计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(7): 1816−1830, 2026

基于混合键合架构的大模型局部性感知解码加速器

王浩然[1,2,3] 许浩博[1,2] 王 颖[1,2] 韩银和[1,2]

[1]（中国科学院计算技术研究所智能计算机研究中心 北京 100190）

[2]（处理器芯片全国重点实验室（中国科学院计算技术研究所） 北京 100190）

[3]（中国科学院大学 北京 100190）

（wanghaoran20g@ict.ac.cn）

Locality-Aware Decoding Accelerator for Large Language Models Based on Hybrid Bonding Architecture

Wang Haoran[1,2,3], Xu Haobo[1,2], Wang Ying[1,2], and Han Yinhe[1,2]

[1]（*Research Center for Intelligent Computing Systems*, *Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190）

[2]（*State Key Lab of Processors* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190）

[3]（*University of Chinese Academy of Sciences*, *Beijing* 100190）

Abstract Large language models (LLMs) have demonstrated remarkable capabilities across various domains, including natural language processing, computer vision, chat bots and content generation. Deploying LLMs on edge devices has attracted significant attention due to its advantages in service stability and data privacy. However, the decoding inference of LLMs is composed of memory-bound operations, and with model sizes growing and complex tasks demanding increasingly longer context lengths, the resulting memory access overhead becomes critical bottleneck limiting the deployment of LLMs on edge platforms. To lower the memory access demand of long context decoding, locality aware attention mechanism is proposed. By leveraging the numerical locality of attention scores during decoding, this mechanism compresses the key-value cache (KV cache), significantly reducing redundant memory access without compromising model accuracy. We present a locality-aware inference accelerator for LLMs with hybrid-bonded 3D DRAM. The accelerator not only reduces the overall memory traffic but also significantly improves the main memory bandwidth. It integrates load-balancing and prediction-based prefetching mechanisms tailored to the structural characteristics of hybrid-bonded 3D DRAM, thereby enhancing the bandwidth utilization. On the evaluated models, the proposed accelerator achieves on average 1 971 token/s, 3 752 token/s and 6 849 token/s throughputs for batch sizes of 1, 2 and 4, respectively, with corresponding energy efficiencies of 0.23 J/token, 0.12 J/token and 0.07 J/token. The achieved throughput corresponds to 50.5 times, 49.4 times, 50.0 times that of the A100 GPU, and 23.7 times, 22.9 times, 21.5 times that of the LAD-HBM accelerator. Meanwhile, the energy consumption amounts to only 5.1%, 5.1%, 4.9% of GPU, and 46.2%, 46.9%, 48.2% of LAD-HBM.

Key words large language model; auto-regressive decoding; hybrid bonding; accelerator; edge device

摘 要 大模型在自然语言处理、计算机视觉、聊天机器人以及内容生成等多个领域中展现出了优秀的能力。大模型在边缘设备上的部署因其在稳定性和数据安全方面的优势受到广泛关注。然而，大模型的解

收稿日期：2025-04-28；**修回日期**：2026-02-06

基金项目：国家自然科学基金项目（62025404, 62495104, 62222411）；国家重点研发计划项目（2023YFB4404400）

This work was supported by the National Natural Science Foundation of China (62025404, 62495104, 62222411) and the National Key Research and Development Program of China (2023YFB4404400).

通信作者：许浩博（xuhaobo@ict.ac.cn）

码推理过程主要由访存受限的算子构成，随着模型规模的扩展以及复杂任务对上下文长度的需求持续增长，访存开销成为了限制大模型在边缘设备上部署的关键瓶颈。为了降低大模型在长上下文解码时的访存开销，局部性感知的注意力机制被提出。该机制基于注意力分数在解码过程中的数值局部性，对键值缓存（KV cache）进行压缩，实现了在不影响模型精度的情况下，有效削减了冗余的访存。设计了一种基于混合键合 3 维 DRAM 的局部性感知大模型推理加速器，在有效降低总体访存量的同时显著提升了主存带宽。该加速器集成了负载均衡与基于预测的预取机制，来适配混合键合 3 维 DRAM 的结构特性，从而提升带宽利用率。在所评估的模型上，该加速器在批处理大小为 1，2，4 时，平均实现了分别为 1 971 token/s，3 752 token/s，6 849 token/s 的吞吐，对应的能耗分别为 0.23 J/token，0.12 J/token，0.07 J/token。该吞吐相当于 A100 GPU 的 50.5 倍、49.4 倍、50.0 倍，LAD-HBM 加速器的 23.7 倍、22.9 倍、21.5 倍，而能耗仅相当于 GPU 的 5.1%，5.1%，4.9%，LAD-HBM 的 46.2%，46.9%，48.2%。

关键词 大模型；自回归解码；混合键合；加速器；边缘设备

中图法分类号 TP311.13；TP309

DOI: 10.7544/issn1000-1239.202550301 **CSTR:** 32373.14.issn1000-1239.202550301

以 Transformer[1] 解码器为核心结构的大模型[2-6]在多个领域的任务中表现出卓越的性能，不仅推动了自然语言处理、计算机视觉、聊天机器人和内容生成[7-10]等技术的突破，还在不断重塑各行业的工作流程。将大模型推理部署在边缘设备上正成为一个重要的需求。部署在边缘设备上的大模型不仅能够更好地满足用户的个性化需求，还能有效保障隐私安全。此外，这种部署方式还能提升模型服务的稳定性，使其不受网络状况和云端并发请求数量等因素的影响，从而为用户提供更可靠的使用体验。

大模型的强大能力源于其参数规模、训练数据规模[11]以及推理上下文的扩展[12]。参数规模和上下文长度的增长显著增加了推理开销，使得在边缘设备上高效部署大模型推理面临挑战。大模型的推理分为预填充（prefill）和解码（decode）两个阶段，其中后者需要多次自回归的解码计算，每次只生成 1 个新的词元（token）。在解码过程中，模型的注意力机制层会将每次生成的词元所对应的键值（key value，KV）向量存储起来，形成键值缓存（KV cache）。键值缓存记录了解码过程中的上下文历史，其大小随着上下文长度的增加而线性增长。与模型权重不同，键值缓存是特定于单个请求的，其访存无法在不同请求之间复用。因此，在长上下文解码任务中，对键值缓存的访存开销成为了性能瓶颈[13-14]。此外，在边缘设备的应用场景中，推理负载通常采用较小的批处理规模，甚至以单请求的方式运行。这种特性显著降低了线性层权重的访存复用机会，使得线性层也成为访存密集型计算。单请求的大模型解码负载的计算访存比[15]是 BERT 和 VGG 网络的近 1/100。现有边缘设备访存带宽难以支撑大模型解码推理的有效加速，即便具备足够的算力，仍会因访存带宽低而使计算资源利用率低。

为了缓解大模型长序列解码时的访存压力，Wang 等人[13]提出了局部性感知的注意力机制加速技术，显著减少了键值缓存的访存量并实现端到端加速。该方法通过统计已生成的上下文中各位置注意力分数最常落入的数值区间，将键值缓存压缩至固定大小的中间缓存。在每次解码时，仅需访问少数注意力分数未落入其最常出现区间的位置（活跃位置）所对应的键值，从而使得注意力机制的访存量仅随上下文长度呈现亚线性增长。在序列长度达到 4 096 时，该技术将访存量降低至原始注意力机制的 10%。然而，该技术并未对负载均衡进行优化，导致在进行细粒度任务划分的并行计算时，局部性感知的注意力机制的性能受到负载不均的限制。此外，该工作提出的加速器采用 HBM2 作为主存，难以满足大模型日益增长的带宽需求。

为解决传统 DRAM 在带宽与能效方面的瓶颈，基于混合键合（hybrid bonding）技术的 3 维 DRAM 架构被提出[16]。该架构将控制、电源管理等外围电路集成在逻辑芯片上，并与存储阵列芯片采用铜-铜键合技术连接。这种技术可以实现高密度低电阻的 I/O，显著提升了 DRAM 的带宽和能效。但是，混合键合的 DRAM 通常以较细的粒度（通常为 bank）进行独立控制以实现高访存并行度，且不采用传统的内存交错技术，因此确保访存负载在各 banks 之间的均衡至关重要。如何在基于混合键合的架构上实现局部性感知的大模型稀疏推理，同时兼顾访存的高并

行度与均衡性,是一个具有挑战性的研究问题。

本文设计了一种基于混合键合 3 维 DRAM 的局部性感知大模型加速器,在显著降低长文本解码的访存量的同时,提供比现有的 GPU/加速器高得多的访存带宽,有效解决了上面提到的问题。本文的主要贡献为 3 个方面:

1)提出了一种面向局部性感知注意力机制的活跃位置均衡算法,该算法根据大模型解码过程中活跃位置的动态稀疏性进行负载划分。实验表明,该算法可将划分后的平均负载与最大负载之比提升至 0.96 以上。该算法开销低且硬件友好,适合在硬件加速器中轻量化地实现。

2)提出了一种活跃位置预测算法,用于在当前解码计算时预测下一次解码将访问的活跃位置。实验表明,该算法的平均活跃位置未命中率仅为 1.96%。基于该算法的硬件预测器能够高效地支持活跃位置预取和访存调度优化,从而提升访存带宽的利用率。

3)设计了一种面向边缘部署场景的大模型推理加速器。在所评估的大模型上,该加速器在批处理大小为 1, 2, 4 时,分别实现了平均 1 971 token/s, 3 752 token/s, 6 849 token/s 的吞吐,相当于 A100 GPU[17] 的 50.5 倍、49.4 倍、50.0 倍,LAD-HBM 加速器[13] 的 23.7 倍、22.9 倍、21.5 倍。同时,其平均能耗分别为 0.23 J/token, 0.12 J/token, 0.07 J/token,相当于 GPU 的 5.1%, 5.1%, 4.9%, LAD-HBM 的 46.2%, 46.9%, 48.2%。

1 相关工作

1.1 混合键合的 3 维 DRAM

DRAM 因其较低的访存延迟和较高的存储密度,被广泛采用为各类硬件系统的主存介质。然而,面对近年来大模型、推荐系统及图分析等访存密集型任务的发展,传统 DRAM 在容量与带宽方面均难以满足快速增长的需求。混合键合 3 维 DRAM 为应对上述挑战提供了有效的解决方案,其主要结构如图 1 所示。

