FlashAttention 优化机制：深入解析“计算快于存储”原理下的显存访问优化

I. 引言：注意力机制的挑战与“计算与存储”范式

A. 无处不在的注意力机制及其扩展性问题

自注意力（Self-Attention）机制已成为 Transformer 模型的核心组成部分，广泛应用于自然语言处理、计算机视觉等多个领域 [1]。然而，其核心优势也伴随着一个显著的挑战：注意力计算的时间和空间复杂度均与输入序列长度 N 成二次方关系，即 O(N2) [5]。这意味着当序列长度翻倍时，运行时间和内存需求将增加四倍。随着模型处理日益增长的上下文长度的需求，这种二次方复杂性使自注意力机制成为长序列处理的主要瓶颈 [5]。

这种二次方复杂性并非仅是理论上的担忧，它直接导致了在处理长序列时产生过量的高带宽显存（High Bandwidth Memory, HBM）访问。具体而言，注意力机制涉及到 N×N 规模的计算，例如查询矩阵 Q 和键矩阵 K 的转置 KT 的乘积 QKT。当 N 非常大时，由此产生的中间矩阵（如 N×N 的注意力得分矩阵 S）会变得异常庞大，远超 GPU 核心旁快速片上静态随机存取存储器（Static Random-Access Memory, SRAM）的容量。因此，这些巨大的中间矩阵必须存储在相对较慢的 HBM 中，并在计算过程中反复读写。正是这些对 HBM 的大规模读写操作主导了执行时间，使得标准注意力机制成为一个典型的“内存带宽受限”（memory-bound）而非“计算受限”（compute-bound）的操作。

B. 现代 GPU 中“计算快于存储访问”的原理

现代图形处理器（GPU）的计算能力（以每秒浮点运算次数 FLOPs衡量）的增长速度已显著超过了显存带宽和延迟的改进速度 [5]。这一发展趋势导致了一个核心现象：在已加载到高速片上存储器（如 SRAM）的数据上执行算术运算，远比从相对较慢的片外 HBM 中获取数据要快得多。简而言之，“计算通常快于存储访问”。

这种性能差异是 FlashAttention 等优化技术利用的基础。如果存储访问速度与计算速度相当甚至更快，那么像 FlashAttention 中采用的重计算等策略可能就不会带来显著的性能提升 [5]。

C. 报告目标与结构

本报告旨在深入技术层面，详细阐释 FlashAttention 如何通过其核心的分块计算（Tiling）和重计算（Recomputation）策略，有效减少对 HBM 的访问次数，从而充分利用现代 GPU 中“计算快于存储”的特性。报告将依次剖析 GPU 的存储层级、标准注意力机制的瓶颈、FlashAttention 的具体实现机制，以及这些机制如何体现“计算快于存储”的原理，并最终分析其性能影响和深远意义。

“计算快于存储”的趋势并非注意力机制所独有，而是现代硬件发展的一个普遍特征。

FlashAttention 的成功因此为其他受限于 IO 的深度学习操作提供了一种重要的设计范式：通过重新设计算法使其具备 IO 感知能力，即便以增加部分计算量为代价，也要最大限度地减少对慢速存储的访问。GPU 架构师倾向于增加更多的计算单元，因为许多工作负载可以并行化。而显存带宽虽然也在增长，但受到物理和功耗的限制，其扩展速度往往不及计算能力。这种差距的日益扩大意味着，那些未针对内存访问模式进行优化的算法将越来越频繁地遭遇“内存墙”瓶颈。FlashAttention 的实践证明，通过重新思考算法以适应存储层次结构（甚至不惜重计算某些值），可以有效地保持强大的计算单元持续工作，从而释放硬件的全部潜能。这预示着深度学习中的其他操作也可以从类似的 IO 感知优化中受益。

II. 理解 GPU 存储体系与计算瓶颈

A. GPU 存储层级：SRAM、HBM 及 DRAM 概述

GPU 的性能在很大程度上依赖于其复杂的多层级存储系统。理解这些层级之间的特性差异对于认识 FlashAttention 等优化技术的原理至关重要。主要的存储层级包括：

- **SRAM (Static Random-Access Memory)**：位于 GPU芯片上，通常集成在每个流式多处理器（Streaming Multiprocessor, SM）中。SRAM 的访问速度极快，延迟极低（例如，数十个时钟周期），并提供极高的带宽（例如，NVIDIA A100 GPU 上的 SRAM 带宽可达 19 TB/s [9]，远高于 HBM [10]）。然而，其容量非常有限（例如，A100 上每个 SM 的 L1 缓存/共享内存为数十至数百KB，总 L2 缓存为数MB [8]）。SRAM 主要用作可编程的高速缓存或共享内存，供 SM 内的计算单元直接访问。
- **HBM (High Bandwidth Memory)**：通常与 GPU 核心封装在同一基板上，但物理上位于核心之外。HBM 的容量远大于 SRAM（例如，现代 GPU 可配备 8GB 至 80GB 甚至更多的 HBM [9]），带宽也非常高（例如，A100 的 HBM2e 带宽可达 1.5-2 TB/s [9]）。尽管 HBM 提供了巨大的数据吞吐能力，但其访问速度和延迟相较于 SRAM 仍然较慢（延迟可达数百个时钟周期）。HBM 是 GPU 的主显存（VRAM）。
- **DRAM (Dynamic Random-Access Memory)**：指系统主内存，位于 GPU 之外，通过 PCIe 等总线与 GPU 连接。DRAM 容量最大，但对于 GPU 而言，其访问带宽最低，延迟也最高 [9]。

