

面向大模型推理的存算协同理论综述



A Survey on Compute-Storage Co-Optimization Theory for Large Model Inference

汪俊峰/Wang Junfeng¹, 刘璇/Liu Xuan¹,
李亚奇/Li Yaqi^{1,2}, 石亮/Shi Liang^{1,2,3}

(1. 华东师范大学, 中国 上海 200062;

2. 上海创智学院, 中国 上海 200233;

3. 华中科技大学武汉光电国家研究中心, 中国 武汉 430074)

(1. East China Normal University, Shanghai 200062, China;

2. Shanghai Innovation Institute, Shanghai 200233, China;

3. Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China)

DOI: 10.12142/ZTETJ.202604006

网络出版地址: <https://link.cnki.net/urlid/34.1228.TN.20260907.1604.002>

网络出版日期: 2026-08-28

收稿日期: 2026-05-16

摘要: 大模型推理在长上下文、高并发与多轮复用条件下, 受算力、存储容量与跨层数据移动共同约束。量化、缓存、卸载与阶段解耦常组合部署, 但压缩率、命中率、显存占用下降等局部指标改善, 并不等价于端到端关键路径缩短。从权重、键值 (KV) 缓存与激活的生命周期出发, 以关键路径净收益为判据, 系统梳理以算换存、以存换算与存算路径3类方法, 并归纳其适用条件、附加开销、质量约束与失效边界, 为面向负载的存算协同部署提供可复用的分析视角。认为未来应从生命周期控制、压缩表征、跨引擎接口与真实负载评测4个方面开展存算协同相关研究。

关键词: 大模型推理; 存算协同; 数据移动; KV缓存; 资源置换; 瓶颈迁移

Abstract: Under long context, high concurrency, and multi-turn reuse, large model inference is constrained by computation, storage capacity, and cross-tier data movement. Quantization, caching, offloading, and stage disaggregation are often combined, but improvements in compression ratio, hit rate, or memory footprint do not imply a shorter end-to-end critical path. Starting from the lifetimes of weights, key-value (KV) caches, and activations, and using critical-path net benefit as the evaluation criterion, this paper systematically reviews three categories of methods: computation for storage, storage for computation, and storage-computation pathways. It further summarizes their applicability, overheads, quality constraints, and failure modes, and provides a reusable analytical lens for workload-oriented compute-storage co-optimization. This paper holds that future research related to compute-storage co-optimization should be conducted from four aspects: lifecycle control, compressed representation, cross-engine interfaces, and real-workload evaluation.

Keywords: large model inference; compute-storage co-optimization; data movement; KV cache; resource exchange; bottleneck migration

引用格式: 汪俊峰, 刘璇, 李亚奇, 等. 面向大模型推理的存算协同理论综述 [J]. 中兴通讯技术, 2026, 32(4): 32-40. DOI: 10.12142/ZTETJ.202604006

Citation: Wang J F, Liu X, Li Y Q, et al. A survey on compute-storage co-optimization theory for large model inference [J]. ZTE technology journal, 2026, 32(4): 32-40. DOI: 10.12142/ZTETJ.202604006

Transformer 自回归推理需反复访问共享权重, 并为活跃序列保存随上下文增长的键值 (KV) 缓存^[1]。随着上下文长度、并发请求数及多轮复用次数的增加, 算力、高带宽内存 (HBM) 容量与跨层数据移动共同成为首词元时延

(TTFT)、逐词元时延 (TPOT) 及服务等级协议 (SLO) 达成的瓶颈。已有研究从服务系统与KV管理的角度进行了梳理^[2-3], 但压缩率、命中率、显存占用下降等局部指标的改善, 并不必然缩短端到端关键路径; 当量化、缓存、卸载与阶段解耦叠加使用时, 单项收益也不能直接累加。

本文以权重、KV与激活的生命周期为线索, 按3类资源置换组织讨论: 以算换存、以存换算与存算路径。评价准

基金项目: 华中科技大学武汉光电国家实验室开放课题计划项目 (2023 WNLOKF004)

则是关键路径净收益——仅当节省落在当前瓶颈且新增开销不高时，局部优化才转化为服务收益。图1展示了面向大模型推理的资源置换与瓶颈迁移统一决策框架。

1 数据对象与边界

推理侧的核心对象为权重、KV与激活：权重可跨请求共享，在解码（Decode）阶段易受内存带宽约束；KV在Prefill阶段生成，在Decode阶段主要被重复读取，同时持续追加新生成Token对应的KV状态，是长上下文、高并发场景下的主要容量压力来源；激活的生命周期较短，但长Prefill过程与大批量处理会抬升资源峰值。HBM需要同时承载上述对象及运行时工作区，因此压缩、缓存与跨层迁移技术，只有在削减当前主导开销，且新增的编解码、数据恢复、传输代价更低时，才能带来端到端收益。3类资源置换的主要边界如表1所示。

2 以算换存

以算换存通过增加或重组计算，减少需物化、驻留或实

际访问的数据量。其典型手段包括注意力输入/输出（I/O）重组、权重与KV的可计算压缩，以及词元裁剪与查询相关访问。该方法的收益取决于额外计算是否作用于当前带宽或流量瓶颈，且编解码开销、索引维护与质量损失是否可控。

2.1 I/O 重组

标准注意力机制会在片上存储与HBM之间多次读写中间结果。FlashAttention通过分块与在线Softmax避免物化完整注意力矩阵；FlashAttention-2优化了任务划分策略；FlashAttention-3借助异步流水线与低精度计算路径提升硬件利用率^[4-6]。FlashInfer进一步适配变长序列、分页机制与Decode内核^[7]。上述方法以更精细的计算组织来削减HBM写回开销，通常不改变计算所得的数学结果，可作为各类优化方案的基线。