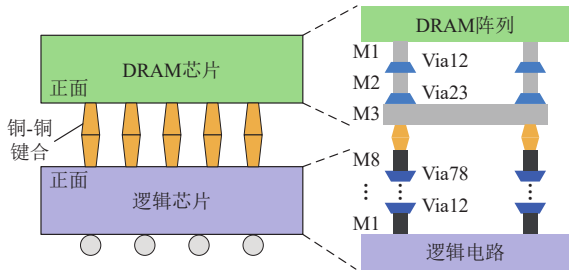


Fig. 1 The structure of hybrid bonding[18]

图 1 混合键合结构[18]

Bai 等人[16] 设计制造了一款基于 LPDDR4/4X 的混合键合 3 维堆叠 DRAM。将逻辑芯片与 DRAM 芯片分别制造后以超高密度、低阻抗的铜-铜键合方式进行面对面堆叠,实现了每 1 Gb 高达 34 GBps 的访存带宽与 0.88 pJ/b 的低能耗。他们基于该技术开发出可量产的 4Gb LPDDR4/4X 芯片,该芯片由 4 颗 1Gb DRAM 芯片与 1 颗逻辑芯片组成,内部集成了超 64 000 个混合键和连接。Yue 等人[18] 研究了如何在基于混合键合 DRAM 的加速器上,利用视觉数据在空间上的聚类相似性来加速视觉 AI 任务。他们将专用加速逻辑单元与 DRAM bank 紧密集成,实现了高效的稀疏检测,并提出稀疏差值表示和稀疏均衡技术来提升 bank 间负载的均衡性。Niu 等人[19] 设计了面向推荐系统的混合键合加速器。该设计将 25 nm DRAM 芯片与 55 nm 逻辑芯片通过 3 μm 间距的铜-铜键合进行连接,实现了 64 Mb/mm^2 的存储密度,提供高达 2.4 GBps/mm^2 的带宽密度。逻辑芯片内集成了匹配引擎与神经网络引擎,其中匹配引擎包含距离计算模块和基于最大堆的 top-k 模块来支持推荐系统的匹配阶段,而神经网络引擎则集成了 INT8 脉动阵列以支持多层感知机(MLP)的计算。

然而混合键合 DRAM 技术在实际部署中仍存在着一些困难。1)混合键合 DRAM 在提升了存储系统并行度和总带宽的同时,引入了更强的分布式特性,这使得确保混合键合 DRAM 系统的利用率成为了新的挑战。2)混合键合 DRAM 的制造成本高,存在着对准精度、晶圆翘曲、良率以及 I/O 面积挤占有效存储面积等多方面的挑战。通常,键合设备的对准精度需为焊盘直径(pad diameter)的 0.1~0.25 倍。例如对于 1 μm 直径的焊盘,对准精度就需要在 100~250 nm 之间。3)逻辑层位于 DRAM 层和封装基板之间,通过铜柱向上方冷却系统导热,需控制逻辑层的厚度在 10 μm 左右[18] 以解决积热问题。

1.2 大模型解码推理加速器

大模型解码推理具有较低的计算强度,即计算量与对主存储器的访存数据量之比较低。随着模型参数规模和上下文长度的不断增长,大模型解码推理的访存开销日益突出。Park 等人[20] 提出了存内计算的 AttAcc 加速系统来加速大模型解码阶段的注意力计算。该系统在高带宽存储器(high bandwidth memory, HBM)内部集成矩阵向量乘计算单元,并构建 GPU 和 AttAcc 之间的流水线协同处理机制来加速端到端的大模型推理。Yu 等人[15] 提出一种基于

芯粒技术的混合异构架构，结合具备高计算能力的NPU与具备近存处理能力的NAND闪存芯片，通过高带宽芯粒间互联实现NPU与近存处理单元的高效协同。该架构引入硬件感知的分块策略以及轻量级NAND闪存片上纠错机制，有效提升了带宽利用率，并减少了闪存数据错误对模型精度的影响。Wang等人[13]利用大模型解码过程中注意力分数的数值局部性，提出了一种将键值缓存压缩为固定大小的中间缓存的算法。他们分模块设计了所提出的算法中的主要步骤，并实现了专用加速器，能够高效地维护中间缓存，从而显著降低访存开销，实现端到端大模型解码推理的吞吐和能效的提升。

2 局部性感知的注意力解码与混合键合DRAM

2.1 大模型推理的原理

大模型的结构由多层Transformer层堆叠而成，每个Transformer层主要由注意力机制和线性层组成。大模型的推理分为2个阶段，即预填充阶段和解码阶段。在预填充阶段，模型接收提示(prompt)，并行地计算提示中所有词元之间的注意力关系，生成第1个输出词元。由于该阶段主要涉及计算强度较高的矩阵乘法运算，通常不会产生访存瓶颈。

在解码阶段，计算以自回归迭代的方式进行，每次解码仅生成1个新的词元，且该词元的生成依赖于先前已生成的所有词元(包括提示中的词元)。对所有已生成词元的记忆能力是大模型性能的关键，而这一能力依赖于注意力机制。下面详细解释注意力机制在解码过程中的计算。记已生成的词元的键值缓存为$\boldsymbol{K},\boldsymbol{V}$，它们都是大小为$n\times d$的张量($n$和$d$分别是已生成的词元数量和单个注意力头的嵌入维度)。在新的一次解码时，注意力机制以新词元的查询、键和值为输入，分别记为$\boldsymbol{q}_{n+1},\boldsymbol{k}_{n+1},\boldsymbol{v}_{n+1}$，它们均为$1\times d$的张量。首先，注意力机制需访存键值缓存，并通过将当前新的键值拼接至末尾来更新键值缓存：

$$\boldsymbol{K}\leftarrow\left(\boldsymbol{K}^{\mathrm{T}},\boldsymbol{k}_{n+1}^{\mathrm{T}}\right)^{\mathrm{T}},\ \boldsymbol{V}\leftarrow\left(\boldsymbol{V}^{\mathrm{T}},\boldsymbol{v}_{n+1}^{\mathrm{T}}\right)^{\mathrm{T}}。\tag{1}$$

之后，计算新查询和更新后的$\boldsymbol{K},\boldsymbol{V}$之间的注意力机制：

$$\begin{aligned}\boldsymbol{s}&=(s_1,s_2,\cdots,s_{n+1})=\frac{\boldsymbol{q}_{n+1}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}},\\ \boldsymbol{o}&=\frac{(e^{s_1},e^{s_2},\cdots,e^{s_{n+1}})\cdot\boldsymbol{V}}{\sum_i e^{s_i}}。\end{aligned}\tag{2}$$

从式(1)(2)可见，注意力机制在解码阶段的主要计算为矩阵-向量乘法，其计算强度低会产生访存瓶颈。同时，键值缓存的规模随着已生成的词元数量线性增长，在长序列生成任务中，对庞大键值缓存的频繁访存会造成显著的延迟。

2.2 局部性感知注意力机制的原理

分段线性方法[21-22]是加速器和边缘设备中实现超越函数的常用技术。在注意力机制中，主要的超越函数是*softmax*中的指数运算。局部性感知的注意力解码机制[13]LAD(locality aware decoding)提出将注意力机制中的指数函数定义域分为若干个子区间，在每个子区间内用线性函数做近似计算，如图2所示，并验证了这种近似方法几乎不影响模型的精度。

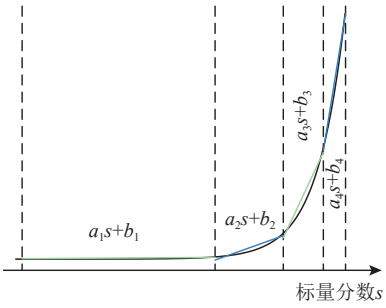


Fig. 2 Exponential function computed by piecewise linear approach

图2 分段线性方法计算指数函数

在采用了分段线性方法后，单次解码中的注意力机制计算可以表达为：

$$\boldsymbol{o}=\frac{\sum_i\left(a_i\left(\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}-m\right)+b_i\right)\boldsymbol{v}_i}{\sum_i\left(a_i\left(\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}-m\right)+b_i\right)},\tag{3}$$

其中i为遍历上下文中所有位置的索引，$\boldsymbol{q}$为查询向量，$\boldsymbol{k}_i,\boldsymbol{v}_i$为位置$i$的键值向量，$m$为注意力分数$\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}$中的最大值，而$a_i$，$b_i$为分段线性系数。在不同的解码步骤中，$a_i$，$b_i$随着注意力分数的变化而变化。局部性是指同一个位置在多次解码步骤中，其注意力分数以较高概率落入某一特定区间。利用这种局部性，LAD统计每个位置最常落入的区间，即众数区间(mode interval)，记位置i的众数区间的系数为a_i^*，b_i^*。在1次解码当中，注意力分数落在众数区间之外的位置被定义为活跃位置，记活跃位置构成的集合为J，位置i的真实区间和活跃区间的系数差为$\alpha_i=a_i-a_i^*$，$\beta_i=b_i-b_i^*$(非活跃位置的$\alpha_i=0,\beta_i=0$)。式(3)可等价地改写为

$$\Delta_{num}=\sum_{i\in J}\alpha_i\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{v}_i-m\sum_{i\in J}\alpha_i\boldsymbol{v}_i+\sum_{i\in J}\beta_i\boldsymbol{v}_i,$$
$$\Delta_{den}=\sum_{i\in J}\alpha_i\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}-m\sum_{i\in J}\alpha_i+\sum_{i\in J}\beta_i。\tag{4}$$

$$\boldsymbol{o}=\frac{\boldsymbol{q}\boldsymbol{A}-m\boldsymbol{B}+\boldsymbol{C}+\Delta_{num}}{\boldsymbol{q}\boldsymbol{D}-mE+F+\Delta_{den}}。\tag{5}$$