FlashAttention 的核心策略在于最大化利用 SRAM 的高速特性，并最大限度地减少与 HBM 之间的数据传输。SRAM 与 HBM 之间在速度和带宽上存在的巨大差异（通常是一个数量级以上 [9]），使得这种优化策略切实有效。如果 HBM 的速度与 SRAM 相差无几，那么分块和重计算所带来的额外开销可能无法被节省的访存时间所抵消。

下表总结了 GPU 主要存储层级 SRAM 和 HBM 的关键特性对比：

表 **1: GPU** 存储层级比较 **(SRAM vs. HBM)**

特性	片上 **SRAM**	高带宽显存 **(HBM)**

通常位置	GPU 芯片上，每个流式多处理器 (SM) 内	GPU 基板上，所有 SM 共享
通常容量	小 (例如，每个 SM 数十至数百 KB，总 L2 缓存数 MB)	大 (例如，8GB - 80GB+)
通常带宽	极高 (例如，10-20+ TB/s 聚合带宽)	非常高 (例如，1-3+ TB/s)
通常延迟	非常低 (数十时钟周期)	低 (数百时钟周期)
关键角色	最快缓存，计算临时存储区	GPU 主显存 (VRAM)

此表清晰地展示了 SRAM 和 HBM 之间的核心差异，这对于理解 FlashAttention 策略的有效性至关重要。它直观地强调了 SRAM 提供巨大的速度优势但容量受限，而 HBM 提供容量但速度相对较慢。这直接支持了“计算快于存储”的论点，揭示了 FlashAttention 所解决的瓶颈正是由这些存储特性差异造成的。

B. 算术强度与内存墙

算术强度 **(Arithmetic Intensity)** 是衡量一个算法计算量与访存量之间比例的指标，定义为算法执行的算术操作次数（FLOPs）除以访问的内存字节数 [14]。这是一个与具体硬件无关的算法固有属性。

基于算术强度，可以将计算任务分为两类 [14]：

- 计算受限 **(Compute-bound)**：操作的执行时间主要由算术运算的数量决定，而访存时间占比较小。典型的例子是具有较大内积维度的矩阵乘法。
- 内存带宽受限 **(Memory-bound)**：操作的执行时间主要由内存访问的次数和速度决定，而计算时间占比较小。例如，逐元素操作（如激活函数、Dropout）、归约操作（如求和、Softmax），以及标准注意力机制（当序列长度 N 较大时）。

Roofline 模型 [11] 提供了一种直观的分析工具，它将一个计算核心（kernel）的算术强度与硬件的峰值计算性能和峰值内存带宽相关联，从而判断该核心是计算受限还是内存带宽受限。

标准注意力机制由于需要读写巨大的 N×N 注意力矩阵 S 以及其他中间张量到 HBM，其算术强度通常较低，因此很容易成为内存带宽受限的操作 [7]。例如，针对 Llama 2 7B 模型的分析显示，其注意力机制的算术强度远低于 GPU 的峰值 FLOPs/Byte 比率，表明其在自回归生成阶段是内存带宽受限的 [15]。这清晰地揭示了标准注意力机制缓慢的根源：并非计算单

元不够快，而是它们常常在等待数据从 HBM 中加载。

尽管 FlashAttention 解决了当前注意力机制面临的内存墙问题，但计算性能增长持续快于内存带宽提升的趋势 [5] 表明，类似的 IO 感知优化将在深度学习的各个领域变得越来越关键。今天看来是计算受限的操作，如果其设计没有充分考虑 IO 感知，未来在更新的硬件上也可能变为内存带宽受限。硬件供应商持续在 GPU 中集成更多的计算单元，而 HBM 带宽的提升虽然也在进行，但其增长速度往往不及理论峰值 FLOPs 的指数级增长。因此，硬件的 FLOPs/Byte 比率（即每传输一字节内存数据所能执行的计算量）趋于增加。一个具有固定算术强度的算法，在旧硬件上可能是计算受限的，但在新硬件上，如果其内存带宽需求相对于硬件提供的计算能力而言过高，就可能转变为内存带宽受限。这要求算法创新持续关注内存访问模式的优化，而不仅仅是减少 FLOPs。

此外，虽然 SRAM 速度极快，但其容量小是一个硬性约束 [8]。例如，仅注意力得分矩阵 S 就可能非常庞大，轻易超过 SRAM 的容量。这一限制使得像 FlashAttention 这样能够以小数据块（分块）方式处理数据的智能算法变得至关重要，而不是简单地试图将所有数据都塞进 SRAM。SRAM 单元比 DRAM/HBM 单元更大、更复杂，导致大容量 SRAM 缓存在芯片面积和功耗方面成本高昂 [10]。因此，GPU 只配备了有限的 SRAM 容量。对于长序列注意力这样的大规模问题，中间数据（如 N×N 的注意力矩阵）无法完全放入 SRAM。因此，仅仅“使用 SRAM”并非解决方案；算法必须被精心设计以高效利用有限的 SRAM，通过将数据切分成适合 SRAM 处理的块，这正是 FlashAttention 中分块策略的核心。