I/O重组的优化收益具有负载依赖性。在短序列与小批量场景下，分块加载、数据同步及专用内核调度开销将占据主导地位。当推理瓶颈转移至权重读取、KV带宽或采样环

节时，持续降低注意力中间写回流量的效果对TPOT的优化作用将趋于有限。相关性能评价需同步考量实际HBM流量、算力利用率、内核耗时占比及端到端TTFT指标，不宜仅依赖注意力内核加速比开展单一维度评估。

长Prefill阶段亦可通过缩减参与计算的数据量降低物化开销。MInference依据注意力模式筛选稀疏计算块^[8]；Ring Attention通过分块通信将长序列分摊至多设备^[9]。前者依赖注意力模式预测，存在预测误判与输出质量下降的风险；后者能够保留更为完整的上下文，但会带来额外的通信与同步开销。当注意力具备稳定稀疏特性时，选择性计算方案更具优势；当单卡存储容量受限且设备互联条件充

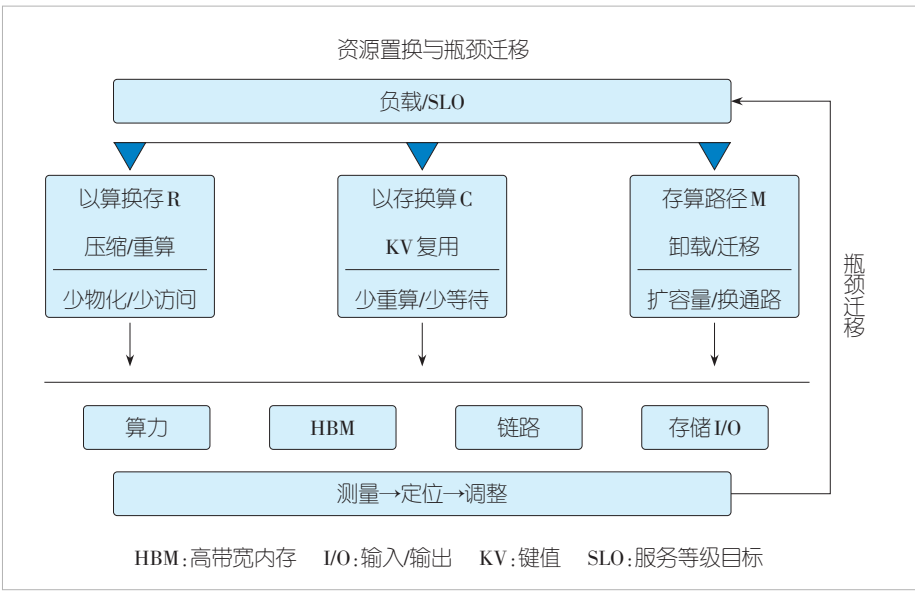


图1 面向大模型推理的资源置换与瓶颈迁移统一决策框架

表1 3类资源置换动作比较

维度	以算换存(R)	以存换算(C)	存算路径(M)
目标	减少物化、存储或读取	避免重复Prefill	扩展容量与共享范围
主要手段	I/O重组、压缩、裁剪与重算	KV分页、前缀/片段复用	卸载、预取与分布式KV
主要代价	编解码、索引与质量风险	驻留、恢复与一致性开销	I/O、链路争用与目录开销
适用条件	带宽或流量主导，附加计算可控	复用率高且恢复快于重算	跨层驻留收益大于迁移开销
正确性	布局无损，压缩或裁剪可能有损	精确且兼容的复用无损	完整迁移无损，选择性恢复有风险

I/O:输入/输出 KV:键值

足时,序列分摊方案更为适用。针对超长输入场景,还可将稀疏筛选与多设备分摊相结合:在单设备内部削减无效块计算,在多设备之间分摊有效块算力。该方案要求统一分块边界与通信调度策略,否则索引不一致将引发数据重复传输或者计算漏算问题。

2.2 权重与KV压缩

权重低比特表示可降低驻留容量与Decode阶段的读取量。SmoothQuant将激活离群值迁移至权重侧;AWQ依据激活统计信息保护重要通道^[10-11];PowerInfer根据激活稀疏性将热、冷神经元分配到不同处理器^[12]。需要区分模型文件大小与实际计算格式:仅当矩阵乘内核可直接消费低比特权重时,容量下降才较有可能转化为设备带宽的降低;若每层先恢复为FP16或BF16,则缩放元数据、反量化读写及工作区开销可能抵消收益。

KV的数值分布与权重不同。KIVI与KVQuant利用Key、Value及通道差异实现低比特表示^[13-14];MiniCache跨层合并相似状态;Palu以低秩投影缩减缓存维度^[15-16]。压缩后占用应计入缩放因子、异常值、残差窗口与临时缓冲。可直接参与计算的表示可同时降低容量与HBM读取;仅用于固态硬盘(SSD)或网络传输的格式,其收益条件为慢链路字节节省大于编码、传输、解码与临时写回开销。

以算换存的目标不是最小化位宽,而是使增加的计算作用于当前瓶颈。若HBM带宽占主导,应优先采用可直接读取的压缩布局;若高速串行计算机扩展总线(PCIe)或网络占主导,可将编码置于慢层并以流式恢复;若算力饱和,增加复杂编解码会延长TTFT。表示格式会影响后续缓存与卸载粒度,因此需与Block边界、预取接口及错误回退一并设计。

对于短生命周期的激活以及可由Prompt快速重建的KV,释放存储并在需要时重算,可能优于长期保留;而对于长上下文或跨轮次状态,重算会再次消耗Prefill并延长恢复时间。是否采用重算,需综合权衡对象大小、重建层数、当前图形处理器(GPU)排队情况与下次使用时间。局部可逆重算也可作为近似选择或缓存失效后的回退手段。需要强调的是,重算路径会与在线Decode争用同一GPU算力。在排队较长时,即使对象可由Prompt重建,优先恢复已有KV仍可能更有利于满足SLO。

