成了设备驱动移植、基础库编译与 PyTorch 框架适配. 在此基础上, 进一步设计了一种轻量级多线程优化策略, 提升注意力机制等核心算子在多核体系结构下的执行效率. 实验结果表明, 在 SG2042+MLU370-S4 平台上部署多个主流大模型时, 该方法在不依赖其他优化策略下, 实现最高达 52.3 倍的端到端推理加速, 验证了其在 RISC-V 异构平台上的可行性与通用性.

关键词: RISC-V 架构; MLU370 加速卡; 大语言模型; 异构计算优化

引用格式: 沈郑东,刘雨冬,于佳耕,田青.基于 RISC-V 异构平台的大语言模型推理加速.计算机系统应用,2025,34(12):16–25. http://www.c-s-a.org.cn/1003-3254/10067.html

Inference Acceleration for Large Language Models on RISC-V Heterogeneous Platform

SHEN Zheng-Dong[1,3], LIU Yu-Dong[2], YU Jia-Geng[2], TIAN Qing[1]

[1](School of Software, Nanjing University of Information Science & Technology, Nanjing 210044, China)
[2](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[3](University of Chinese Academy of Sciences, Nanjing, Nanjing 211135, China)

Abstract: With the widespread deployment of large language models (LLMs) across various generative tasks, their high computational demands impose stringent performance requirements on the underlying hardware. RISC-V, an emerging open-source instruction-set architecture, shows great potential owing to its excellent customizability and extensibility. Nevertheless, when deploying mainstream LLMs, the RISC-V ecosystem still faces challenges such as an incomplete software stack and limited compute capability. This study proposes an inference acceleration method for LLMs on RISC-V heterogeneous platforms. By establishing a heterogeneous runtime environment that integrates the Cambricon MLU370 accelerator, the device driver is ported, essential libraries are compiled, and the PyTorch framework is adapted. Building on this foundation, a lightweight multi-threading optimization scheme is further designed to improve the efficiency of core operators—especially the attention mechanism—on multi-core architectures. Experimental results on the SG2042+MLU370-S4 platform show that, without relying on any additional optimizations, the proposed method achieves up to 52.3 times end-to-end inference speedup for several mainstream LLMs, thus demonstrating both the feasibility and broad applicability of the approach on RISC-V heterogeneous systems.

Key words: RISC-V architecture; MLU370 accelerator card; large language model (LLM); heterogeneous computing optimization

① 基金项目: 国家重点研发计划 (2023YFB4503902)
收稿时间: 2025-04-30; 修改时间: 2025-06-24, 2025-07-21; 采用时间: 2025-09-05; csa 在线出版时间: 2025-11-04
CNKI 网络首发时间: 2025-11-05

近年来, 无论在生成式任务、推理任务还是上下文理解等方面, 大语言模型 (large language model, LLM) 都表现出了显著的能力, 并服务了工业界和学术界的广泛应用场景. 作为第 3 代人工智能技术的典型代表, 大语言模型通过其数万亿参数的深层 Transformer 架构, 正在重塑自然语言处理的技术范式. 2018–2024 年间模型参数规模年均增长率达 350%, 现代 LLM 普遍具有 10^9–10^{12} 量级的参数规模, 导致其计算复杂度呈超线性增长态势, 故在实际部署中的计算资源消耗和硬件依赖性成为重大挑战[1]. 如何在不同硬件平台上高效移植和优化 LLM, 是当前研究的热点.

RISC-V 架构作为一种新兴的开源指令集架构, 以其开放、简洁、稳定、免授权、先进和模块化等特性[2]受到了广泛关注. 和传统封闭的指令集架构相比, RISC-V 架构赋予开发者更大的设计自主权, 使开发者能够根据需求灵活定制指令集, 以提升计算性能. 目前, RISC-V 架构提高计算能力的方式主要聚焦于向量扩展机制 (如 RVV)、异构协处理器集成 (如 SiFive Intelligence X280 和 Andes NX27V) 以及软硬件协同设计等方向, 旨在提升对大规模数据并行计算的支持能力, 并通过编译器优化与指令集级调整实现低功耗、高效率的算子执行路径.

与此同时, 大语言模型的推理优化已成为研究热点. 一些研究工作[3,4]通过优化大模型的存储结构, 有效提升了大模型的推理和训练速度. 此外混合精度与张量并行[5,6]技术的应用, 在提高了模型吞吐量的同时, 也进一步加快了大模型推理效率. FlashAttention 系列算法[7–9]采用算子融合思想, 将多个算子合并执行, 从而减少了冗余计算步骤, 从而提升了整体计算效率. 除此之外, 部分主流推理引擎 (如 DeepSpeed-Inference[10]) 基于 GPU 架构提出了多线程与 Kernel 级优化策略, 以实现大模型的高效部署.

然而, 上述推理优化方法多构建于 x86/GPU 平台, 在迁移至 RISC-V 架构等新兴指令集架构平台时将面临 3 方面挑战: (1) 指令集异构性导致的算子重定向开销显著; (2) 硬件生态尚未成熟, 主流模型架构难以直接部署; (3) 缺乏高效的硬件加速资源与接口支持.

针对以上问题, 本文面向 RISC-V 平台下的大语言模型推理场景, 提出了一种异构推理加速方案, 该方案主要是通过将国产加速卡与 RISC-V 平台进行适配, 重点解决了驱动兼容、基础软件移植问题, 同时使用多线程推理策略共同优化大模型在 RISC-V 平台上的推理效率.