III. FlashAttention：一种 IO 感知的注意力机制重设计

A. 核心原则：最小化对慢速 HBM 的内存访问

FlashAttention 的核心目标是避免将庞大的中间 N×N 注意力矩阵（以及其他大型张量）读写到 HBM [5]。它是一种“IO 感知的精确注意力算法” [14]，即它不采用近似计算，而是计算出与标准注意力完全相同的结果，但通过更少的内存操作来实现 [6]。这一定位明确了 FlashAttention 是一种优化计算方式而非计算内容的技术。

B. 分块策略：将计算引入高速 SRAM

FlashAttention 的分块（Tiling）策略是其实现 HBM 访问优化的关键手段：

1. 输入划分：将输入的查询（Q）、键（K）和值（V）矩阵沿序列长度维度分割成较小的块（block）[2]。
2. 加载至 **SRAM**：算法通过嵌套循环处理这些分块。在外层循环中，K 和 V 矩阵的块从 HBM 加载到高速的片上 SRAM。在内层循环中，Q 矩阵的块也被加载到 SRAM [5]。这些块的大小经过精心选择，以确保它们以及计算过程中产生的中间结果能够完全容纳在有限的 SRAM 容量之内 [7]。
3. **SRAM** 内的分块计算：针对当前 SRAM 中的 Q、K、V 块，执行注意力计算的关键步骤（如 QKT、Softmax 和与 V 的乘积）完全在 SRAM 内部完成 [5]。

4. 在线 **Softmax (Online Softmax / Numerically Stable Softmax)**：这是 FlashAttention 的一项核心算法创新。标准 Softmax 需要在计算出所有 QKT 得分后才能进行归一化。然而，为了避免在 HBM 中实例化完整的 N×N 得分矩阵 S，FlashAttention 采用了一种“在线”或“增量式”的 Softmax 计算方法。当处理 Q、K、V 的分块时，它会为 Q 块中的每一行（对应一个查询向量）维护 Softmax 计算所需的运行统计量（当前块内得分的最大值以及指数和的累积值）[7]。每当处理一个新的 K 块时，这些统计量会被更新，并且之前计算的部分输出会根据新的统计量进行正确地重新缩放。这种方法巧妙地避免了在 HBM 甚至 SRAM 中完整存储 N×N 的 S 矩阵。相关的数学细节在多篇文献中有详细阐述，包括通过减去行最大值来保证数值稳定性的标准技巧 pi=Σjexp(sj−m)exp(si−m)，其中 m=maxjsj [20]。
5. 输出写回：计算得到的当前块的输出结果（最终输出矩阵 O 的一部分）被写回到 HBM [5]。

一个形象化的描述是 [5]：FlashAttention 通过分块来避免在相对较慢的 GPU HBM 上实例化庞大的 N×N 注意力矩阵。在外层循环中，FlashAttention 遍历 K 和 V 矩阵的块，并将它们加载到高速的片上 SRAM。在每个 K、V 块内部的循环中，再遍历 Q 矩阵的块，将其加载到 SRAM，计算注意力，并将输出写回 HBM。分块策略直接解决了 SRAM 容量有限的问题，而在线 Softmax 则是使分块策略在数学上对注意力计算保持精确的关键算法创新。

C. 重计算策略：以计算换存储（尤其在反向传播中）

标准的注意力反向传播过程需要前向传播中计算得到的中间注意力矩阵 S（一个 N×N 的大矩阵）以及其他值来计算梯度。在 HBM 中存储这个巨大的 S 矩阵会消耗大量显存，并且在反向传播时将其读回也非常耗时。

FlashAttention 采用了重计算策略来应对这一挑战：它不在前向传播后保存完整的 S 矩阵（或其他大型中间值，如未经传统检查点技术保存的 Q, K, V），而是在反向传播过程中，在 SRAM 内部按需即时重新计算注意力计算的必要部分（例如 S 的分块或 Softmax 归一化所需的统计量）[5]。

选择重计算的根本原因在于“计算快于存储”的原理。在 SRAM 中重新计算一个 S 分块所需的浮点运算量，其时间成本通常低于从 HBM 读写完整 S 矩阵所产生的延迟和带宽成本。FlashAttention 的论文明确指出，通过在反向传播中动态重算注意力值，可以避免从 HBM 读取注意力矩阵，从而加速计算并减少内存使用 [5]。

重计算不仅节省了 HBM 的容量（使得激活值相关的显存占用随序列长度 N 线性增长，而非二次方增长），更重要的是节省了 HBM 的带宽和延迟。让 GPU 的计算单元在 SRAM 中执行额外的 FLOPs，通常比让它们空闲等待数据从 HBM 传输要高效得多。

分块策略使得重计算变得可行且高效。由于数据在前向传播时已经被组织成适合 SRAM 处理的块状结构，反向传播过程可以复用这种块状结构，仅在需要时重新计算中间数据的相