2.3 KV裁剪与按需访问

词元选择将“保留全部历史”改为“保留或读取必要历史”。H2O与Scissorhands依据历史重要性驱逐非必要的Token或KV状态;StreamingLLM保留注意力汇点与滑动窗

口^[17-19];SnapKV、PyramidKV与ChunkKV分别依据观察窗口、层间预算与语义块选择位置^[20-22]。此类方法可同时降低容量与读取量,但永久删除后通常无法恢复。

Quest依据当前查询(Query)与页级统计选择本步需加载的KV页^[23];DuoAttention区分检索头与流式头并设置不同历史范围^[24];ShadowKV以低秩Key表示辅助稀疏恢复^[25]。选择粒度越细,潜在节省越大,索引、预测与不规则访存开销就越高。误删会造成不可逆质量损失,并可能在后续多步持续影响生成。

因此,I/O重组、数值压缩与信息裁剪应分别按语义等价、数值误差与信息删除进行评价。工程上可先建立无损布局基线,再验证可直接计算的压缩表示,最后在质量预算明确时引入动态选择。

此外,以算换存还受控制开销约束。稀疏模式识别、查询打分、词元重要性估计与格式转换均引入额外计算。当请求很短、模式变化快或批量很小时,控制开销难以被摊薄。在线服务宜把复杂分析置于Prefill或后台阶段,将Decode中的决策限制为少量可预测操作;离线长上下文任务则可接受更高分析成本以换取更大的访问削减。同一算法在两类负载上的结论不宜直接迁移。

3 以存换算

以存换算是指用存储容量换取计算时间,在保留历史KV后可避免重复Prefill与状态重建。请求内分页布局决定可接纳批量,跨请求前缀与片段复用决定能否缩短TTFT;但命中并不自动等价于端到端收益,还需权衡恢复开销、容量机会成本与兼容边界。

3.1 KV布局与复用

KV缓存以存储容量换取计算时间:保留历史Key与Value后,Decode无需重新处理已生成的词元。PagedAttention将逻辑序列映射到非连续物理块,降低动态长度请求的碎片;vAttention利用虚拟内存保持地址连续^[26-27]。二者主要改善驻留布局,不减少KV信息量或注意力读取量,但通过碎片率、地址映射与写时复制影响可接纳批量。

跨请求复用可避免重复Prefill。SGLang用基数树匹配公共前缀;PromptCache以预定义模块组织可复用状态^[28-29];CacheBlend面向非前缀检索增强生成(RAG)片段,通过选择性重算修复片段间交叉注意力^[30]。从精确词元前缀、预定义模块到组合片段,复用自由度增大,位置约束、键匹配与补算成本亦相应增加。LMCache将KV扩展为可在GPU、CPU、磁盘与远端访问的数据层^[31],使一次Prefill结果可跨

请求、跨实例或跨时段使用。

缓存命中并不自动等价于端到端收益。精确前缀恢复开销低、正确性边界清晰，适合作为共享缓存的基础机制；模块化片段便于编排，但要求模板与位置一致；RAG 片段组合更灵活，但需修复跨片段注意力。缓存位置同样影响收益：本地命中主要支付查找与恢复开销，远程命中还需支付网络与排队开销；对低频短前缀，直接重算可能更快。

3.2 缓存收益与生命周期

缓存的预期收益取决于复用概率、可避免的 Prefill 计算量与再次访问时间。相关开销包括插入、索引、查找、取回、片段修复、一致性维护，以及占用显存带来的容量机会成本。命中率应按复用词元数或避免的计算时间加权，而不宜仅按请求数统计。在 HBM 中保留大量冷 KV 可降低未来恢复开销，但这可能会挤占活跃请求的批处理空间。将冷 KV 下沉虽然可以释放一定容量，但仍需在恢复截止时间前完成传输。因此，缓存策略应综合考虑下次使用时间、重算代价、共享概率与当前队列，而不宜仅依赖最近最少使用算法 (LRU)。

缓存复用需满足多维度兼容性约束，包括模型、适配器、分词器、提示模板、位置编码、KV 存储布局与数值精度的完全匹配。多租户场景下，还需将访问权限、资源所有权、版本信息与安全属性纳入缓存键的构建维度；跨版本迭代更新时，历史 KV 缓存需隔离至独立命名空间或采用渐进失效策略。本文将缓存净收益定义为：预估规避的重算耗时，扣除缓存恢复耗时与存储容量机会成本；同时在队列长度动态增长、缓存容量水位抬升时，主动降低低收益缓存任务的调度优先级。

此外，前缀共享还存在收益递减问题。系统级 Prompt 被多个请求复用时，首次请求承担构建与写入开销，后续请求才获得节省。当共享度下降或模板频繁变更时，目录、哈希与失效维护仍持续发生。RAG 片段组合可能出现部分命中、重复片段与位置不连续，此时可复用比例不宜用字节命中率代替。运行时应记录共享前缀长度、命中后补算词元数与恢复时延分位数，以判断缓存是否真正缩短 Prefill 关键路径。对于暂停的 Agent 会话，还应保留原始 Prompt、工具状态与版本元数据，以便 KV 失效时能够重算或部分恢复。

3.3 推测解码与缓存

推测解码由小模型或附加头生成候选，再由主模型并行验证，从而减少串行 Decode 步数^[32]。该方法增加候选计算与临时状态，以换取主模型串行步数减少，其收益取决于候

选接受率、验证批量与 KV 提交方式。接受长度波动会改变批量形状与 KV 增长速度。若分配器仍按固定步长预留，峰值容量压力可能上升。

共享前缀可降低候选阶段的输入处理开销，但候选分支产生的临时 KV 通常不宜跨请求复用，验证失败后还需回收未提交状态。分块 Prefill 通过混合调度降低长输入对 Decode 的阻塞^[33]，同时使活跃请求、预填充块与缓存恢复共享调度队列。因此，缓存管理需感知词元提交节奏，不宜将生成速度视为常数。推测解码与前缀缓存同时启用时，还应区分共享状态与请求私有状态：共享前缀可降低草稿阶段开销，但未提交候选 KV 不应写入跨请求缓存目录。