1 相关工作

近年来, LLM 因其在自然语言处理任务中的出色表现, 受到了学术界和工业界的广泛关注. 其推理优化的研究体系不断扩展, 涵盖了模型压缩、推理框架、硬件加速等多个方向. 本节围绕 RISC-V 架构下的 AI 加速技术、专用 AI 加速卡在 LLM 推理中的应用, 以及 RISC-V 架构下的异构计算协同挑战 3 个方面, 系统回顾并总结相关研究进展与发展趋势.

1.1 RISC-V 架构的 AI 加速技术

RISC-V 架构凭借其开源特性与高度模块化的设计理念, 在 AI 推理加速场景中展现出高度的可扩展性与设计灵活性. 当前, 研究主要集中于通过指令集扩展与专用协处理器设计来增强其原生算力.

在指令集扩展方面, RISC-V 向量扩展 (RVV) 因其高效的数据并行支持成为研究热点. RVV1.0[11]由 RISC-V International 于 2021 年正式发布, 旨在为多样化处理器提供统一的向量扩展标准. Perotti 等人[12]制造了支持 RVV1.0 的开源处理器, 并通过实验验证了 RISC-V 向量扩展在数据并行计算任务中的高效性. Chander 等人[13]进一步在边缘设备场景评估了 RVV 的适用性, 其方案相较于 ARM 异构方案显著降低了延迟, 凸显了 RISC-V 架构在边缘 AI 场景在的潜力.

在专用协同处理器架构方面, 产业界已经推出多款高性能 RISC-V 架构的 AI 解决方案. SiFive Intelligence X280 系列[14]集成标量 RISC-V 核心与可编程 AI 引擎, 通过共享内存架构降低数据搬运开销, 提高了计算能力. Andes Technology 公司的 NX27V 系列[15]处理器通过扩展 RISC-V RV64GC 指令集 (支持 BF16 浮点运算与动态向量位宽), 实现了面向 Transformer 模型的高效 AI 计算加速, 为边缘至云端 AI 负载提供了灵活的可扩展硬件基础. 此外, 国际上也涌现了如 Celerity、Esperanto ET-SoC-1 等探索提升计算能力的 RISC-V 项目.

当前, RISC-V 在 AI 推理加速方面取得了一系列技术突破, 既有架构标准的规范化, 也有产业实现的逐步落地. 但其生态仍不成熟, 在与现有的异构加速卡的软硬件适配与优化方面仍需进一步完善. 因此, 需要结合国产加速卡具体需求, 研究相应的异构加速卡的软

硬件适配方案.

1.2 专用 AI 加速卡及其在 LLM 推理中的应用

专用 AI 加速卡凭借其强大的并行计算能力和优化的内存子系统, 已成为加速 LLM 推理的关键硬件. 其软件生态的成熟度直接影响部署效率.

国际主流加速卡 (如 NVIDIA GPU, Google TPU, AMD Instinct MI 系列, Intel Habana Gaudi) 拥有成熟的软件栈和优化框架. 例如, DeepSpeed-Inference 和 FasterTransformer 等框架广泛采用多线程、CUDA Kernel 融合等技术, 显著优化了 Transformer 模型在 GPU 上的推理性能 (如 TensorRT-LLM). 这些方案在 x86/ARM 平台上取得了卓越成效.

国内加速卡发展迅速, 寒武纪 MLU370 加速卡是代表性产品之一. 它基于第 4 代 MLUarch 架构, 集成大量张量处理单元 (TPU) 和高带宽内存 (HBM2e), 特别适配 Transformer 类模型的矩阵运算特征. 其配套的 CNRT 运行时库、高性能算子库 (CNNL, CNCL 等) 以及 PyTorch 扩展 (CATCH, torch_mlu), 为模型移植提供了支持, 已广泛应用于视频分析、图像识别、自然语言处理等领域. 其他国产加速卡如华为昇腾 Ascend 系列、天数智芯 BI 系列等也在 LLM 推理领域积极布局.

当前 AI 加速卡在 LLM 推理中的作用已十分关键, 国际方案在性能与生态上更为成熟, 国内加速卡则在特定模型与场景中不断突破, 但在 RISC-V 架构上, 整体软件兼容性与通用性相比于成熟的 ARM 和 x86 生态仍存在差距, 这对其实际部署与应用构成了一定限制.

2 基于 RISC-V 平台的 LLM 推理优化方法

本文提出的优化方法包括两个核心方法: (1) 在 RISC-V 平台上适配寒武纪 MLU370 加速卡, 构建从设备识别、驱动移植、软件栈部署到 PyTorch 框架支持的完整异构计算环境; (2) 针对 LLM 推理过程中的计算瓶颈, 引入了多线程矩阵乘法优化方法, 提升了 LLM 的推理速度.

2.1 基于 RISC-V 平台的 MLU370 加速卡的适配

目前, 针对 RISC-V 架构在专用计算加速硬件 (如 GPU/NPU/TPU) 的移植工作主要面临两大难点, 即硬件兼容和定制化软件栈的支持. 硬件兼容需要确保 RISC-V 平台能够通过驱动正确地查看加速卡, 并使得加速卡能够正常运行. 由于这些加速卡的开发公司都有定制化的软件栈, 为了适配新的架构, 需要重新编译和调整软件栈. 虽然 MLU370 加速卡也是 NPU 加速卡, 但作为国产先进的 AI 推理芯片, 其有大量的自研软件, 包括运行时库、框架、分析工具等. 在 RISC-V 架构上移植 MLU370 时, 需要重新编写或者调整这些软件, 以适配新的架构和平台. 此外, MLU370 加速卡的原生软件环境基于 x86 架构, 移植时需要解决这些架构特定优化的支持. 最后, MLU370 原生开发环境基于 x86 架构的早期 v4.19 内核版本, RISC-V 架构的 Linux 内核支持始于 v5.7, 这种内核版本代际差异要求对设备驱动接口进行跨版本适配. 综上所述, 硬件兼容和不同硬件平台的定制化适配导致 MLU370 的 RISC-V 移植难度更高.

