https://crad.ict.ac.cn
63(3): 782−797, 2026

SW-Inf: 基于申威智能加速卡的 Transformer 高效推理方法

沈文渊[1] 汤凌韬[1] 何昊天[1] 费天成[1] 赵玉龙[1] 张鲁飞[1] 秦晓军[1] 刘 鑫[2] 陈左宁[2]
[1](数学工程与先进计算国家重点实验室 江苏无锡 214125)
[2](国家并行计算机工程技术研究中心 北京 100083)
(wenyuan17@yeah.net)

SW-Inf: Efficient Transformer Inference Method for Sunway AI Accelerator Card

Shen Wenyuan[1], Tang Lingtao[1], He Haotian[1], Fei Tiancheng[1], Zhao Yulong[1], Zhang Lufei[1], Qin Xiaojun[1], Liu Xin[2], and Chen Zuoning[2]
[1](*State Key Laboratory of Mathematical Engineering and Advanced Computing*, *Wuxi*, *Jiangsu* 214125)
[2](*National Parallel Computer Engineering Technology Research Center*, *Beijing* 100083)

Abstract Transformer has fundamentally revolutionized various fields of natural language processing through its self-attention mechanism that effectively models long-range contextual relationships. It overcomes traditional neural networks' limitations in handling long sequences, and has emerged as the mainstream architecture for modern large language models. Currently, Transformer inference systems generally face a shortage of computing power and are commonly confronted with challenges like substantial scheduling latency, concurrency execution difficulties, and severe memory access limitations during deployment. The Sunway AI accelerator card, featuring a domestically-developed next-generation intelligent processor, has successfully empowered diverse edge inference deployments. However, conventional inference frameworks struggle to effectively explore the architectural potential in three critical dimensions: robust system control capabilities, multi-level parallel computing, and rich memory hierarchy. We propose SW-Inf: a high-efficiency Transformer inference method based on the Sunway AI accelerator card. First, an on-chip management processing element (MPE)-orchestrated computation graph scheduling mechanism is proposed to minimize kernel launch latency and inter-process data transfer overhead through graph aggregation and MPE coordination. Second, an LDM-prioritized heuristic tiling strategy is introduced to enable load-balanced parallel computation across processing elements. Finally, a hardware-aware tightly-coupled attention computation implementation is developed to eliminate redundant intermediate data movement between registers and main memory. We implement an end-to-end inference system on the Sunway AI accelerator card. Experimental results demonstrate that compared with the vLLM inference system on NVIDIA V100S which has higher memory bandwidth than the Sunway AI accelerator card, SW-Inf achieves 15% higher throughput and 20% better memory utilization rate.

Key words Sunway AI accelerator card; large model inference; scheduling optimization; parallel strategy; operator fusion

摘 要 Transformer 通过自注意力机制高效捕捉全局依赖,突破了传统神经网络处理长序列信息时的限制,推动了自然语言处理多个领域的革命性进展,成为现代大语言模型的核心架构。目前,国内 Transformer 推理应用整体上处于算力紧缺状态,且在部署时往往面临调度开销大、计算并发难、访存瓶颈明显等问题。申威智能加速卡搭载了国产自主研发的新一代智能处理器,已有效支撑各类端侧 Transformer 推理应用,然而常规推理框架难以发挥申威异构众核架构的管理功能完备、多级并行计算和存储层次丰富等特点。对此,提出一种基于申威智能加速卡的 Transformer 高效推理方法 SW-Inf。首先,提出了片上主核驱动的

收稿日期: 2025-03-01; **修回日期**: 2025-09-24
通信作者: 刘鑫(yyylx@263.net)

整图下沉调度机制，通过计算图打包和片上主核调度，减少主机与设备间的核函数启动开销和设备内各流程数据传输开销；其次，设计了 LDM 优先的启发式分块并行策略，实现计算部件负载均衡，提高了硬件资源利用率；最后，实现了体系结构紧耦合的注意力计算方法，避免推理的中间结果频繁读写主存和寄存器，缓解端侧推理应用的访存瓶颈问题。为验证所提方法的效果，基于申威智能加速卡实现了一套端到端推理系统，测试结果表明，相较于访存带宽更高的英伟达 V100S 上的 vLLM 推理系统，SW-Inf 的吞吐率和主存利用率分别提高了 15 个百分点和 20 个百分点。

关键词 申威智能加速卡；大模型推理；调度优化；并行策略；算子融合

中图法分类号 TP311

DOI: 10.7544/issn1000-1239.202550153 **CSTR:** 32373.14.issn1000-1239.202550153

近年来，随着人工智能技术的飞速发展，以 ChatGPT 为代表的大语言模型在自然语言处理、计算机视觉和语音识别等通用人工智能领域取得了突破性进展[1-3]。凭借其强大的表示与学习能力，能够高效处理各类复杂任务[4]，如高质量文本生成[5]、精准机器翻译[6]和智能问答[7]等。与此同时，智能监控、自动驾驶和工业自动化等端侧应用场景对人工智能模型的推理提出包括低延迟、低功耗和高资源利用率在内的更高要求。

Transformer[8]作为目前大语言模型的典型架构，凭借注意力机制高效处理长距离依赖问题，同时增强并行计算度显著提升了训练效率，同时 Transformer 架构在跨模态应用的广泛适应性，使其在多模态任务中展现出强大的整合能力。然而，随着模型规模的不断扩大，Transformer 架构的注意力机制的计算复杂度与序列长度呈二次方关系增长，导致在处理长序列数据时计算资源消耗显著增加。此外，在推理阶段，Transformer 架构对内存和计算资源的需求极高，尤其是当序列长度增加时，内存使用量会急剧上升。这种高计算复杂度和内存需求使其在边缘部署场景下面临严峻挑战[9-10]。

各界为此作了许多探索。硬件方面，英伟达（Nvidia）GPU 凭借强大的并行计算能力、高带宽显存和专为深度学习优化的 TensorCore，在大规模模型智能推理应用中表现出色，成为 Transformer 推理的首选设备。软件方面，SGLang[11]是一个用于高效执行复杂语言模型程序的开源系统，由前端语言简化编程，运行时利用 RadixAttention 和压缩有限状态机等优化技术提升效率，经多种任务评估表现优异。vLLM 通过 PagedAttention 技术、连续批处理技术，并结合英伟达 Triton 的动态张量并行，在 A100 GPU 上实现 1 750 token/s 的 GPT-3 吞吐量，较 PyTorch 提升 6.2 倍，成为推理系统的行业标杆。

申威智能处理器是申威系列面向智能计算加速的国产处理器，硬件上具备了包括环网总线、从核阵列和脉动阵列等，软件上包含了自研的驱动运行时、算子库、动态内存管理单元和自适应编译优化器等，已具备一定的 Transformer 推理能力。但是相较于 Nvidia GPU，申威智能加速卡（SW-AI）采用异构众核架构，具备巨大的软硬协同优化空间，可弥补其在矩阵加速数据类型和片上网络带宽等方面等存在的限制。其面临的具体挑战与机遇如下：

1）Transformer 推理流程中的大量算子均需发起主机和设备间跨 PCIe 的交互，使得调度开销成为推理耗时中不可忽视的一部分。而申威智能处理器具备异构众核架构，利用主核强大的自主控制能力，将整个 Transformer 推理过程转移到主核上，把主机-设备间的 PCIe 交互转化为芯片内主从核间的交互。

2）申威智能加速卡的计算和存储层次丰富、特征明显，需要设计有效的任务划分和并行计算策略，同时各部件的使用在核内局存受限的情况下有各自的数据对齐和并行设计要求，加剧内存管理复杂度。这要求“LDM 优先”设计必须兼顾计算单元适配、存储限制突破及通信优化 3 重目标，在硬件特性与软件易用性间寻求平衡。

3）申威智能加速卡拥有高计算能力，但在面临访存瓶颈的 Decoder-Only 模型推理场景下，访存带宽无法很好地匹配计算能力，因此如何减少算子间冗余访存，同时提高计算访存并发度，是提高模型推理效率的关键。

针对上述挑战与机遇，本文研究基于申威智能加速卡的 Transformer 高效推理方法 SW-Inf，主要贡献有 3 点：

1）提出了基于片上主核整图下沉调度机制，利用申威处理器主核的调度管理能力，降低推理过程中主机与设备间的交互次数，减少调度开销。

2)设计了 LDM 优先的启发式并行策略，实现申威众核架构下计算和存储资源的负载均衡，通过计算、访存、通信相互掩盖的方式，减少不同部件间的数据等待，提高整体资源利用率，降低推理时延。

3)实现了多层内存高效的注意力机制，从算子、指令集和寄存器 3 个层面解决了 FlashAttention 算法在申威智能处理器上算子内冗余访存高、计算效率低和寄存器利用率不高等问题。

结合上述优化方法而实现的推理系统 SW-Inf 与申威处理器的计算单元深度对接，确保指令级兼容性，具有较强的推理效率和稳定性，可直接整合至申威云端服务器与超算平台，强化生态 AI 能力。

1 相关工作

大语言模型因其在各类任务中展现出强大的理解和推理能力而受业界广泛关注。近年来，大模型参数规模的快速增长带来了更强的模型能力，同时也极大增加了模型推理复杂度，如何提高硬件利用效率和满足大模型在不同推理场景下的计算和存储需求，成为研究热点。

模型算法方面，FlashAttention[12] 基于循环分块和计算融合的思想，结合 online Softmax 技术，提出 IO 感知的精确注意力算法，大幅减少了中间结果对主存的冗余访存，从而提升注意力计算性能。MLA(multi latent attention)是 DeepSeek-V2[13] 采用的创新注意力机制，通过低秩键值联合压缩，在性能上超越传统的多头注意力模块，同时大幅降低 KV Cache 存储量。