4 KV 存储路径

当 HBM 无法同时容纳权重、热 KV、激活与工作区时，KV 需在 CPU 内存、NVMe SSD 与远程存储池之间进行存放与恢复。对于多轮 Agent 与长会话负载，Prefill 命中后的性能往往由 KV 回到计算单元的数据通路决定，而非仅由容量是否足够决定。本节从节点内放置、I/O 通路选择、双路径加载、选择性恢复与分布式数据层几个方面展开讨论，并给出跨层组合与服务评价建议。

4.1 节点内放置

FlexGen 联合规划权重、KV 与激活的放置，并通过大批量与流水重叠来降低 CPU 与 SSD 的传输开销。FlexInfer 面向端侧异构设备动态选择卸载对象^[34-35]。此类方法首先解决了部署可行性问题，并不自动降低请求时延。较大的批量处理可扩大计算覆盖窗口，但会增加 KV 占用、排队及 TTFT。离线吞吐配置不宜直接外推到在线低时延服务。

对 KV 路径而言，关键约束是恢复截止时间：查找、解码与数据移动必须在首次使用前完成。阻塞下一 Decode 步骤的热 KV 优先级高于可延迟的权重预取；可由 Prompt 重建的临时激活优先级低于长会话状态。若权重卸载占满 PCIe，KV 恢复将被拖慢。PowerInfer 表明迁移粒度可由层细化到神经元或结构单元^[12]：粒度越细，空间利用率越高，但预测、小事务与同步开销会增加。系统应依据 I/O 队列深度、事务启动开销与预测置信度选择块大小，而非固定采用最小块。

4.2 KV I/O 通路

外层 KV 进入 GPU HBM 的常见通路包括：1) 存储经主机 DRAM 再由 cudaMemcpyAsync 写入 HBM；2) GPUDirect 存储器 (GDS) 由 NVMe 经 PCIe 直达 HBM，减少主机 CPU

拷贝^[36]；3）经计算网卡（CNIC）发起GPU远程直接内存访问（GPUDirect RDMA），将远端数据直接写入HBM，使传输进入网卡侧服务质量（QoS）调度。三者峰值带宽、CPU开销以及集合通信的隔离能力等方面有所不同。

GDS比较适用于以吞吐为主且PCIe上模型执行通信压力较低的场景：绕过主机中转可降低CPU占用并缩短存储到HBM的路径。但在Prefill/Decode（P/D）解耦与张量并行的常见部署中，同一PCIe链路上同时存在集合通信与KV加载，GDS与CUDA拷贝引擎通常无法纳入计算网卡侧的虚拟通道仲裁，KV突发可能会抬高通信尾延迟。此时，先由存储网卡读入主机DRAM，再经网卡以RDMA Write完成主机到设备的数据传输，可把KV流量置于可调度路径上，以稍长的主机中转路径换取与延迟敏感通信的隔离。而对于大量小块KV，用户态提交RDMA工作请求并批量触发doorbell，往往比逐次调用cudaMemcpyAsync更易摊薄提交开销。

路径选择应同时报告有效带宽、CPU占用、PCIe争用与集合通信P99。若评测仅给出GDS峰值吞吐而忽略通信干扰，则会高估其在在线服务中的收益。相应地，新生成KV落盘亦可经CNIC先到主机、再经存储网络写入后端，使读写两侧共用同一套QoS规则。

4.3 双路径加载

在P/D解耦架构中，传统KV加载仅采用“存储→Prefill实例”的单路径传输模式。多轮Agent负载场景下，Prefill侧存储网卡极易出现带宽饱和，而Decode侧存储网卡往往处于空闲状态，形成显著的带宽资源不对称问题。DualPath方案在保留存储至Prefill原生路径的基础上，新增存储至Decode的传输路径：将KV数据优先加载至Decode实例，再通过计算网络RDMA转发至Prefill实例，依托全局调度器在双路径间动态分配流量，实现双侧存储NIC带宽的聚合利用^[37]。该方案的核心并非扩充存储介质资源，而是重构KV数据进入计算平面的拓扑入口。

双路径机制的有效运行需满足多项约束条件：计算网络可用带宽需高于存储网络尾部带宽，确保转发流程不会显著干扰Decode通信；调度器可依据Prefill与Decode双侧的存储队列、计算队列状态及业务SLO自适应选择传输路径；加载路径与执行通信可在PCIe/NIC层级实现硬件隔离。若无法满足上述条件，经由Decode的绕行传输将同步增加Decode尾延迟与Prefill等待时延。性能评价需全面统计双侧存储网卡（NIC）利用率、路径切换占比、KV数据在存储网络与计算网络的流量分布，以及TTFT、吞吐相较单路径架构的

优化幅度，而非仅仅关注聚合带宽峰值指标。

Prefill/Decode架构解耦本身也会重构KV数据的交接链路。DistServe、Splitwise与WindServe分别从有效吞吐配置、异构资源匹配、流式动态调度维度实现了推理阶段解耦^[38-40]。Prefill阶段结束后需完成KV数据跨实例交接，网络拥塞将直接纳入TTFT关键路径。在短输入、低负载或设备互联带宽不足的场景下，阶段解耦带来的性能收益可能小于KV传输与远端排队产生的开销，此时推理共置执行方案更具优势。当P/D解耦与双路径机制叠加部署时，需严格区分“阶段间KV交接流量”与“外层存储KV加载流量”，对两类流量分别统计计量并设置差异化调度优先级。

