CD-PIM：面向边缘设备低批量大语言模型加速的高带宽、高计算效率LPDDR5存内处理架构

Ye Lin[1]，Chao Fang[1]，Xiaoyong Song[2],Qi Wu[1],Anying Jiang[1],Yichuan Bai[1],Li Du[1],†南京大学电子科学与工程学院，中国 [2]中国移动研究院，中国 电子邮箱： {yelin, fantasysee, qiwu, jay, yicbai}@smail.nju.edu.cn, songxiaoyong@chinamobile.com, ldu@nju.edu.cn

arXiv:2601.12298v1 [cs.AR] 18 Jan 2026

摘要—在边缘部署低批量大语言模型时，执行内存密集型通用矩阵-向量乘法（GEMV）运算会面临严重的内存带宽瓶颈。尽管数字存内处理（PIM）架构有望加速GEMV运算，但现有配备PIM的边缘设备仍存在三项主要局限：带宽提升有限、混合工作负载下组件利用率不足，以及计算单元（CUs）的计算能力较低。本文提出CD-PIM，通过三项创新应对这些挑战。首先，我们提出高带宽高计算效率模式（HBCEM），通过采用分段式全局位线将每个存储体划分为四个伪存储体，从而提升带宽。其次，我们提出低批量交错模式（LBIM），通过将GEMV运算与GEMM运算重叠执行来提高组件利用率。第三，我们设计了一种高计算效率计算单元，通过将权重数据依次送入计算核心，以流水线方式执行增强型GEMV运算。第四，我们对键缓存矩阵采用按列映射，对值缓存矩阵采用按行映射，从而充分利用计算单元资源。评估结果表明，与仅使用GPU的基准及当前最先进的PIM设计相比，在HBCEM模式下，CD-PIM在单批次场景中的平均加速比分别达到11.42×和4.25× 。此外，在低批量规模下，与HBCEM相比，CD-PIM在LBIM模式下的平均加速比为1.12× 。

I. 引言

基于 Transformer 的大语言模型（LLM） [1] 在问答 [2] 和代码补全 [3]等各类应用中展现出了卓越的能力。与处理需要高计算能力的大批次请求的云服务 [4]，相比，端侧大语言模型推理 [5] 通常处理低批次请求，并需要高带宽，以在实时应用 [6]中实现低延迟交互。

大语言模型的推理过程通常分为两个阶段：预填充阶段和解码阶段 [7]。在预填充阶段，大语言模型会一步处理整个输入词元序列，主要依赖计算密集型的通用矩阵-矩阵乘法（GEMM）操作 [8]。相比之下，解码阶段以自回归方式一次生成一个词元，这依赖于内存密集型通用矩阵-向量乘法（GEMV）操作 [9]。然而，边缘设备有限的内存带宽会导致解码延迟增加 [10]。

为缓解解码阶段的内存带宽瓶颈，数字存内处理（PIM）架构正日益被用作硬件加速器。根据计算单元（CU）的布局，PIM 设计可分为子阵列级 [11]、存储体级 [12]、存储体组（BG）级 [13]和输入/输出级 [14]架构。其中，存储体级 PIM 通过利用存储体级并行性，在面积成本与内部内存带宽之间实现了良好权衡，其计算单元集成在内存存储体内 [15]。这种方法在近期发布的量产级 PIM 设备中十分常见 [16]。在此类架构中，配备存内处理的处理器由主机处理器执行通用矩阵-矩阵乘法运算，并将通用矩阵-向量乘法运算卸载至 PIM，从而提升大语言模型推理 [17]–[20]的整体性能。

为实现高效的端侧推理，当前存内处理设计主要面临三项挑战：(1) 先前的工作 AttAcc [13] 利用大量高带宽内存（HBM）存储体，为云服务实现了 242 TB/s 的内部带宽。然而，边缘设备的内存存储体数量要少得多，因此现有存内处理架构对于在边缘端高效加速大语言模型推理至关重要。资源受限使端侧推理难以满足严格的低延迟要求。此外，现有存内处理架构使用计算能力相对较低的计算单元执行通用矩阵-向量乘法运算，因为其内部内存时钟频率低于所支持的最高数字时钟频率 [15]，且每个存储体仅集成一个计算单元来执行通用矩阵-向量乘法运算 [21]。(2) 当前存内处理架构采用阻塞模式，即处理器和存内处理无法同时执行，从而限制了存内处理的效率。尤其对于输入序列较长、输出序列较短的计算密集型工作负载，处理器上的预填充延迟会成为端到端推理延迟的主要因素。一个直接的解决方案是将存内处理用作配备专用内存的独立加速器，从而使存内处理执行通用矩阵-向量乘法运算的同时，处理器执行通用矩阵-矩阵乘法运算 [22] [23]。然而，由于内存资源受限，这种方案不适用于边缘设备。因此，必须将存内处理集成到现有内存系统中，以支持并发

†通讯作者。本研究部分受中国国家重点研发计划项目（项目编号：2022YFB4400900）、江苏省战略性新兴产业和关键技术项目（项目编号：BE2023020-3）、江苏省基础研究计划（项目编号：BK20243042）以及南京大学—中国移动通信集团有限公司联合研究院资助。

执行通用矩阵-向量乘法和通用矩阵-矩阵乘法；(3) 在注意力层中，键（K）缓存矩阵和值（V）缓存矩阵采用固定映射，由于计算流程不同，会导致计算单元资源利用不均衡 [13]。具体而言，如果按行划分 K缓存矩阵 (H_{dim}, L)，其中 H_{dim} 表示嵌入维度，L 表示从预填充阶段到当前解码步累计的词元数量，那么追加的 (H_{dim}, 1) 列向量仅由一个计算单元处理，导致存算一体计算单元的利用率不均衡。然而，对于 V缓存矩阵，按行进行数据映射可以充分利用计算单元资源。