式(5)中$\boldsymbol{A}$~F代表基于各位置的众数计算出的中间缓存: $\boldsymbol{A}=\sum_i a_i^*\boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{v}_i$, $\boldsymbol{B}=\sum_i a_i^*\boldsymbol{v}_i$, $\boldsymbol{C}=\sum_i b_i^*\boldsymbol{v}_i$, $\boldsymbol{D}=\sum_i a_i^*\boldsymbol{k}_i^{\mathrm{T}}$, $E=\sum_i a_i^*$, $F=\sum_i b_i^*$。当某些位置的众数更新时, 中间缓存也应相应地更新。设需要更新众数的位置的集合为U, 则更新公式为

$$\boldsymbol{A}=\boldsymbol{A}+\sum_{i\in U}\alpha_i\boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{v}_i,\ \boldsymbol{B}=\boldsymbol{B}+\sum_{i\in U}\alpha_i\boldsymbol{v}_i,$$
$$\boldsymbol{C}=\boldsymbol{C}+\sum_{i\in U}\beta_i\boldsymbol{v}_i,\ \boldsymbol{D}+=\sum_{i\in U}\alpha_i\boldsymbol{k}_i^{\mathrm{T}},\tag{6}$$
$$E=E+\sum_{i\in U}\alpha_i,\ F=F+\sum_{i\in U}\beta_i。$$

总结来说, LAD 将注意力机制解码计算分解为先利用固定大小的中间缓存计算初始分子$\boldsymbol{q}\boldsymbol{A}-m\boldsymbol{B}+\boldsymbol{C}$和分母$\boldsymbol{q}\boldsymbol{D}-mE+F$, 再根据活跃位置的系数差计算修正项$\Delta_{num}$和$\Delta_{den}$以对分子和分母进行修正。这种做法的优势在于, 每次解码仅需访存固定大小的中间缓存和少量活跃位置的键值向量, 相较于原始注意力机制, 在长序列解码时可减少大量冗余访存。

2.3 混合键合 DRAM 的机会与挑战

为了应对 DRAM 性能滞后于计算部件访存需求的挑战, 混合键合技术被引入, 以同时提升访存带宽和能效[16,18,20,23]。混合键合技术将 DRAM 的控制电路、DFT 模块等从 DRAM 芯片分离出来, 集成在了逻辑芯片上, 并利用铜-铜键合技术在垂直方向上将 DRAM 芯片和逻辑芯片堆叠连接。混合键合技术具有 3 种优势[16]:

1)高 I/O 密度。相邻 I/O 间可实现 3 μm 间距, 实现高达 110 000/mm^2 的 I/O 密度, 该密度较 HBM 中使用的微凸点技术高出数百倍。

2)低寄生电阻。DRAM 数据通过铜互连直接传输至逻辑单元, 有效降低寄生电阻至 0.5 Ω 以下, 同时将数据存取能效提升至 0.88 pJ/b。

3)存储与逻辑芯片分离。DRAM 芯片与逻辑芯片可采用相互独立的制程工艺, 从而能够针对各自的性能需求分别进行设计与工艺优化, 可以在逻辑芯片上实现除传统 DRAM 外围电路之外的更复杂的逻辑功能。

混合键合 DRAM 的上述优势对于构建高性能边缘侧推理平台具有重要意义。在 2.2 节中介绍的 LAD 机制显著降低了注意力机制解码阶段的访存需求, 但其主要计算负载仍是低计算强度的特殊算子和矩阵向量乘法。因此, LAD 机制仍然是访存受限的。实现混合键合 3 维 DRAM 的架构来加速基于 LAD 的大模型推理, 不仅可以大幅减少长序列解码时的访存量, 还能显著提升访存性能与能效, 这为边缘侧大模型推理提供了强有力的支撑。然而, 在混合键合 DRAM 的架构上部署基于 LAD 的大模型面临着一些问题, 本文将这些问题归纳为 2 个方面, 即访存的空间均匀性和时间均匀性问题。

1)空间均匀性。混合键合技术显著提升了 I/O 密度, 从而实现了更高的访存并行度, 并支持更细粒度的独立访存操作。与传统 DRAM 结构相比, 这种变化在数据映射优化和高效利用带宽方面引入了新的挑战。实际上, 现有混合键合 DRAM 可实现独立访存每个 bank[18], 因此当负载在 bank 间分布不均时, 会导致部分 bank 成为瓶颈, 显著降低整体带宽利用率, 此即访存的空间均匀性问题。在 LAD 机制中, 访存和计算围绕着活跃位置进行, 但活跃位置在上下文序列中的分布是不均匀的, 且在解码过程中动态变化, 这使得保持访存的空间均匀性具有挑战性。图 3 展示了采用朴素的均分策略将上下文分配至所有 LAD-3D 计算单元(参考 3.1 节)的情况下, 各模型

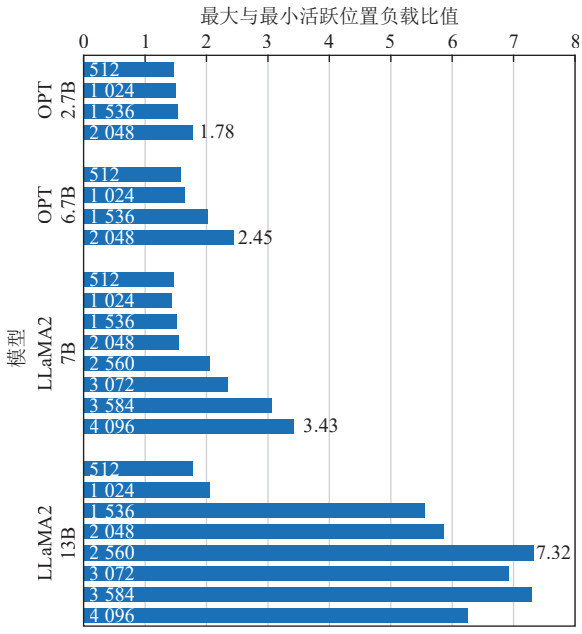


Fig. 3 The ratio of maximum to minimum active position load for each model under different context lengths

图 3 不同上下文长度下各模型最大与最小活跃位置负载比值

最大与最小活跃位置负载的比值。可以观察到，在所有测试模型中，活跃位置的分布均存在不均衡现象，且其严重程度随解码上下文长度的增加而加剧。

2)时间均匀性。LAD 机制中的活跃位置是根据每次解码时注意力分数所在的区间动态确定的，其随着查询向量的不同不断变化。因此，活跃位置的访存模式是动态变化且在每次解码计算发生之前不能确定的。此外，虽然线性层权重的访存模式是静态的，但在不同的批处理大小下，单位时间内的访存需求也存在差异。如何保障访存的时间均匀性，使 DRAM 持续处于高带宽利用率的状态，是部署基于 LAD 的大模型的另一个挑战。

3 基于混合键合架构的局部性感知大模型解码加速器

3.1 加速器总体架构及其并行策略

本文提出的加速器架构由 8 个 LAD-3D 芯片组成，每个 LAD-3D 芯片内含 32 个 LAD-3D 计算单元，这些计算单元通过环形拓扑 NoC 互连。同时，LAD-3D 芯片之间通过 D2D 互连形成链式拓扑。图 4 展示了 LAD-3D 加速器的总体架构。

与此种互连拓扑相适应地，本文提出芯片间流水线并行与芯片内张量并行相结合的并行策略。流水线并行策略将大模型的所有 Transformer 层均分为 8 组，并分别映射到 8 个 LAD-3D 芯片上执行。后一个 LAD-3D 芯片接收前一个 LAD-3D 芯片的输出作为输入，由于在解码过程中，Transformer 层之间传递的张量仅为单个词元(在批处理大小大于 1 时，为批大小个词元)，因此 D2D 互连上的通信压力小，可以采用开销较低的 D2D 互连设计。

本文根据大模型中每个算子的特性，采用相应的算子内张量并行策略，以高效利用 LAD-3D 计算单元之间的环形 NoC。线性算子采用列并行的策略，将权重矩阵沿着列维度均分为 32 份，并分别映射至不同的 LAD-3D 计算单元。每个 LAD-3D 计算单元仅计算线性层输出中对应于自身权重的通道，因此需通过 All-Gather 聚合得到完整的输出，而 All-Gather 在环形拓扑上效率很高。

对于式(4)(5)(6)所示的 LAD 注意力算子(下文记为算子 1, 2, 3)，本文采用不同的并行策略，以优化它们的通信开销，如图 5 所示。算子 1 不适合采用头并行，因为不同头的活跃位置数量不同，若采用头并行则会使计算单元之间的活跃位置数量难以平衡，从而引发负载不均，影响性能。为解决这一问题，本文提出一种针对活跃负载均衡优化的序列并行(sequence parallelism)策略，将上下文序列划分为多个子集，并分配至不同的计算单元，使得每个计算单元所分配到的位置中，活跃位置的数量大致相当(详见 3.3 节)。设划分后，式(4)中的活跃位置集合 J 被划分为子集 $J_1, J_2 \cdots, J_m$，第 i 个计算单元负责计算 J_i 对应的分子与分母修正项：

$$\begin{aligned}\Delta_{\boldsymbol{num}}^{J_i} &= \sum_{i\in J_i}\alpha_i \boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{v}_i - m\sum_{i\in J_i}\alpha_i\boldsymbol{v}_i + \sum_{i\in J_i}\beta_i\boldsymbol{v}_i\,,\\ \Delta_{den}^{J_i} &= \sum_{i\in J_i}\alpha_i \boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}} - m\sum_{i\in J_i}\alpha_i + \sum_{i\in J_i}\beta_i\,。\end{aligned} \tag{7}$$

需要注意的是，这里的公式仅展示了对单个注意力头的计算，而每个计算单元需对所有注意力头执行该计算。这样，每个计算单元算得所有注意力头的分子分母修正项的一个部分和。由于算子 2 采