应分块，同样在 SRAM 内部完成。例如，为了计算 Q、K 或 V 某个块的梯度，通常需要对应块的 S 或 P（Softmax 概率）。FlashAttention 可以将相应的 Q 块和 K 块（这些块本身较小）重新加载到 SRAM，并在 SRAM 中重计算出所需的 S 块或 P 块。这种重计算是局部化的，使用的是已经按块管理的数据。如果没有分块，重计算将缺乏结构性，如果重计算本身无法在 SRAM 中完成，可能仍需要大量的中间 HBM 访问。

D. 核函数融合：一种协同优化（简述）

FlashAttention 的实现通常还涉及到将多个独立的操作（例如，矩阵乘法、Softmax、逐元素操作）融合成一个单一的 GPU 核函数（Kernel Fusion）[2]。这样做的好处是减少了启动多个核函数的开销，更重要的是，它允许被融合操作之间的中间数据保留在寄存器或 SRAM 中，避免了往返 HBM 的开销。虽然分块和重计算是减少 HBM 访问的主要策略，但核函数融合通过最小化分块内部计算的中间结果的内存流量，对这些主要策略起到了补充作用。

值得注意的是，虽然 FlashAttention 的核心原理（分块、重计算、在线 Softmax）是算法层面的改进，但其最佳块大小和融合策略的实现会依赖于具体的硬件特性（如 SRAM 大小、SM 数量、warp 调度、特定指令的成本等）。这就是为什么我们会看到 FlashAttention、FlashAttention-2 [3] 和 FlashAttention-3 [23] 等版本不断演进，以更好地利用特定 GPU 架构（如 Ampere、Hopper）的特性。例如，FlashAttention-2 针对 Ampere 架构改进了并行性和工作划分，而 FlashAttention-3 则进一步针对 Hopper 架构利用了其异步执行能力和 FP8 支持。这表明，尽管“计算快于存储”的原则普适存在，但如何最好地利用这一原则会随着硬件细节的演变而发展。

IV. FlashAttention 如何体现“计算快于存储”

A. HBM 读写操作的显著减少

FlashAttention 通过其精心设计的策略，极大地减少了对 HBM 的读写操作。标准注意力机制中，对 N×N 的注意力得分矩阵 S 的 HBM 访问（读和写）复杂度为 O(N2)，根据具体实现，Q, K, V 的访问也可能贡献相当的 HBM 流量。相比之下，FlashAttention 将与 S 相关的 HBM 访问有效地从 O(N2) 降低到接近 O(Nd) 或 O(Nd/Mblock_size) 的级别（其中 Mblock_size 指 SRAM 中处理的块大小），因为它在主要计算循环中仅以分块形式读写 Q, K, V 以及最终的输出 O [7]。至关重要的是，完整的 S 矩阵从未被完整地写入 HBM 或从 HBM 读出。有报告指出，FlashAttention 由于不向 HBM 读写大型 N×N 注意力矩阵，仅在注意力计算部分就实现了高达 7.6 倍的加速 [5]。

下表概念性地对比了标准注意力和 FlashAttention 在 HBM 访问上的差异：

表 **2: HBM** 访问比较：标准注意力 **vs. FlashAttention (**概念性**)**

操作阶段	标准注意力 **HBM** 访问	**FlashAttention HBM**	注意

	(概念性)	访问 (概念性)	
读取 Q, K, V	O(Nd)	O(Nd) (分块读取)	两者都需要从 HBM 初始读取输入。
计算 S=QKT	写 S 到 HBM: O(N2)	无完整 S 写入 HBM	S 在 SRAM 中逐块计算。
计算 P=softmax(S)	从 HBM 读 S: O(N2)	无完整 S 从 HBM 读取	在线 Softmax 使用 SRAM 中的 Sblock。
计算 O=PV	从 HBM 读 P (若未融合), 读 V	使用 SRAM 中的 Pblock, Vblock	
写入 O	O(Nd)	O(Nd) (分块写入)	输出写入 HBM。
反向传播:			
读取 S (或 P, Q, K, V)	从 HBM 读 S: O(N2)	在 SRAM 中重计算 Sblock (或 Pblock)	关键区别：避免了对大型 S 矩阵的 HBM 读取。再次分块读取 Q,K,V。

此表直观地展示了 FlashAttention 如何减少 HBM 访问。它对比了注意力机制关键阶段的内存访问模式，清晰地表明标准注意力中与 S 矩阵相关的 O(N2) 级别的 HBM 访问在 FlashAttention 中被消除或显著减少，因为这些计算被控制在 SRAM 内或通过重计算完成。

B. 最大化 **SRAM** 中常驻数据的算术运算

通过将数据块加载到 SRAM 中，FlashAttention 能够在这些数据离开 SRAM 或其计算结果被写回 HBM 之前，对它们执行多次算术运算（QKT 的一部分、Softmax 计算、与 Vblock 的乘积等）[5]。这显著提高了在 SRAM 局部执行的操作的算术强度。重计算策略进一步强化了这一点：与其执行一次缓慢的 HBM 读取，不如在 SRAM 中执行多次快速的浮点运算。正如相关分析所指出的，FlashAttention 的核心思想是利用 GPU 高速的片上内存（SRAM）来存储中间计算结果，并最小化不同存储层级之间的数据移动 [9]。