为应对这一挑战，我们提出了跨分区存算一体（CD-PIM），这是一种高带宽、高计算效率的存算一体加速器，专为边缘设备上的低批次大语言模型推理而优化。CD-PIM架构将每个内存存储体划分为四个伪存储体（Pbanks），只需对布线和外围电路进行少量改动，便可提升内部带宽。此外，它能够无缝集成到现有存储系统中，并支持两种工作模式：高带宽计算效率模式（HBCEM）和低批量交错模式（LBIM）。在HBCEM下，CD-PIM通过同时激活四个P存储体，最大化内部内存带宽，以加速通用矩阵-向量乘法运算；而在LBIM下，它将通用矩阵-向量乘法延迟与通用矩阵-矩阵乘法延迟重叠，从而降低端到端推理延迟。此外，我们还设计了一种高计算效率计算单元，以流水线方式执行通用矩阵-向量乘法运算。

我们的主要贡献如下：

- 针对挑战（1），我们提出了一种基于LPDDR5的高带宽、高计算效率CD-PIM架构，可在HBCEM下充分加速通用矩阵-向量乘法运算，适用于输入序列较短、输出序列较长的内存密集型工作负载。CD-PIM通过对全局位线（GBL）进行分段，将每个存储体划分为四个P存储体，从而提升内存带宽 4 × 。我们设计了一种轻量级高计算效率计算单元，通过将独立缓冲区中的权重数据依次馈入计算核心，以流水线方式高效执行通用矩阵-向量乘法运算，从而提升计算能力。（第III-A节）

- 针对挑战（2），我们通过为GEMV运算启用两个P存储体，并为GEMM运算启用另外两个P存储体，使存内处理中的GEMV延迟与处理器中的GEMM延迟重叠，从而降低计算密集型工作负载在LBIM中的端到端推理延迟，并实现存内处理与处理器的同时执行。（第III-B节）

- 针对挑战（3），我们对K缓存矩阵采用按列映射，对V缓存矩阵采用按行映射，通过内积和外积运算充分利用计算单元资源。（第III-C节）

我们使用 NVIDIA Jetson AGX Orin 64GB [24] 和 Apple iPhone 15 Pro[25]，在计算密集型和内存密集型工作负载上评估所提出的架构。评估结果表明，与仅使用 GPU 的基准和最先进的存算一体设计相比，CD-PIM 在 HBCEM 中分别实现了平均 11.42× 和 4.25× 的加速比。对于批量较小的计算密集型工作负载，与 HBCEM 相比，CD-PIM 在 LBIM 中实现了平均 1.12× 的加速比。

II. 背景

A. 基于 *Transformer* 的大语言模型推理

基于 Transformer 的大语言模型由多个级联的 Transformer 解码器层组成 [26]。对于每个请求，推理从预填充阶段开始，在该阶段中，输入序列中的多个词元并行处理，以生成第一个输出词元 [27]。随后，解码阶段迭代生成后续词元，每次都将先前生成的词元作为输入。此过程持续进行，直至生成完整的输出序列。由于这种迭代特性，解码阶段中每个解码器层的输入都是单个向量，因此会产生访存密集型的通用矩阵-向量乘法运算；这与预填充阶段执行的通用矩阵-矩阵乘法运算形成对比。由于每个请求仅执行一次预填充阶段，而解码阶段则必须重复数百次，因此解码延迟主导了端到端推理延迟。

B. 银行级 *PIM* 加速器

银行级 PIM 大语言模型加速器通常将计算单元置于存储体内，以利用比外部内存接口更高的内部内存带宽。为克服器件上带宽提升有限的问题，FOLD-PIM [5]将全局位线分为上、下两段，并采用一个以两倍内部内存时钟频率运行的计算单元。类似地，Pipe-PIM [15]将全局位线分为左、右两段，并采用两个计算单元，每个计算单元均以内部内存时钟频率运行。尽管这些存内处理架构能够高效加速GEMV运算，但它们以阻塞模式运行，因此限制了存内处理利用率。

数字 PIM 集成了用于执行GEMV运算的计算单元，以及用于在存储体边缘存储和复用输入向量与部分和的向量缓冲区。在存储体级数字 PIM 中，计算单元位于存储体区域内。然而，向量缓冲区可以位于存储体区域 [5],、BG 区域 [13], 或 I/O 区域 [14]。向量缓冲区距离计算单元越远，数据传输能耗越高。为尽量降低传输能耗，本研究将向量缓冲区置于计算单元旁边。

端侧大语言模型推理服务通常具有两个关键特征：同时处理的请求数量较少，且要求交互延迟低。例如，AIOS [28]被提出作为一种嵌入大语言模型的管理系统，支持六种接口。然而，当前存内处理架构采用阻塞模式，无法让存内处理与处理器计算并发执行 [29]。这种串行化方式必然导致资源利用不足。尤其是在低批量场景下，阻塞式计算模式会导致推理延迟增加，从而限制存内处理在边缘设备中的应用。尽管先前的工作 [30] 通过双行缓冲区组实现了存内处理与NPU协同处理，但这种方法会带来相当大的面积开销。