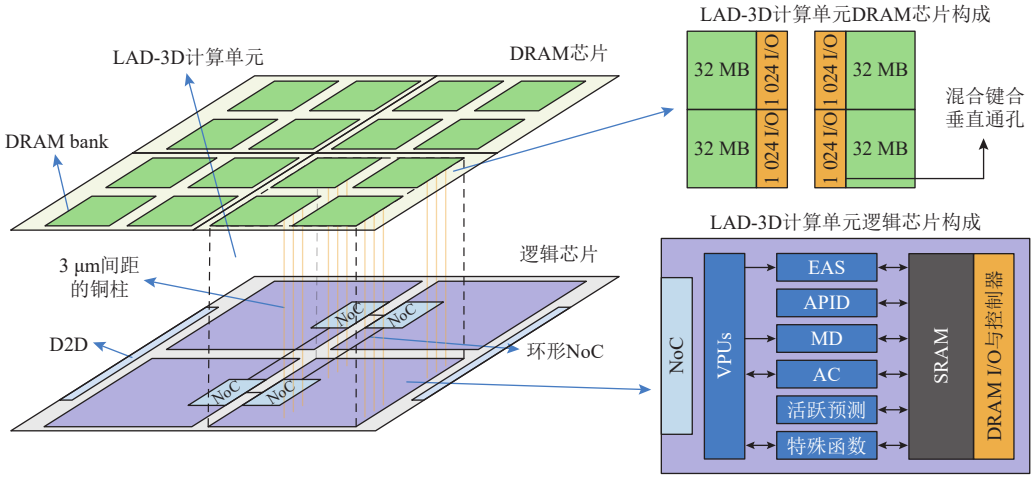
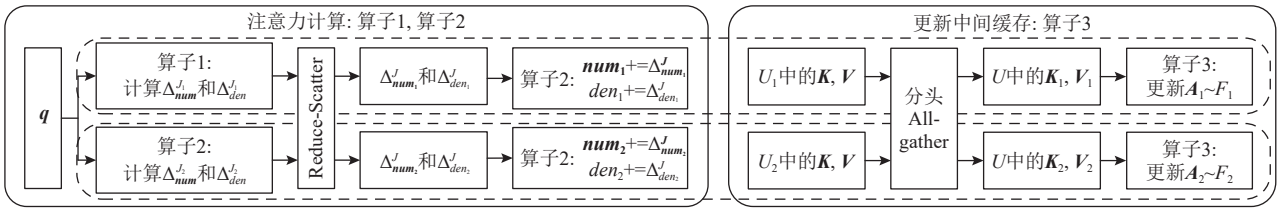


Fig. 4 The overall architecture of the accelerator

图 4 加速器的整体架构



注: 图中变量下标区分注意力头划分后得到的不同子张量, 第1行和第2行的算子分别被映射到2个不同的LAD-3D计算单元。

Fig. 5 The parallelization strategy across LAD-3D computational units

图 5 LAD-3D 计算单元之间的并行策略

用头并行, 因此需在计算单元之间进行 Reduce-Scatter, 使得每个计算单元获得其负责的注意力头对应的修正项的全部和。在环形拓扑上, Reduce-Scatter 能够实现最高的理论通信效率。

算子 2 和算子 3 的计算主要围绕中间缓存展开, 而每个注意力头都拥有独立的中间缓存, 并且不同注意力头上的计算相互独立。因此, 采用头并行策略能够在这 2 个算子上实现最高的计算与通信效率。算子 1 末尾的 Reduce-Scatter 确保每个计算单元获得与其中间缓存匹配的修正项, 可直接按照式(3)加至分子分母中。对于算子 3, 由于采用序列并行, 需要更新众数的位置集合U也被划分为子集$U_1, U_2, \cdots, U_m$。若每个计算单元基于子集U_i计算:

$$\begin{aligned}
&\Delta \boldsymbol{A}^{U_i}=\sum_{i\in U_i}\alpha_i \boldsymbol{k}_i^{\mathrm{T}}\boldsymbol{v}_i,\ \Delta \boldsymbol{B}^{U_i}=\sum_{i\in U_i}\alpha_i \boldsymbol{v}_i,\\
&\Delta \boldsymbol{C}^{U_i}=\sum_{i\in U_i}\beta_i \boldsymbol{v}_i,\ \Delta \boldsymbol{D}^{U_i}=\sum_{i\in U_i}\alpha_i \boldsymbol{k}_i^{\mathrm{T}},\\
&\Delta E^{U_i}=\sum_{i\in U_i}\alpha_i,\ \Delta F^{U_i}=\sum_{i\in U_i}\beta_i,
\end{aligned} \tag{8}$$

则会得到中间缓存更新量$\Delta \boldsymbol{A}$~ΔF的部分和, 需要对它们进行 Reduce-Scatter。然而, $\Delta \boldsymbol{A}$~ΔF的大小较大, 会造成高昂的 Reduce-Scatter 通信开销。例如, $\Delta \boldsymbol{A}$的大小就是算子 1 中分子修正项的d倍(d为单个注意力头的嵌入维度, 通常d=128)。为解决这个问题, 在算子 3 执行之前, 每个计算单元需要执行 All-Gather 来收集其负责的注意力头对应的集合U中所有位置的键值。由于众数的稳定性, 集合U的规模是很小的(在本文的实验中仅占序列长度的10^{-3}量级), 因此这一通信开销极低。在执行完 All-Gather 后, 每个计算单元即可计算其所负责注意力头的更新量, 并完成对应的中间缓存的更新。

3.2 计算单元设计

如图 5 所示, LAD-3D 计算单元包含高效注意力分数(efficient attention score, EAS)计算模块、活跃位置识别(active position identification, APID)模块、众数差异(mode discrepancy, MD)模块、注意力计算(attention computation, AC)模块、特殊函数单元(special function unit, SFU)以及负责核心计算任务的向量处理单元(vector processing unit, VPU)。图 6 展示了这些模块的详细结构。在一个 LAD-3D 计算单元中, EAS, APID, MD, AC 模块共用同一套 VPU, 它们分时调用 VPU 依次执行注意力机制层的各部分计算。SFU 负责处理特殊函数, 包括层归一化(LayerNorm)、旋转位置编码(rotary positional embedding)以及激活函数等。线性层的计算完全由 VPU 负责执行。

EAS 模块的核心功能是计算查询和键中心(key center)之间的注意力分数, 并根据新产生的键实时更新键中心。EAS 计算的注意力分数仅用于判定是否落在众数区间以识别活跃位置, 故而无需精确计算。EAS 模块动态地从键向量序列中选取方向中心作为键中心, 使得即便仅访存这部分占比很小的中心, 依然能够以足够的精度识别出活跃位置。为此, EAS 模块首先计算查询和键中心之间的点积, 并将计算结果进行缩放, 即乘以每个位置的键模长与其对应的键中心模长的比值, 从而得到近似的注意力分数。对于新生成的键, EAS 模块计算其与现有键中心之间的余弦相似度, 并找出相似度绝对值的最大值。随后, 该最大值与预设阈值 0.98 进行比较, 若其大于 0.98, 则新键归属于该最大值对应的中心, 反之新键自身成为一个新的中心。

APID 模块的主要功能包括识别活跃位置、更新活跃概率并维护活跃位置在计算单元间分布的均衡。该模块包含 2 个查找表, 分别存储了注意力分数区间的上下界。APID 模块以 EAS 模块算得的注意力分数, 以及各位置当前的众数区间、众数区间计数为输入。APID 模块从查找表中检索众数区间的上下界并与注意力分数进行比较以判定该位置是否为活跃状态。若判定为活跃, 则将该位置添加到活跃位置 FIFO 中; 若判定为不活跃, 则将该位置的众数区间计数加 1, 以记录其众数区间的又一次出现。活跃位置信号输入至活跃概率计算模块, 以计算各位置

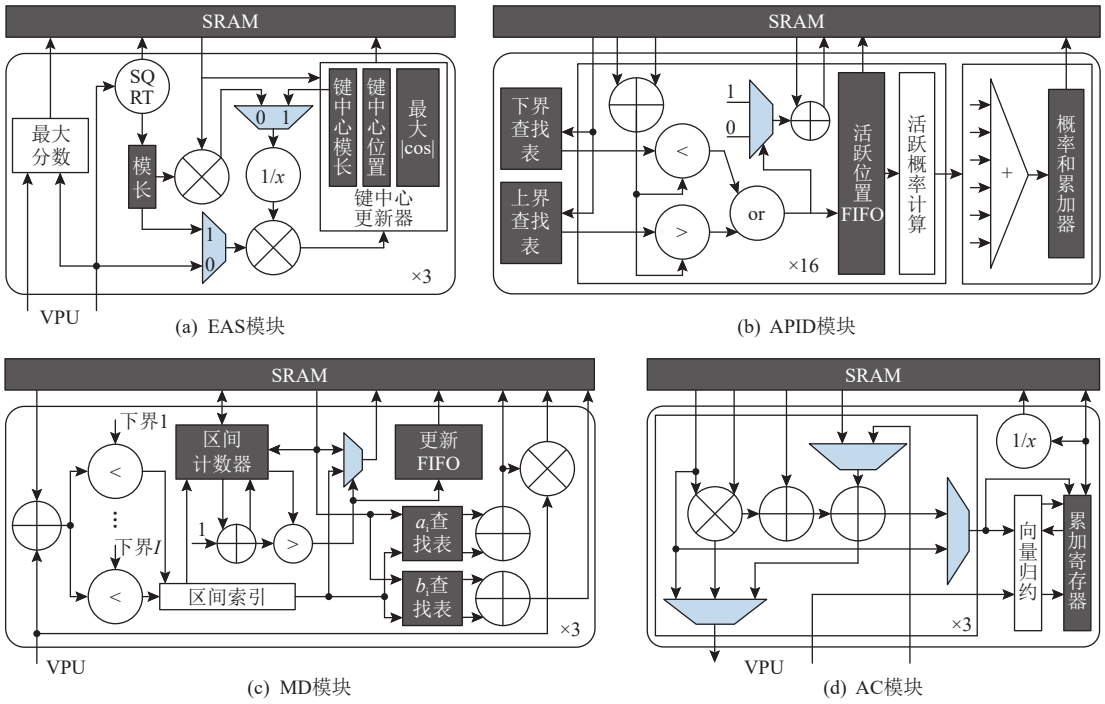


Fig. 6 The architectural details of LAD-3D accelerated modules

图 6 LAD-3D 加速模块的架构细节

的活跃概率，并累加在概率和累加器中，用于维护活跃位置的均衡分布（详见 3.3 节）。

基于 APID 识别出的活跃位置，MD 模块计算这些活跃位置的精确注意力分数，并将所得分数与各区间的下界进行比较，从而确定活跃位置所属的实际区间。对于每个被处理的活跃位置，将其所有区间的计数加载到一组寄存器中。MD 模块将活跃位置的实际区间的计数加 1，而后与其当前的众数区间的计数比较，若实际区间的计数更大，则表明该位置的众数区间发生了改变，将其添加到众数更新 FIFO 中。MD 模块还包含存储了各区间的线性系数的查找表，用以查找活跃位置的精确区间和众数区间对应的系数，进而计算系数差作为后续 AC 模块的输入，以支持对活跃位置进行修正计算。

