arXiv:2611.28702v1 [cs.PF] 21 Nov 2026

HiPerfCC-KVCache：利用分层键值缓存加速机密大语言模型服务

Haoxuan Xu [1] **Shaoyu Wang** [1] **Minghao Lin** [1] **Beijie Liu** [1] **Potung Yu** [1] **Yongqin Wang** [1]
Vincent Quentin Ulitzsch [2] **Seo Jin Park** [1] **Murali Annavaram** [1] **Mengyuan Li** [1]

摘要

分层键值（KV）缓存通过将可复用状态保留在图形处理器内存之外，减少大语言模型服务中的重复预填充计算。然而，在图形处理器机密计算（机密计算）环境下，受保护传输可能会抵消这一优势。我们发现，大批量 KV 复制在长时间的中央处理器端处理期间会持有驱动程序锁，尽管使用了独立的线程和流，却仍会导致其他服务操作停滞。我们在 Hopper 和 Blackwell 图形处理器上的特征分析表明，锁范围、复制粒度和分配所有权会影响前台进度和传输并行性。基于这些发现，我们提出*HiPerfCC-KVCache*，通过管理资源生命周期，在有界的备份提交之间优先处理就绪令牌回读，并借助共享图形处理器暂存区的辅助上下文并行执行 KV 恢复。*HiPerfCC-KVCache*在 SGLang 中实现，使用现有的受保护传输接口，无需更改硬件或驱动程序。在 H200 上针对三个模型和三种工作负载进行测试时，*HiPerfCC-KVCache* 相较于采用暂存式 KV 传输的 HiCache-Workers 基线，在测试负载最高时实现了 1.74× 的几何平均输出令牌吞吐量加速比（最高达 2.72×）。首个令牌平均时间平均减少 36%，最高减少 59%。该方案在 RTX PRO 6000 Blackwell 上同样有效，相较于相同基线，几何平均吞吐量加速比达到 1.28×（最高达 2.20×）。此外，容量分析还对收益差异以及与关闭机密计算的服务之间仍存在的差距进行了建模和解释。

1 引言

托管大语言模型（LLM）服务引发了人们对敏感输入暴露给特权软件和管理员的担忧（Mohan等人，2024年；Tan等人，2025年）。机密计算（CC）通过硬件强制实施的可信执行环境（TEEs）应对这些担忧。当前的机密GPU平台结合CPU和GPU可信执行环境，以保护服务运行时、主机内存和GPU执行（Gu等人，2026年）。先前的研究表明，当计算始终在受保护的GPU内存中进行时，开销较低（Mohan等人，2024年；Yin& Wang，2026年；Asad & Grunseid，2026年；Pethe等人，2026年）。然而，涉及频繁CPU–GPU传输的工作负载可能产生大量的加密和身份验证开销。

这一挑战对于分层键值（*KV*）缓存尤为重要，它是大语言模型服务中广泛使用的一种优化方法。KV 缓存可避免对共享提示前缀或对话历史的请求进行重复计算（Kwon等人，2023年；Zheng等人，2024年；Gao等人，2024年；Wang等人，2025年）。随着上下文变长、并发度提高，GPU内存容量难以满足需求，分层缓存会将键值存储扩展到主机内存中，使先前计算出的KV状态得以恢复，而无需重新计算（Gao等人，2024年；Liu等人，2025年）。这种方式以CPU–GPU传输替代开销高昂的预填充计算，从而提升服务性能。然而，在GPU 机密计算环境下，这些传输会在共享主机内存中引入开销高昂的软件加密弹跳缓冲区（Apsey等人，2023年；Gu等人，2026年），其开销可能超过KV复用带来的收益。

我们使用 SGLang 进行的测量表明，这种性能反转确实存在。在服务于 Qwen2.5-14B 的多轮工作负载中，HiCache 的两种默认后端相较于关闭 CC（CC-off）时仅 GPU 的前缀缓存，可将首个令牌平均时间（TTFT）缩短 64–69%。然而，在开启 CC（CC-on）时，它们的平均 TTFT 2。0–5.3× 比 CC 开启时仅 GPU 基线慢（第3节）。这引出了一个根本问题：*在 GPU 机密计算下，如何才能保留分层 KV 缓存为大语言模型服务带来的优势？*

现有方法通过预加密、传输预测和运行时调度来降低受保护传输开销。LÆGIS 等架构方案支持预加密，但需要进行 GPU硬件改动（Yang& Jog，2026年）。PipeLLM 通过传输预测使加密与计算重叠（Tan等人，2025年），而 KV缓存恢复依赖未来请求的到达情况和 GPU 缓存驻留，因此难以