推理服务系统方面的主要挑战是在高并发请求情况下充分利用硬件资源减少推理气泡，同时优化资源管理，提高系统响应速度和吞吐率。Orca[1] 提出了迭代级调度策略和连续批处理技术，实现了细粒度资源的利用和灵活请求响应，有效减少了请求的平均等待时间，提升系统的整体响应速度与资源利用率。PagedAttention[14] 针对推理服务中 KV Cache 导致的内存碎片问题，借鉴操作系统虚拟内存和分页技术的思路进行缓存管理，实现灵活的内存共享，从而提升内存利用率。上述工作能有效提高系统吞吐率，但由于单批请求负载增大可能导致推理时延变长，对此 Sarathi[15] 将处于预填充阶段的请求拆分为几乎相等大小的块，并构建无停顿调度，在不暂停解码阶段请求的情况下添加新请求，从而平衡吞吐量和延迟之间的矛盾。

端侧推理应用方面的主要挑战是在硬件资源受限的情况下实现模型推理，尽可能提高硬件资源利用率。FlexGen[16] 设计了一种运行在有限 GPU 资源上的 LLM 推理引擎，通过整合 GPU、CPU、磁盘等计算和存储资源进行高效任务卸载，高效调度 I/O 操作。同时，采用细粒度分组量化将权重和缓存压缩至 4 b 以提升性能，实现了在单 GPU 上的千亿参数模型推理。类似的，PowerInfer[17] 研究消费级 GPU 上的 LLM 推理优化方法，借鉴 Deja Vu[3] 中模型推理过程只有部分神经元被激活这一观察，对模型进行动态剪枝，并根据神经元的激活概率将其分为冷热 2 类神经元，通过 GPU 处理热神经元以及 CPU 处理冷神经元，实现了 GPU-CPU 混合推理方法，充分利用了整机资源。类似的工作还有 SparseGPT[2]，Michel[18]，Bansal[19]，LLMinFlash[20] 等。另外，诸如 LLaMA.cpp[21]，MNN[22] 等开发框架有效摆脱了大模型推理对 AI 框架和 GPU 设备的依赖，支持个人主机或各类边缘设备上的推理应用开发与部署。

随着国产智能处理器的发展，基于国产硬件平台优化大模型性能的研究逐渐涌现。例如，SWattention[23] 是针对 SW26010pro 处理器的精确注意力高效计算方法，文献 [24] 基于国产寒武纪 MLU270 芯片提出了一种针对 YOLOv5 模型的推理优化方法。BaGuaLu[25] 是世界上第 1 个支持人脑神经元突触规模的神经网络的训练平台，成功将预训练模型扩展到新一代神威超级计算机的整机规模，支持最大 174 万亿参数量模型的训练。

总体来说，目前大模型性能优化的大量研究工作仍聚焦于 GPU 平台，基于国产智能处理器的研究相对较少，缺乏具有针对性的算法与系统优化方案，难以直接移植现有技术。例如 2024 年 Tri Dao 联合 Meta, Nvidia 等机构提出了 FlashAttention-3[26]，针对 H100 新增的张量内存加速器(tensor memory accelerator, TMA)和 FP8 低精度运算支持，进一步提升了注意力计算效率，该方法为 Hopper 架构 GPU 量身定制。对于上述现状，本文聚焦大语言模型在国产申威平台上的端侧部署问题，首次系统性分析申威智能加速卡在推理流程中的计算与存储瓶颈，并围绕调度控制、并行计算与算子优化 3 个方面，提出一套端到端的高效推理方案 SW-Inf。

2 研究背景

2.1 申威智能处理器

申威异构众核智能处理器采用分层式架构设计，

如图 1 所示，其核心组件包括环形网络总线、1 个管理控制核（management processing element, MPE）和 4 个从核组（core group, CG）。其中，MPE 作为主核，采用 64 位 RISC 指令集架构，承担全局资源调度、任务分配及系统管理等核心功能。每个 CG 包含 1 个高速存储控制器和 32 个运算核心（computing processing element, CPE）。CPE 也称从核，以 4×8 的 2 维 mesh 拓扑结构互连，形成高效的阵列内数据交互网络。每个从核均配备核内局存（local data memory, LDM）、通信引擎（DMA）、向量计算部件以及专用于矩阵运算的脉动阵列。运算核心之间通过阵列内高速网络进行互连。

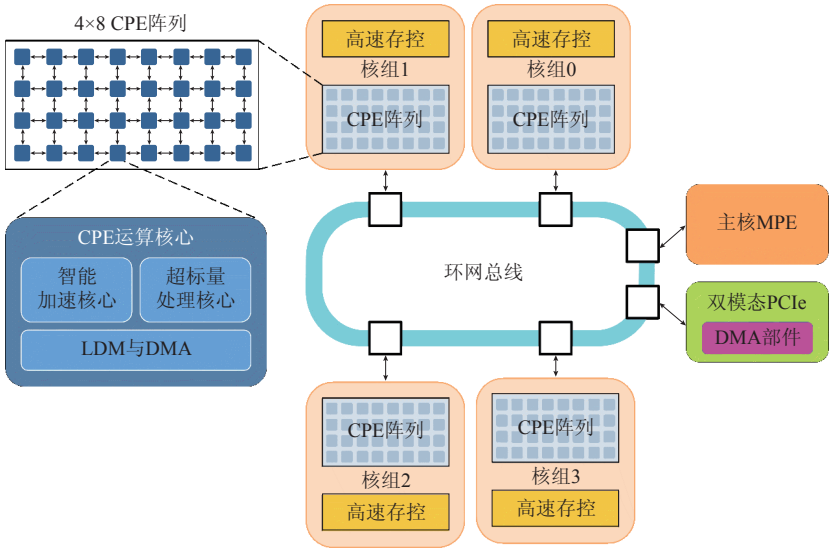


Fig. 1　Structure of Sunway AI processor

图 1　申威智能处理器结构

申威智能处理器的片上存储系统呈现 3 级层次结构：寄存器（register）→LDM→核组共享主存（HBM），访存带宽逐级递减。申威处理器的计算能力与英伟达 A100 相当，但在访存带宽指标上弱于 V100S 和 A100。相较于 V100S/A100，申威加速卡具备以下特征：1）采用大规模脉动阵列替代 tensor core，但需严格满足数据形状与最小计算单元对齐；2）每个从核执行单线程任务，寄存器文件大小远小于 GPU；3）取消 2 级缓存，依赖阵列广播实现数据复用；4）支持基于 RMA 的 LDM 间显式通信。这些架构特性为进一步优化访存能力提供了可能，但同时要求必须严格依赖 LDM 进行合理规划。一方面脉动阵列需依赖 LDM 空间排布实现严格的数据对齐，数据复用同样必须考虑 LDM 空间受限；另一方面 DMA/RMA 通信的高效性同样依赖 LDM 地址对齐。因此，申威异构众核智能处理器的架构特性要求在面对 Transformer 等大规模计算模型推理需求时必须严格依赖 LDM 的空间布局，以满足其容量限制和脉动阵列对齐要求。在此基础上实施的精细化分块策略是实现计算粒度适配、访存开销最小化以及核间通信优化，充分发挥申威处理器众核优势的关键。

2.2　推理的调度开销

以 GPU 为代表的 AI 加速卡通常是控制与计算分离式架构，会导致大模型推理过程存在控制流程冗余、主机设备交互频繁等问题，产生大量计算外的开销，包括核函数启动和数据拷贝等。模型推理流程的算子无论是 cuDNN 还是 cuBLAS 实现，都涉及众多 launch 或 memcpy 操作。例如，仅 1 个 conv1d 算子就涉及多达 7 次操作，包括 1 次 Copy_kernel_Cuda、2 次 nchwToNhwcKernel、1 次 CUDAFunction_Add 和 1 次 computeOffsetsKernel 等。此外，memcpy, getDevice, moduleUnload 等处理 CUDA 资源分配和释放的辅助操作进一步增加了主机和设备之间的交互开销。

本文在 NVIDIA V100S 平台对 Meta-LLaMA 的 LLaMA2-7B 模型进行单批次推理耗时分析。将推理耗时分为模型加载、API 调用、GPU 计算及其他开销。如图 2 所示，在批处理数目（batch size）为 1 的典型端侧推理场景下，API 调用时间占比高达 48.1%，高于 GPU 实际计算耗时。基于此，在未深度优化的推理系统中，主机与加速卡间产生高频次控制流交互，每次 kernel 启动需经历指令下发、状态查询、结果回传

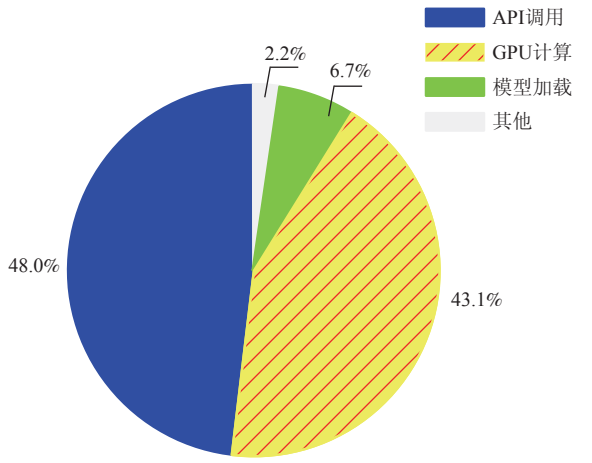


Fig. 2 Inference time consumption breakdown of Meta-LLaMA

图 2 Meta-LLaMA 推理耗时分解

等过程。当处理 LLaMA2-7B 这类包含 120+异构算子的模型时，接近一半的推理时间用于上述交互。

申威智能加速卡片上主核 MPE 具备强大逻辑控制能力。在大模型推理时，将整个 Transformer 推理过程转移到 MPE 上，可以将主机和设备间 PCIe 的数据交互转变为芯片内 MPE 和 CPE 间的数据交互，从而降低主机和设备之间交互的开销。