FlashAttention 有效地重新定义了注意力计算的“原子工作单元”。标准注意力机制通常是“先计算完整的 S，再计算完整的 P，最后计算完整的 O”，而 FlashAttention 将其转变为“对于 Q、K、V 的一个分块，局部地计算出对应 O 的一个分块”。这种局部化是充分利用 SRAM 的关键。标准注意力机制全局处理矩阵，这使得大型中间结果（如 S,P）必须依赖

HBM。而 FlashAttention 的分块处理，将每个 Q, K, V 块加载到 SRAM，并仅使用当前 *SRAM* 中的数据计算相应的输出 O 块。这种高度局部化的计算最大化了 SRAM 数据的复用，并最小化了中间结果对 HBM 的访问需求。

C. I/O 复杂度的转变分析

- 标准注意力 **I/O**：主要受限于对 S 矩阵的 O(N2) 级别的 HBM 读写操作。
- **FlashAttention I/O**：主要操作变为从 HBM 分块读取 Q, K, V（复杂度约为 O(Nd)）以及向 HBM 分块写入 O（复杂度约为 O(Nd)）。关键在于，与 S 相关的二次方 HBM I/O 被消除了。FlashAttention 论文的核心贡献之一便是通过分块减少内存读写次数 [7]。

“计算快于存储”的原则使得 FlashAttention 能够做出一个明确的权衡：通过增加总的 FLOPs（由于重计算以及分块操作可能引入的更复杂的索引和循环控制），来换取对慢速 HBM 内存访问的大幅削减。其净效应是加速，因为避免 HBM I/O 所节省的时间远大于在 SRAM 中执行额外计算所花费的时间。例如，在反向传播中重计算 S 意味着再次执行 QKT 操作，这些是标准注意力（保存 S）不会重复的 FLOPs。然而，这些重计算主要在 SRAM 中进行。另一种选择是从 HBM 读取庞大的 S 矩阵，这非常缓慢。如果（从 HBM 读取 S 的时间）>（在 SRAM 中重计算 S 的时间），那么重计算就更快。现代 GPU 架构使得对于典型的注意力参数，这个不等式是成立的。

V. FlashAttention 的性能影响与优势

A. 实现的加速与显存占用降低

FlashAttention 的优化策略带来了显著的性能提升和显存效率改善。具体而言，在 BERT-Large 模型上实现了 15% 的端到端训练墙钟时间加速，在 GPT-2 模型上实现了高达 3 倍的注意力计算加速 [5]。后续版本如 FlashAttention-2 相较于初代 FlashAttention 又实现了约 2 倍的提速 [3]。在显存占用方面，FlashAttention 成功地将与序列长度相关的激活值存储从二次方复杂度降低到线性复杂度 [6]，并在注意力计算本身实现了例如 7.6 倍的显存降低 [9]。这些具体的量化指标证明了其设计策略在实践中的巨大成功。

B. 支持 **Transformer** 处理更长的序列

由于显存占用的减少和计算速度的提升，FlashAttention 使得在现有硬件上训练和运行具有更长上下文窗口的 Transformer 模型成为可能 [5]。这不仅提升了模型在某些特定任务上的性能（例如，GPT-2 的困惑度改善，长文档分类准确率提升 [5]），还催生了全新的模型能力，例如在 Path-X（序列长度 16K）等极具挑战性的任务上取得了超越随机猜测的性能 [5]。这突显了 FlashAttention 的影响已超越单纯的速度提升，它为探索和实现更强大的模型能力开辟了道路。

速度和显存的优化并非孤立的好处，它们之间存在着复合效应。显存占用的减少意味着可以在 GPU 中同时容纳更多的序列（即更大的批量大小），这通常能提高训练效率和吞吐量，

或支持更长的单个序列。而每一步计算的加速则意味着可以在相同时间内完成更多的训练迭代或更快的推理响应。这些因素共同作用（更大的批量/更长的序列 + 更快的单步计算），显著改善了模型的训练时间和整体能力。

C. 精确性（无近似计算）

与一些通过近似计算来降低复杂度的注意力优化方法不同，FlashAttention 计算的是精确的注意力值，与标准注意力机制的输出在数学上是等价的 [6]。这是一个关键优势，因为它保证了模型质量不会因为优化而受到影响。通过在不牺牲准确性的前提下大幅提升效率，FlashAttention 为长上下文 Transformer 模型的研究和应用普及化做出了重要贡献。以往，由于标准注意力机制对显存和时间的巨大开销，处理极长上下文的模型训练和部署往往局限于拥有海量计算资源的机构，或者不得不依赖可能牺牲准确性的近似技术。FlashAttention 显著降低了这些资源门槛，使得更多的研究人员和开发者能够在相对普通的硬件配置上实验和部署长上下文模型，从而加速了在需要深度理解长文本（如文档摘要、长篇问答、视频理解）等领域的创新。

VI. FlashAttention 的局限性与演进

A. 硬件依赖与局限性

FlashAttention 的最佳性能表现高度依赖于特定的 GPU 硬件特性，例如充足的 SRAM 容量、高效的 Tensor Core 支持以及优化的 warp 级原语 [8]。早期的 FlashAttention 版本主要针对 NVIDIA GPU 进行了深度优化。将其推广到其他类型的计算架构，如神经处理单元（NPU）或其他供应商的 GPU（例如 AMD GPU），则需要大量的移植工作和针对特定架构的重新设计 [25]。例如，FastAttention 论文指出，FlashAttention 系列主要支持高端 GPU 架构（如 Ampere 和 Hopper），目前难以直接迁移到 NPU 和低资源 GPU 上 [25]。这表明 FlashAttention 并非一个可以不考虑底层硬件而即插即用的通用解决方案。

