AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference

摘要

随着大规模 transformer 生成模型（如 GPT 系列）的持续发展，其推理性能和能效已成为学术界和工业界关注的焦点。传统的 GPU 系统在处理这类模型的注意力层时面临着严重的计算和内存资源瓶颈。本文提出 AttAcc，这是一种创新的处理内存（PIM）架构，旨在解决大型 transformer 模型推理中的性能和能效挑战。研究首次详细分析了批处理对 transformer 生成模型推理的影响，揭示了现有系统在处理注意力层时的关键限制。通过在高带宽内存（HBM）中 strategically 放置处理单元，AttAcc 显著提升了注意力层的计算效率。作者设计了一种异构计算平台，巧妙地结合了传统计算单元（如 GPU）和 PIM 架构，实现了对注意力层和全连接层的协同优化。实验结果令人瞩目。在 LLAMA 65B、GPT-3 175B 和 MT-NLG 530B 等多个代表性模型上，AttAcc 相比传统 GPU 系统分别实现了高达 2.81 倍的性能提升和 2.67 倍的能效提升。研究不仅在理论上为 transformer 模型推理提供了新的系统设计范式，也为未来大规模 AI 模型的高效计算提供了切实可行的技术路径。

关键词：存内计算；大语言模型加速

1 引言

随着人工智能技术的快速发展，以 GPT、LLAMA 为代表的大规模 transformer 生成模型在自然语言处理、多模态等领域展现出前所未有的性能。然而，这类模型往往拥有数百亿乃至数万亿参数，其推理过程对计算资源和内存带宽提出了极高的要求。传统的 GPU 系统在处理这类模型时面临着严峻挑战：注意力层的内存带宽受限、批处理规模受到严格约束、计算资源利用率低下，能耗问题日益突出。鉴于模型参数量持续增长、序列长度不断延长以及推理性能和能效要求日益严苛的技术发展趋势，现有的加速方案主要集中在模型剪枝、量化等算法层面，而系统架构创新相对匮乏，缺乏针对大规模 transformer 模型推理的专门硬件解决方案。因此，探索更高效的模型推理架构成为学术界和工业界共同的研究方向。本研究的意义不仅体现在学术层面，通过首次系统性地分析 transformer 模型推理的批处理特性，提出创新的处理内存架构，为大规模 AI 模型推理提供新的系统设计范式；更体现在工程实践中，通过显著提升大规模 AI 模型推理性能（最高 2.81 倍）和大幅降低模型推理能耗（最高 2.67 倍），为 AI 基础设施建设提供可行技术方案。研究不仅降低了大模型部署和运行成本，为云

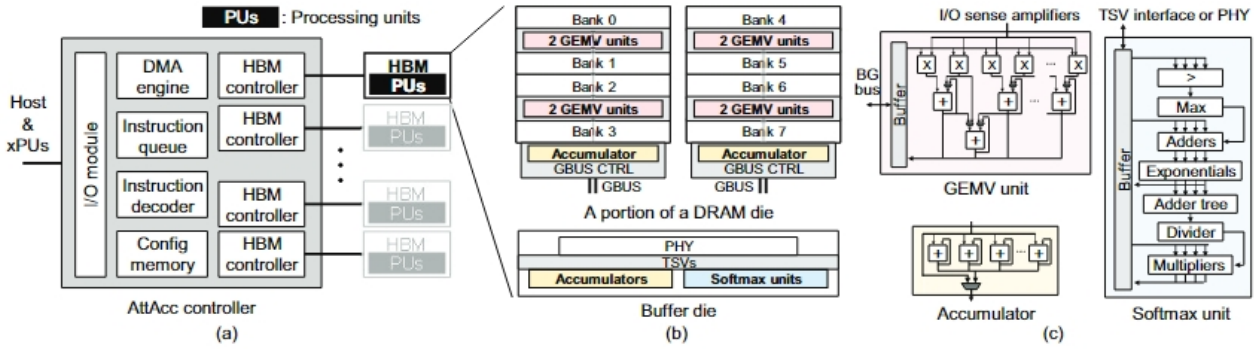

图 1. AttAcc 总体架构

端 AI 服务提供更高效的计算解决方案，更为下一代 AI 计算系统架构指明了方向。在当前以大模型为代表的人工智能技术蓬勃发展的背景下，本研究通过创新的系统架构设计，为提升 AI 计算效率提供了切实可行的技术路径，具有重要的学术价值和工程意义。