2.3 推理的访存瓶颈

Transformer 推理过程分为预填充(prefill)和解码(decoding)两个阶段。预填充阶段处理用户输入的多词元提示词，主要算子为矩阵乘，其需要的计算能力与访存带宽的比值高，且与提示词长度呈正相关；而解码阶段输入的是上一迭代产生的单个词元，主要算子为矩阵向量乘，计算访存比较低，但随着批处理数目的增加会有所提升。表 1 给出了 LLaMA2 模型中多头自注意力(multi-head attention, MHA)和前馈神经网络(feedforward neural network, FFN)两个关键模块在不同输入句长 N 下的计算访存比估计。

Table 1 Arithmetic Intensity of the Core Modules in LLaMA2 Model

表 1 LLaMA2 模型核心模块计算访存比

句长 N	模型选择	MHA 线性映射	MHA 核心计算	FFN
128		120	43	123
512	LLaMA2-7B	409	51	437
4 096		1 365	62	1 727
128		455	43	125
512	LLaMA2-70B	2 048	51	474
4 096		2 048	62	2 493

当前，AI 加速卡针对大规模矩阵乘法中数据高复用的特性，提供了极高的计算能力。例如，英伟达 V100S PCIe 加速卡的半精度计算能力达到 130 TFLOPS，访存带宽为 1 134 GBps，计算访存比约为 117∶1；而 A100 40 GB PCIe 带宽高达 1 555 GBps，计算访存比分别增长到 195∶1。随着 GPU 技术的不断发展，计算能力与访存能力的差距逐渐增大，这使得 AI 加速卡能够充分满足模型预训练和推理服务系统等计算密集型场景对高算力的需求。但在端侧推理场景中，通常只有少量并发请求，“单模型单用户”是一种典型的应用模式。在这种模式下，模型 1 次只处理 1 个推理请求，且自解码阶段的输入句长 N=1，计算访存比较低，导致明显的访存瓶颈，计算单元常常处于等待数据的空闲状态，严重影响端侧推理效率。

申威智能加速卡作为一款适用于训练与推理场景的通用加速卡，其半精度计算能力与访存带宽的比值约 320∶1。在批处理数目较小时，尤其是在单请求推理场景下，计算与访存能力失配问题愈发突出。如图 3 展示了基于申威加速卡 MHA 和 FFN 模块的执行耗时分解，其中注意力头数为 8，序列长度为 256，注意力头维度为 64。可见对于 2 个核心模块，访存耗时都占总体耗时的 90% 以上，导致访存成为推理流程瓶颈。本文重点关注在小批处理数目和单请求推理场景下的性能优化策略，此类场景下访存瓶颈问题最为突出。

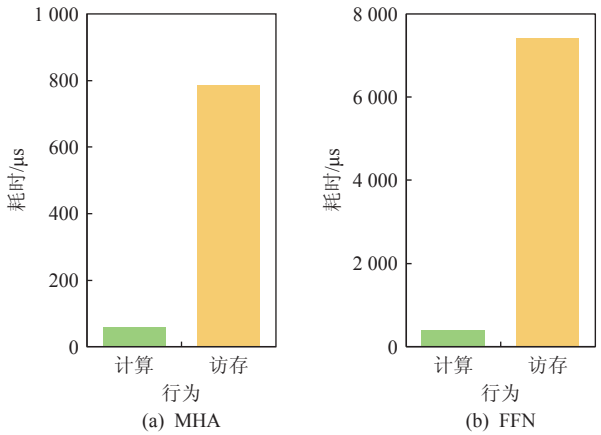


Fig. 3 Analysis of time consumption for MHA and FFN on Sunway AI processor

图 3 申威智能处理器上 MHA 和 FFN 耗时分析

3 SW-Inf 设计实现

3.1 总体架构

本文结合申威智能加速卡的体系结构特点，设计并实现了面向端侧推理的高效 Transformer 推理系

统 SW-Inf，总体架构如图 4 所示。SW-Inf 推理系统的主机侧包含专用算子库、轻量级 AI 框架、申威智能加速卡的运行时系统 SDAA[27]（software defined accelerator architecture）等部分，以支持上层各类推理应用。当主机侧发起一个 Transformer 推理应用时，首先，SW-Inf 将模型参数拷贝至加速卡主存，通过整图下沉技术将单次推理迭代涉及的算子打包成一个核函数进行发起。发起后输入至加速卡由片上主核进行调度，极大减少主机和设备间通过 PCIe 总线的交互次数，降低内核发起开销。其次，结合硬件的特性，设计多级并行策略将计算任务和相关数据分配至各从核，减少循环分块引入的冗余访存，并实现众核间的负载均衡。最后，对每个从核计算任务中的算子进行深度融合，在受限的 LDM 空间下尽可能合并相邻操作符，避免中间结果频繁读写 HBM 和寄存器，从而减少访存开销，提高执行效率。

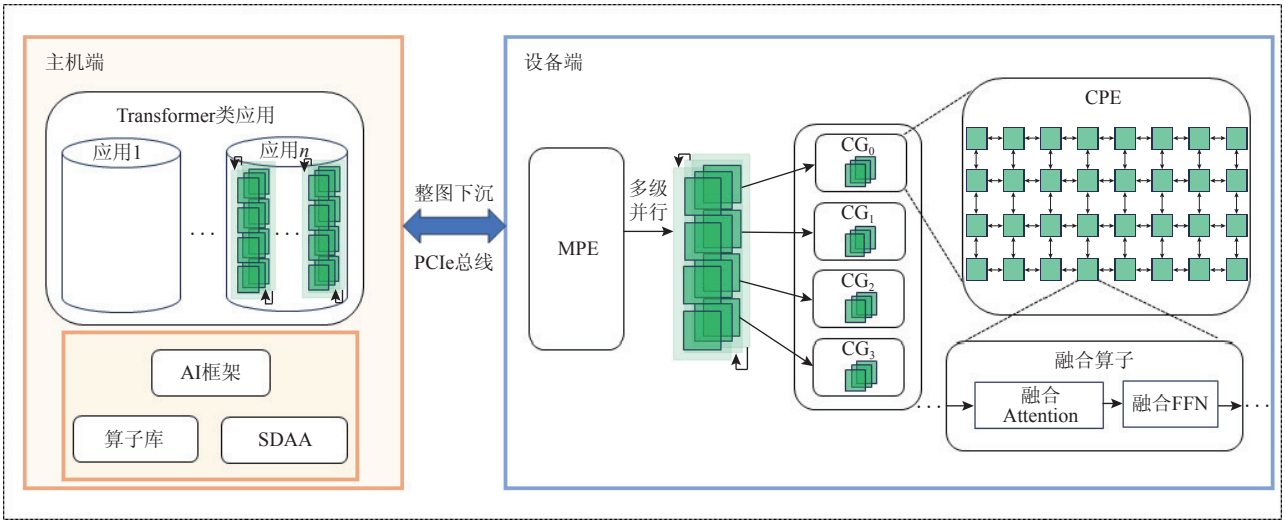


Fig. 4　Overall architecture of SW-Inf

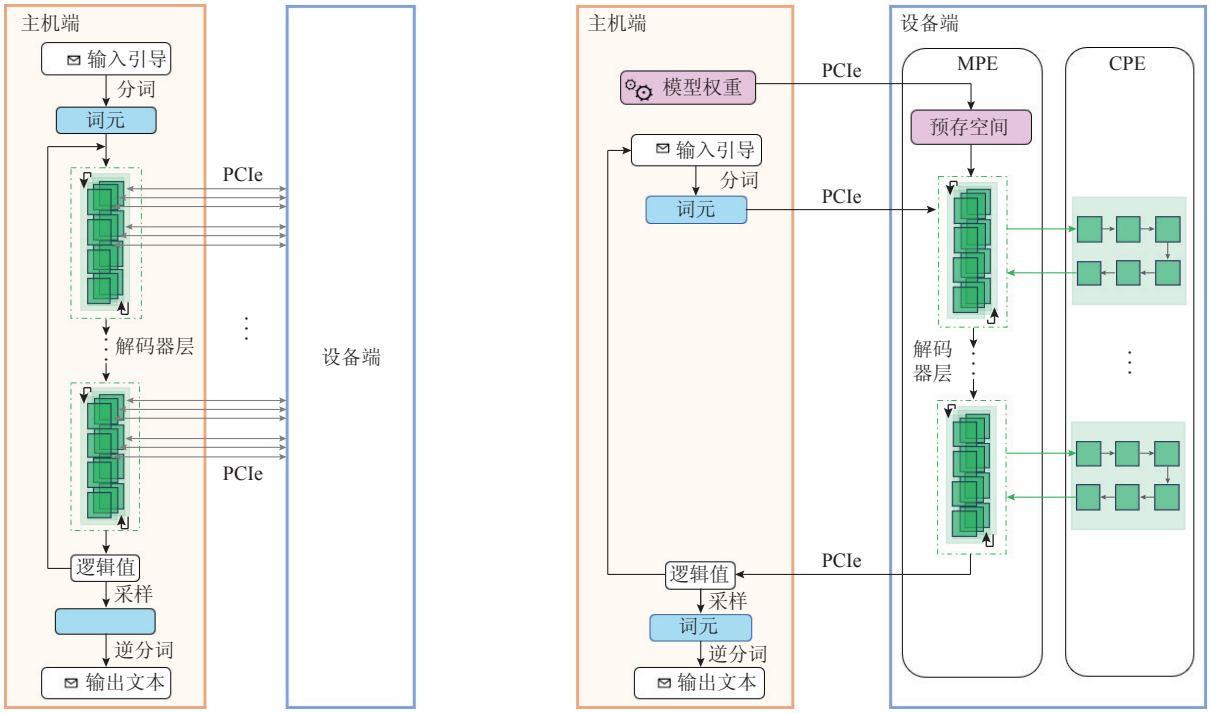
图 4　SW-Inf 总体架构

3.2　片上主核驱动的整图下沉调度机制

Transformer 推理过程涉及大量算子，如图 5(a)所示，每次核函数启动均需要通过 PCIe 总线进行主机-设备交互，单次交互时间在微秒级别。针对此问题，GPU 采用分层控制模式：CPU 通过 CUDA Graphs 预编译静态计算图以降低 Host 干预，但其固化执行路径难以适配动态条件分支，必须将分支拆解为独立 Kernel，且跨设备协调需依赖全局内存原子操作，底层由流式多处理器（streaming multiprocessor，SM）的 Warp 调度器执行线程级资源分配。然而，主从架构中主核的集中式控制与 GPU 的异构分层调度存在本质差异，使得前者在复杂逻辑密集型推理任务中更具扩展性与适应性。因此，本节主要结合申威架构主从核异构特点减少核函数启动开销。