虽然 FlashAttention 是一种强大的算法，但其对特定硬件特性的深度依赖意味着实现峰值性能是一项复杂的核函数工程任务。这可能加剧所谓的“软件彩票”现象 [1]，即那些具备为特定硬件编写高度优化核函数技能的研究者能够获得显著优势，这可能会减缓新算法思想的采纳速度，如果这些新思想无法轻易地被“Flash 化”。FlashAttention 的有效性源于针对 GPU 存储层次和计算单元的细致底层优化，实现此类核函数绝非易事，需要深厚的 CUDA/GPU 编程专业知识。新的注意力变体可能在理论上具有优势，但如果它们无法实现与 FlashAttention 同等级别的 IO 感知优化，其实际加速效果可能会受限甚至不存在。这为新的注意力机制在实践中保持竞争力设置了很高的门槛，因为它们不仅需要在算法上合理，还需要在核函数层面进行专家级的实现。

B. 演进：**FlashAttention-2** 与 **FlashAttention-3**

FlashAttention 的核心思想在后续版本中得到了持续发展和完善。FlashAttention-2 针对 NVIDIA Ampere 等架构，通过改进并行性、优化工作划分以及减少非矩阵乘法（

non-matmul）的 FLOPs，进一步提升了性能 [3]。FlashAttention-3 则更进一步，针对 NVIDIA Hopper 架构的特性，例如利用 Tensor Core 和张量内存加速器（Tensor Memory Accelerator, TMA）的异步性来实现计算与数据传输的重叠，交错执行分块的矩阵乘法和 Softmax 操作，并引入了块量化和非相干处理技术以充分利用 FP8 低精度计算的优势 [23]。

从 FlashAttention 到 FA2 和 FA3 的演进清晰地表明，IO 感知的优化并非一劳永逸。随着 GPU 架构的不断发展（例如，新的内存类型、不同的缓存大小、新的异步特性、新的数据类型），最佳利用"计算快于存储"原则的具体方法也在发生变化，这要求对关键核函数进行持续的重新优化。FlashAttention v1 针对 V100/A100 等 GPU 进行了优化。FlashAttention-2 通过改进并行性和工作划分，进一步提升了在 A100 上的性能 [4]。Hopper GPU (H100) 引入了 TMA 和更强大的 FP8 Tensor Core 等新特性。FlashAttention-3 则是专门为利用这些 Hopper 特性（异步性、FP8）而开发的，以进一步提升性能 [23]。这展示了硬件进步与软件（核函数）优化之间的协同进化：硬件不断发展，软件必须随之调整以理解并利用这些与计算和内存相关的新架构特性，从而榨取最大的性能。

VII. 结论：IO 感知算法设计的深远意义

FlashAttention 通过其核心的分块计算和重计算策略，深刻地体现了"计算快于存储访问"这一现代硬件设计原则，并成功地缓解了注意力机制中的 HBM 带宽瓶颈。它不仅显著提升了 Transformer 模型的训练和推理效率，还使得处理更长序列成为可能，从而推动了自然语言处理等领域的发展。

FlashAttention 的成功是算法与硬件协同设计的一个典范。它清晰地表明，在设计高性能计算算法时，必须充分考虑底层硬件的特性，特别是存储层次结构的差异。FlashAttention 不仅仅是加速了注意力机制本身，更重要的是，它引领了一种优化思路的转变：从单纯追求减少理论 FLOPs 转向更加关注实际的内存访问模式和数据移动效率。

随着模型规模和数据集的持续增长，IO 感知的算法设计理念将变得愈发重要。未来的深度学习系统优化，将更加强调最小化跨越存储层级的数据传输，而不仅仅是计算量的优化。FlashAttention 所展示的原理和方法，为应对未来更大规模、更复杂模型的挑战，提供了宝贵的经验和启示。未来的 GPU 和加速器可能会拥有更复杂的内存系统（例如，多级缓存、不同类型的封装内存），"计算与存储"之间的差距可能会以更细微的方式持续扩大或变化。那些将内存视为扁平、同质实体的算法将无法充分发挥硬件性能。因此，未来成功的算法很可能需要从一开始就内在地具备 IO 感知能力，明确考虑数据局部性、分块以及最小化跨越内存层级中最慢链路的数据传输，正如 FlashAttention 为 SRAM-HBM 接口所做的那样。

VIII. 参考文献

14 Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, Christopher Ré. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135.
1 Bowen Yang, Jianan Li, Bohan Zhuang, Hexiang Hu, Kaining Zhang, Zixuan Liu, Jianfei Cai,