4.4 选择性恢复与预取

即便确定数据加载通路，推理过程仍无需每次搬移完整KV数据。InfiniGen依托部分查询信息与已有Key缓存预判关键KV内容，实现选择性按需拉取^[41]，将完整KV数据留存于低速存储层，仅在预测偏差时补充加载。面向多轮大模型服务的以内存为中心的架构（MCAM）用于管理多级KV缓存的分布位置，异步KV预取机制则可利用计算间隙提前完成缓存状态恢复^[42-43]。预取策略存在明显的权衡关系：预取时机过早会占用宝贵的HBM资源，过晚则会造成GPU计算等待；预取粒度过细会受限于单次事务时延，粒度过粗则会冗余搬运无效数据。选择性数据恢复与GDS、双路径机制相互正交：前者精简数据传输字节量，后者决定剩余数据的传输拓扑路径。

CacheGen通过定制化编码方案削减KV数据跨存储、跨网络的传输字节开销^[44]。当低速链路的字节节省收益高于编码、传输、解码及临时写回的综合开销时，该机制可有效压缩推理关键路径时延。压缩传输主要优化存储网络与PCIe链路流量，可计算压缩还能进一步降低HBM与计算单元间的数据交互流量。需要说明的是，两类优化机制需加以区分，不可混为一谈。缓存迁移无须完全被计算时延掩盖即可产生优化价值，仅当容量释放带来的请求准入提升、排队时延降低等收益，超过迁移暴露开销与链路争用损耗时，能够切实提升服务性能；反之，该类优化主要体现为部署可行性层面的增益。

在故障恢复与调度公平性层面，可依据任务重算耗时与业务优先级划分缓存恢复等级。I/O调度可将缓存迁移任务拆解为可抢占式事务，在保障Decode任务截止时间的前提下，为不同请求分配固定带宽份额，同时将超时请求退回本地重算或切换至备用加载路径。

4.5 分布式 KV

分布式 KV 缓存将缓存覆盖范围从单节点拓展至集群维度。Mooncake 构建以 KV 数据为核心的分布式数据层；MemServe 依托弹性内存池与统一访问接口，支撑跨实例 KV 缓存复用^[45-46]。缓存容量与共享域的扩张，使得目录维护、权限归属、副本管理、版本迭代与故障恢复成为系统必需的基础性开销。针对推理请求，调度器需综合比对本地图中、本节点 SSD/GDS 恢复、远程内存池读取、双路径加载与重新 Prefill 等多种方式的预估完成时延，不可预设单一路径的绝对最优性。

缓存感知路由需要在数据局部性与集群负载均衡之间实现权衡。Preble 依托提示词共享特征提升前缀缓存局部性；Llumnix 通过请求动态迁移消解节点热点；LoongServe 借助弹性序列并行机制适配长上下文推理场景^[47-49]。单纯追求高缓存命中率，易使共享前缀与存储 I/O 集中于少数节点，引发负载倾斜。片面依赖队列负载均衡，则会破坏数据局部性、放大跨节点 KV 传输字节量。面向长期缓存与跨节点缓存部署，需明确缓存兼容规则与安全边界，并依据 KV 重算成本动态配置副本数量。

随着集群规模扩张，控制面调度开销将成为推理路径耗时的重要组成部分。目录更新、路径决策与副本回收流程均会消耗 CPU 资源与网络消息开销。控制周期过短会引发频繁路径切换，抵消缓存复用收益；控制周期过长则无法及时感知并消解存储热点。实际部署中需区分 GPU 资源分配、节点预取调度与集群路径调整的时间尺度差异，通过流量限速、租约管理与版本校验机制，规避多控制器对同一 KV 对象的调度争用问题。

4.6 跨层优化与评价

短上下文、高并发 Decode 负载下，系统优化优先级通常聚焦于分页机制、批处理策略与权重带宽瓶颈；而超长提示词与多轮 Agent 负载场景，需优先统计 KV 外层命中率与

恢复路径时延，并据此择优部署 GDS 加速、主机中转、双路径传输或选择性预取等优化方案。针对显存资源受限场景，可在保障数据格式兼容的前提下，先对 KV 数据进行压缩处理再落地存储，同步缩减缓存容量占用与链路传输字节量。多机制组合部署需按照不可逆优化程度逐级递增推进，同时分别统计存储 I/O、PCIe、计算网络与 GPU 计算的耗时占比，实现精细化性能拆解。

在 KV 存储路径的性能评价体系中，除 TTFT、TPOT 与有效吞吐等常规指标外，还需纳入多维度量化参数，包括各传输路径的字节占比、存储与计算网卡利用率、GDS 及主机中转引入的 CPU 与 PCIe 开销、集合通信 P99 时延，以及路径切换、路径回退的触发次数。消融实验需遵循科学递进顺序：首先固定计算策略与缓存机制，再替换底层 I/O 传输通路，最后叠加双路径调度策略，以此精准区分“精简数据搬运量”与“更换传输通路优化资源争用”两类核心收益。系统成本核算不能仅依托 GPU 数量统计，需要全面涵盖 SSD 带宽、存储网卡、计算网卡与主机内存中转产生的综合开销。典型负载场景的优化优先级与核心评价指标汇总如表 2 所示。

5 未来发展方向

从生命周期控制、压缩表征、跨引擎接口与真实负载评测 4 个维度分析，大模型推理存算协同领域仍存在若干亟待解决的开放性问题，具体如下：

第一，运行时管控机制亟需从单一访问历史统计，拓展至全生命周期管理、使用时限约束与传输路径自适应选择。共享前缀、暂停智能体任务、临时激活值与模型权重具备差异化的重算开销与下次访问时序特征；GPU 内存分配、节点预取调度、I/O 通路切换与集群副本管理需适配不同时间尺度完成精细化管控，规避多控制环路对同一缓存对象的重复迁出、路径调整与重复取回问题，减少无效系统开销。