基于上述分析，下面将详细介绍本文提出的基于整图下沉的多级调度技术，如图 5(b)所示。该技术将输入文本转换为词元向量后，仅需进行 1 次 PCIe 启动，然后将任务转移到 MPE 执行。在 MPE 内，每个编码和解码步骤仅涉及 1 次启动操作，并且 MPE 利用计算图将计算任务分配给 CPE。随后，CPE 根据计算图的内部结构执行这些任务，生成文本，并将结果返回给主机进行输出。

基于整图下沉多级调度机制分为模型预加载和模型执行 2 个阶段。在模型预加载阶段，多级调度机制利用了 MPE 强大的逻辑控制能力，引入分层弹性拆分机制将整个 Transformer 模型按层或算子粒度拆分为多个子图单元，并根据加速卡存储余量，动态选择子图粒度，最终将逻辑计算图转移到 MPE 上，把主机和设备间的 PCIe 交互转化为芯片内 MPE 和 CPE 间的交互。这种转化充分发挥了设备端的计算能力，减少了因主机端与设备端之间的协调而产生的额外开销。在模型执行阶段，为了减少过多的启动次数，完成模型预加载后，在 MPE 和 CPE 之间调度同样采用整图下沉方法。具体做法是利用 MPE 构建一个包含多个算子的计算图。对于参数量较大的模型，MPE 会进一步将计算图按层拆分为多个子图，并通过异步流水线实现子图预取、参数换入换出。构建完成后，通过执行启动操作，以计算图的形式将多个计算任务分配给 CPE，而不是针对每个单独的计算操作进行启动。根据计算图的规模，可将启动



(a) 传统调度机制 (b) 基于整图下沉3级调度机制

Fig. 5 Illustration of the full-graph descent scheduling principle

图 5 整图下沉调度原理示意图

操作数量缩减至原来的千分之几甚至万分之几, 显著提升了 Transformer 推理在核组和从核之间的片上交互效率, 使系统资源得到更充分的利用。

整图下沉方法在图构建过程中会产生一定的开销。为了减少这种开销, 本文采用了一种异步执行方法。具体来说, 在 CPE 执行当前图的计算时, MPE 同时构建后续的图, 这有效地掩盖了部分图构建开销。由于 CPE 和 MPE 相互独立, 本文在 CPE 进行计算的同时, 利用 MPE 进行图的构建。针对超大模型, MPE 会进一步将后续图拆分为更细粒度的子图, 并动态调整预加载顺序。在理想情况下, 可实现图构建与计算的完全并行化。例如, 可在当前 Transformer 层计算过程中, 构建下一个 Transformer 层的计算图。从算子层面来看, 算子内部数据交互可优化为整体数据搬运, 一次性将该算子中各数据整体打包传输到 CPE。在极端情况下, 整个推理流程可合并为 1 张计算图, 从而将单次完整推理的核函数启动次数降至 2 次。这不仅减少片内总体访存量, 而且大数据量带来的传输带宽往往更高。

3.3 LDM 优先的启发式分块并行策略

在 2.1 节中, 通过与 GPU 架构的对比分析, 指出深度挖掘申威架构计算潜力的关键在于利用片上局部存储器 LDM。为此, 本文提出了一种以最小化推理访存量为目标的 LDM 优先启发式分块并行策略, 旨在显著提高带宽利用率并完善计算流水线。

以矩阵乘计算 $\boldsymbol{Y}=\boldsymbol{X}\times\boldsymbol{W}$, $\boldsymbol{X}\in\mathbb{R}^{m\times d}$, $\boldsymbol{W}\in\mathbb{R}^{d\times n}$, $\boldsymbol{Y}\in\mathbb{R}^{m\times n}$ 为例, 本文深入探讨了该策略的应用。在典型的 Transformer 架构中, 注意力层和全连接层通常涉及大量矩阵乘运算, 通常有 $n=d$ 或 $n=4d$。在端侧推理场景下, 由于并发请求较少, 推理过程主要集中在自解码阶段, 此时有 $m\ll d$。特别地, 单请求情况下, $m=1$。图 6 展示该策略整体流程, 其 3 个子图分别阐述了策略在不同维度上的考量、具体步骤及其设计动机。

针对跨核组通信开销显著高于核组内通信开销的特性, 分块策略的首要目标是将数据同步约束在核组内部, 从而减少跨核组通信的频率和开销。通过将权值矩阵 $\boldsymbol{W}$ 的 n 维切分, 实现 4 个核组间的数据独立性, 如图 6(a)所示。在处理单请求场景时, 优先将 $\boldsymbol{W}$ 的 n 维划分至各核组 HBM 中, 核组内矩阵乘运算维度为

$$(m,d)\times\left(d,\frac{n}{4}\right)=\left(m,\frac{n}{4}\right)。\tag{1}$$

完成核组间划分后, 需在核组内进行细粒度任务分配, 以充分发挥申威架构中 32 个从核的并行计算能力。假设 m, d, n 被平均划分为 k_1, k_2, k_3 块, 每个从核中处理的矩阵形状为

$$\left(\frac{m}{k_1},\frac{d}{k_2}\right)\times\left(\frac{d}{k_2},\frac{n}{4k_3}\right)=\left(\frac{m}{k_1},\frac{n}{4k_3}\right),\tag{2}$$

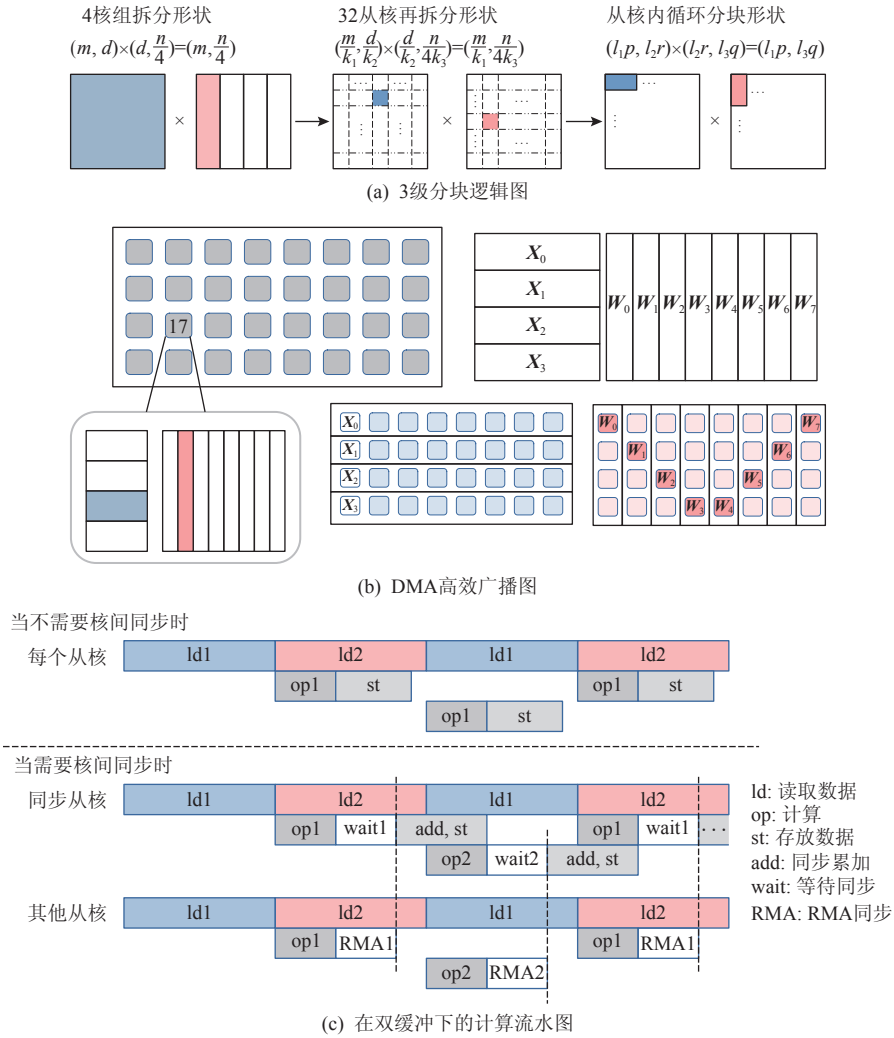


Fig. 6 Multi-level parallel computing strategy

图 6 多级并行计算策略

$k_1, k_2, k_3 \in \mathbb{N}_+$, $k_1k_2k_3 = 32$。

当块大小仍然超过 LDM 空间，需在从核内设计循环分块策略。考虑以下因素：首先，计算部件对输入数据形状和存储模式的要求存在最小计算单元的限制；其次，为实现访存与计算并行，需考虑流水操作对存取行为和内存分配的影响；最后，考虑内积计算数据换入换出频次高、外积计算 LDM 空间占用大，需采用混合策略以平衡访存量和空间限制，合理设计分块大小。所以设脉动阵列要求的最小计算单元维度如式(3)所示，通过循环分块策略(l_1, l_2, l_3)确定单次存取任务规模如式(4)所示。在半精度条件下，分块大小需满足式(5)的双重约束：首先，内存占用不超过 LDM 可用空间 M。采用双缓冲流水机制时，输入矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 需双份存储以重叠计算与访存，而结果矩阵通过实时累加仅需单份存储。其次，各维度不突破扩展上限。其中 $up(x, y)$ 函数表示将 x 扩充到 y 的倍数，用于处理维度划分余项，在核组划分时需扩展到最小计算单元的整数倍；从核划分时则需对齐循环分块维度，下文推导的分块最优策略公式默认满足倍数关系。

$$(p,r)\times(r,q)=(p,q), \tag{3}$$

$$(l_1p,l_2r)\times(l_2r,l_3q)\rightarrow(l_1p,l_3q), \tag{4}$$

$$\begin{cases} 2l_1l_2pr+2l_2l_3qr+l_2l_3pq\leqslant M/2, \\ l_1p\leqslant up\left(\dfrac{m}{k_1},p\right), \\ l_2r\leqslant up\left(\dfrac{d}{k_2},r\right), \\ l_3q\leqslant up\left(\dfrac{n}{4k_3},q\right), \\ l_1,l_2,l_3\in\mathbb{N}_+。 \end{cases} \tag{5}$$