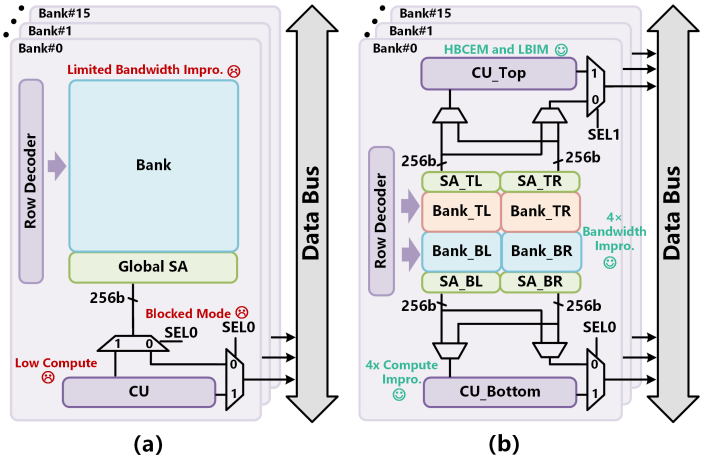

图 1. (a) 采用单个计算单元的传统存内处理架构；(b) 采用两个计算单元执行通用矩阵-向量乘法的 CD-PIM 架构。

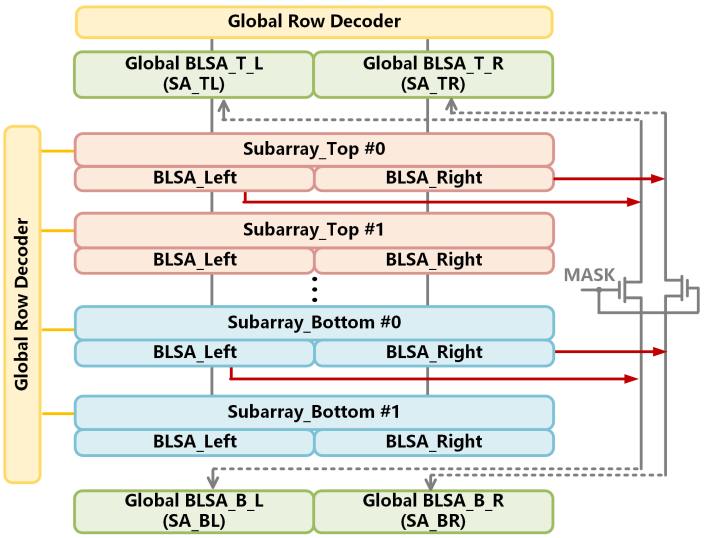

图 2. CD-PIM 架构的细节：通过分离全局位线和隔离晶体管，将每个存储体划分为四个 Pbank。

与面向云端设备的基于高带宽内存的存内处理相比，后者每个秩集成四个裸片、每个裸片包含128个以上的存储体；LPDDR5存内计算仅提供数量有限的存储体，限制了用于提升内部内存带宽的存储体级并行性。因此，扩展内部内存带宽对于高效加速边缘设备上的GEMV运算至关重要。此外，变化的工作负载分布 [31] 以及边缘设备之间计算资源和内存资源的差异 [32] ，在很大程度上决定了首令牌生成时间（TTFT）在整体推理延迟中的占比。尤其是对于输入序列较长、输出序列较短的计算密集型工作负载，受限的计算资源会使首令牌生成时间主导端到端推理延迟。

在传统的动态随机存取存储器芯片中，由于采用突发传输方案，外部内存接口的带宽高于内部时钟频率 [33]。然而，以往的存储体级

表 I 两种边缘设备规格

设备	DRAM	吞吐量 (TFLOPS)	带宽 (GB/s)
Jetson AGX Orin	LPDDR5	42.5	204.8
iPhone 15 Pro	LPDDR5	4.29	51.2

数字PIM设计 [34]以内部时钟频率执行GEMV运算，从而限制了计算单元的计算能力。这种降低的计算能力使此类存内处理架构无法充分加速低批量LLM推理。

III. CD-PIM：高带宽、高计算效率的PIM架构

本节介绍CD-PIM，这是一种高带宽、高计算效率的PIM架构，通过利用存储体级并行性，以较低的实现成本加速GEMV运算。CD-PIM利用其高内存带宽，加速HBCEM中的解码阶段。此外，为支持低批量、计算密集型工作负载，配备CD-PIM的处理器会同时运行CD-PIM以执行LBIM中的解码阶段，并由处理器执行预填充阶段，从而重叠解码延迟。值得注意的是，本文中的输入和权重数据均采用8位精度表示，不会导致大语言模型推理准确率出现明显下降。此外，我们还引入了三条新指令来支持PIM操作。

A. 所提出的*CD-PIM*

与传统DRAM-PIM架构相比，如图1(a)所示，CD-PIM将额外的计算单元直接集成到每个存储体中。图1(b)展示了CD-PIM的整体架构，其中包含两个用于加速GEMV运算的计算单元。顶部和底部计算单元通过左、右全局BLSA（SA TL、SA TR、SA BL、SA BR）从存储体接收权重数据，并以流水线方式执行INT8乘加（MAC）运算。每个计算单元每个计算周期可处理32 B的数据宽度。为增强计算能力，计算单元设计为以400 MHz运行，是LPDDR5 [5]内部时钟频率的两倍。因此，计算单元可在单个内存周期内，对来自左侧SA和右侧SA的（64,1）向量数据进行流水线处理。每个计算单元都配备一个64 B输入缓冲区，用于存储输入向量；以及一个128 B输出缓冲区，用于存储部分和。通过在每个存储体中采用两个流水线计算单元，CD-PIM相较于配备相同DRAM芯片的DH-PIM，提供了 2× 计算优势 [34]。此外，得益于针对K缓存和V缓存矩阵的专用映射策略，计算单元能够支持内积和外积运算。

图2展示了CD-PIM架构的细节。首先，通过将全局位线（GBL）、位线感应放大器（BLSA）和全局BLSA分别均分为左侧GBL/右侧GBL、左侧BLSA/BLSA-right以及全局BLSA-