Yong Liu, Ming-Ming Cheng, Qixiang Ye. (2024). FlashAttention on a Napkin: A Diagrammatic Approach to Deep Learning IO-Awareness. arXiv:2412.05496v1.
5 HKAIFT. (N.D.). FlashAttention: Revolutionizing Transformers by Overcoming Hardware Performance Bottlenecks. hkaift.com.
2 Hopsworks AI. (N.D.). Flash Attention. hopsworks.ai.
6 Tri Dao. (2023). Flashier Attention. adept.ai.
19 Alfredo Deza. (2023). Understanding Flash Attention and Flash Attention 2: The Path to Scale the Context Lenght of Language Models. towardsai.net.
12 Exxact Corp. (2024). GDDR6 vs HBM GPU Memory: What's the Difference? exxactcorp.com.
13 DigitalOcean. (2025). GPU Memory Bandwidth and Its Impact on Performance. digitalocean.com.
18 Shaped.ai. (2024). Jagged Flash Attention Optimization. shaped.ai.
17 Reddit r/LocalLLaMA. (2024). How much does flash attention affect intelligence? reddit.com.
27 Wevolver. (N.D.). High Bandwidth Memory (HBM): Revolutionizing Data Processing. wevolver.com.
13 DigitalOcean. (2025). GPU Memory Bandwidth and Its Impact on Performance. digitalocean.com. 13
15 Baseten. (2025). A guide to LLM inference and performance. baseten.co.
16 DataCrunch.io. (N.D.). Multi-Head Latent Attention: Benefits in Memory and Computation. datacrunch.io.
11 Samuel Williams, Andrew Waterman, David Patterson. (2014). A Practical Tool for Architectural and Program Analysis. crd.lbl.gov.
9 Continuum Industries. (2022). Flash Attention. training.continuumlabs.ai.
7 Anish Dubey. (2024). Flash attention(Fast and Memory-Efficient Exact Attention with IO-Awareness): A deep dive. towardsdatascience.com.
10 Reddit r/chipdesign. (2024). Why dont make sram-based hbm? reddit.com.
7 Anish Dubey. (2024). Flash attention(Fast and Memory-Efficient Exact Attention with IO-Awareness): A deep dive. towardsdatascience.com. 7
20 Georgios Tziantzioulis, Christos-Savvas Bouganis. (2025). FLASH-D: A Transformed FlashAttention Kernel with Hidden Softmax Division and Inherent Numerical Stability. arXiv:2505.14201.
25 Haoran Lin, Xianzhi Yu, Kang Zhao, Zhelong Li, Tongxin Li, Mingbao Liu, Zeyu Dai, Senzhang Wang, Li Lyna Zhang, Lawrence S. Kim, Yu Wang, Huazhong Yang. (2024). FastAttention: Extend FlashAttention to NPUs and Low-resource GPUs for Efficient LLM Inference. arXiv:2410.16663v1.
3 Modular. (N.D.). FlashAttention-2: Revolutionizing Attention Mechanisms. modular.com.
4 E2E Networks. (2023). Shades of Attention: FlashAttention vs FlashAttention-2: A Comprehensive Study. e2enetworks.com.
23 Anonymous Authors. (2024). FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. OpenReview.
24 NVIDIA GTC. (2025). FlashAttention-3: Fast and Accurate Attention With Asynchrony and Low Precision. nvidia.com.

8 DZone. (2024). Accelerating AI: A Dive into Flash Attention and Its Impact. dzone.com.
21 Christian Mills. (N.D.). CUDA Mode Notes: Lecture 12 - Flash Attention. christianjmills.com.
22 YouTube. (N.D.). Flash Attention Hardware Basics and Kernel Fusion. youtube.com. (Conceptual explanation)
26 Haoran Lin, Xianzhi Yu, Kang Zhao, Zhelong Li, Tongxin Li, Mingbao Liu, Zeyu Dai, Senzhang Wang, Li Lyna Zhang, Lawrence S. Kim, Yu Wang, Huazhong Yang. (2024). FASTATTENTION: EXTEND FLASHATTENTION TO NPUS AND LOW-RESOURCE GPUS FOR EFFICIENT LLM INFERENCE. OpenReview. (Corresponds to arXiv:2410.16663)
5 HKAIFT. (N.D.). FlashAttention: Revolutionizing Transformers by Overcoming Hardware Performance Bottlenecks. 5
6 Tri Dao. (2023). Flashier Attention. 6
9 Continuum Industries. (2022). Flash Attention. 9
7 Anish Dubey. (2024). Flash attention(Fast and Memory-Efficient Exact Attention with IO-Awareness): A deep dive. 7
7 Anish Dubey. (2024). Flash attention(Fast and Memory-Efficient Exact Attention with IO-Awareness): A deep dive. 7
20 Georgios Tziantzioulis, Christos-Savvas Bouganis. (2025). FLASH-D: A Transformed FlashAttention Kernel with Hidden Softmax Division and Inherent Numerical Stability. 20
4 E2E Networks. (2023). Shades of Attention: FlashAttention vs FlashAttention-2: A Comprehensive Study. 4
23 Anonymous Authors. (2024). FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. 23
25 Haoran Lin, et al. (2024). FastAttention: Extend FlashAttention to NPUs and Low-resource GPUs for Efficient LLM Inference. 25
8 DZone. (2024). Accelerating AI: A Dive into Flash Attention and Its Impact. 8
20 Georgios Tziantzioulis, Christos-Savvas Bouganis. (2025). FLASH-D: A Transformed FlashAttention Kernel with Hidden Softmax Division and Inherent Numerical Stability. 20
25 Haoran Lin, et al. (2024). FastAttention: Extend FlashAttention to NPUs and Low-resource GPUs for Efficient LLM Inference. 25