在满足上述 LDM 容量限制和各部件对齐要求后，最小化各从核的访存复杂度以及从核之间的通信复杂度，进一步确定分块最优策略。若 LDM 空间能容纳式(2)中的输入输出分块，则优化问题可表

示为

$$\arg\min\frac{2d}{k_2}\left(\frac{m}{k_1}+\frac{n}{4k_3}\right)+\gamma\frac{mn}{2k_1k_3}(k_2-1), \quad (6)$$

$$\text{s.t. } k_1,k_2,k_3\in\mathbb{N}_+,\ k_1k_2k_3=32。$$

反之，当受 LDM 容量限制时，分块策略需在细粒度层面优化。当输入形状(m, n, d)和硬件参数(最小计算单元 p, q, r 及 LDM 空间 M)固定时，从核拆分策略(k_1, k_2, k_3)在 32 从核下仅有 21 种组合可能。循环分块策略(l_1, l_2, l_3)受 LDM 空间约束取值有限，通过穷举所有排列组合，计算对应推理访存量并选取最小值，即可获得 3 级并行架构下以访存最优为目标的分块策略。如式(7)所示，该优化问题即为在多重硬件限制中的全局最优解。

$$\arg\min\frac{mnd}{64}\left(\frac{1}{l_1}+\frac{1}{l_3}\right)+\gamma\frac{mn}{2k_1k_3}(k_2-1), \quad (7)$$

$$\text{s.t. } k_1,k_2,k_3\in\mathbb{N}_+,\ k_1k_2k_3=32。$$

式(6)和式(7)中γ表示从核间数据同步速率与本地 HBM 访存带宽的比值，与具体的访存行为相关，包括数据量、数据跨步情况、同步核数、参与同步的从核的物理位置等。γ一般指 DMA 广播行为与 RMA 行为速率的比值。

在上述 LDM 优先考量和最小化访存量之后，需进一步实现带宽优化与脉动阵列计算流的深度绑定。本文基于 LDM 私有化设计，通过 DMA 广播和流水操作实现硬件资源最大化利用。如图 6(b)所示，采用 DMA 广播代替 GPU 的 2 级缓存的透明机制，实现核间数据复用；图 6(c)通过双缓冲流水中的 RMA 广播同步数据，取代 GPU 共享内存操作。

为实现高效通信和复用相同子矩阵，采用行多播和列多播模式 DMA 广播操作。该操作需结合硬件层面上的数据传输逻辑，合理设计广播行为。例如，在 4 行 8 列的广播策略下，如图 6(b)所示，17 号从核的 $\boldsymbol{X}_2$ 数据由 16 号从核行广播提供，$\boldsymbol{W}_1$ 由 9 号从核列广播提供，以实现数据复用并减少单从核模式导致的重复访存。硬件层面通过通道单向传输约束和动态起始点选择，例如交错选择列广播起始点，进一步优化数据传输效率。

为实现高效的计算访存流水，使用双缓冲机制协调独立计算与同步需求。如图 6(c)所示，在非同步场景下，采用非阻塞 DMA 实现计算与预取的并行化：首个访存结束后立即启动计算，同时预取下一任务数据。在同步场景下，部分从核需通过 RMA 完成累加同步：非同步核保持独立流程，仅将存回操作替换为 RMA 传输；同步核需等待关联从核 RMA 完成后方可执行累加。每个等待队列元素的空间占用通常为最小计算单元的大小。通过时间重叠与操作解耦，该机制实现了计算的连续性并降低了同步开销。

在上述分析的基础上，算法 1 给出了完整的基于 LDM 优先的启发式分块并行过程。算法 1 的行①~③根据从核拆分因子将矩阵初步拆分，行④~⑰为内部循环分块过程，其中通过 2 个缓冲区交替预取和 DMA 异步传输来实现双缓冲流程；通过脉动阵列计算每个小块；根据不同情况通过 RMA 同步结果或是直接异步 DMA 返回结果。

算法 1. LDM 优先的启发式分块并行策略。

输入：矩阵 $\boldsymbol{X}$ 维度 [m, d]，矩阵 $\boldsymbol{W}$ 维度 [d, n]，从核拆分因子 k_1, k_2, k_3，循环分块因子 l_1, l_2, l_3；

输出：矩阵 $\boldsymbol{Y}$ 维度 [m, n]。

术语定义：下标 part 指矩阵 2 级分块状态，block 指 3 级分块状态；ldm 标识 LDM 空间数据；next 表征双缓冲机制；***Queue*** 为任务等待队列。

① for 对每个核组并行执行 do
② for 对组内每个从核并行执行 do
③ 将输入矩阵拆分为局部块 $\boldsymbol{X}_{\text{part}}[m/k_1][d/k_2]$，$\boldsymbol{W}_{\text{part}}[d/k_2][n/4k_3]$；
④ for 网格划分范围 ($m/(k_1\times l_1\times p)$, $n/(k_3\times l_3\times q)$) 内遍历 $i\times j$ do
⑤ 初始化 $\boldsymbol{Y}_{\text{ldm}}$ 为全 0；
⑥ DMA 异步传输($\boldsymbol{X}_{\text{block}}$, $\boldsymbol{W}_{\text{block}}\rightarrow\boldsymbol{X}_{\text{ldm}}$, $\boldsymbol{W}_{\text{ldm}}$)；
⑦ for 对维度方向分块($d/(k_2\times l_2\times r)$)遍历 k do
⑧ 等待 DMA 传输完成；
⑨ 异步预取下一块($\boldsymbol{X}_{\text{next_block}}$, $\boldsymbol{W}_{\text{next_block}}\rightarrow\boldsymbol{X}_{\text{next_ldm}}$, $\boldsymbol{W}_{\text{next_ldm}}$)；
⑩ 矩阵乘计算($\boldsymbol{X}_{\text{ldm}}$, $\boldsymbol{W}_{\text{ldm}}\rightarrow\boldsymbol{Y}_{\text{ldm}}$)；
⑪ if 该从核进行同步 do
⑫ RMA 规约($\boldsymbol{Y}_{\text{ldm}}$, $\boldsymbol{Queue}_{\text{sync}}\rightarrow\boldsymbol{Y}_{\text{block}}$)；
⑬ else
⑭ DMA 异步传输($\boldsymbol{Y}_{\text{ldm}}$, $\boldsymbol{Y}_{\text{block}}$)；
⑮ end if
⑯ 交换地址($\boldsymbol{X}_{\text{ldm}}$, $\boldsymbol{W}_{\text{ldm}}\leftrightarrow\boldsymbol{X}_{\text{next_ldm}}$, $\boldsymbol{W}_{\text{next_ldm}}$)；
⑰ end for
⑱ end for
⑲ end for
⑳ end for

3.4 体系结构紧耦合的注意力计算方法

3.3 节提出的启发式策略通过将大规模矩阵运算拆分到从核上进行分块并行运算，有效缓解了计算压力，并在 Transformer 流程中展现出一定的适用性，但访存受限问题仍是制约性能提升的关键因素。为此，本节提出一种与申威架构紧耦合的注意力机制 SWAI-Attention，该机制严格遵循两大核心原则，对 Transformer 推理过程进行了深度适配与优化：一是减少访存通信，通过优化数据存储和访问模式降低访存开销；二是适配硬件架构，通过指令优化和并行化策略，充分发挥申威架构的计算能力。通过这种软硬件协同优化的方式，进一步解决访存瓶颈问题。

SWAI-Attention 借鉴了 Flash-Attention 的设计思路，以多头注意力机制算子为基础。通过算子融合技术，在并行策略的驱动下将中间操作符合并至单个从核内。这一过程通过引入少量线性存储开销，避免了中间结果的反复读写，直接计算最终结果，从而显著减少了冗余的访存开销。

在此基础上，进一步结合申威硬件特性，针对 Attention 算子中的超越函数 exp 展开优化。exp 函数涉及复杂的指数运算，不适合单指令多数据（single instruction multiple data, SIMD）处理的除法运算，导致计算和访存开销较大。本节通过并行分段线性查表法、并行指令硬件适配和寄存器汇编控制 3 个步骤优化非线性函数，有效解决了非线性函数在计算和访存上的高开销问题。具体优化措施如下：

1）并行分段线性查表法。采用并行分段线性查表法，通过 2 张表分别记录线段的斜率和截距以及通过乘加运算拟合非线性函数，如图 7 的步骤 1 优化 HBM 访存开销，显著减少了对 HBM 的读写操作，降低了访存开销。