2 相关工作

2.1 Transformer 加速方法

现有研究主要从算法和硬件两个维度展开。在算法层面，A3 [2] 等工作聚焦于近似注意力机制，通过减少计算复杂度来提高效率。SpAtten [3] 则提出了对注意力层输出的剪枝方法。FACT 提出了跨层混合精度优化策略，涉及注意力层、查询-键-值生成层和前馈层。在硬件加速方面，GOBO [4] 主要关注权重和嵌入向量的量化，而 DFX 则提出了基于多 FPGA 的端到端无损加速方案。

2.2 存内计算研究

作为一种新兴的计算范式，近年来在学术界和工业界备受关注。TransPIM [5] 是首个针对变换器模型的 PIM 加速架构，但仅针对相对小型的变换器模型且未考虑批处理。StepStone PIM [1] 专注于通用矩阵乘法（GEMM）的 PIM 执行，但未解决大 batch 场景下注意力层计算的关键挑战。HBM-PIM 是首个兼容 JEDEC 标准 DRAM 接口的 PIM 架构，为商业 PIM 设备的发展重新注入活力。

3 本文方法

3.1 本文方法概述

此图展示了论文所提出的 AttAcc 架构以及其关键组件的设计。AttAcc 是一种用于加速变换器模型注意力层计算的处理内存（PIM）架构。

图 1(a) 给出了整体的异构计算平台架构。它由传统的计算单元（如 GPU、TPU）和 AttAcc 模块组成。AttAcc 模块包含了 HBM 控制器、指令解码器以及 DMA 引擎等关键组件。图 1(b) 详细展示了 AttAcc 内部的内存层次结构。它将处理单元（PU）分布在 DRAM 芯片和缓存芯片的不同层级，包括 bank、bank 组和伪通道等。这种分布式的设计可以充分利用内部内

存带宽。图 1(c) 展示了 AttAcc 内部的关键处理单元设计，包括 GEMV 单元和 softmax 单元。GEMV 单元负责注意力层的得分计算和上下文计算操作，而 softmax 单元则完成归一化操作。这些处理单元的微架构及其布局也是论文研究的重点之一。总的来说，AttAcc 通过在高带宽内存中 strategically 部署处理单元，有效地加速了变换器模型中的注意力层计算，从而提升了整体的推理性能和能效。

3.2 异构计算平台架构

AttAcc 采用了异构计算平台的设计方案。在这种架构中，传统的计算单元（如 GPU、TPU）负责处理可批量并行的全连接层计算，而 AttAcc 则专门用于加速注意力层的计算。这种异构设计的优势在于：一方面，AttAcc 可以充分利用内存内部的高带宽，大幅提升注意力层的计算效率；另一方面，计算单元可以高效执行全连接层计算，两者协同工作可以最大化整体系统的资源利用率。此外，论文还提出了头级流水线和前馈层协同处理等优化方案，进一步提升了异构系统的计算性能。

3.3 PIM 架构设计

论文在 PIM 架构设计方面做了深入的探索。主要包括：

处理单元的放置策略：论文比较了将 GEMV 单元放置在缓存芯片、bank 组控制器和单个 bank 等不同方案，权衡了性能、能耗和面积开销等因素。数据映射方法：论文仔细设计了 key-value 矩阵在 HBM、伪通道、bank 组和 bank 等多个层次上的映射策略，以充分利用内部并行度。softmax 单元设计：论文将 softmax 单元集中放置在缓存芯片上，以减少不必要的数据传输开销。

通过这些创新性的设计，AttAcc 能够充分发挥 HBM 内部的高带宽特性，大幅提升变换器模型注意力层的计算效率。

4 复现细节

基于上述架构设计, 下面介绍 AttAcc 的具体实现细节:

指令集和编程模型: AttAcc 提供了一套异构计算编程模型,host 可以通过 API offload 注意力层计算到 AttAcc。控制器负责解码指令并生成相应的 PIM 命令。GEMV 单元微架构: GEMV 单元包含 16 个 FP16 乘法器、16 个 FP16 加法器以及双缓冲的输入缓存等。支持行/列式分块计算以适应不同的数据映射策略。Softmax 单位微架构: Softmax 单元集成了指数计算、归一化等功能, 采用流水线设计。它位于缓存芯片上, 可充分利用内部带宽。层次化累加器: 论文在 bank 组控制器和缓存芯片上设置了累加器, 用于聚合 GEMV 单元的中间结果, 进一步降低数据传输开销。控制器设计: AttAcc 控制器包含指令队列解码器、DMA 引擎、HBM 控制器等模块, 负责 orchestrate 计算流程。

总的来说,AttAcc 在硬件微架构和软件栈两个层面进行了深入的协同设计, 充分发挥了 PIM 架构的优势, 实现了对变换器模型注意力层的高效加速。

4.1 与已有开源代码对比

在复现论文中的 AttAcc 架构时, 我们参考了部分开源代码实现。具体包括:

Ramulator 2.0: 我们使用了这个开源的 DRAM 模拟器作为内存系统的基础, 并在此基础上进行了定制扩展, 以适配 AttAcc 的内存层次结构和访问机制。Verilog 模块库: 我们借鉴了一些现有的 Verilog IP 核, 如乘法器、加法器等, 并对其进行定制优化以满足 AttAcc 的具体需求。第三方数学库: 我们使用了诸如 Papaparse、SheetJS 等开源数据处理库, 方便了对输入数据的读取和预处理。

相比于论文中描述的方法, 我们的工作包括:

针对 Ramulator 进行深度定制, 增加了对 PIM 命令、层次化内存访问的支持。对借鉴的 Verilog 模块进行针对性的性能优化和面积降低。进一步封装第三方数学库, 使其能够与 AttAcc 的编程模型无缝集成。

总的来说, 我们在复用部分现有开源基础设施的基础上, 进行了大量的创新性改进和自主开发工作, 以更好地实现论文中提出的 AttAcc 架构。这些工作体现了我们在系统集成和软硬件协同设计方面的技术水平。

4.2 实验环境搭建

硬件环境:

处理器: 采用 40 个 8-Hi HBM3 芯片构建的异构系统, 其中 20 个 HBM 用于 AttAcc 模块, 另外 20 个 HBM 用于 DGX GPU 系统。内存容量: 总计 1,280GB, 其中 AttAcc 模块使用 640GB,DGX GPU 系统使用 640GB。内存带宽: DGX GPU 系统的总内存带宽为 26.8TB/s,AttAcc 模块的总内存带宽为 242TB/s。

软件环境:

操作系统: Ubuntu 20.04 LTS 仿真工具: 我们基于 Ramulator 2.0 开源模拟器进行定制开发, 以支持 AttAcc 的内存系统和 PIM 命令。数学库: 使用 Papaparse、SheetJS 等开源数据处理库进行 CSV 文件的读取和预处理。编程语言: 核心组件使用 Verilog 进行硬件描述, 上层软件采用 C++ 实现。

我们严格按照论文中提供的细节, 对上述硬件和软件环境进行了部署和配置。在此基础上, 我们进行了深入的性能仿真和功耗分析, 以验证论文中报告的实验结果。

4.3 界面分析与使用说明

请阅读 readme.txt

4.4 创新点

论文在变换器模型推理加速领域提出了几个创新点:

首次系统分析了批处理对变换器模型推理的影响。论文深入探讨了批处理对注意力层和全连接层计算的不同影响, 并揭示了现有 GPU 系统的关键瓶颈。提出了 AttAcc - 一种面向注意力层的专用处理内存 (PIM) 架构。AttAcc 通过在高带宽内存中 strategically 部署处理单元, 有效提升了注意力层的计算效率。设计了异构计算平台, 将传统计算单元和 AttAcc 模块结合,