为了解决上述挑战, 本文提出基于 RISC-V 平台的 MLU370 运行环境架构, 由硬件适配、驱动移植、基础软件适配与 PyTorch 框架适配这 4 个关键层次构成, 整体架构图如图 1 所示.

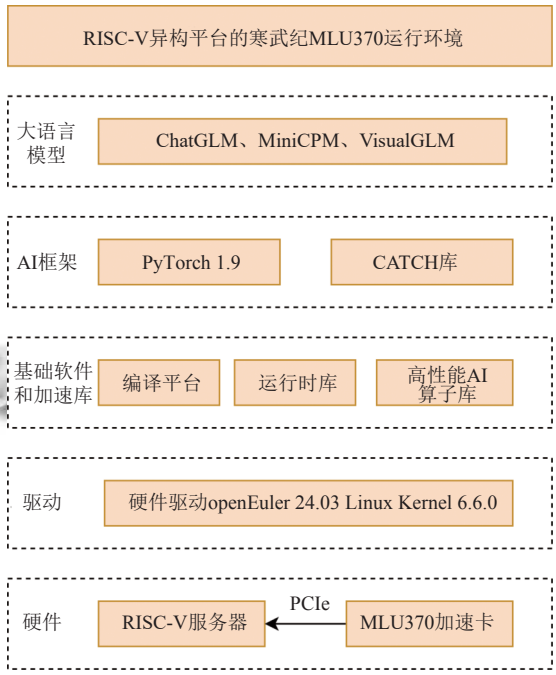

图 1 基于 RISC-V 平台的 MLU370 运行环境架构图

首先, 面对硬件适配中遇到的 BAR (基地址寄存器) 空间不足问题, 提出并实现了基于设备树与动态重映射机制的解决方案. 其次, MLU370 加速卡在 RISC-V 架构上的运行支持主要集中于与架构强相关的软件栈, 包括驱动、运行时库、高性能算子库及其他依赖组件. 最后, 在 RISC-V 平台上成功适配了 MLU370 系

列加速卡, 并利用加速卡运行了大语言模型 (如 TeleChat-12B).

2.1.1 硬件适配

为实现 MLU370 加速卡在 RISC-V 架构平台上的运行, 首要工作是确保 MLU370 加速卡能够被 RISC-V 平台正确识别, 从而支持运行大模型. 然而, MLU370 作为一款 PCIe (peripheral component interconnect express) 设备, 在不同平台间移植时, 需要特别关注 PCIe BAR (base address register) 空间问题. 这一问题通常涉及平台硬件资源分配、操作系统配置以及驱动程序实现. 此外, 由于 MLU370 是专为 x86 和 ARM 指令集架构设计的, 其在 RISC-V 平台上与这两种指令集架构的实现存在差异, 可能导致 BAR 空间不足的问题, 从而使 MLU370 无法正常映射内存地址.

针对这一问题, 我们通过调整 PCIe 设备树 (device tree) 文件中的 PCIe BAR 范围 (ranges 属性) 来支持 MLU370 加速卡对 PCIe 资源的需求. 在现代 Linux 操作系统中, 系统通过设备树了解硬件设备及其属性, 而无需将这些信息硬编码到内核中. PCIe 设备树的 ranges 属性用于定义父总线地址和子总线或设备地址之间的映射. 采用动态 BAR 重映射机制, 通过修改 PCIe 设备树的 ranges 属性, 实现 3×256 MB 连续地址空间分配, 满足 MLU370-S4 的 DMA 传输需求. 调整后的设备树文件的其中一个 PCIe 控制节点如代码 1 所示.

代码 1. 设备树中 PCIe 控制节点配置

```
// PCIe 控制器配置
pcie: pcie@706000000 {
    compatible = "sophgo, cdns-pcie-host";
    device_type = "pci";
    #address-cells = <0x03>;
    #size-cells = <0x02>;
    bus-range = <0x00 0x3f>
    reg = <0x70 0x60000000 0x00 0x2000000 0x40 0x00 0x00 0x1000>;
    reg-name = "reg\ocfg";
    ranges = <0x1000000 0x00 0xc0000000 0x40 0xc0000000 0x00
0x400000 0x42000000 0x00 0xd0000000 0x40 0xd0000000 0x00
0x10000000 0x2000000 0x00 0xe0000000 0x40 0xe0000000 0x00
0x20000000 0x43000000 0x42 0x00 0x00 0x02 0x00 0x3000000 0x41
0x00 0x01 0x00>;
    status = "okay";
```

2.1.2 驱动移植

寒武纪 MLU370 系列加速卡的驱动 (cambricon-mlu-driver) 采用 GCC 编译器和 Makefile 进行自动化编译. GCC 编译器已支持 RISC-V 指令集架构. 因此, 驱动移植的核心步骤是利用 GCC 编译器完成源代码的编译与安装适配.