表2 典型负载场景的优化优先级与核心评价指标

负载类型	首要约束	优先动作	核心指标	主要回退策略
交互式短问答	TTFT/TPOT、权重或 KV 带宽	I/O 重组、可计算压缩、分页与小批调度	TTFT/TPOT 分位数、HBM 带宽、批量大小	降低压缩等级或关闭选择性恢复
长上下文与 RAG	Prefill 算力、KV 写入与恢复	前缀/片段缓存、选择性恢复、预取	命中词元比例、恢复截止时间、TTFT、质量	重新 Prefill 或完整 KV 恢复
多轮 Agent	冷 KV 容量、跨轮恢复与路径拥塞	分层 KV、缓存感知路由、双路径	重用概率、路径字节、P99、回退次数	下沉冷 KV 并限制并发
批量离线生成	吞吐、单位成本与存储带宽	大批量放置、压缩传输、流水重叠	Token/s、GPU 利用率、GiB/Token、能耗	降低并行度或改用本地重算
GPU: 图形处理器 I/O: 输入/输出 RAG: 检索增强生成 TTFT: 首 Token 延迟/首词元延迟 HBM: 高带宽内存 KV: 键值 TPOT: 每输出 Token 时间/每输出词元时间				

第二,亟需构建可直接计算、可跨层传输的统一数据表征体系,并实现与多级存储通路的深度协同。优化压缩格式需支持矩阵乘、注意力等核心计算内核直接读取解析,无需额外格式转换;同时向缓存调度、数据迁移层级暴露位宽特性、分块边界与运行工作区资源需求,有效抑制格式转换、临时副本带来的性能损耗,避免抵消存算协同的路径优化收益。

第三,跨引擎KV缓存复用亟需标准化接口与可靠的异常回退机制。标准接口需统一规范模型与适配器参数、分词器配置、位置编码方式、分块边界、存储布局、数值精度、资源所有权与版本信息;当出现格式不兼容、目录失效、网络超时等异常场景时,系统需具备可靠的主动重算能力。针对数据持久化与多租户共享场景,还需明确访问控制、数据加密、操作审计与资源销毁的语义规范。若无上述统一约束,跨实例缓存复用将局限于单引擎局部优化,无法构建可规模化、可运营的分层缓存数据底座。

第四,性能评测体系需从传统峰值吞吐指标,转向真实负载下的精细化瓶颈解析。测试场景需全面覆盖长上下文推理、RAG检索增强、智能体迭代交互、请求突发到达与多租户混部等真实业务轨迹,评测指标需纳入网络尾时延、预测误差代价、迁移回滚次数、故障恢复耗时与安全管控开销。硬件架构研究表明,存储容量、数据迁移开销与集群可扩展性仍是大模型推理的长期约束条件^[50],单一峰值性能无法等价于真实端到端服务能力。此外,跨模态编码器状态、工具调用中间结果、外部知识库缓存的全生命周期管控规则尚未形成统一标准,其与文本KV缓存的资源置换、收益权衡机制仍有待深入探究。

6 结束语

大模型推理存算协同的核心逻辑,是在数据对象生命周期、输出质量、硬件资源与业务SLO的多重约束下,完成精细化资源置换策略的自适应选择。以算换存依托计算重组、数据压缩与选择性计算机制,降低推理过程的数据物化与内存访问开销;以存换算依靠KV缓存留存与复用策略,规避重复Prefill计算损耗;存算路径优化则通过跨层数据放置、多路径恢复通路与分布式KV数据层部署,扩充有效缓存容量并压缩数据恢复时延。3类优化机制支持叠加组合,其综合收益受HBM容量、CPU算力、PCIe通道、存储网络、计算网络及集合通信的资源争用状态共同制约。针对差异化推理负载,需先精准定位Prefill与Decode阶段的核心约束瓶颈,结合KV外层缓存命中特征,择优采用计算侧压缩、缓存复用或传输通路调优方案。

各类存算协同优化均存在适用边界:容量扩容仅当数据可按时序约束送达计算单元时,方可转化为真实推理性能增益;缓存复用仅在数据恢复开销显著低于重算开销时具备优化价值;跨层数据恢复与推理阶段解耦,仅在路径隔离机制与带宽收益能够覆盖绕行传输、数据交接的额外开销时,可实现正向收益。因此,需围绕数据对象特征、传输路径特性与资源置换规则构建分析框架,在每一轮优化迭代后重新拆解端到端关键路径,将局部技术改进转化为可量化、可解释的推理服务增益。

致谢

感谢华东师范大学、上海创智学院、华中科技大学武汉光电国家研究中心对本研究的支持!感谢华东师范大学闫晗同学、许珂同学、王婧老师在研究过程中给予的协助!