AC 模块负责执行 LAD 注意力计算的主要步骤，即式(4)(5)(6)。该模块包含 1 个向量累加寄存器和 1 个向量归约模块，支持在单个周期内计算累加寄存器中的向量与另外 3 个向量的和，并将结果存至累加寄存器。AC 模块首先基于中间缓存计算初始的注意力输出分子 $\boldsymbol{qA}-m\boldsymbol{B}+\boldsymbol{C}$ 和分母 $\boldsymbol{qD}-mE+F$，并将计算结果存入向量累加寄存器中。随后，AC 模块开始对分子和分母进行修正计算。在实际计算中，修正项主要由 2 部分组成：一部分是来源于活跃位置的系数差，应根据 MD 模块算得的活跃位置系数差，分别计算分子和分母的修正项 $\sum_{i\in J}\left(\alpha_i\left(\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}-m\right)+\beta_i\right)\boldsymbol{v}_i$ 和 $\sum_{i\in J}\left(\alpha_i\left(\boldsymbol{q}\boldsymbol{k}_i^{\mathrm{T}}-m\right)+\beta_i\right)$。另一部分修正是来源于尚未纳入中间缓存的位置，该部分的计算形式同第 1 部分，只是系数差需要修改为这些位置当前所在区间的系数。修正计算分多周期完成，每周期计算修正项的 1 个部分和并通过归约模块更新到向量累加寄存器中。待所有部分和计算完成后，寄存器中便累加了最终的分子和分母，可直接用于计算注意力输出。为了更新中间缓存，同一芯片上的每个 LAD-3D 计算单元通过 NoC 收集所有 MD 模块，众数更新 FIFO 中记录的位置的键值向量中自己负责处理的注意力头对应的部分（见 3.1 节并行策略）。随后，每个 LAD-3D 计算单元的 AC 模块即可执行式(6)以更新其负责的注意力头对应的中间缓存。在 AC 模块中，更新计算采用逐向量的方式进行。具体而言，中间缓存被拆解为一系列向量，每次将 1 个待更新向量加载到向量累加寄存器中，完成累加更新后，再处理下一个向量，直至更新完成。

EAS, MD, AC 模块的硬件并行度与单个 LAD-3D 计算单元内的 VPU 数量保持一致，以确保这些模块能够恰好调用所有的 VPU 资源。APID 模块不进行向量运算，故而不需要调用 VPU，其硬件并行度不受 VPU 数量的限制。为了在较少的周期内完成活跃位置识别，APID 的硬件并行度需要足够高。在实验

中，本文设定 EAS, MD, AC 模块的并行度皆为 3，而 APID 的并行度为 16。

3.3 访存的空间均匀性：基于活跃概率的活跃位置均衡机制

如 3.1 节所述，本文对 LAD 注意力算子采用了序列并行的策略。设解码上下文的长度为n，将词元位置序号的集合$S=\{1,2,\cdots,n\}$分为m个不相交的子集$S_1,S_2,\cdots,S_m$，且$S_1\cup S_2\cup\cdots\cup S_m=S$。通过这种划分将词元序列分配到$m$个计算单元上，即计算单元$i$处理$S_i$中位置的对应词元。现有的序列并行策略通常按照位置顺序将序列均分为m块，然而，这种简单的划分方式在 LAD 机制中会造成性能损失。具体而言，不同位置的活跃概率存在差异，均分的划分方法可能导致各子集内位置的活跃概率之和差异较大。在解码过程中，活跃概率和较高的子集往往会产生更多的活跃位置，从而使对应的计算单元有更高的计算和访存负载。如图 7 所示，在 LLaMA2-13B 模型的解码过程中，采用均分划分方式会导致不同的 LAD-3D 计算单元的活跃负载差距较大，且该不均衡程度呈现随着上下文长度的增加而加剧的趋势。由于 LAD 计算过程中存在同步需求(3.1 节的 Reduce-Scatter)，系统的整体性能受限于负载最高的计算单元，活跃位置分布的不均衡会成为制约性能的关键瓶颈。

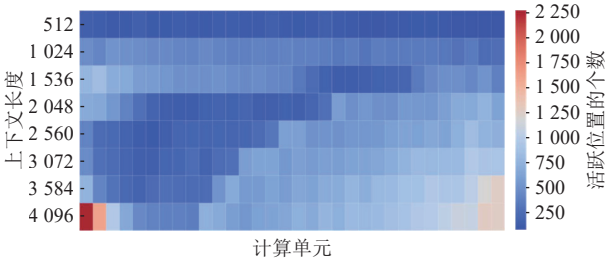


Fig. 7 The heatmap of the distribution of active positions across LAD-3D computational elements

图 7 活跃位置在各 LAD-3D 计算单元间分布的热力图

为了解决上述问题，需要维护各子集的活跃概率之和的动态平衡。受负载均衡(MakeSpan)问题的启发，本文提出一种基于活跃概率的贪心算法，如算法 1 所示。

算法 1. 活跃位置均衡算法。

输入：第j次解码的活跃状态$y_1^j,\cdots,y_{n_j}^j$(其中$j=1,2,\cdots,k$(k为解码次数)，$y_i^j=1$表示活跃)；

输出：位置序号集合S的划分$S_1,S_2,\cdots,S_m$。

① 初始化：$T_i=0,p_i=0,S_i=\varnothing,c_i^j=0$；

② for $j=1$ to k do

③ for $i=1$ to n_j do

④ $p_i\leftarrow\left(p_ic_i^j+\left(y_i^j==1\right)\right)/\left(c_i^j+1\right)$；

⑤ $c_i^{j+1}\leftarrow c_i^j+1$；

⑥ end for

⑦ for $i=1$ to m do

⑧ $T_i\leftarrow 0$；

⑨ for all k in S_i do

⑩ $T_i\leftarrow T_i+p_k$；

⑪ end for

⑫ end for

⑬ $i^*\leftarrow\arg\min T_i$；

⑭ if $j==1$ then

⑮ 将$\{1,2,\cdots,n_1\}$平均分配到$S_1,S_2,\cdots,S_m$中；

⑯ else

⑰ $S_{i^*}\leftarrow S_{i^*}\cup\left\{n_{j-1},n_{j-1}+1\cdots,n_j\right\}$；

⑱ end if

⑲ end for

在第j次解码时，利用最新的活跃状态y_i^j更新每个位置i的活跃概率p_i，根据已有的划分计算每个子集S_i对应的概率和T_i，并找出所有概率和中最小的T_{i^*}。若是预填充阶段(算法 1 的行⑦)则将输入提示中的所有位置平均分配到各集合S_i中。若是解码阶段(算法 1 的行⑧)，则将新生成的位置并入S_{i^*}。

控制均衡机制的硬件模块集成在 APID 模块中，其中包含活跃概率计算模块、概率求和模块以及用于维护子集划分的相关逻辑(图 6(b))。概率计算以及概率求和模块的硬件并行度与 APID 的活跃位置识别并行度保持一致(见 3.2 节)，当待处理的位置总数超出硬件并行度时，采用分批次处理方式多周期完成计算。每个时钟周期识别出的活跃位置信号被送入概率计算模块，该模块实现算法 1 行③的逻辑，更新当前批次位置的活跃概率。由于一个 APID 模块仅负责处理其所属的 LAD-3D 计算单元所分配到的位置，即当前 LAD-3D 计算单元对应的子集中的所有位置(见 3.1 节并行策略)，因此概率求和模块直接将概率计算模块的输出进行累加。概率求和模块的主要结构为一个加法树和加法树输出的累加器。概率求和模块的输出在同一芯片上的各个 LAD-3D 计算单元之间通过环形 NoC 进行 reduce-min 运算。概率和最小的计算单元将当前解码步骤新生成的位置序号添加到其子集中。

3.4 访存的时间均匀性：活跃位置预测与预取机制

在自回归解码的过程中，每次解码的活跃位置是不固定的，这种不确定性对访存效率构成挑战。具体而言，某个位置在当前解码步骤中是否活跃，取

决于其注意力分数所处的区间，而该分数由当前查询向量和键向量之间的点积决定。若总是在计算出注意力分数后才对活跃位置进行访存操作，访存延迟会显著影响 LAD 注意力机制的性能。故而，本文提出一种高精度的活跃位置预测器，如图 8 所示。被预测为活跃位置的键值向量可提前从 DRAM 预取至片上缓存，这为联合调度线性层和注意力层的访存请求提供了优化空间，从而实现解码过程中访存负载的时间均匀性，以提升系统的整体性能。

算法 2 展示了预测器的预测逻辑，其分为预测和更新 2 个部分。该预测器是基于活跃历史记录来预测下一次解码的活跃状态的。对于上下文中的每个位置i，记录最近$hlen$次解码的活跃状态，其中“1”表示活跃，“−1”表示不活跃。这样，每个位置i仅需维护一个$hlen$比特大小的向量$\boldsymbol{h}_i$，其带来的存储和访存开销影响很小。例如，按照本文的实验设定$hlen=32$，一个 32 头的注意力层在上下文长度为 4 096 时，其活跃历史仅占 0.5 MB。预测器的核心在于利用感知机学习活跃历史中的模式[24]，而这些模式很大程度上受活跃概率的影响。因此，预测器按照活跃概率对活跃模式进行分类学习：概率值被分为[0,0.1]，[0.1,0.2]，⋯，[0.9 ~ 1.0]这 10 个区间，所有位置按照其活跃概率所在的区间被分为 10 类，分别用$w_1,w_2\cdots$，w_{10}这 10 个独立的感知机权重进行学习和预测。