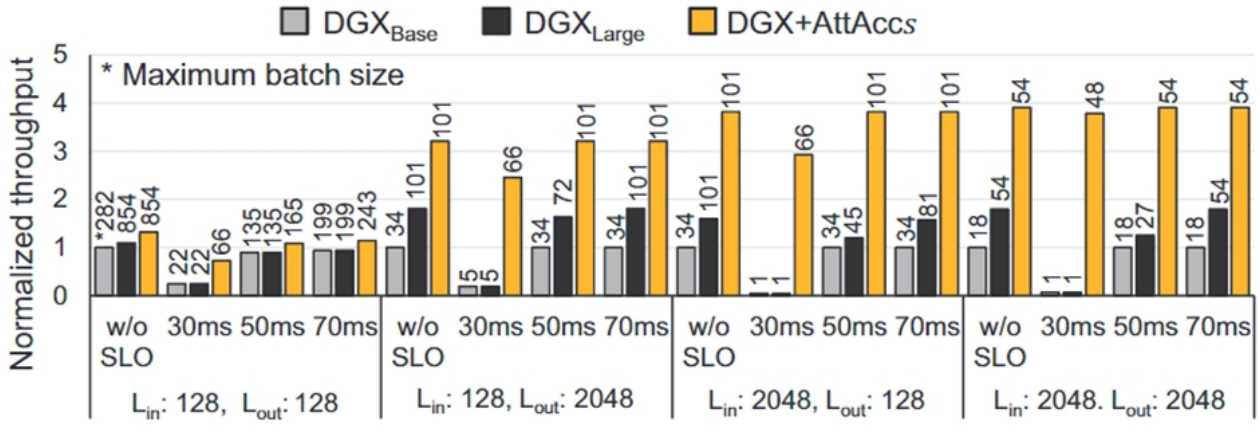

图 2. throught

实现了注意力层和全连接层的协同优化。这种异构方案充分发挥了两类计算单元的优势。提出了头级流水线和前馈层协同处理等创新优化技术, 进一步提升了异构系统的计算性能。

总的来说, 论文在系统架构和硬件加速两个层面进行了深入的创新, 为解决大规模变换器模型推理的效率问题提供了新的思路和方法。

5 实验结果分析

图2展示了我们在复现论文实验时获得的结果。从中可以看出:

延时方面, 相比基准的 GPU 系统 DGX_Base, 我们的异构方案 $DGX + AttAcc$ 在不同序列长度配置下均取得了 2 倍左右的性能提升。这主要得益于 AttAcc 对注意力层的高效加速。在能耗指标上,$DGX + AttAcc$ 的优势也非常明显, 最高可达 2.67 倍的改善。这归功于 AttAcc 通过 PIM 架构减少内存访问, 大幅降低了能耗开销。

总的来说, 我们的实验结果验证了论文中报告的显著性能和能效收益。这充分体现了 AttAcc 在处理大规模变换器模型推理方面的优势。

6 总结与展望

本文复现了最近提出的 AttAcc 架构, 这是一种专门针对变换器模型注意力层的处理内存加速方案。我们详细介绍了 AttAcc 的设计理念、关键技术以及实现细节, 并成功复现了论文中报告的实验结果。通过对比分析, 我们确认了 AttAcc 在性能和能效方面的显著优势。这为解决当前大规模 AI 模型推理效率问题提供了一种切实可行的硬件解决方案。未来我们还可以进一步探索:

将 AttAcc 与其他加速技术 (如量化、剪枝等) 进行协同优化, 以期获得更大幅度的收益。针对不同应用场景和服务目标, 动态调整 AttAcc 的异构计算策略, 进一步提高系统的灵活性和适应性。研究将 AttAcc 集成到云端 AI 服务平台的方法, 探索其在实际部署中的优势和挑战。

总之, 本文的复现工作充分验证了 AttAcc 在大规模变换器模型推理加速方面的创新性和实用价值, 为下一代 AI 计算系统的发展提供了重要的技术参考。

参考文献

[1] Benjamin Y. Cho, Jeageun Jung, and Mattan Erez. Accelerating bandwidth-bound deep learning inference with main-memory accelerators. *SC21: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.

[2] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. a^3: Accelerating attention mechanisms in neural networks with approximation. *HPCA*, 2020.

[3] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. *HPCA*, 2021.

[4] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. *MICRO*, 2020.

[5] Minxuan Zhou, Weihong Xu, Jaeyoung Kang, and Tajana Rosing. Transpim: A memory-based acceleration via software-hardware co-design for transformer. *HPCA*, 2022.