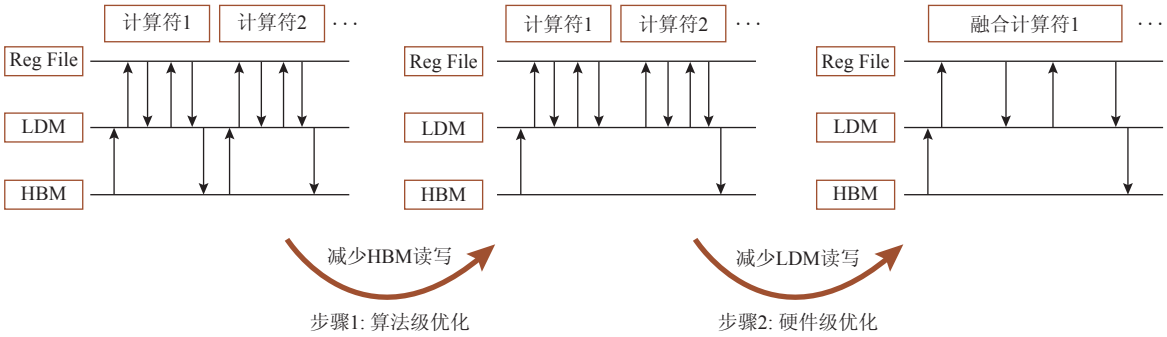


Fig. 7 Optimization logic for nonlinear functions

图 7 非线性函数优化逻辑

2）并行指令硬件适配。利用申威架构的并行查表汇编指令，结合 SIMD 特性，实现单条指令同时处理多个数据的计算合并。这一优化贯穿于图 7 的步骤 1 和步骤 2，显著提升了计算效率。

3）寄存器汇编控制。通过多寄存器控制数据流向，避免批处理查表操作中的冗余表加载和寄存器读取，如图 7 所示，步骤 2 优化 LDM 访存开销，确保数据不相互依赖并实现寄存器间的并行计算。

针对申威架构中 online Softmax 算子难以利用 SIMD 并行化的难题，提出了一种整体转置算法，有效解决了访存效率低下的问题，并显著提升了计算性能。

在传统实现中，计算矩阵每一行的最大值时，理想的并行化方案是利用 SIMD 一次性装载 n 行的某一列元素，然后逐列遍历 n 行所有列的元素，从而实现 n 行的 rowmax 并行计算。然而，如图 8(a)所示，矩阵在 HBM 中默认按行主序存储，这与按列装载的要求存在冲突，导致 rowmax 算子难以高效利用 SIMD 并行化。此外，主存 DMA 不支持转置取数，进一步限制了矩阵原地快速转置的实现。

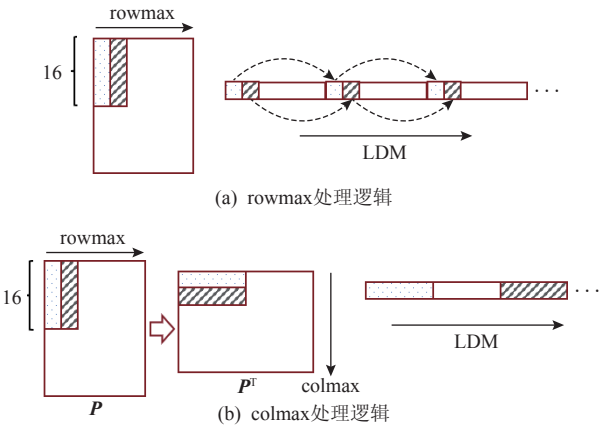


Fig. 8 Transforming rowmax operators to colmax in the transposed MHA algorithm

图 8 MHA 转置算法中算子形态变化

为解决这一问题,本节提出了一种整体转置算法。如图 8(b)所示,该算法在不改变注意力模块核心计算逻辑的前提下,仅将中间结果以转置形式呈现,并通过 DMA put 的转置存储功能将最终输出存回 HBM。通过这种方式,将 rowmax 算子转换为 colmax 算子,使得行向量在 LDM 中连续存储,从而满足硬件的装载要求,并能够高效利用 SIMD 进行并行计算。

4 实验评估

本节对 SW-Inf 的关键技术和整体优化效果进行实验验证与分析。测试硬件环境为 1 台型号 NF5468M5 的浪潮英信服务器,其 PCIe 插槽上分别安装一块 Tesla-V100S 32 GB 加速卡,NVIDIA -A100 40 GB PCIe 加速卡和 1 块申威智能加速卡,以确保主机侧环境一致性。Tesla-V100S 和 NVIDIA -A100 的 CUDA 版本号为 12.2,申威智能加速卡的 SDAA 版本号为 1.1。

4.1 SW-Inf 端到端推理性能测试

本节对 SW-Inf 推理系统的性能进行整体评估,以 LLaMA2-7B Instruct Inference 为评估对象,并在 V100S, A100 和申威加速卡上对 4 个推理系统进行对比测试,包括一个标准推理系统(面向 GPU 的 Meta-LLaMA(git commit: b8348da))、一个申威加速卡上直接移植 cuda 生态后的推理系统(Vanilla),以及 2 个高度优化的先进推理系统:面向 GPU 的 VLLM(版本为 0.6.1)和本文提出的 SW-Inf。这种对比设置能够更直观地反映在一定硬件性能差距的情况下不同推理系统之间的性能和效率差异。

实验的参数设置如下:batch size 为 1,上下文长度分别设置为 50→500, 500→1 000, 1 000→2 000, 其中 $m \rightarrow n$ 表示提示词长度为 m,提示词与完整回答总长度为 n, $m<n$。主要选取吞吐率 *TPS*(tokens per second)和主存带宽利用率(memory bandwidth usage, *MBU*)作为测试指标。*MBU* 定义如下:

$$MBU = \left(\frac{(\text{模型参数大小} + \text{KV Cache})/TPOT}{\text{峰值访问带宽}} \right), \quad (8)$$

其中 *TPOT*(time per output token)是单个 token 的生成时间。*TPS* 表示系统除首词外在单位时间内输出的词元数目,用于衡量推理系统基于特定硬件平台的实际性能,而 *MBU* 用于衡量推理系统能否充分利用硬件主存带宽的能力。

由图 9 和图 10 所示,SW-Inf 在性能上展现出显著优势。相比于直接移植的 Vanilla 系统,SW-Inf 的 *TPS* 提升 1.83 倍, *MBU* 都提升 35% 以上;与在 V100S, A100 上运行的典型推理系统 Meta-LLaMA 相比,SW-Inf 的 *TPS* 提升 2 倍, *MBU* 都提升 35% 以上;相比于 V100S 上最先进的推理系统 VLLM,在 3 种不同的上下文长度下,SW-Inf 的 *TPS* 提升了 1.12 倍, *MBU* 都提升 15% 以上。此外,尽管 SW-Inf 的 *TPS* 受限于主存访问带宽不如运行在 A100 上的 VLLM。但在 3 种不同的上下文长度下, *MBU* 都提升 9% 以上。

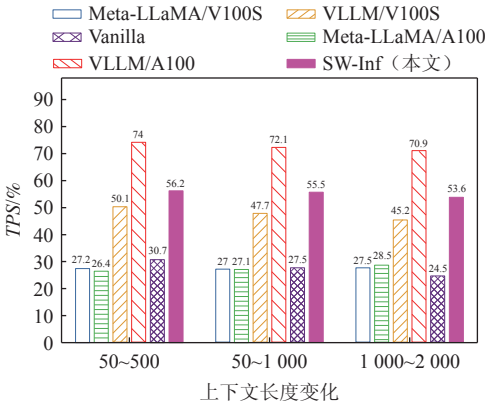


Fig. 9 End-to-end inference throughput in different scenarios

图 9 不同场景下的端到端推理吞吐量

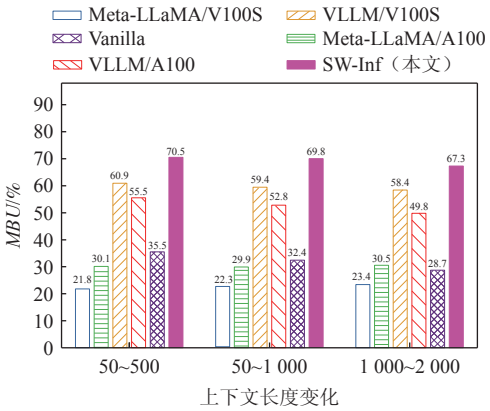


Fig. 10 Memory bandwidth usage with end-to-end inference in different scenarios

图 10 不同场景下的端到端推理主存带宽利用率

实验结果表明,尽管申威加速卡在硬件性能上与 V100S 和 A100 存在差距,尤其是在主存带宽方面,SW-Inf 通过综合应用 3 种针对性优化策略,成功弥补了硬件性能的不足,基于申威智能加速卡实现了较高的推理吞吐率,在某些场景下的推理性能甚至超过了基于 V100S 的先进推理系统。这一结果不仅证明了 SW-Inf 在资源受限条件下的高效性,还为国产硬件在大模型推理领域的应用提供了有力支持,展示了其在实际部署中的巨大潜力。

4.2 有效性验证

4.2.1 整图下沉机制效果验证

为验证整图下沉调度机制对申威智能加速卡架构推理性能的优化效果，本节以主机端与设备端的交互次数和交互总耗时（具体涉及 PCIe 传输）为核心指标，设计了对照实验。整图下沉前的对照组采用传统的实时交互模式，控制流程由主机端主导；下沉后的实验组则采用整图下沉调度，通过计算图打包与片上主核调度优化流程。

由表 2 可见，整图下沉优化方案带来了显著增益：1）交互次数显著降低，单次词生成的交互次数从平均千次级降至固定的 2 次，包括 1 次预处理和 1 次词元解码；2）总耗时大幅减少，下沉后总耗时较传统模式减少 1~2 个数量级，当词数为 500~1 500 时，耗时从 11.75 s 降至 84.44 ms，效率提升了 139 倍。

Table 2 Comparison of Kernel Launch Overhead Between Meta-LLaMA and SW-Inf

表 2 Meta-LLaMA 和 SW-Inf 的核函数启动开销对比

词数量	Meta-LLaMA		SW-Inf	
	交互次数	交互耗时/s	交互次数	交互耗时/ms
50~500	567 280	5.181	900	30.870
500~1 000	630 208	5.346	1 000	54.345
500~1 500	260 208	11.750	2 000	84.441