表 II 命令选择表

命令描述	SEL0	SEL1
PIM_MAC_FM	1	1
MACT_LDB	0	1
MACB_LDT	1	0

left/全局BLSA-right，将每个存储体划分为左存储体和右存储体。由于这种划分，列译码器可以同时访问左存储体和右存储体。此外，每个左侧GBL和右侧GBL还通过隔离晶体管进一步划分为上侧GBL和下侧GBL，从而将左存储体和右存储体分别划分为上存储体和下存储体。行译码器可以同时访问这些上、下存储体。因此，CD-PIM将每个存储体划分为四个P存储体（Bank TL、Bank TR、Bank BL、Bank BR），所有存储体均可同时激活以执行GEMV运算，因此与 FOLD-PIM相比具有更高的内存带宽 [5]。

B. HBCEM 和 LBIM

表 II 总结了通过控制 SEL0 和 SEL1 信号实现的三条 PIM指令。PIM MAC FM 指令同时激活四个 P存储体，以充分加速通用矩阵-向量乘法运算。MACT LDB 指令激活顶部计算单元以执行通用矩阵-向量乘法运算，同时处理器通过访问底部存储体中的数据来执行通用矩阵-矩阵乘法运算。类似地，MACB LDT 指令激活底部计算单元以执行通用矩阵-向量乘法运算，同时访问顶部存储体中的数据。因此，通过同时在存内处理和处理器中分别执行通用矩阵-向量乘法和通用矩阵-矩阵乘法，通用矩阵-向量乘法延迟与通用矩阵-矩阵乘法延迟得以重叠。

对于传统边缘设备，大语言模型推理的预填充和解码阶段均在处理器上执行。然而，对于内存密集型工作负载，有限的内存带宽会导致显著的推理延迟，如图 4(a) 所示。对于在 NVIDIA Jetson AGX Orin 64 GB 上运行的 LLAMA-1B，当 (Lin, Lout) = (128, 2048) 时，端到端推理延迟达到 35.7s，而处理器计算利用率为 85%。为加速解码阶段，我们在 HBCEM 中将解码阶段卸载到存内处理，激活四个 P存储体，并使用 PIM_MAC_FM 指令充分加速通用矩阵-向量乘法运算，如图 4(b) 所示。借助存内处理的高内部内存带宽，与处理器执行相比，解码阶段延迟降低了 90.2%，使总推理延迟降至 3.53s。

对于计算密集型工作负载，尽管存内处理加速了通用矩阵-向量乘法运算，但首令牌生成时间主导着端到端推理延迟；对于计算能力较低的边缘设备，这一问题尤为突出，如图4(b)所示。对于 LLAMA-13B 的 (输入长度, 输出长度) = (2048, 128)，将解码阶段卸载到存内处理会使首令牌生成时间在总延迟中的占比从 NVIDIA Jetson AGX Orin 64 GB 上的 53.7% 增至 Apple iPhone 15 Pro 上的 65.2%。为解决这一问题，CD-PIM 使用 MACT = LDB 和 MACBLDT 指令并发执行通用矩阵-向量乘法和通用矩阵-矩阵乘法运算：激活两个 P存储体执行通用矩阵-向量乘法运算，并激活另外两个 P存储体执行

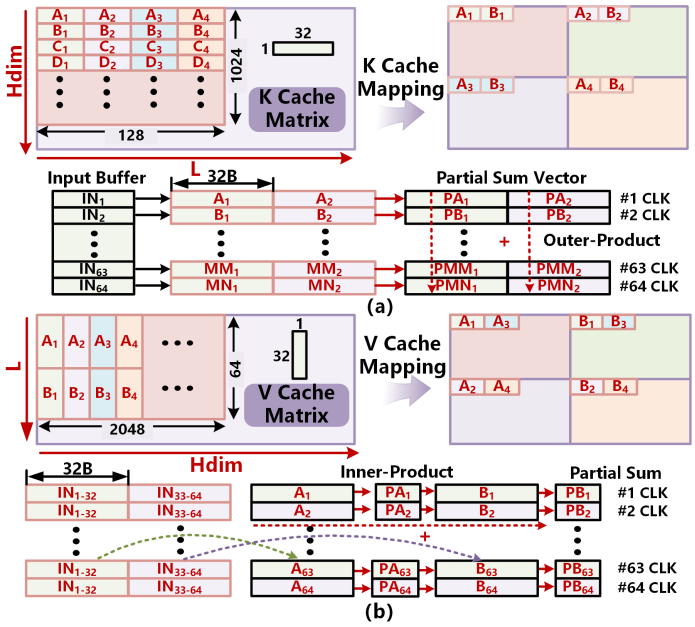

图 3. CD-PIM 的详细数据映射：(a) K缓存矩阵数据映射与外积计算流程；(b) V缓存矩阵数据映射与内积计算流程。

LBIM 中的通用矩阵-矩阵乘法运算。尽管与 HBCEM 相比，LBIM 中的存内处理计算能力减半，但通用矩阵-向量乘法延迟可与通用矩阵-矩阵乘法延迟有效重叠，如图4(c)所示。

C. 数据映射

在解码阶段，计算流程由 QKV生成层、注意力层和前馈网络（FFN）组成。首先，将输入词元送入 QKV生成层，以生成尺寸为 (1，H_{dim}) 的查询（Q）、K 和 V 向量。新生成的 K 和 V 向量分别与 K缓存矩阵（H_{dim}，L）和 V缓存矩阵（L，H_{dim}）拼接。随后，将 Q 向量传入注意力层，与 K缓存矩阵执行一次通用矩阵-向量乘法。所得向量经 softmax函数归一化，生成注意力权重向量（1，L）。接着，将该权重向量与 V缓存矩阵相乘，计算得到注意力输出（1，H_{dim}）。最后，将注意力输出传入 FFN 层，以生成下一个词元。