参考文献

- [1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [PP/OL]. arXiv (2017-06-12)[2026-05-26]. <https://arxiv.org/abs/1706.03762>
- [2] Miao X P, Oliaro G, Zhang Z H, et al. Towards efficient generative large language model serving: a survey from algorithms to systems [J]. ACM computing surveys, 2026, 58(1): 1-37. DOI: 10.1145/3754448
- [3] Li H, Li Y, Tian A, et al. A survey on large language model acceleration based on KV cache management [PP/OL]. arXiv (2024-12-27)[2026-05-26]. <https://arxiv.org/abs/2412.19442>
- [4] Dao T, Fu D Y, Ermon S, et al. FlashAttention: fast and memory-efficient exact attention with IO-awareness [PP/OL]. arXiv (2022-05-27)[2026-05-26]. <https://arxiv.org/abs/2205.14135>
- [5] Dao T. FlashAttention-2: faster attention with better parallelism and work partitioning [PP/OL]. arXiv (2023-07-17)[2026-05-26]. <https://arxiv.org/abs/2307.08691>
- [6] Shah J, Bikshandi G, Zhang Y, et al. FlashAttention-3: fast and accurate attention with asynchrony and low-precision [PP/OL]. arXiv (2024-07-11)[2026-05-26]. <https://arxiv.org/abs/2407.08608>
- [7] Ye Z H, Chen L Q, Lai R H, et al. FlashInfer: efficient and customizable attention engine for LLM inference serving [PP/OL]. arXiv (2025-01-02)[2026-05-26]. <https://arxiv.org/abs/2501.01005>
- [8] Jiang H Q, Li Y C, Zhang C, et al. MInference 1.0: accelerating pre-filling for long-context LLMs via dynamic sparse attention [PP/OL]. arXiv (2024-07-02)[2026-05-26]. <https://arxiv.org/abs/2407.02490>
- [9] Liu H, Zaharia M, Abbeel P. Ring attention with blockwise transformers for near-infinite context [PP/OL]. arXiv (2023-10-03)[2026-05-26]. <https://arxiv.org/abs/2310.01889>
- [10] Xiao G X, Lin J, Seznec M, et al. SmoothQuant: accurate and efficient post-training quantization for large language models [PP/OL]. arXiv (2022-11-18)[2026-05-26]. <https://arxiv.org/abs/2211.10438>
- [11] Lin J, Tang J M, Tang H T, et al. AWQ: activation-aware weight quantization for LLM compression and acceleration [PP/OL]. arXiv (2023-06-01)[2026-05-26]. <https://arxiv.org/abs/2306.00978>
- [12] Song Y X, Mi Z Y, Xie H T, et al. PowerInfer: fast large language model serving with a consumer-grade GPU [C]//Proceedings of