在预测阶段，对于每个位置i，预测器首先确定其活跃概率p_i所属的第$j[i]$个区间，并据此选取对应的权重$w_{j[i]}$，随后计算该权重与活跃历史向量之间的点积x_i。若点积$x_i>0$或位置i在当前解码步骤中是活跃的，则预测其在下一次解码是活跃的（算法 2 的行③）。

在更新阶段，预测器根据预测状态和真实状态之间的差异来更新感知机权重。当位置i的输出x_i的绝对值小于阈值δ时，表明感知机预测的稳定性较低；而若预测状态y_i和真实状态$\hat{y}_i$不一致，则说明感知机预测错误。只有在稳定性低且预测错误的情况下，预测器才会更新感知机权重（算法 3 的行①）。位置i将自身模式$y_i\boldsymbol{h}_i$加到权重中来更新感知机（算法 3 的行②）。若为$y_i=1$（活跃），则加$\boldsymbol{h}_i$使得未来该权重与$\boldsymbol{h}_i$的点积增大，提高将$\boldsymbol{h}_i$预测为活跃的可能性；而若$y_i=-1$（不活跃），则加$-\boldsymbol{h}_i$使得点积减小，提高预测为不活跃的可能性。在运行时，同一权重会被多个位置共享。如果直接累加所有位置的更新，会由于更新向量的模长过大而导致权重不稳定。因此，预测器采用投票机制来更新权重。如算法 3 的行⑦所示，对累加的更新向量取符号函数，确保更新向量的每个元素值为±1，并且更新方向符合大多数位置的模式。

算法 2. 预测器算法。

输入：位置i的活跃历史$\boldsymbol{h}_i\in\{\pm1\}^{hlen}$，位置$i$的活跃概率$p_i$，预测器权重$\boldsymbol{w}_j\in\mathbb{Z}^{hlen}$（其中$j=1,2,\cdots,10$）；

输出：位置i的预测值x_i以及预测活跃状态$\hat{y}_i$。

① 确定p_i所在的区间$I_{j[i]}$，其中$I_{j[i]}=[0.1(j[i]-1),0.1j[i]]$；

② $x_i\leftarrow\langle\boldsymbol{h}_i,\boldsymbol{w}_{j[i]}\rangle$；

③ if $(x_i>0)|(\boldsymbol{h}_i[-1]==1)$ then

④ $\quad\hat{y}_i\leftarrow1$；

⑤ else

⑥ $\quad\hat{y}_i\leftarrow-1$；

⑦ end if

算法 3. 更新算法。

输入：预测算法的输出x_i，$\hat{y}_i$，真实活跃状态y_i，阈值δ；

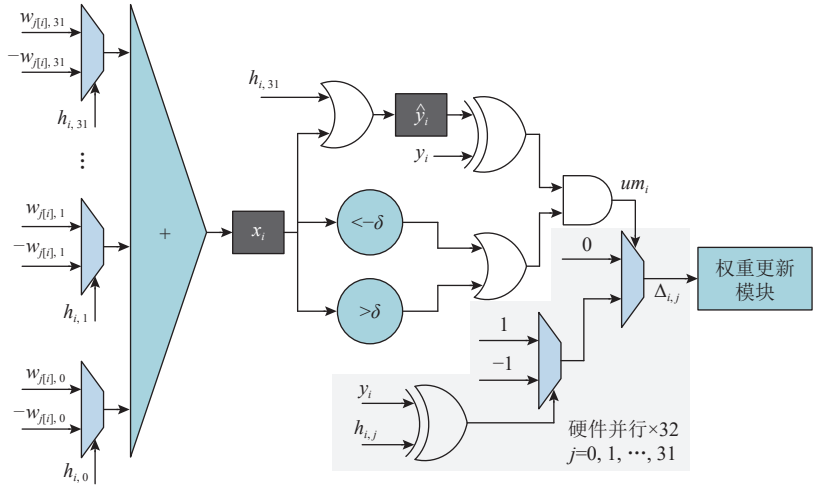


Fig. 8 The architectural details of the active position predictor

图 8 活跃位置预测器的架构细节

输出: 更新后的权重$\boldsymbol{w}_j$。

① $um_i \leftarrow (|x_i| \leqslant \delta) \& (y_i \neq \hat{y}_i)$;

② $\Delta_i \leftarrow y_i \boldsymbol{h}_i um_i$;

③ for $j = 1$ to 10 do

④ $\quad \boldsymbol{update} \leftarrow \boldsymbol{0}$;

⑤ $\quad$ for all i in $\{i | p_i \in I_j\}$ do

⑥ $\quad\quad \boldsymbol{update} \leftarrow \boldsymbol{update} + \Delta_i$;

⑦ $\quad$ end for

⑧ $\quad \boldsymbol{w}_j \leftarrow \boldsymbol{w}_j + \mathrm{sgn}(\boldsymbol{update})$;

⑨ end for

在硬件实现方面, 由于历史$\boldsymbol{h}_i$的元素仅取值为±1, 预测阶段计算点积$\langle \boldsymbol{h}_i, \boldsymbol{w}_{j[i]} \rangle$时并不需要乘法器, 可以通过加法树来实现该点积计算, 省去了乘法器的硬件开销, 如图 8 所示。预测器中负责更新的硬件包括用于根据预测结果来生成更新掩码um_i的逻辑单元、生成更新量Δ的逻辑单元以及权重更新模块。预测器根据um_i, y_i, $\boldsymbol{h}_i$, 通过 2 级选择器生成Δ_i的取值, 可为$\boldsymbol{0}$, $+\boldsymbol{h}_i$或$-\boldsymbol{h}_i$。权重更新模块由权重更新累加器和符号函数逻辑组成, 在计算每个感知机的更新量累加值后取其符号函数作为最终更新值。

基于活跃位置预测器, 加速器可引入预取机制: 在执行当前层计算的同时, 从 DRAM 预取下一层预测活跃位置对应的键值向量、线性层权重等必要数据到片上 SRAM。在实验中, 预测未命中的活跃位置平均仅占 1.96%, 故绝大部分访存都在预取阶段完成, 能够充分进行调度优化。

4 实验评估

4.1 实验方法

1)测试基准与对比方案

本文测试了各个硬件加速器在 OPT 2.7B, OPT 6.7B, LLaMA2 7B, LLaMA2 13B[4,6] 这 4 个具有代表性的大语言模型上的表现。本文选取英伟达 A100-40 GB PCIe[17] GPU 以及 LAD-HBM 加速器[13] 作为对比平台。其中, A100-40GB 部署了流行的大模型推理框架 vLLM[25], 该框架集成了包括 PagedAttention 在内的多种优化技术, 能够充分发挥 GPU 的大模型推理性能。LAD-HBM 加速器采用其论文中推理吞吐最高的版本(3.5 MB 片上缓存), 其采用了和 A100-40GB 配置相同的 HBM 作为主存, 访存带宽为 1555 GBps。

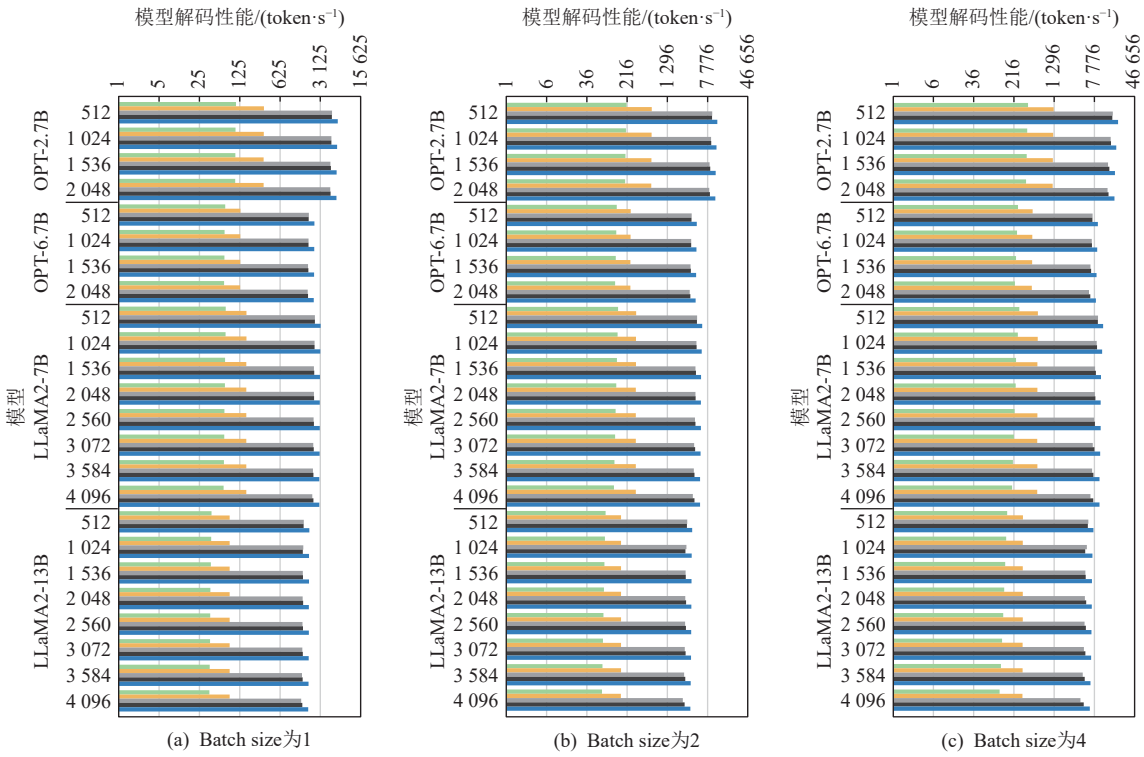
2)硬件实现

本文基于 Verilog 实现了 LAD-3D 加速器, 且所有功能模块均通过功能验证。随后, 使用 Synopsys Design Compiler 基于 TSMC 22 nm 工艺库对 RTL 代码进行了综合, 综合结果表明 LAD-3D 计算单元满足了 1ns 的时序约束, 从而确保 LAD-3D 逻辑芯片能够运行于 1GHz 的工作频率。LAD-3D 的计算部件采用 fp16 的数值表示, 与作为对比测试的 GPU 以及 LAD-HBM 采用相同的数值表示。本文采用 CACTI[26] 对 SRAM 进行模拟, 通过配置阵列大小、端口数等参数来满足 LAD-3D 计算部件的访存需求, 并模拟出功耗与面积数据。本文实现了精确到周期的 LAD-3D 模拟器, 并拓展了开源的模拟器 DRAMSim-3[27] 以模拟混合键合 DRAM, 其 DRAM 参数采用了文献 [18] 中提供的数据配置。每个 DRAM bank 大小为 32 MB, 拥有 1 024 个工作在 500 MHz 的 I/O, 提供 59.6 GBps 的峰值带宽。每个 LAD-3D 计算单元的算力为 0.768 TFLOPS, 垂直方向上混合键合了 4 个 DRAM banks, 提供 238.4 GBps 的峰值带宽以及 128 MB 的 DRAM 存储。为了准确模拟混合键合 DRAM 访存的延迟, 本文在受测的 PyTorch 代码中插入访存记录(memory trace)抓取函数以生成每个测试例运行时的访存命令(包含访存命令类型和地址信息)序列。拓展的 DRAMSim-3 模拟器以所记录的访存命令序列为输入, 模拟出混合键合 DRAM 的访存延迟。由于本文实现的加速器系统由 8 个 LAD-3D 芯片组成的流水线构成, 每个 LAD-3D 芯片集成了 32 个计算单元, 所以系统的峰值算力为 196.608 TFLOPS, 峰值访存带宽为 59.6 TBps, 总 DRAM 容量为 32 GB。