实验表明，整图下沉调度通过计算图预编译与片上主核自主调度，将动态控制逻辑从主机端迁移至设备端，有效减少跨端交互频率。同时，利用申威架构的硬件并行性掩盖数据搬运延迟，为大规模语言模型推理提供了低延迟保障。

4.2.2 多级并行策略效果评估

本节以矩阵乘运算为例评估多级并行计算策略的有效性，主要验证最小化 LDM 访存量的启发式策略能否搜索出高效并行方式。测试矩阵规模（m, d, n）为（2 048, 4 096, 4 096）或（4 096, 4 096, 8 192）两种典型维度的矩阵乘法，最小计算单元（p, r, q）设置为（128, 32, 32），可用的本地高速缓冲空间 M=200 KB，并统一使用 DMA/RMA 广播机制优化数据传输与同步来对比不同并行分块方式的访存量和实际耗时。

如表 3 所示，在同一矩阵形状下，拆分和分块后的访存量越小，实际的耗时往往也越小。在相近的总访存量时，涉及需要 RMA 同步的流程往往对应的平均带宽要比核组间独立计算的流程要低，实际在处理 RMA 同步问题时，往往同步会在一定程度上影响流水操作。最后表明在实际场景，尽管不同传输模式有速率差异比，但总体还是以访存量为推理流程性能的主要因素，在多级并行策略下，往往选取最小化访存量的情况为实际最优结果。

Table 3 Memory Traffic and Execution Time of Matrix Multiplication Under Different Parallelization Strategies

表 3 不同并行策略下的访存量和实际耗时

矩阵规模	从核拆分策略	循环分块策略	访存量/KB	实际耗时/μs
m=2 048 d=4 096 n=4 096	4，2，4	2，2，4	6 656	2 213.74
	8，1，4	2，2，4	6 144	1 997.44
	4，2，4	2，1，8	4 352	1 580.41
	8，1，4	2，1，8	4 096	1 232.98
m=4 096 d=4 096 n=8 192	2，8，2	4，1，2	37 888	13 166.83
	4，1，8	4，1，2	36 864	10 732.33
	2，8，2	2，1，8	17 408	5 961.80
	4，1，8	2，1，8	16 384	4 862.31

4.2.3 SWAI Attention 性能测试

本节实验讨论注意力机制在与申威体系结构紧耦合前后的性能变化，重点对非线性函数优化和整体转置优化进行了消融实验，并最终进行了整体性能测试。

1）非线性函数优化实验

本节验证了非线性函数优化对算子性能的提升效果，主要测试了 softmax 算子中的指数函数 exp。表 4 给出了在单核组相同输入规模下 3 种 exp 实现方法的性能对比：MathLib 基于 C 语言标准数学库的标量计算实现；TBLExp 采用并行分段线性查表法，并结合并行查表指令，以适配硬件加速计算；FusedExp 在 TBLExp 基础上，进一步结合寄存器汇编控制，减少寄存器交互，合并指令层的计算，从而提升整体效率。

Table 4 Performance Improvement of the Fused exp Operator

表 4 exp 融合算子性能提升

向量维度	单次运行平均耗时/μs		
	MathLib	TBLExp	FusedExp
1 024	885.49	1.14	0.21
2 048	1 783.62	2.27	0.42
4 096	3 567.24	4.52	0.83
6 144	5 350.85	6.77	1.23
8 192	6 364.42	9.02	1.64

由表 4 可见，在多种输入规模下，MathLib 的计算耗时显著高于其他 2 种方法。这是由于 C 语言标准数学库中的函数实现主要是为了保证通用性，未

针对硬件特性进行优化。相比之下, TBLExp 通过分段线性拟合和硬件并行查表指令, 大幅减少了寄存器和 LDM 之间的数据传输。而 FusedExp 进一步优化了寄存器分配和指令流水, 根据单从核计算任务指定所需寄存器, 从而更有效地实现指令流水化和平均性能提升, 且在某些场景下表现更优。

2)整体转置实验

本节验证了整体转置算法在注意力机制中对 max 相关计算的性能提升效果。由于分块策略的约束, 矩阵 $\boldsymbol{S}$ 形状由矩阵 $\boldsymbol{Q}$ 和 $\boldsymbol{K}$ 的分块决定, 例如, $\boldsymbol{Q}$ 分块为 (192, 128), $\boldsymbol{K}$ 的转置为 (128, 32), 此时 $\boldsymbol{S}$ 为 (192, 32), 结果向量 $\boldsymbol{M}$ 为 (192, 1), 如表 5 所示, 实验覆盖了 3 种典型分块对应的形状, 并测试了 max 步骤的完整流程, 包括读取、计算和存回。

Table 5 Performance Improvement of the Transposed MHA Algorithm

表 5 MHA 转置算法性能提升

$\boldsymbol{S}\rightarrow\boldsymbol{M}$	单次运行平均耗时/μs	
	rowmax	colmax
(32, 32)→(32, 1)	24.86	0.20
(32, 128)→(32, 1)	32.91	0.50
(192, 32)→(192, 1)	146.51	0.85

SIMD 向量化指令实现了多元素的并行计算, 显著提升了计算效率。如表 5 所示, 在不同矩阵形状下, colmax 相比于 rowmax 的加速比可达 124~172。特别的, 在矩阵形状为 (32, 32) 时, 计算耗时从 24.86 μs 显著降低至 0.20 μs。此外, 随着矩阵列数的增加, 加速比进一步提升, 充分体现了 SIMD 向量化规模的优势。实验结果表明, 整体转置算法通过将 rowmax 转换为 colmax, 显著适配了申威架构的 SIMD 硬件特性, 为注意力机制的分块策略提供了坚实的硬件环境。

3)MHA 算子整体实验

本节实验将 SWAI Attention 与文献 [12] 中的 FlashAttention 在 Tesla-A100 40 GB 下的多种主流注意力机制, 包括 PyTorch Attention, Megatron, Local Attention, Longformer Attention, FlashAttention 进行性能对比, 并且这些主流注意力机制数据与原论文一致。旨在衡量在计算能力相当的情况下, 访存带宽存在一定差距时, SWAI Attention 在缓解访存瓶颈方面的效果。评估对象为注意力模块的单次正向计算。实验参数设置如下: 批样本数为 16, 注意力头数目为 8, 注意力头维度为 64, 且不进行 mask 和 dropout 处理。

如表 6 所示, 尽管申威智能加速卡在访存带宽上相对 A100 存在较大差距, 导致其在长句长任务中的性能表现劣于对比方案。但在短句长场景下, SWAI Attention 的计算耗时显著低于其他方法。特别是在句长为 256 时, SWAI Attention 在 SW-AI 架构上的耗时相较于 Tesla-A100 上的先进方法 FlashAttention 减少了 50%, 充分证明了体系结构紧耦合的注意力计算方法的有效性。

Table 6 Performance Comparison of Different Attention Calculation Methods

表 6 不同注意力计算方法的性能对比

Attention 方法	硬件环境	耗时/ms	
		seq_len = 256	*seq_len* = 4 096
PyTorch Attention	Tesla-A100 40 GB	0.22	16.47
Megatron		0.26	-
Local Attention		0.57	11.56
Longformer Attention		0.98	11.07
FlashAttention		0.09	8.42
SWAI Attention	SW-AI	0.045	21.37

注: 在 Tesla-A100 40 GB 下的多种 Attention 方法的数据与文献 [12] 一致, 所以有一处为空。*seg_len* 为句长。

4.3 消融实验

本节主要开展消融实验, 探究本文所提的 3 个优化方法对总体推理性能给出了不同情况下的端到端推理 *TPS* 和相对于基准方法的性能提升比。其中“原始”场景与 4.1 节的 Vanilla 意义相同, 表示申威平台上未经优化的推理系统, “+1”表示在 Vanilla 基础上进行整图下沉优化, “+2”表示在 Vanilla 基础上使用 LDM 优先的多级并行优化, “+3”表示在 Vanilla 基础上进行 SWAI-Attention 优化, 按此记号记录不同组合下的 *TPS* 和性能提升比。

由图 11 可见, 3 种优化方法在单独引入时可带来 60%~80% 的性能提升, 其中, 多级并行策略在减少总体访存量的同时实现负载均衡, 充分利用所有

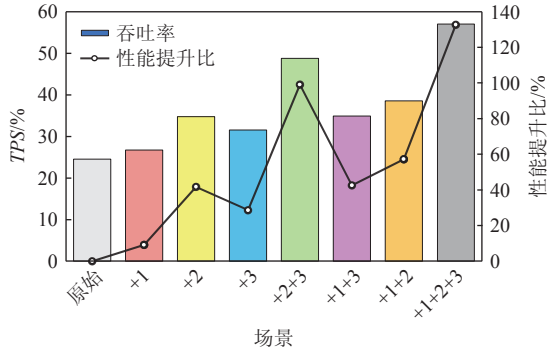


Fig. 11 Ablation study in end-to-end inference

图 11 端到端推理消融实验

从核的计算能力，构成性能提升的核心基础；整图下沉与注意力优化通过降低数据搬运开销和计算冗余度进一步提高性能，最终从存储、计算、通信 3 个层面进行协同优化，总体可带来超过 120% 的端到端推理 *TPS* 提升。

4.4 不同模型和参数维度下的适应性分析

本节验证所提方法在不同模型和参数规模的情况下的适应性，针对 Transformer 模型的核心模块 MHA 和 FFN，测试 LLaMA 和 Qwen 这 2 类模型，共 4 种参数规模下 MHA 和 FFN 解码阶段的推理耗时。