- the ACM SIGOPS 30th Symposium on Operating Systems Principles. ACM, 2024: 590–606. DOI: 10.1145/3694715.3695964
- [13] Liu Z, Yuan J, Jin H, et al. KIVI: a tuning-free asymmetric 2 bit quantization for KV cache [PP/OL]. arXiv (2024-02-05) [2026-05-26]. <https://arxiv.org/abs/2402.02750>
- [14] Hooper C, Kim S, Mohammadzadeh H, et al. KVQuant: towards 10 million context length LLM inference with KV cache quantization [PP/OL]. arXiv (2024-01-31) [2026-05-26]. <https://arxiv.org/abs/2401.18079>
- [15] Liu A, Liu J, Pan Z Z, et al. MiniCache: KV cache compression in depth dimension for large language models [PP/OL]. arXiv (2024-05-23) [2026-05-26]. <https://arxiv.org/abs/2405.14366>
- [16] Chang C C, Lin W C, Lin C Y, et al. Palu: compressing KV-cache with low-rank projection [PP/OL]. arXiv (2024-07-30) [2026-05-26]. <https://arxiv.org/abs/2407.21118>
- [17] Zhang Z Y, Sheng Y, Zhou T Y, et al. H₂O: heavy-hitter oracle for efficient generative inference of large language models [PP/OL]. arXiv (2023-06-24) [2026-05-26]. <https://arxiv.org/abs/2306.14048>
- [18] Liu Z C, Desai A, Liao F S, et al. Scissorhands: exploiting the persistence of importance hypothesis for LLM KV cache compression at test time [PP/OL]. arXiv (2023-05-26) [2026-05-26]. <https://arxiv.org/abs/2305.17118>
- [19] Xiao G X, Tian Y D, Chen B D, et al. Efficient streaming language models with attention sinks [PP/OL]. arXiv (2023-09-29) [2026-05-26]. <https://arxiv.org/abs/2309.17453>
- [20] Li Y H, Huang Y B, Yang B W, et al. SnapKV: LLM knows what you are looking for before generation [PP/OL]. arXiv (2024-04-22) [2026-05-26]. <https://arxiv.org/abs/2404.14469>
- [21] Cai Z F, Zhang Y C, Gao B F, et al. PyramidKV: dynamic KV cache compression based on pyramidal information funneling [PP/OL]. arXiv (2024-06-04) [2026-05-26]. <https://arxiv.org/abs/2406.02069>
- [22] Liu X, Tang Z H, Dong P J, et al. ChunkKV: semantic-preserving KV cache compression for efficient long-context LLM inference [PP/OL]. arXiv (2025-10-14) [2026-05-26]. <https://arxiv.org/abs/2502.00299>
- [23] Tang J M, Zhao Y L, Zhu K, et al. Quest: query-aware sparsity for efficient long-context LLM inference [PP/OL]. arXiv (2024-06-16) [2026-05-26]. <https://arxiv.org/abs/2406.10774>
- [24] Xiao G X, Tang J M, Zuo J W, et al. DuoAttention: efficient long-context LLM inference with retrieval and streaming heads [PP/OL]. arXiv (2024-10-14) [2026-05-26]. <https://arxiv.org/abs/2410.10819>
- [25] Sun H S, Chang L W, Bao W L, et al. ShadowKV: KV cache in shadows for high-throughput long-context LLM inference [PP/OL]. arXiv (2024-10-28) [2026-05-26]. <https://arxiv.org/abs/2410.21465>
- [26] Kwon W, Li Z H, Zhuang S Y, et al. Efficient memory management for large language model serving with PagedAttention [PP/OL]. arXiv (2023-09-12) [2026-05-26]. <https://arxiv.org/abs/2309.06180>
- [27] Prabhu R, Nayak A, Mohan J, et al. vAttention: dynamic memory management for serving LLMs without PagedAttention [PP/OL]. arXiv (2024-05-07) [2026-05-26]. <https://arxiv.org/abs/2405.04437>
- [28] Zheng L M, Yin L S, Xie Z Q, et al. SGLang: efficient execution of structured language model programs [PP/OL]. arXiv (2023-12-12) [2026-05-26]. <https://arxiv.org/abs/2312.07104>
- [29] Gim I, Chen G, Lee S S, et al. Prompt Cache: modular attention reuse for low-latency inference [PP/OL]. arXiv (2023-11-07) [2026-05-26]. <https://arxiv.org/abs/2311.04934>
- [30] Yao J Y, Li H C, Liu Y H, et al. CacheBlend: fast large language model serving for RAG with cached knowledge fusion [C]// Proceedings of the Twentieth European Conference on Computer Systems. ACM, 2025: 94–109. DOI: 10.1145/3689031.3696098
- [31] Liu Y H, Cheng Y H, Yao J Y, et al. LMCACHE: an efficient KV cache layer for enterprise-scale LLM inference [PP/OL]. arXiv (2025-10-08) [2026-05-26]. <https://arxiv.org/abs/2510.09665>
- [32] Leviathan Y, Kalman M, Matias Y. Fast inference from transformers via speculative decoding [PP/OL]. arXiv (2022-11-30) [2026-05-26]. <https://arxiv.org/abs/2211.17192>
- [33] Agrawal A, Kedia N, Panwar A, et al. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve [PP/OL]. arXiv (2024-03-04) [2026-05-26]. <https://arxiv.org/abs/2403.02310>
- [34] Sheng Y, Zheng L, Yuan B, et al. FlexGen: high-throughput generative inference of large language models with a single GPU [PP/OL]. arXiv (2023-03-13) [2026-05-26]. <https://arxiv.org/abs/2303.06865>
- [35] Du H C, Wu S Y, Kharlamova A, et al. FlexInfer: breaking memory constraint via flexible and efficient offloading for on-device LLM inference [C]// Proceedings of the 5th Workshop on Machine Learning and Systems. ACM, 2025: 56–65. DOI: 10.1145/3721146.3721961
- [36] NVIDIA. GPUDirect storage overview guide. [EB/OL]. [2026-05-26]. <https://docs.nvidia.com/gpudirect-storage/overview-guide/>
- [37] Wu Y T, Chen S Y, Zhong Y M, et al. DualPath: breaking the storage bandwidth bottleneck in agentic LLM inference [PP/OL]. arXiv (2026-02-25) [2026-05-26]. <https://arxiv.org/abs/2602.21548>
- [38] Zhong Y M, Liu S Y, Chen J D, et al. DistServe: disaggregating prefill and decoding for goodput-optimized large language model serving [PP/OL]. arXiv (2024-01-18) [2026-05-26]. <https://arxiv.org/abs/2401.09670>
- [39] Patel P, Choukse E, Zhang C, et al. Splitwise: efficient generative LLM inference using phase splitting [PP/OL]. arXiv (2023-11-30) [2026-05-26]. <https://arxiv.org/abs/2311.18677>
- [40] Feng J Q, Huang Y K, Zhang R, et al. WindServe: efficient phase-disaggregated LLM serving with stream-based dynamic scheduling [C]// Proceedings of the 52nd Annual International Symposium on Computer Architecture. ACM, 2025: 1283–1295. DOI: 10.1145/3695053.3730999
- [41] Lee W, Lee J, Seo J, et al. InfiniGen: efficient generative inference of large language models with dynamic KV cache management [PP/OL]. arXiv (2024-06-28) [2026-05-26]. <https://arxiv.org/abs/2406.19707>
- [42] Chu, K, Shen, Z, Cheng, S, et al. MCaM: efficient LLM inference with multi-tier KV cache management [C]// Proceedings of 2025 IEEE 45th International Conference on Distributed Computing Systems. IEEE, 2025: 571–581. DOI: 10.1109/ICDCS63083.2025.00062
- [43] Dong Y H, Miao Y B, Li W N, et al. Accelerating LLM inference throughput via asynchronous KV cache prefetching [J]. Proceedings of the AAAI conference on artificial intelligence, 2026, 40(25): 20844–20851. DOI: 10.1609/aaai.v40i25.39224
- [44] Liu Y, Li H, Cheng Y, et al. CacheGen: KV cache compression and streaming for fast large language model serving [C]// Proceedings of the ACM SIGCOMM 2024 Conference. ACM, 2024: 38–56
- [45] Qin R Y, Li Z M, He W R, et al. Mooncake: a KVCache-centric disaggregated architecture for LLM serving [J]. ACM transactions on storage, 2026, 22(4): 1–38. DOI: 10.1145/3773772
- [46] Hu C C, Huang H Y, Hu J H, et al. MemServe: context caching for disaggregated LLM serving with elastic memory pool [PP/OL]. arXiv (2024-06-25) [2026-05-26]. <https://arxiv.org/abs/2406.17565>
- [47] Srivatsa V, He Z, Abhyankar R, et al. Preble: efficient distributed

- prompt scheduling for LLM serving [PP/OL]. arXiv (2024-05-08) [2026-05-26]. <https://arxiv.org/abs/2407.00023>
- [48] Sun B, Huang Z, Zhao H, et al. Llumnix: dynamic scheduling for large language model serving [PP/OL]. arXiv (2024-06-05) [2026-05-26]. <https://arxiv.org/abs/2406.03243>
- [49] Wu B Y, Liu S Y, Zhong Y M, et al. LoongServe: efficiently serving long-context large language models with elastic sequence parallelism [PP/OL]. arXiv (2024-04-15) [2026-05-26]. <https://arxiv.org/abs/2404.09526>
- [50] Ma X Y, Patterson D. Challenges and research directions for large language model inference hardware [J]. Computer, 2026, 59(5): 55-64. DOI: 10.1109/mc.2026.3652916

作者简介



汪俊峰，华东师范大学在读硕士研究生；主要研究方向为大模型存储系统和推理性能优化。



刘璇，华东师范大学在读硕士研究生；主要研究方向为大模型存储系统和多模态数据湖仓。



李亚奇，华东师范大学、上海创智学院在读博士研究生；主要研究方向为智能存储系统、高带宽闪存。



石亮，华东师范大学教授、博士生导师，上海创智学院全时导师；主要研究方向为存储系统和操作系统，涵盖大模型与人工智能、大数据与云计算、嵌入式系统与物联网等领域。