在注意力层中，会根据计算流程调整KV缓存矩阵的数据映射。**对于按列进行的K缓存矩阵映射，每个数据块对应一个大小为(1 × 32)的向量，Pbank可通过一次突发操作访问该向量**。在存储体级别，每个存储体都能够访问组织为(2 × 64)数据块的矩阵。在GEMV运算之前，Q向量被加载到CU输入缓冲区中。Q向量被划分为大小为(1 × 1024)的子向量，并依次传输到LPDDR5裸片。每个(1 × 1024)子向量进一步划分为十六个(1 × 64)子向量，分别对应LPDDR5器件的16个存储体。在

每个存储体内，一个(1×64)子向量会广播到两个计算单元的输入缓冲区中。

在 GEMV运算期间，每个计算单元都会对存储在输入缓冲区中的 1B 输入数据与一个（1 × 32B）权重向量执行外积，生成一个（1 × 32）部分和向量。随后，同一输入数据与另一个（1 × 32B）权重向量相乘，以获得新的（1 × 32）部分和向量。因此，在单个内部存储器时钟周期内，计算单元将 1B 输入数据与（1 × 64B）权重数据相乘，从而生成一个（1 × 64）部分和向量。在下一个周期中，将生成的（1 × 64）部分和向量与前一步产生的结果累加，如图3(a)所示。例如，输入数据 IN_1 与权重向量 A_1 相乘，生成部分和向量 PA_1。同一输入数据 IN_1 也与权重向量 A_2,相乘，得到部分和向量 PA_2。接着，输入数据 IN_2 与权重向量 B_1 相乘，并与 PA_1 相加，生成更新后的部分和向量 PB_1。类似地，IN_2 与权重向量 B_2 相乘，并与 PA_2，累加，生成更新后的部分和向量 PB_2。按照这一计算流程，每个存储体都能在单个存储器内部时钟周期内执行 ((1, 1) × (1, 128)) GEMV运算。对于每个（1 × 1024）输入向量段，输入缓冲区容量支持每个存储体处理一个（64 × 128）矩阵。在裸片级，可以处理一个（1024 × 128）矩阵。随后，下一个（1 × 1024）输入向量段被传输至DRAM芯片，从而实现连续计算。

对于按行的V缓存矩阵映射，与K缓存矩阵相比，可归纳出三个主要差异，如图3(b)所示。首先，每个数据块对应一个(32 × 1)向量，每个存储体可访问一个以(64 × 2)数据块组织的矩阵，而K缓存矩阵访问的是(2 × 64)数据块。其次，注意力向量(1,L)被划分为多个(1 × 64)子向量。每个子向量都会广播到计算单元的所有输入缓冲区。第三，在GEMV运算期间，每个计算单元计算一个(1 × 32)注意力向量IN_{1-32}与一个(32 × 1)权重向量A_1 的内积，以生成部分和PA_1。随后，将后续的(1 × 32B)注意力数据IN_{33-64} 与一个(32 × 1B)权重向量B_1 相乘，并与PA_1 相加，得到新的部分和PB_1。按照这一计算流程，每个存储体可在单个内部时钟周期内执行一次((1,64) × (64,2)) GEMV运算。对于每个(1 × 64)注意力向量，输出缓冲区容量足以支持在一个存储体内处理一个(64 × 128)矩阵；在裸片级，则可处理一个(64 × 2048)矩阵。随后，下一个(1 × 64)注意力向量将传输至DRAM芯片，以持续进行计算。

IV. 评估

A. 方法

我们通过修改开源动态随机存取存储器模拟器 Ramulator 2 [35], , 在 4GB LPDDR5 的基础上实现了 CD-PIM。每个 LPDDR5 裸片提供 16 个数据引脚，每个引脚的运行速率为 6.4 Gbps。在我们的实验设置中，我们采用 Apple iPhone15 Pro 和 NVIDIA Jetson AGX Orin 作为

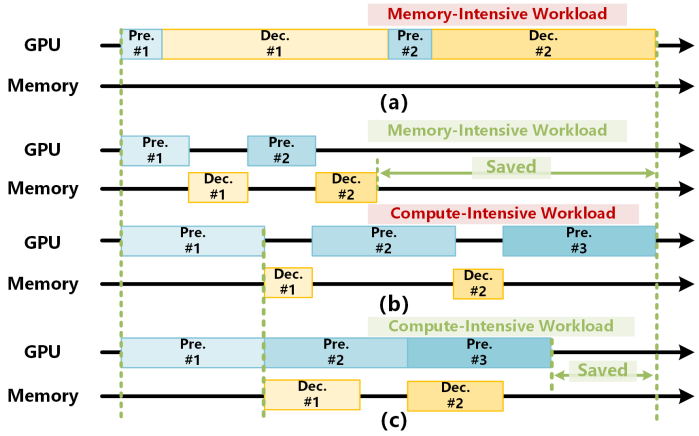

图 4. 大语言模型推理的时序图：(a) 运行内存密集型工作负载的 GPU；(b) 在 HBCEM 中配备 CD-PIM 、运行内存密集型和计算密集型工作负载的 GPU ；(c) 在 LBIM 中配备 CD-PIM、运行计算密集型工作负载的 GPU。

边缘设备来评估 CD-PIM 的性能，这两种设备分别配备 4 个和 16 个 LPDDR5 裸片，提供的总内存容量分别为 16 GB 和 64 GB。在性能评估中，我们将 CD-PIM 与仅使用 GPU 的基准以及 Apple iPhone 15 Pro 和 NVIDIA Jetson AGX Orin 上的 AttAcc [13]基准执行进行比较。此外，我们还评估了 CD-PIM 在批量大小为 1 和 4 时的表现，并针对不同的大语言模型参数考察了 HBCEM 和 LBIM。