在面向 RISC-V 架构进行移植时, MLU370 系列计算卡通常选择使用 openEuler 操作系统. openEuler 当前使用的 Linux 内核版本为 6.x.x 或更高. 然而, 寒武纪官方经过驱动编译测试的 Linux 操作系统内核版本范围为 3.10.0–5.15.67. 因此, 在驱动程序编译过程中, 由于内核版本过高可能会导致兼容性报错. 此类错误通常源于内核版本升级导致的 API 变更, 例如函数接口的参数数量、位置或类型发生改变, 致使驱动源码无法通过编译. 为解决这些编译报错问题, 我们通过在相应头文件中为当前内核版本添加正确的函数定义, 并在调用函数时传入正确的参数来优化驱动代码. 驱动成功适配后, 即可查看 MLU370 的 PCIe 设备信息, 驱动移植成功如图 2 所示.

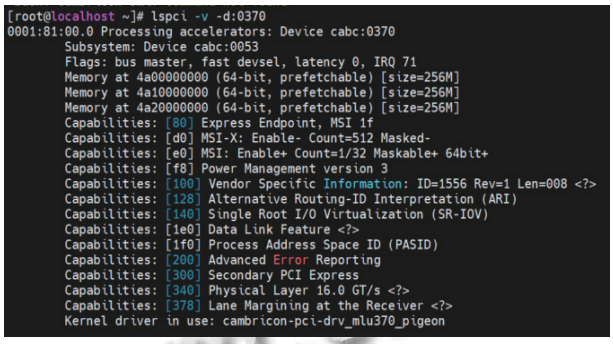

图 2 驱动移植成功示意图

2.1.3 基础软件适配

寒武纪 MLU370 加速卡的基础软件从功能维度划分, 可以分为运行时库、框架、分析工具、BANG 语言及工具库、算子库和视觉应用工具. 其中, 运行时库、框架、高性能算子库以及 BANG 语言及工具库这 4 大部分对顶层应用 (如大模型) 的运行起着关键作用.

运行时库主要由 CNRT (cambricon runtime) 和 CNDrv (cambricon driver) 组成. 这些库为开发者提供了操作 MLU 硬件的接口, 涵盖设备管理、内存分配和任务调度等功能. 高性能算子库包括 AI 算子库 CNNL、通信库 CNCL 和视觉库 CNCV, 覆盖了视觉、语音、自然语言处理、搜索推荐和自动驾驶等典型深度学习应用场景, 为用户提供了强大的计算能力. BANG 语言是寒武纪公司为其 AI 芯片开发的一种编程语言, 提供了通用的 C 语言接口, 降低了用户的学习和使用难度.

BANG 语言拥有一套完善的异构开发工具与环境, 包括 CNCC 编译工具和 CNGDB 调试工具.

目前, 寒武纪基础软件的编译系统在 RISC-V 平台上已经完成了适配. 在 RISC-V 平台上移植寒武纪基础软件的首要工作是修改构建配置文件 (比如 CMakeLists.txt 文件) 以支持 RISC-V 平台. 此外, 为了与 ARM 架构在功能和性能上对齐, 需要参考基础软件中已支持的 ARM 架构, 对部分汇编源码和脚本进行适配, 解决因指令集差异导致的兼容性问题.

2.1.4 PyTorch 框架适配

PyTorch 是当前主流的开源深度学习框架之一, 支持 Python、C++等多语言接口, 凭借其动态图机制、高效的计算图调度、良好的模块化设计, 在研究与工业部署中被广泛应用. PyTorch 的底层通过 Python C 扩展模块将前端 API 与共享核心 C 库连接, 并依托 ATen 张量计算模块适配多种计算后端 (如 CPU、GPU、TPU 等), 具备良好的异构可移植性.

为了支持寒武纪 MLU 系列加速卡, 寒武纪公司开发了 Cambricon PyTorch 框架. 该框架基于 PyTorch 提供的设备扩展接口, 将 MLU 后端库中定义的张量类型和算子操作动态注册到 PyTorch 系统中, 使其能够在 MLU 硬件上完成训练与推理过程. 从 PyTorch 1.3.0 版本起, 寒武纪通过封装 Python 扩展包 (如 torch_mlu) 的方式支持原生 PyTorch, 将所有 MLU 相关操作集中于独立模块, 避免对主框架的侵入式修改, 从而提升了跨平台兼容性和可维护性.

在本文中, 为实现大语言模型在 RISC-V 架构上的异构推理支持, 我们选择以 PyTorch 1.9 为基础版本, 完成了面向 RISC-V+MLU 平台的跨架构构建与适配. 具体适配流程包括以下两个关键步骤.

(1) 跨架构构建 PyTorch 主体框架

首先配置 RISC-V 工具链, 并对 PyTorch 源码中的平台识别宏、依赖路径 (如第三方库路径、编译器配置等) 进行必要调整, 以保证其在 RISC-V 架构下能够顺利完成源码构建与模块链接.

(2) MLU 后端支持扩展包编译与部署

随后, 在 RISC-V 根文件系统中交叉编译寒武纪提供的 PyTorch 扩展模块 (如 torch_mlu 和配套库), 生成可在目标平台上安装与运行的 Python 扩展包, 并完成相关运行时库 (如 CNNL、CNRT 等) 的集成配置.

通过上述适配过程, 本文在 RISC-V 平台上成功运行了完整的 PyTorch 框架, 并实现了对 MLU 硬件的支持. 该适配为后续构建大语言模型推理管线、实现异构协同计算提供了基础性保障, 验证了 PyTorch 框架在国产 RISC-V+MLU 平台下的可移植性与可部署性.

2.2 基于多线程的大语言模型推理优化

在大语言模型推理中, 注意力机制作为核心计算模块, 其矩阵乘法操作对整体性能影响显著. 随着序列长度和批处理规模的增长, 计算复杂度迅速上升, 成为限制推理效率的主要瓶颈.