如表 7 所示，模型参数规模（隐藏层维度从 2 048 扩展至 8 192）与多头注意力机制的结构复杂度（注意力头数从 16 增至 64）的同步提升，会导致 FFN 和 Attention 的计算时延显著增长，其运行耗时分别增长 12.6 倍与 13.3 倍，可见 SW-Inf 推理耗时与 Transformer 规模成近线性关系，随着模型规模增大，各核心模块的算子规模增大，如 GEMM 等关键算子的计算访存比提升及数据复用度增大，如 2.3 节分析，由于申威智能加速卡在端侧推理场景下存在计算能力溢出的现象，因此模型规模的增长不会导致实际推理耗时等比例增长。

Table 7 Time Consumption of MHA and FFN Under Different Models and Parameter Dimensions

表 7 不同模型和参数维度下 MHA 和 FFN 的耗时

模型	隐藏层维度	注意力头数	耗时/ms	
			MHA	FFN
Qwen-1.8B	2 048	16	0.34	3.06
LLaMA-7B	4 096	32	1.01	10.34
LLaMA-13B	5 120	40	1.66	15.85
Qwen-72B	8 192	64	4.29	40.81

5 结 论

申威智能加速卡的众核架构、主从核特性以及丰富的片上计算资源和存储层次，为端侧大模型智能应用的性能优化提供了坚实基础，同时也对推理系统设计提出了新的挑战。本文提出了一种基于申威智能加速卡的 Transformer 高效推理方法 SW-Inf，通过整图下沉调度机制减少主机与设备间的频繁交互开销，设计多级并行计算策略充分利用片上所有计算资源以及利用深度算子融合技术缓解推理过程中的访存瓶颈问题。实验结果表明，与英伟达 V100S 上的先进推理系统 vLLM 相比，SW-Inf 在推理吞吐率和主存利用率方面均取得了显著提升。这一成果不仅验证了本文提出的 3 种针对性优化策略的有效性，更凸显出国产硬件在边缘侧大模型推理部署中的广阔前景和巨大潜力。本文工作主要面向单卡端侧推理场景，在未来的工作中，将进一步研究面向推理服务系统和模型训练场景的软硬协同优化方法与技术，以充分发挥申威加速卡的潜力，推动其在更广泛场景中的应用。

作者贡献声明：沈文渊参与理论分析、优化全过程和整体代码编写以及撰写论文；汤凌韬负责研究构思设计与理论分析，完成跨平台性能测试和论文撰写；何昊天、费天成撰写论文、制作图表和修订关键段落；赵玉龙、张鲁飞指导研究框架和协调实验设备；秦晓军、刘鑫和陈左宁负责质量控制、数据复核和方法论改进。

参 考 文 献

[1] Yu G, Jeong S, Kim G, et al. Orca: A distributed serving system for Transformer-based generative models[C]//Proc of the 16th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2022: 521−538

[2] Frantar E, Alistarh D. SparseGPT: Massive language models can be accurately pruned in one-shot[J]. arXiv preprint, arXiv: 2301.00774, 2023

[3] Liu Zichang, Wang Jue, Dao T, et al. Deja Vu: Contextual sparsity for efficient LLMs at inference time[C]//Proc of the 40th Int Conf on Machine Learning. Cambridge, MA: PMLR, 2023: 919

[4] Wang Xiaochen, Zhang Kun, Zhang Peng. Multi-perspective view of large model security and practice[J]. Journal of Computer Research and Development, 2024, 61(5): 1104−1112 (in Chinese)
(王笑尘, 张坤, 张鹏. 多视角看大模型安全及实践[J]. 计算机研究与发展, 2024, 61(5): 1104−1112)

[5] Nallapati R, Zhou B, dos Santos C, et al. Abstractive text summarization using sequence-to-sequence RNNs and beyond[C]// Proc of the 20th SIGNLL Conf on Computational Natural Language Learning. Stroudsburg, PA: ACL, 2016: 280−290

[6] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint, arXiv: 1409.0473, 2014

[7] Adiwardana D, Luong M T, So D R, et al. Towards a human-like open-domain chatbot[J]. arXiv preprint, arXiv: 2001.09977, 2020

[8] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30: 5998−6008

[9] Kim S, Hooper C, Wattanawong T, et al. Full stack optimization of Transformer inference: A survey[J]. arXiv preprint, arXiv: 2302.

14017, 2023

[10] Samuel W, Andrew W, David P. Roofline: An insightful visual performance model for multicore architectures[J]. Communications of the ACM, 2009, 52(4): 65−76

[11] Zheng Lianmin, Yin Liangsheng, Xie Zhiqiang, et al. SGLang: Efficient execution of structured language model programs[J]. Advances in Neural Information Processing Systems, 2024, 37: 62557−62583

[12] Dao T, Fu D, Ermon S, et al. FlashAttention: Fast and memory-efficient exact attention with IO-awareness[J]. Advances in Neural Information Processing Systems, 2022, 35: 16344−16359

[13] DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model[J]. arXiv preprint, arxiv: 2405.04434, 2024

[14] Kwon W, Li Zhuohan, Zhuang Siyuan, et al. Efficient memory management for large language model serving with PagedAttention[C]//Pro of the 29th Symp on Operating Systems Principles. New York: ACM, 2023: 611–626

[15] Agrawal A, Kedia N, Panwar A, et al. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve[C]//Proc of the 18th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2024: 117−134.

[16] Sheng Ying, Zheng Lianmin, Yuan Binhang, et al. FlexGen: High-throughput generative inference of large language models with a single GPU[C]//Proc of the Int Conf on Machine Learning. Cambridge, MA: PMLR, 2023: 31094−31116

[17] Song Yixin, Mi Zeyu, Xie Haotong, et al. PowerInfer: Fast large language model serving with a consumer-grade GPU[C]//Proc of the 30th ACM SIGOPS Symp on Operating Systems Principles. New York: ACM, 2024: 590−606

[18] Michel P, Levy O, Neubig G. Are sixteen heads really better than one?[J]. Advances in Neural Information Processing Systems, 2019, 32: 14014−14024

[19] Bansal H, Gopalakrishnan K, Dingliwal S, et al. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale[J]. arXiv preprint, arXiv: 2212.09095, 2022

[20] Alizadeh K, Mirzadeh I, Belenko D, et al. LLM in a flash: Efficient large language model inference with limited memory[C]//Proc of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg, PA: ACL, 2024: 12562−12584

[21] GGML-org. LLaMA. cpp[EB/OL]. [2024-10-01]. https://github.com/ggml-org/llama.cpp

[22] Jiang Xiaotang, Wang Huan, Chen Yiliu, et al. MNN: A universal and efficient inference engine[C]//Proc of Machine Learning and Systems. Miami, FL: MLSys Foundation, 2020: 1−3

[23] Wu Ruohan, Zhu Xianyu, Chen Junshi, et al. SWattention: Designing fast and memory-efficient attention for a new Sunway supercomputer[J]. The Journal of Supercomputing, 2024, 80(10): 13657−13680

[24] Liu Kunyang. Optimization and application of YOLOv5 model based on Cambrian MLU270[D]. Chengdu: Sichuan Normal University, 2023 (in Chinese).
(刘锟阳. 基于寒武纪 MLU270 的 YOLOv5 模型优化与应用研究[D]. 成都: 四川师范大学, 2023)

[25] Ma Zixuan, He Jiaao, Qiu Jiezhong, et al. BaGuaLu: Targeting brain scale pretrained models with over 37 million cores[C]//Proc of the 27th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2022: 192−204

[26] Shah J, Bikshandi G, Zhang Ying, et al. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision[J]. Advances in Neural Information Processing Systems, 2024, 37: 68658−68685

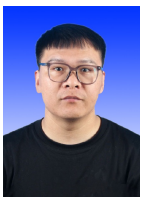
[27] Zhao Yulong, Zhang Lufei, Xu Guochun, et al. SDAA: Runtime system for Sunway intelligent acceleration card[J]. Journal of Software, 2024, 35(12): 5710−5724 (in Chinese)
(赵玉龙, 张鲁飞, 许国春, 等. SDAA: 面向申威智能加速卡的运行时系统[J]. 软件学报, 2024, 35(12): 5710−5724)



Shen Wenyuan, born in 2000. Undergraduate. His main research interests include high performance computing and operating systems.
沈文渊, 2000 年生。本科生。主要研究方向为高性能计算、操作系统。



Tang Lingtao, born in 1994. PhD. His main research interests include machine learning, privacy protection, and intelligent operating systems.
汤凌韬, 1994 年生。博士。主要研究方向为机器学习、隐私保护、智能操作系统。



He Haotian, born in 1997. Master. His main research interests include high performance computing and operating systems.
何昊天, 1997 年生。硕士。主要研究方向为高性能计算、操作系统。



Fei Tiancheng, born in 1999. Master. His main research interests include high performance computing and operating systems.
费天成, 1999 年生。硕士。主要研究方向为高性能计算、操作系统。



Zhao Yulong, born in 1984. PhD candidate. His main research interests include artificial intelligence and operating system.
赵玉龙, 1984 年生。博士研究生。主要研究方向为人工智能、操作系统。



Zhang Lufei, born in 1986. PhD, senior engineer. His main research interests include high-performance computing, operating system, and machine learning.

张鲁飞, 1986 年生。博士，高级工程师。主要研究方向为高性能计算、操作系统、机器学习。



Qin Xiaojun, born in 1975. PhD, senior engineer. His main research interests include system security, software vulnerability analysis, and artificial intelligence.

秦晓军, 1975 年生。博士。高级工程师。主要研究方向为系统安全、软件脆弱性分析、人工智能。



Liu Xin, born in 1979. PhD, professor. Her main research interest includes parallel algorithms and applications.

刘　鑫, 1979 年生。博士，研究员。主要研究方向为并行算法和应用。



Chen Zuoning, born in 1957. Academician of Chinese Academy of Engineering. Her main research interests include software theory, operating systems, and information security.

陈左宁, 1957 年生。中国工程院院士。主要研究方向为软件理论、操作系统、信息安全。