B. 性能结果

图 5 展示了单批次场景下 GPU 和 AttAcc 相对于 HBCEM 与 LBIM 中 CD-PIM 的归一化性能评估。在 NVIDIA Jetson AGX Orin 上，与仅使用 GPU 的基准方案 [36] 相比，HBCEM 中的 CD-PIM 在 LLAMA-1B 、LLAMA- 7B 和 LLAMA-13B 上分别实现了 4.48×–10.51× 、6.71×–13.74× 和 7.47×–14.6× 的加速比。这些提升源于 PIM 相较于 GPU 外部内存带宽所具备的更高内部内存带宽。此外，随着模型规模从 LLAMA-1B 增至 LLAMA-13B，解码阶段的延迟会更高。借助 PIM 加速通用矩阵-向量乘法运算，性能提升会更加显著。

对于苹果 iPhone 15 Pro，HBCEM 中的 CD-PIM 相较于 NVIDIA Jetson AGX Orin 实现了更高的加速比。由于 iPhone 的内存带宽低于 Jetson，解码阶段的延迟更长，导致工作负载更加受内存限制。因此，使用存内处理加速通用矩阵-向量乘法，能在 iPhone 上带来更大的性能提升。尤其是在 (Lin, Lout) = (128, 2048) 这一内存密集型场景中，LLAMA-1B 在 iPhone 15 Pro 上的加速比从 Jetson AGX Orin 上的 10.1× 提升至 18.6× 。

然而，由于计算能力有限，与内存密集型工作负载相比，HBCEM 中的 CD-PIM 在计算密集型工作负载上的效率较低。图 6 展示了在四批次场景下，NVIDIA Jetson AGX

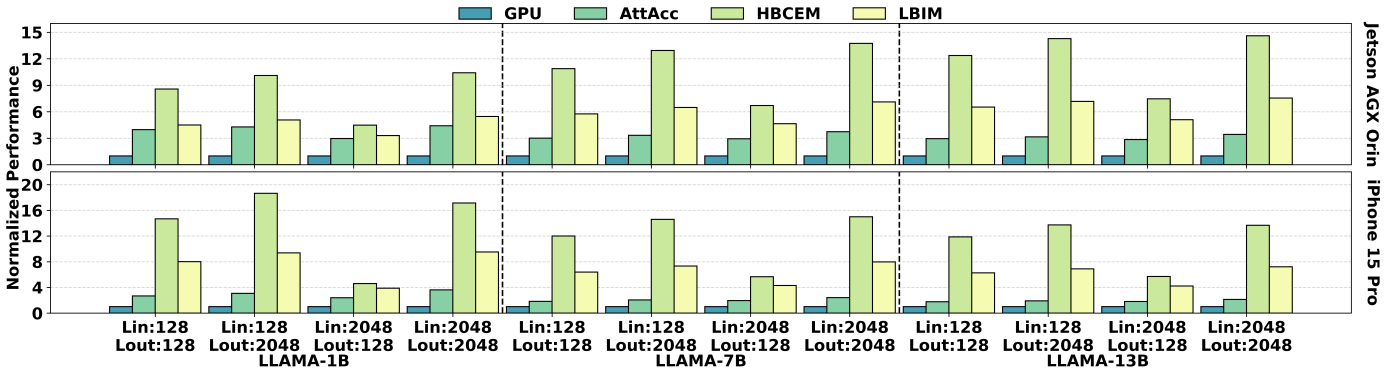

图 5. NVIDIA Jetson AGX Orin 和 Apple iPhone15Pro 上，LLaMA-1B、-7B 和 -13B 在不同输入长度和输出长度下的归一化性能（批量大小 = 1）。

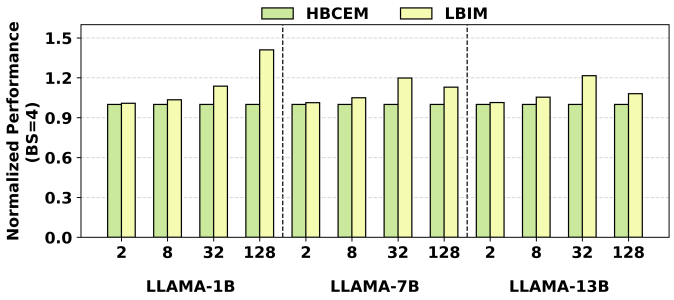

图 6. 配备 CD-PIM 的 NVIDIA Jetson AGX Orin 在 HBCEM 和 LBIM 下的归一化性能（批量大小 = 4）。

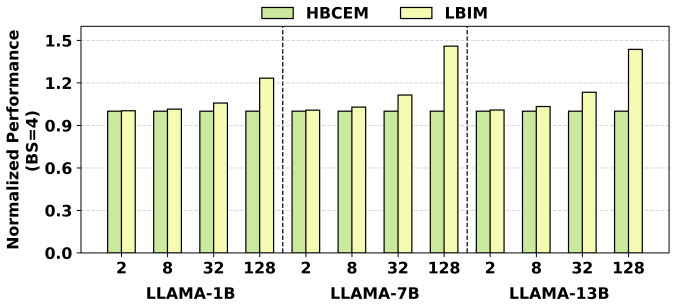

图 7. 配备 CD-PIM 的 Apple iPhone 15 Pro 在 HBCEM 和 LBIM 下的归一化性能（批量大小 = 4）。

Orin 平台上 HBCEM 和 LBIM 中 CD-PIM 的性能评估，其中输入长度固定为 2048。对于 LLAMA-1B，当输出长度从 2 增加到 128 时，LBIM 中的 CD-PIM 通过重叠解码阶段的延迟，相较于 HBCEM 基线实现了 1.01×–1.41× 的加速比。相比之下，对于 LLAMA-7B 和 LLAMA-13B，LBIM 的性能加速比最初随输出长度从 2 增至 32 而提升，但随后开始下降，因为当输出长度为 = 128 时，解码阶段的延迟超过了首令牌生成时间延迟。