在如 RISC-V 等多核平台上, 串行计算模式难以高效利用多核资源, 导致处理器利用率低、推理延迟高. 因此, 本文针对注意力模块提出一种轻量级多线程优化策略, 利用线程池并行执行矩阵乘法任务, 在不改动模型结构的前提下提升推理吞吐率.

方法概述: 本方法主要针对 LLM 推理中注意力机制中的矩阵乘法操作. 该操作计算密集且数据依赖性低, 具备高度可并行化特性. 本文提出一种基于线程池的轻量级优化策略, 将矩阵划分为多个子块, 并以线程为调度单元, 实现并行计算与结果重整, 整体过程包括 4 个阶段, 具体流程图如图 3 所示.

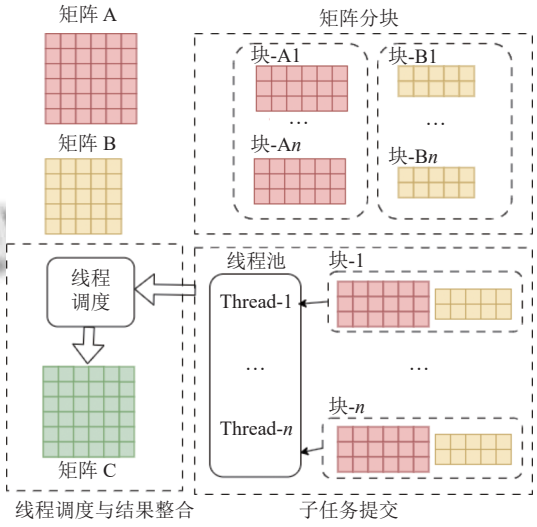

图 3 多线程推理优化方法示意图

2.2.1 矩阵分块

矩阵分块的方式有多种, 本文根据大模型数据量大的特点, 采用基于批次 (batch) 维度的分块策略, 将矩阵乘法操作按照 batch 等分, 以生成多个子矩阵. 该分块方式具备良好的通用性, 特别适用于大模型中大批量输入场景. 分块数量可根据当前任务负载和硬件

资源 (如可用 CPU 核心数) 进行动态调整, 确保各线程间计算负载尽可能均衡. TeleChat-12B 中 query 和 key 的矩阵分块实现如代码 2 所示.

代码 2. 矩阵分块实现

```
def function(self, query, key, num_parts)
  if num is None:
    total_size = query.shape[0]
    num_cores = self.num_threads
    num_parts = min(num_cores, total_size // 8)
  def _matmul_partition(q_part, k_part):
    return torch.matmul(q_part, k_part.transpose(-2, -1))
  q_partitions = torch.chunk(query, num_parts, dim=0)
  k_partitions = torch.chunk(key, num_parts, dim=0)
```

2.2.2 线程池创建

考虑到线程的频繁创建与销毁会带来较高的系统开销, 为降低线程管理成本并提升整体运行效率, 本策略引入线程池机制对线程进行统一管理. 线程池中的线程采用长期驻留方式, 避免了重复创建和销毁带来的资源浪费. 线程池的大小则根据平台的 CPU 核心数来进行确定.

实现过程中, 可通过 Python 标准库中的 ThreadPoolExecutor 创建固定规模的线程池, 相关配置如代码 3 所示.

代码 3. 线程池创建

```
# 添加线程池配置
self.num_threads = 32 #设置线程数
self.thread_pool = ThreadPoolExecutor(max_workers=self.num_threads)
```

2.2.3 线程调度

线程池创建后, 采用基于任务队列的调度策略来调度矩阵乘法子任务. 线程池内部维护一个线程安全的任务队列 (FIFO 队列), 所有子任务会先提交给线程池, 线程池会将这些任务放入工作队列中, 然后线程池中的空闲的工作线程再从队列中轮询拉取任务并执行. 该调度策略可以使得线程复用充分.

实现过程中, 可以通过 Python 标准库中的 thread_pool.submit 接口实现子任务的提交到线程池.

2.2.4 结果汇总与整合

所有子任务在被线程池异步执行后, 其计算结果会以对象的形式返回. 系统通过轮询的方式收集这些结果, 随后将各分块的结果按照分块的维度拼接, 最终还原出完整的计算后的矩阵. 这一结果汇总过程在主线程中完成, 确保了整体计算的一致性与正确性.

3 实验结果及分析

3.1 实验环境

目前, RISC-V 平台采用 SOPHGO SG2042[16]进行移植, 其主要优势体现在: 1) 采用 12 nm 工艺集成 64 个 XuanTie C920 核心; 2) 支持 RVV 0.7 向量扩展指令集; 3) 提供双通道 PCIe Gen4 x16 高速互连. SG2042 作为一款基于 RISC-V 架构的处理器, 其开源特性和灵活性使得它在定制化计算任务、科学研究和工程仿真等场景中表现出色. 具体硬件配置如表 1 所示.

表 1 SG2042 硬件配置表

硬件	硬件参数
处理器核	XuanTie C920
架构	RV64GCV
CPU核心数	64
CPU主频	2.0 GHz
内存	128 GB
PCIe	2个Gen4 x16, 支持CCIX

寒武纪加速卡采用的是 MLU370-S4, 具体硬件配置如表 2 所示. 本研究选择该设备进行适配主要基于 3 重技术优势: 首先, 架构设计在计算效率与能耗比方面表现出显著优势, 特别适用于对计算密度和能源效率有严格要求的绿色数据中心场景; 其次, 紧凑型硬件设计 (物理尺寸缩减 30% 的同时保持 128 TOPS 算力输出) 实现了业界领先的部署密度 (支持 8 卡/2U 的标准机架配置), 这使其在空间受限环境下展现出独特优势. 这些技术特性共同构成了该加速卡在高密度 AI 推理场景中的核心竞争力.