4.2 性能结果

本节展示了 LAD-3D, LAD-HBM, A100 GPU 的端到端大模型解码推理性能对比。其中, LAD-3D 还做了无基于活跃概率的活跃位置均衡机制以及无活跃预取机制的消融实验, 以评估这 2 项技术的有效性。为评估不同上下文长度下的模型性能, 本文测试了在上下文长度为 512, 1 024, …, 4 096 时的解码吞吐性能(其中 OPT 模型仅支持最长 2 048 的上下文长度)。

如图 9 所示, LAD-3D 在所有测试例下都取得了显著好于 GPU-vLLM 和 LAD-HBM 的解码性能。在所有测试例子中, LAD-3D 相较于 GPU-vLLM 在 OPT-2.7B, OPT-6.7B, LLaMA2-7B, LLaMA2-13B 模型上分别实现了 55.2 倍、35.7 倍、44.5 倍、50.0 倍的几何平均加速比。相较于 LAD-HBM, 也实现了平均 17.6 倍、18.5 倍、17.9 倍、22.6 倍的加速比。在执行 LLaMA2-13B 的解码推理时, LAD-3D 在批处理大小分别为 1,



注: 每簇柱子从上至下分别为GPU-vLLM、LAD-HBM、LAD-3D无均衡机制、LAD-3D无预取机制、LAD-3D的数据。

Fig. 9 Decoding performance of LLMs under different context lengths

图 9 大模型在不同上下文长度下的解码性能

2, 4 的情况下, 平均实现了 1 971 token/s, 3 752 token/s, 6 849 token/s 的吞吐, 而 GPU-vLLM 在相同负载下的平均吞吐仅为 39 token/s, 76 token/s, 137 token/s, LAD-HBM 的平均吞吐也仅为 83 token/s, 164 token/s, 319 token/s。无活跃位置均衡机制的 LAD-3D 解码性能平均下降了 24%, 而无活跃位置预测预取机制的 LAD-3D 解码性能平均下降了 22%。LAD-3D 的性能优势来源于 3 个方面: 一是活跃位置的稀疏性; 二是混合键合的 DRAM 拥有远高于 HBM 的带宽; 三是通过活跃位置的均衡调度以及预取机制, 有效提升了混合键合 DRAM 的带宽利用率。

4.3 活跃位置均衡与预测结果

表 1 展示了在不同上下文长度下的各模型测试中, 所有 LAD 计算单元的活跃位置的平均值与最大值的比值。该比值越接近于 1, 表示活跃位置负载在计算单元之间分布得越均衡。在所有测试场景中, 该比值均高于 0.96, 且平均值达到 0.98, 这证明了 LAD-3D 引入的活跃位置均衡机制的有效性。表 2 展示了活跃位置预测器的活跃位置未命中的概率。在所有测试场景下, 未命中概率均小于 4%, 平均值为 1.96%。少数未命中的活跃位置所引起的访存开销对加速器性能的影响极小。

Table 1 Balance Ratios of Active Position

表 1 活跃位置的均衡比值

上下文长度	模型			
	OPT-2.7B	OPT-6.7B	LLaMA-13B	LLaMA-7B
512	0.979 0	0.964 9	0.967 6	0.968 5
1 024	0.973 2	0.973 2	0.981 1	0.985 0
1 536	0.973 9	0.984 8	0.981 3	0.986 6
2 048	0.982 3	0.982 3	0.985 2	0.988 9
2 560	-	-	0.987 6	0.987 2
3 072	-	-	0.989 7	0.986 3
3 548	-	-	0.990 7	0.988 1
4 096	-	-	0.991 9	0.989 4

注: "-"表示该模型不支持该上下文长度。

4.4 面积和功耗结果

表 3 列出了单个 LAD-3D 计算单元的逻辑芯片中主要模块的面积和功耗。除表中所列模块外, 单个 LAD-3D 计算单元还包含 NoC 以及 DRAM I/O 与控制器, 其面积分别为 0.192 5 mm^2 和 2.187 5 mm^2。单个 LAD-3D 芯片包含面积为 57.305 6 mm^2 的 D2D 模块。值得注意的是, 混合键合 DRAM 的 I/O 与控制器集成在了逻辑芯片而非 DRAM 芯片上, 且为实现更高的 I/O 数量面积较大。

Table 2 Predicting Missing Probability of Active Positions

表 2 活跃位置的预测未命中概率

上下文长度	模型			
	OPT-2.7B	OPT-6.7B	LLaMA-13B	LLaMA-7B
512	0.028 9	0.026 2	0.010 4	0.032 2
1 024	0.021 8	0.016 1	0.010 8	0.028 2
1 536	0.024 6	0.018 1	0.017 2	0.026 2
2 048	0.016 8	0.020 1	0.019 1	0.020 9
2 560	-	-	0.014 9	0.017 3
3 072	-	-	0.013 2	0.021 2
3 548	-	-	0.015 2	0.022 3
4 096	-	-	0.013 8	0.015 1

注: “-”表示该模型不支持该上下文长度。

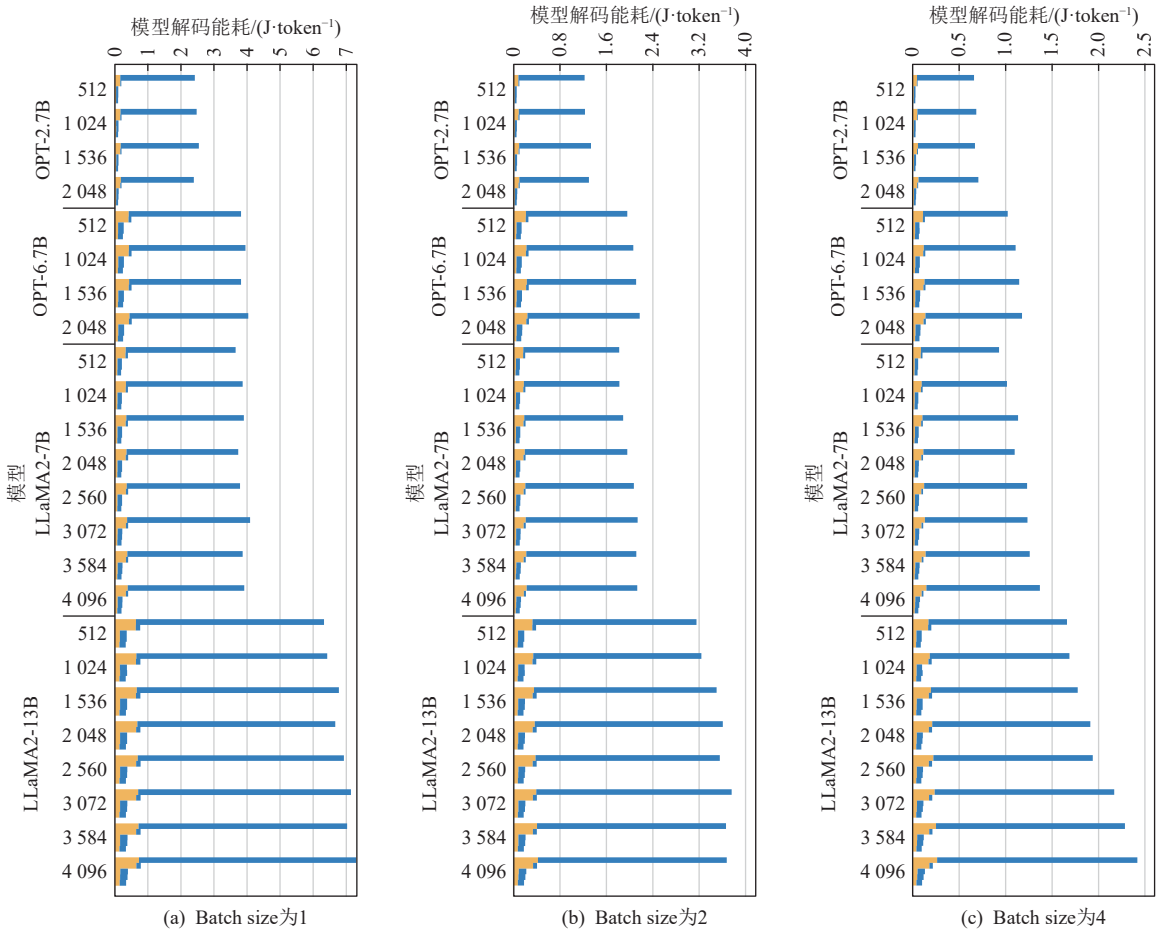
Table 3 Area and Power of Logic Modules of LAD-3D Computational Elements