对于 Apple iPhone 15 Pro，在 LLAMA-1B 上，LBIM 中的 CD-PIM 相较于 HBCEM 的加速比为 1.01×–1.23× ，低于在 NVIDIA Jetson AGX Orin 上取得的性能加速比，因为对于计算密集型工作负载，计算能力降低所造成的影响大于内存带宽降低所造成的影响。因此，与 Jetson AGX Orin 相比，首令牌生成时间延迟对 iPhone 15 Pro 端到端推理延迟的贡献更大。对于 LLAMA-7B 和 LLAMA-13B，LBIM 中的 CD-PIM 所取得的加速比为

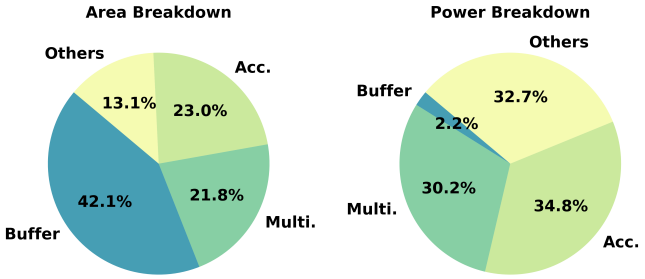

图 8. 计算单元的面积和功耗细分。

当输出长度从 2 到 128 时，相较于 HBCEM 分别为 1.01×–1.46× 和 1.01×–1.44×，这是因为解码阶段的延迟上升至总推理延迟的近一半，如图 7 所示。

C. 面积和功耗开销

与仅支持内积的计算单元相比，所提出的计算单元同时支持内积和外积。图 8 展示了计算单元的面积和功耗细分。我们使用 Synopsys Design Compiler 评估计算单元的面积和功耗，结果表明，每个处理单元占用 14，941 μm^2，在台积电 28 纳米工艺下功耗为 4。5 mW。这相当于 32 Gb LPDDR5 芯片面积的 0。8%，总计增加 144 mW，证实了 CD-PIM 的成本效益。

V. 结论

本文提出了 CD-PIM，以解决内存密集型工作负载中 GEMV 加速效率低下，以及计算密集型工作负载中首令牌生成时间延迟较长的问题。CD-PIM 将每个存储体划分为四个 P存储体，以提高内部内存带宽，并配备两个计算单元，以实现较高的计算能力。对于内存密集型工作负载，HBCEM 中的 CD-PIM 可全面加速 GEMV 运算；对于计算密集型工作负载，LBIM 中的 CD-PIM 通过同时使用 CD-PIM 执行解码阶段、使用处理器执行预填充阶段，来隐藏解码延迟。本文设计了一个高计算能力的计算单元，以流水线方式执行 GEMV。我们还分析了 CD-PIM 的面积和功耗开销，结果表明，所提出的架构仅需极低的硬件成本，即可显著提升性能。