同时, 移植的基础软件环境内核版本已升级适配到 Linux 6.6.0, 操作系统版本使用 openEuler 24.03. 在软件环境方面, 重点考虑了对 RISC-V 指令集架构的支持. 具体软件环境配置如表 3 所示.

3.2 MLU370-S4 在 RISC-V 平台的功能性适配分析

本节旨在对 MLU370-S4 加速卡在 RISC-V 平台上的功能适配性进行系统化评估, 具体验证目标包括: (1) 评估 MLU370-S4 加速卡在 RISC-V 平台上对主流大模型推理能力的适配效果, 及适配后能否正常支持各类典型大模型的加载与推理运行; (2) 验证其在高负载以及长时间运行场景下的系统稳定性和可靠性, 确保加速卡能够满足实际应用场景对连续性和安全性的

要求. 针对上述目标, 分别从典型大模型的运行验证和系统稳定性两个方面开展实验.

表 2 MLU370-S4 配置表

配置项	配置详情
制程工艺	7 mm
计算精度支持	INT4、INT8、INT16、FP16、BF16、FP32
峰值性能	INT8 192 TOPS INT16 96 TOPS FP16 72 TFLOPS BF16 72 TFLOPS FP32 18 TFLOPS
内存类型	LPDDR5
内存容量/位宽	24 GB/384-bit
内存带宽	307.2 GB/s
系统接口	PCIe Gen4 x16
最大功耗	75 W
视频解码	支持

表 3 基础软件环境

基础软件	类型版本
架构	RISC-V 64
操作系统	openEuler 24.03 (LTS)
Python	3.10.6
GCC	12.3.1
G++	12.3.1
PyTorch	1.13.1
内核	Linux 6.6.0

3.2.1 典型大模型运行验证

为验证 MLU370 加速卡在 RISC-V 平台完成适配后, 是否能够支持其原有的大模型推理能力, 本文选取了 5 个主流的大语言模型在适配后的平台上进行部署与运行测试. 这些模型涵盖了 ChatGLM 系列的不同版本、轻量级语言模型 MiniCPM, 以及支持多模态的 VisualGLM, 具有较强的代表性和多样性. 其中, ChatGLM 系列模型具有双向自回归结构, 支持中英双语对话, 具有广泛的适配性[17]. MiniCPM 是参数规模较小的轻量级模型, 更适用于资源受限的场景[18]. VisualGLM 则融合了图文输入能力, 代表了当前多模态语言模型的发展趋势[18].

测试内容包括模型能否成功加载、推理流程是否完整以及其在实际运行过程中的性能表现, 重点记录了每个模型的参数规模、运行时功耗、内存占用及所用推理精度等关键指标, 以综合评估适配后的表现与适用性. 相关结果如表 4 所示.

从验证结果可以看出, 各类大模型均可在适配后的平台上运行, 并成功完成推理任务. 而且在所使用的推理精度为 FP16 的情况下, 大模型运行所占内存和大模型的参数量呈现正相关并在合理的范围内, 同时运行时的功耗也在范围内波动, 验证了 MLU370 在 RISC-V 平台适配后的有效性, 并且在能耗和内存方面达到了较为平衡的表现.

表 4 大模型运行情况表

LLM	参数量 (B)	推理精度	是否正常完成推理	功耗 (W)	内存占用 (GB)
ChatGLM	6	FP16	是	27	14.04
ChatGLM2	6	FP16	是	27	13.94
ChatGLM3	6	FP16	是	28	13.94
MiniCPM	2	FP16	是	25	6.74
VisualGLM	6	FP16	是	27	16.39

注: FP16 (floating point 16-bit) 表示16位浮点数精度

3.2.2 稳定性测试与分析

为验证进行了 MLU370 适配后, 本方案在 RISC-V 平台上的稳定性, 我们采用寒武纪官方提供的 complex 压力测试用例进行实验. 该测试用例通过并行执行主机与设备、设备内部以及不同设备间的内存拷贝操作, 模拟在高负载条件下的大规模数据流动, 旨在评估硬件系统在长时间运行和高负载情况下的稳定性. 需要注意的是, complex 测试用例通过持续的压力操作来观察设备是否能稳定运行.

本文合理设定了测试参数, 包括内存拷贝大小、执行线程数以及操作的重复次数. 此外, 为全面评估设备在不同运行时长下的稳定性, 本文分别进行了 4 h、8 h 和 12 h 这 3 个批次的压力测试.

测试结果表明, 在整个测试过程中, MLU370 设备持续进行高负载的数据拷贝操作, 未出现系统宕机或异常中断现象. 尤其在最长 12 h 的连续运行中, 设备始终保持稳定, 未发生任何故障情况, 充分证明了其在 RISC-V 平台下的可靠性和稳定性. 6 h 测试结果如图 4 所示, 进一步验证了 MLU370 设备在高负载下的稳定运行能力.

3.3 CPU 平台下多线程优化分析

为测试多线程优化策略的实际效果, 采用 TeleChat-12B 大模型, 在不使用 MLU370 加速卡的场景下, 分别在不同线程数配置下进行相同的问答任务, 测量总耗时作为性能指标. 从图 5 可以看到在不使用多线程优化策略的情况下, 模型推理总耗时为 3 966.47 s. 当线程数提升至 8 时, 推理时间略有下降, 耗时约为 3 500 s, 性能提升不明显, 表明在低并发度下, 任务调度与线程

管理的开销尚未被充分摊薄. 随着线程数进一步增至 16, 耗时降至约 3 200 s, 推理效率逐步提升. 最终在 32 线程配置下, 模型总推理耗时显著下降至 2 503.34 s, 相比未使用多线程推理优化策略加速比达 1.58 倍.