表 3 LAD-3D 计算单元逻辑模块的面积与功耗

模块名	面积/mm^2	动态功耗/mW	静态功耗/mW
VPU	0.170 6	125.048 6	33.257 1
SFU	0.068 6	43.293 4	16.881 0
EAS	0.004 8	3.231 0	1.167 3
APID	0.005 9	2.713 9	1.211 6
MD	0.002 5	1.752 9	0.485 5
AC	0.085 6	60.205 4	19.829 1
SRAM	6.155 6	1 004.828 8	466.369 6
预测器	0.051 9	18.972 5	9.329 5

DRAM 芯片中，每个 bank 面积为 2.74 mm^2，容量为 32 MB，有 1 024 个 I/O，运行频率为 500 MHz，工作主电压 VDD 为 1.2V±0.02 V。各类时序参数包括：行到列延迟 16.4 ns，CAS 延迟 1.7 ns，行激活保持时间 33.6 ns，预充电时间 11.7 ns，行周期 45.3 ns。

图 10 中展示了 A100 GPU、LAD-HBM、LAD-3D（无活跃位置均衡机制）、LAD-3D(无活跃位置预取机制)和 LAD-3D 在大模型解码任务中的能耗表现。在批处理大小为 1，2，4 时，LAD-3D 的 DRAM 平均能



(a) Batch size为1 (b) Batch size为2 (c) Batch size为4

注: 每簇柱子从上至下分别对应A100 GPU、LAD-HBM、LAD-3D (无活跃位置均衡机制)、LAD-3D (无活跃位置预取机制)和LAD-3D, 每个横向柱子靠近左侧坐标轴的部分表示HBM/DRAM能耗, 远离坐标轴的部分表示逻辑计算部件的能耗。

Fig. 10 Decoding energy of LLMs under different context lengths

图 10 大模型在不同上下文长度下的解码能耗

耗分别为 0.09 J/token, 0.05 J/token, 0.03 J/token, 而计算部件的平均能效分别为 0.14 J/token, 0.07 J/token, 0.04 J/token。在批处理大小为 1, 2, 4 时 LAD-3D 的解码单个词元的能耗分别是 GPU 的 5.1%, 5.1%, 4.9%, LAD-HBM 的 46.2%, 46.9%, 48.2%。无活跃位置均衡机制的 LAD-3D 推理能耗平均上升了 14%, 而无活跃位置预测预取机制的 LAD-3D 推理能耗平均上升了 13%。在本文的实验中, 混合键合 DRAM 展现出 4~5 倍于 HBM 的能耗, 这主要得益于铜-铜键合的低电阻以及显著低于 TSV 的寄生电容。

5 总　　结

本文详细分析了大模型的推理机制及其高访存需求的特性。混合键合 DRAM 因其高 I/O 密度、低寄生电阻的特性, 提供了较传统 DRAM 显著更高的总带宽和能效。本文提出了一种基于混合键合 DRAM 的端侧大模型推理加速器 LAD-3D。LAD-3D 能够利用大模型推理过程中注意力分数的数值局部性减少不必要的键值缓存的访存, 并以混合键合 DRAM 为主存实现了高能效高性能的端侧大模型推理。LAD-3D 实现了基于活跃概率的活跃位置均衡机制和活跃位置预测预取机制, 有效解决了访存的空间和时间均匀性问题, 确保 LAD-3D 能够充分利用混合键合 DRAM 的带宽。

作者贡献声明：王浩然设计了大模型局部性感知解码的混合键合加速器, 并进行了性能和功耗实验；许浩博指导了实验的细节, 搜集了混合键合 DRAM 的架构参数, 制定了论文写作大纲；王颖、韩银和指导了论文总体的技术方向和实验方法。

参　考　文　献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30: 6000−6010

[2] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in Neural Information Processing Systems, 2020, 33: 1877−1901

[3] Achiam J, Adler S, Agarwal S, et al. Gpt−4 technical report[J]. arXiv preprint, arXiv: 2303.08774, 2023

[4] Touvron H, Martin L, Stone K, et al. LLaMA 2: Open foundation and fine-tuned chat models[J]. arXiv preprint, arXiv: 2307.09288, 2023

[5] Liu Aixin, Feng Bei, Wang Bin, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model[J]. arXiv preprint, arXiv: 2405.04434, 2024

[6] Zhang Susan, Roller S, Goyal N, et al. OPT: Open pre-trained transformer language models[J]. arXiv preprint, arXiv: 2205.01068, 2022

[7] He Kaiming, Chen Xinlei, Xie Saining, et al. Masked autoencoders are scalable vision learners[C]//Proc of the 2022 IEEE/CVF Conf on Computer Vision and Pattern Recognition. 2022: 16000−16009

[8] Roziere B, Gehring J, Gloeckle F, et al. Code LLaMA: Open foundation models for code[J]. arXiv preprint, arXiv: 2308.12950, 2023

[9] Liu Aixin, Feng Bei, Xue Bing, et al. Deepseek-v3 technical report[J]. arXiv preprint, arXiv: 2412.19437, 2024

[10] Guo Daya, Zhu Qihao, Yang Dejian, et al. DeepSeek-Coder: When the large language model meets programming–The rise of code intelligence[J]. arXiv preprint, arXiv: 2401.14196, 2024

[11] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv preprint, arXiv: 2001.08361, 2020

[12] Wei J, Wang Xuezhi, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models[J]. Advances in Neural Information Processing Systems, 2022, 35: 24824−24837

[13] Wang Haoran, Li Yuming, Xu Haobo, et al. LAD: Efficient accelerator for generative inference of LLM with locality aware decoding[C]//Proc of the 31st IEEE Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2025: 1482−1495

[14] Ge Xuran, Ou Yang, Wang Bo, et al. Survey of storage optimization techniques in large language model inference[J]. Journal of Computer Research and Development, 2025, 62(3): 545−562 (in Chinese)
(葛旭冉, 欧洋, 王博, 等. 大语言模型推理中的存储优化技术综述[J]. 计算机研究与发展, 2025, 62(3): 545−562)

[15] Yu Zhongkai, Liang Shengwen, Ma Tianyun, et al. Cambricon-LLM: A chiplet-based hybrid architecture for on-device inference of 70b LLM[C]//Proc of the 57th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2024: 1474−1488

[16] Bai Fujun, Jiang Xiping, Wang Song, et al. A stacked embedded DRAM array for LPDDR4/4X using hybrid bonding 3D integration with 34GB/s/1Gb 0.88 pJ/b logic-to-memory interface[C]//Proc of the 68th IEEE Int Electron Devices Meeting (IEDM). Piscataway, NJ: IEEE, 2020: 6.6. 1−6.6. 4

[17] Choquette J, Gandhi W, Giroux O, et al. Nvidia a100 tensor core GPU: Performance and innovation[J]. IEEE Micro, 2021, 41(2): 29−35

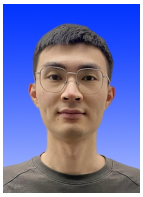
[18] Yue Zhiheng, Wang Huizheng, Fang Jiahao, et al. Exploiting similarity opportunities of emerging vision AI models on hybrid bonding architecture[C]//Proc of the 51st ACM/IEEE Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2024: 396−409

[19] Niu Dimin, Li Shuangchen, Wang Yuhao, et al. 184QPS/W 64Mb/mm^2 3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system[C]//Proc of the 2022 IEEE Int Solid-State Circuits Conf (ISSCC). Piscataway, NJ: IEEE, 2022, 65: 1−3

[20] Park J, Choi J, Kyung K, et al. AttAcc! Unleashing the power of PIM for batched transformer-based generative model inference[C]//Proc of the 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems, Volume 2. New York: ACM, 2024: 103−119

[21] Amin H, Curtis K M, Hayes-Gill B R. Piecewise linear approximation applied to nonlinear function of a neural network[J]. IEE Proceedings-Circuits, Devices and Systems, 1997, 144(6): 313−317

[22] Khan H, Khan A, Khan Z, et al. NPE: An FPGA-based overlay processor for natural language processing[J]. arXiv preprint, arXiv: 2104.06535, 2021
[23] Jiang Xiping, Zuo Fengguo, Wang Song, et al. A 1596-GB/s 48-Gb stacked embedded DRAM 384-core SoC with hybrid bonding integration[J]. IEEE Solid-State Circuits Letters, 2022, 5: 110−113
[24] Jiménez D A, Lin C. Dynamic branch prediction with perceptrons[C]//Proc of the 7th HPCA Int Symp on High-Performance Computer Architecture. Piscataway, NJ: IEEE, 2001: 197−206
[25] Kwon W, Li Zhuohan, Zhuang Siyuan, et al. Efficient memory management for large language model serving with pagedattention[C]//Proc of the 29th Symp on Operating Systems Principles. New York: ACM, 2023: 611−626
[26] Muralimanohar N, Balasubramonian R, Jouppi N P. CACTI 6.0: A tool to model large caches[J]. HP Laboratories, 2009, 27: 28
[27] Li Shang, Yang Zhiyuan, Reddy D, et al. DRAMSim3: A cycle-accurate, thermal-capable DRAM simulator[J]. IEEE Computer Architecture Letters, 2020, 19(2): 106−109



Wang Haoran, born in 1997. PhD. His main research interests include computer architecture, large language model system and accelerator design, and memory architecture based on advanced packaging.

王浩然, 1997 年生。博士。主要研究方向为计算机体系结构、大模型系统与加速器设计、基于先进封装的存储架构。



Xu Haobo, born in 1990. PhD, associate professor. Member of CCF. His main research interests include domain-specific processor chips and 2.5D/3D chiplet-based integrated chips.

许浩博, 1990 年生。博士, 副研究员。CCF 会员。主要研究方向为领域专用处理器芯片、2.5 维/3 维芯粒集成芯片。



Wang Ying, born in 1985. PhD, professor. Senior member of CCF. His main research interests include dedicated processor architecture and integrated chip systems.

王　颖, 1985 年生。博士, 研究员。CCF 高级会员。主要研究方向为专用处理器体系结构、集成芯片系统。



Han Yinhe, born in 1980. PhD, professor. Senior member of CCF. His main research interests include computer architecture and chips, and data computing chips and computing systems.

韩银和, 1980 年生。博士, 研究员。CCF 高级会员。主要研究方向为计算机体系结构和芯片、数据计算芯片和计算系统。