参考文献

[1] C. Li 等，“Specpim：通过架构-数据流协同探索加速存内处理系统上的推测推理”，载于第 29 届美国计算机协会编程语言与操作系统架构支持国际会议（ASPLOS）论文集，第 3 卷，2024年，950 965页。–。[2] S. Zhang 等，“Opt：开放的预训练 Transformer 语言模型”，2022 [ ] 2205 01068 。在线。网址：[3] https://arxiv.org/abs/。E. Nijkamp 等，“Codegen：支持多轮程序合成的开放式代码大语言模型”，2023年。 [在线]。网址：https://arxiv.org/abs/2203 13474 。L. Liu 等，“让所有人都能负担得起大语[4] 言模型推理：通过 NDP-DIMM 扩展 GPU 内存”，载于 IEEE 高性能计算机体系结构国际研讨会（HPCA），2025年，第 1751–1765 页。[5] K. Jeun 等，“Fold-pim：一种面向设备端小语言模型、成本高效的 LPDDR5 存内处理方案”，《IEEE 计算机体系结构快报》（CAL），第 24 卷，第 1 期，第 185–188 页，2025年。[6] Z. Zhou 等，“Dimm-link：实现高效 DIMM 间通信的近存处理方案”，载于 IEEE 高性能计算机体系结构国际研讨会（HPCA），2023年，第 302–316 页。[7] V. Pulavarthi 等，“大语言模型是否已准备好实际用于断言生成？”，载于欧洲设计、自动化与测试会议（DATE），2025年，第 1–7 页。[8] R. Qin 等，“Nvcim-pt：一种由 NVCIM 辅助的边缘大语言模型提示调优框架”，载于欧洲设计、自动化与测试会议（DATE），2025年，第 1–7 页。[9] Y. Hou 等，“Nora：模拟存内计算加速器上大语言模型的噪声优化重缩放”，载于 2025年欧洲设计、自动化与测试会议（DATE），2025年，第 1–7 页。[10] F. Liu 等，“Aster：基于马尔可夫决策过程，通过自适应动态跳层实现高效 Transformer 推理”，载于第 33 届美国计算机协会多媒体国际会议（MM）论文集，2025年，第 11 853–11 861 页。[11] Q. Deng 等，“Dracc：一种用于精确卷积神经网络推理的动态随机存取存储器加速器”，载于第 55 届美国计算机协会/欧洲设计与自动化协会/电气电子工程师学会设计自动化会议（DAC），2018年，第 1–6 页。[12] S. Cho 等，“Mcdram v2：动态随机存取存储器内脉动阵列加速器，旨在解决边缘深度神经网络中的大型模型问题”，《IEEE Access》，第 8 卷，第 135 223–135 243 页，2020年。[13] J. Park 等，“Attacc!：释放存内处理在批量 Transformer 生成模型推理中的强大能力”，载于第 29 届美国计算机协会编程语言与操作系统架构支持国际会议（ASPLOS）论文集，第 2 卷，2024年，第 103–119 页。[14] S.-S. Park 等，“一种基于 LPDDR 的 CXL-PNM 平台，用于高性价比地推理基于 Transformer 的大语言模型”，载于 IEEE 高性能计算机体系结构国际研讨会（HPCA），2024年，第 970–982 页。[15] T. Jeong 等，“Pipepim：通过流水线化和双缓冲，最大限度提高面向机器学习的数字 PIM 中的计算单元利用率”，《IEEE 集成电路与系统计算机辅助设计汇刊》（TCAD），第 43 卷，第 12 期，第 4585–4598 页，2024年。[16] S. Lee 等，“一种基于 1ynm 1.25v 8gb、16gb/s/pin GDDR6 的存内加速器，支持 1tflops 乘加运算及多种深度学习应用激活函数”，载于 IEEE 国际固态电路会议（ISSCC），第 65 卷，2022年，第 1–3 页。[17] L. Wu 等，“Pimoe：通过感知节流的任务卸载，在 NPU-PIM 系统上实现高效混合专家 Transformer 部署”，载于 2025年第 62 届美国计算机协会/电气电子工程师学会设计自动化会议（DAC）。电气电子工程师学会，2025年，第 1–7 页。[18] Y. Hu 等，“Plain：通过混合精度量化，利用 PIM 的高内部带宽加速大语言模型推理”，载于 2025年 IEEE/ACM 国际计算机辅助设计会议（ICCAD）。电气电子工程师学会，2025年，第 1–9 页。[19] Y. Chen 等，“P3-llm：一种使用混合数值格式的大语言模型推理集成式 NPU-PIM 加速器”，arXiv 预印本 arXiv:2511 06838，2025年。。[20] Y. Zhao 等，“Um-pim：具有统一共享内存空间的动态随机存取存储器存内处理”，载于 2024年 ACM/IEEE 第 51 届年度计算机体系结构国际研讨会（ISCA）。电气电子工程师学会，2024年，第 644–659 页。[21] H. Lee 等，“Paise：面向基于 Transformer 的大语言模型的存内处理加速推理调度引擎”，载于 IEEE 高性能计算机体系结构国际研讨会（HPCA），2025年，第 1707–1719 页。[22] Y. Kwon 等，“基于 SK 海力士领域专用内存的以内存为中心的计算”，载于 *IEEE Hot Chips 35* 研讨会 *(HCS)*，2023年，第 1–26 页。[23] M. Seo 等，“Ianus：基于 NPU-PIM 统一内存系统的集成加速器”，载于 第 *29* 届美国计算机协会编程语言与操作系统架构支持国际会议论文集（*ASPLOS*），第 *3* 卷，2024年，第 545–560 页。[24] 英伟达，“Nvidia Jetson AGX Orin 系列数据手册”，https://www.diamondsystems.com/files/binaries/Jetson%20AGX%20Orin%20DS10662-001%20v1.2.pdf，2022年，访问日期：2025年9月14日。[25] R. Smith，“苹果发布 M3 SoC 系列：M3、M3 Pro 和 M3 Max 各展风采”，https://www.anandtech.com/show/21116/appleannounces-m3-soc-family-m3-m3-pro-and-m3-max-make-their-marks，2023年，访问日期：2025年9月14日。[26] C. Fang等，“Anda：通过可变长度分组激活数据格式释放高效的大语言模型推理能力”，载于 *IEEE* 高性能计算机体系结构国际研讨会（*HPCA*），2025年，第1467–1481页。[27] S. Ma 等，“Apt-llm：利用任意精度张量核心计算加速大语言模型”，*IEEE* 集成电路与系统计算机辅助设计汇刊（*TCAD*），2025年。[28] K. Mei 等，“AIOS：大语言模型智能体操作系统”，载于第二届语言建模会议（*COLM*），2025年。 [在线]。网址：https://openreview.net/forum?id=L4HHkCDz2x [29] J. Ahn 等，“支持存内处理的指令：一种低开销、具备局部性感知的存内处理架构”，载于美国计算机协会/电气电子工程师学会第*42*届计算机体系结构年度国际研讨会（*ISCA*），2015年，第336–348页。[30]G. Heo 等，“NeuPIMs：用于批量大语言模型推理的 NPU-PIM 异构加速”，载于第*29*届美国计算机协会编程语言与操作系统架构支持国际会议论文集（*ASPLOS*），第*3*卷，2024年，第722–737页。[31] C. Li 等，“H2-LLM：面向异构混合键合低批量大语言模型推理的硬件与数据流协同探索”，载于第*52*届计算机体系结构年度国际研讨会（*ISCA*）论文集，2025年，第194–210页。[32] S. H. Seo 等，“Facil：面向 SoC-PIM 协同端侧大语言模型推理的灵活 DRAM 地址映射”，载于*IEEE* 高性能计算机体系结构国际研讨会（*HPCA*），2025年，第1720–1733页。[33] S.-Y. Cho 等，“一款采用 PAM3 优化的 TRX 均衡和 ZQ 校准的 16-Gb、37-Gb/s GDDR7 DRAM”，*IEEE* 固态电路期刊（*JSSC*），第60卷，第1期，第184–196页，2025年。[34] B. J. Kim 等，“DH-PIM：通过双半模式扩展，最大限度提高数字存内处理中的计算单元利用率”，*IEEE* 集成电路与系统计算机辅助设计汇刊（*TCAD*），2025年。[35] H. Luo 等，“Ramulator 2.0：现代化、模块化且可扩展的 DRAM 模拟器”，*IEEE* 计算机体系结构快报（*RAL*），第23卷，第1期，第112–116页，2024年。[36] H. Touvron 等，“Llama：开放且高效的基础语言模型”，2023年。 [在线]。网址：https://arxiv.org/abs/2302.13971