Works cited

1. 1 Introduction - arXiv, accessed May 24, 2025, https://arxiv.org/html/2412.05496v1
2. What is Flash Attention? - Hopsworks, accessed May 24, 2025, https://www.hopsworks.ai/dictionary/flash-attention
3. FlashAttention-2 - AI Resources - Modular, accessed May 24, 2025, https://www.modular.com/ai-resources/flashattention-2
4. FlashAttention vs FlashAttention-2 - an Analysis. - E2E Networks, accessed May 24, 2025, https://www.e2enetworks.com/blog/shades-of-attention-flashattention-vs-flashattention-2-a-comprehensive-study
5. FlashAttention: Revolutionizing Transformers by Overcoming ... - AIFT, accessed May 24, 2025, https://hkaift.com/flashattention-revolutionizing-transformers-by-overcoming-ha

rdware-performance-bottlenecks/
6. FlashAttention: Fast Transformer training with long sequences, accessed May 24, 2025, https://www.adept.ai/blog/flashier-attention
7. Flash attention(Fast and Memory-Efficient Exact Attention with IO ..., accessed May 24, 2025, https://towardsdatascience.com/flash-attention-fast-and-memory-efficient-exact-attention-with-io-awareness-a-deep-dive-724af489997b/
8. Accelerating AI: A Dive into Flash Attention and Its Impact - DZone, accessed May 24, 2025, https://dzone.com/articles/accelerating-ai-flash-attention-impact
9. Flash Attention | Continuum Labs, accessed May 24, 2025, https://training.continuumlabs.ai/inference/why-is-inference-important/flash-attention
10. Why dont make sram-based hbm? : r/chipdesign - Reddit, accessed May 24, 2025, https://www.reddit.com/r/chipdesign/comments/1fxcy0n/why_dont_make_srambased_hbm/
11. Roofline Model Toolkit: A Practical Tool for Architectural and Program Analysis - Computing Sciences Research - Lawrence Berkeley National Laboratory, accessed May 24, 2025, https://crd.lbl.gov/assets/pubs_presos/PMBS14-Roofline.pdf
12. GDDR6 vs HBM - Different GPU Memory Types | Exxact Blog, accessed May 24, 2025, https://www.exxactcorp.com/blog/hpc/gddr6-vs-hbm-gpu-memory
13. GPU Memory Bandwidth and Its Impact on Performance - DigitalOcean, accessed May 24, 2025, https://www.digitalocean.com/community/tutorials/gpu-memory-bandwidth
14. [2205.14135] FlashAttention: Fast and Memory-Efficient Exact ..., accessed May 24, 2025, https://ar5iv.labs.arxiv.org/html/2205.14135
15. A guide to LLM inference and performance | Baseten Blog, accessed May 24, 2025, https://www.baseten.co/blog/llm-transformer-inference-guide/
16. Multi-Head Latent Attention: Benefits in Memory and Computation — Blog - DataCrunch, accessed May 24, 2025, https://datacrunch.io/blog/multi-head-latent-attention-benefits-in-memory-and-computation
17. How much does flash attention affect intelligence in reasoning models like QwQ - Reddit, accessed May 24, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1jcsvys/how_much_does_flash_attention_affect_intelligence/
18. Jagged Flash Attention Optimization | Shaped Blog, accessed May 24, 2025, https://www.shaped.ai/blog/jagged-flash-attention-optimization
19. Understanding Flash-Attention and Flash-Attention-2: The Path to Scale The Context Lenght of Language Models | Towards AI, accessed May 24, 2025, https://towardsai.net/p/artificial-intelligence/understanding-flash-attention-and-flash-attention-2-the-path-to-scale-the-context-lenght-of-language-models
20. www.arxiv.org, accessed May 24, 2025, https://www.arxiv.org/pdf/2505.14201
21. GPU MODE Lecture 12: Flash Attention - Christian Mills, accessed May 24, 2025, https://christianjmills.com/posts/cuda-mode-notes/lecture-012/

22. ELI5 FlashAttention: Understanding GPU Architecture - Part 1 - YouTube, accessed May 24, 2025, https://www.youtube.com/watch?v=wwXHfSJQOag
23. FlashAttention-3: Fast and Accurate Attention with Asynchrony and ..., accessed May 24, 2025, https://openreview.net/forum?id=tVConYid20&referrer=%5Bthe%20profile%20of%20Tri%20Dao%5D(%2Fprofile%3Fid%3D~Tri_Dao1)
24. FlashAttention-3: Fast and Accurate Attention With Asynchrony and Low Precision | GTC 25 2025 | NVIDIA On-Demand, accessed May 24, 2025, https://www.nvidia.com/en-us/on-demand/session/gtc25-S71368
25. arxiv.org, accessed May 24, 2025, https://arxiv.org/html/2410.16663v1
26. FASTATTENTION: EXTEND FLASHATTENTION TO NPUS AND LOW-RESOURCE GPUS FOR EFFICIENT LLM INFERENCE - OpenReview, accessed May 24, 2025, https://openreview.net/pdf?id=76NYyOrnfk
27. High Bandwidth Memory: Concepts, Architecture, and Applications - Wevolver, accessed May 24, 2025, https://www.wevolver.com/article/high-bandwidth-memory