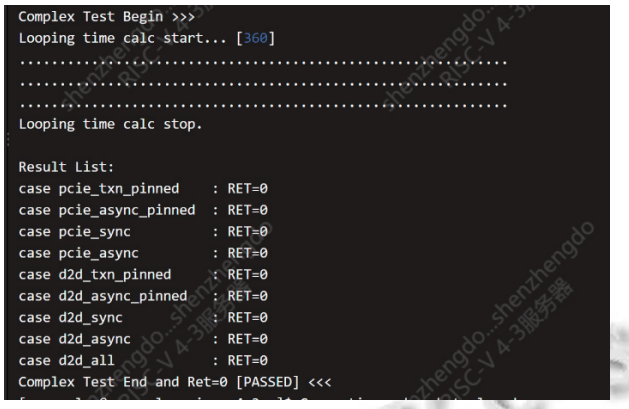

图 4 Complex 测试用例运行 6 h 结果图

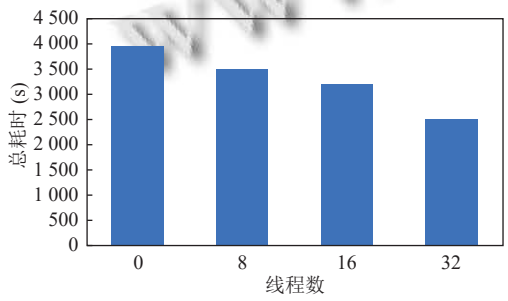

图 5 不同线程数下 TeleChat-12B 推理总耗时对比

该结果表示, 多线程优化能有效提升推理性能, 尤其在高并发配置下表现出良好加速潜力. 然而观察其平均推理吞吐率 (tokens/s) 仅从未优化前的 0.05 tokens/s 提升到了启用 32 线程优化时的 0.08 tokens/s, 未呈现显著提升. 这一现象表明, 总耗时优化并未完全转化为推理吞吐率的线性提升. 后续将针对这一问题进行相应研究分析.

3.4 多模型的推理优化性能分析

图 6 展示了 6 个不同参数量级的大语言模型在 CPU 平台推理与结合 MLU370-S4 加速卡推理两种条件下的文本生成速度 (字/s) 对比情况. 为确保测试的公平性与结果的可比性, 所有模型均采用相同的 FP16 精度配置, 未采用任何量化、剪枝或 KV Cache 等性能优化技术. 同时, 在测试过程中统一使用一组具有代表性的问答类问题作为输入, 问题内容涵盖一般性知识问答、基础逻辑推理与开放式对话生成等典型场景, 确保各模型在相同语义任务下进行推理操作.

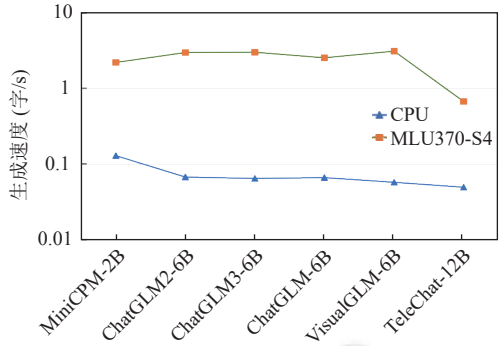

图 6 不同参数量级大语言模型在 CPU 与 MLU370-S4 平台的文本生成速度对比

从整体结果来看, 无论是参数量较小的 2B 模型, 还是规模较大的 12B 模型, 在结合 MLU370-S4 加速卡后, 其文本生成速度均显著优于纯 CPU 推理. 数据表明, 在不同模型规模下, MLU370-S4 始终展现出稳定的性能优势, 表明其在 RISC-V 架构上对大模型推理任务的良好适应性. 随着模型参数规模的扩大, 这一加速优势愈加明显. 然而, 当模型规模进一步扩展至 12B 时, 受限于 MLU370-S4 的显存带宽及 RISC-V CPU 单核计算能力, 其加速效果相较 6B 模型略有下降, 但依然大幅优于纯 CPU 推理速度.

图 7 展示了在使用 MLU370-S4 加速卡的推理条件下, 各大语言模型相较于纯 CPU 推理的文本生成速度加速比. 与图 6 中展示的响应速度变化不同, 图 7 采用定量方式呈现各模型在实际推理效率方面的提升程度, 更加直观地反映了加速卡所带来的性能增益.

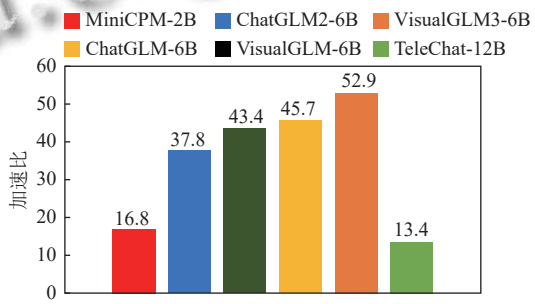

图 7 不同参数大模型在 CPU 与 MLU370-S4 平台上的文本生成加速比

实验结果表明, 所测试的 6 个模型在引入 MLU370-S4 加速后, 文本生成速度均实现数量级提升. 其中, MiniCPM-2B 在纯 CPU 推理条件下的生成速度为 0.13 字/s, 已是 6 个模型中表现最优者; 但在结合加速卡后, 其推理速度提升约 16 倍, 显示出良好的加速潜力. 尤其值

得注意的是, VisualGLM-6B 在加速后取得了最高的加速比, 端到端推理加速比达 52.3 倍, 其 95% 置信区间为[51.6, 53.1], 说明该加速效果在多次实验中保持高度稳定.

相比之下, TeleChat-12B 的加速比为 13.4, 显著低于 6B 量级模型的提升幅度. 这主要是由于其超大参数规模导致加速卡内部缓存压力显著增加, 频繁触发片外访存, 降低了算力利用率; 同时, RISC-V CPU 与加速卡之间的带宽及调度开销在大模型中被进一步放大, 使得 Host-Device 通信成为新的性能瓶颈. 此外, 大模型算子链路更长, 部分算子尚无针对 MLU370-S4 的专用实现, 需回退至 CPU 或通用计算核执行, 从而进一步稀释了整体加速效果. 这些因素共同作用, 使得在超大参数量模型下加速性能提升出现一定程度的边际减弱.

综上, 本文所提出的优化策略在 RISC-V 平台上展现出良好的适应性与广泛的适用性, 显著提升了不同规模大语言模型的推理性能. 该方案的实现也为加速卡在多样化模型场景下的高效部署提供了技术保障, 也为 RISC-V 生态在智能计算领域的深入拓展奠定了基础.

4 结论与展望

本文面向 RISC-V 架构下的大语言模型推理场景, 完成了对寒武纪 MLU370 加速卡的适配工作, 并在此基础上提出了一套结合硬件加速与多线程并行计算的优化方案. 实验证明, 该方案可显著提升大语言模型推理速度, 部分模型最高提速超过 50 倍. 研究结果验证了 RISC-V 平台与国产加速卡协同优化的可行性与有效性, 为推动国产异构计算体系在大模型推理中的应用提供了有力支持. 虽然采用国产加速卡使得大语言模型推理速度获得有效提升, 但当前方案在计算访存比和指令级并行度方面仍存在优化空间, 后续研究将聚焦于: 1) 基于 RVV 向量指令集的算子融合优化; 2) 面向稀疏注意力的动态调度策略; 3) 混合精度计算的能效比提升, 实现大语言模型在 RISC-V 平台上推理性能的进一步提升.

参考文献

1 Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 159.

2 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]

3 Devoto A, Zhao Y, Scardapane S, *et al*. A simple and effective L_2 norm-based strategy for KV cache compression. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Miami: Association for Computational Linguistics, 2024. 18476–18499.

4 Zhao YL, Gu A, Varma R, *et al*. PyTorch FSDP: Experiences on scaling fully sharded data parallel. Proceedings of the VLDB Endowment, 2023, 16(12): 3848–3860. [doi: 10.14778/3611540.3611569]

5 Frantar E, Castro RL, Chen JL, *et al*. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. Las Vegas: ACM, 2025. 239–251.

6 Ai X, Yuan H, Ling ZY, *et al*. NeutronTP: Load-balanced distributed full-graph GNN training with tensor parallelism. Proceedings of the VLDB Endowment, 2024, 18(2): 173–186. [doi: 10.14778/3705829.3705837]

7 Dao T, Fu DY, Ermon S, *et al*. FLASHATTENTION: Fast and memory-efficient exact attention with IO-awareness. Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 1189.

8 Dao T. FlashAttention-2: Faster attention with better parallelism and work partitioning. Proceedings of the 12th International Conference on Learning Representations. Vienna: OpenReview.net, 2024.

9 Shah J, Bikshandi G, Zhang Y, *et al*. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. Proceedings of the 38th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2024. 2193.

10 Aminabadi RY, Rajbhandari S, Awan AA, *et al*. DeepSpeed-Inference: Enabling efficient inference of Transformer models at unprecedented scale. Proceedings of the 2022 International Conference for High Performance Computing, Networking, Storage and Analysis. Dallas: IEEE, 2022. 1–15.

11 RISC-V International. The RISC-V vector extension version 1.0. https://github.com/riscv/riscv-v-spec/release/tag/v1.0. (2021-12-01)[2025-04-15].

12 Perotti M, Cavalcante M, Wistoff N, *et al*. A “new ara” for vector computing: An open source highly efficient RISC-V V1.0 vector processor design. Proceedings of the 33rd IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP). Gothenburg: IEEE, 2022. 43–51.

13 Chander VN, Varghese K. A soft RISC-V vector processor for Edge-AI. Proceedings of the 35th International Conference on VLSI Design and 21st International Conference on Embedded Systems (VLSID). Bangalore: IEEE, 2022. 263–268.

14 SiFive. SiFive intelligence X280 core IP product brief. https://www.sifive.com/cores/intelligence-x280. [2025-04-15].

15 Andes Technology. AndesCore NX27V: A RISC-V vector processor IP for AI/DSP workloads. https://www.andestech.com/. [2025-04-15].

16 Brown N, Jamieson M, Lee J, *et al*. Is RISC-V ready for HPC prime-time: Evaluating the 64-core Sophon SG2042 RISC-V CPU. Proceedings of the 2023 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis. Denver: ACM, 2023. 1566–1574.

17 Zeng AH, Xu B, Wang BW, *et al*. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools. arXiv:2406.12793, 2024.

18 Hu SD, Tu YG, Han X, *et al*. MiniCPM: Unveiling the potential of small language models with scalable training strategies. Proceedings of the 1st Conference on Language Modeling. Philadelphia: OpenReview.net, 2024.

(校对责编: 张重毅)