[1]南加州大学 [2]麻省理工学院。

预印本。

预测。其他工作通过同步调度减少传输争用，并将令牌回读卸载到后台工作线程以恢复重叠（Yin & Wang，2026年；Asad & Grunseid，2026年）。据我们所知，此前没有论文系统地优化 GPU 机密计算下的分层 KV 缓存。为了解现有技术是否足够，我们构建了 *HiCache-Workers*，这是一个强大的基线，整合了以往工作中适用的软件级优化。具体而言，它结合了 GPU 暂存、专用 CPU 工作线程和 CUDA 流，用于 KV 备份和恢复；同时采用调度策略，在继续服务就绪请求的同时，延后处理等待恢复的请求。这些优化使平均首令牌时间相较于 CC 下 SGLang 的原版 HiCache 直接后端降低了 60%，从 3.39 秒降至 1.35 秒。尽管在我们评估的 CC开启 SGLang 配置中，HiCache-Workers 实现了最高吞吐量和最低平均首令牌时间，但其吞吐量仅达到关闭 CC 的 HiCache 的 77%，且仍然导致 2.6× 相较于关闭 CC 时的 HiCache，平均首令牌时间（第 3 节）。

为了解释这一差距，我们在 Hopper 和 Blackwell 图形处理器上追踪受保护的 KV 传输以及前台服务操作。我们发现，批量受保护复制会在持有与资源管理、流操作和对延迟敏感的前台传输共享的驱动程序锁时，执行持续时间较长的 CPU 端处理（第4节）。因此，即使在不同线程和 CUDA 流之间，后台复制也可能使无关请求陷入停滞。我们的分析进一步表明，复制粒度和分配所有权是影响传输干扰的重要因素。因此，仅提高传输带宽或将复制操作移至后台工作线程并不足够；高效的分层缓存还必须最大限度地减少对前台服务的干扰。

基于这些发现，我们提出了*HiPerfCC-KVCache*：一种面向图形处理器机密计算环境下分层 KV 缓存的争用感知运行时。我们不再仅将受保护传输视为带宽问题，而是 *HiPerfCC-KVCache* 围绕所识别的驱动程序锁结构对其进行组织，使后台 KV 移动既不会阻塞就绪请求，也不会彼此串行化。该方案通过三种机制实现这一点，每种机制分别针对一个争用来源。首先，*HiPerfCC-KVCache* 复用传输资源，并协调其在备份与恢复路径中的生命周期，以减少调度器侧的资源管理停顿。其次，它限制 KV 备份提交，并优先处理延迟敏感的令牌回读，从而限制前台干扰，同时保留传输与计算重叠。第三，它利用辅助 CUDA 上下文来减少争用并并行化 KV 恢复，扩展了此前针对模型加载探索的上下文级并行性（Yin & Wang，2026年）。通过协调内存分配所有权并在上下文之间共享 GPU 暂存缓冲区，*HiPerfCC-KVCache*还消除了恢复期间额外的组装复制。总体而言，这些机制同时解决了受保护传输开销及其对正在进行的服务的干扰。

我们使用现有的受保护传输接口，在 SGLang 中实现了 *HiPerfCC-KVCache* ，并在 H200 和 RTX PRO 6000 Blackwell 上，针对三种工作负载评估了三个模型。在 H200 的最高测试负载下，*HiPerfCC-KVCache* 相较于 HiCache-Workers 实现了 1.74× 的几何平均吞吐量加速比（最高达 2.72×），并将平均首令牌时间平均降低了 36%。一个近似容量模型将计算需求（包括额外的预填充和启动开销）与恢复需求（复制服务和准入等待）分离开来，从而解释了相较于关闭 CC 的性能提升以及剩余差距。在 SWE-bench 上，更长的生成过程使恢复能够与计算重叠，而不会限制吞吐量（恢复未饱和状态），*HiPerfCC-KVCache* 达到了关闭 CC 时吞吐量的 91–102%。NarrativeQA 的高恢复需求使其处于恢复饱和状态：*HiPerfCC-KVCache* 相较于 HiCache-Workers 实现了 2.31–2.72× 的吞吐量加速比，但仅达到关闭 CC 时吞吐量的 65–73%。同样的区别也解释了 Blackwell 上的结果，在那里，*HiPerfCC-KVCache*相较于 HiCache-Workers 实现了 1.28× 的几何平均吞吐量加速比。

本文的贡献如下：

- **CC 下的 HiCache 开销诊断**。我们揭示了受保护的键值传输如何通过驱动程序锁竞争阻塞无关的服务工作，并确定了 Hopper 和 Blackwell 上的复制粒度与分配所有权是另外两个约束因素。

- **竞争感知型 KV 备份**。我们协调资源生命周期并限制备份提交，同时优先处理延迟敏感的令牌回读。

- **上下文并行 KV 恢复**。我们结合辅助 CUDA 上下文与跨上下文缓冲区共享，实现恢复并行化，并避免额外的组装复制。

- **端到端收益与容量分析**。*HiPerfCC-KVCache* 相较于 HiCache-Workers，在 H200 和 Blackwell 上分别实现了 1.74× 和 1.28× 的几何平均吞吐量加速比，并使 H200 上的平均首令牌时间平均降低 36%。消融实验与容量建模解释了这些收益，以及与关闭 CC 相比仍存在的差距。

我们的代码位于 `github.com/sept-usc/HiPerfCC-KVCache`。

工作范围。*HiPerfCC-KVCache*面向现有 GPU 机密计算平台上的单图形处理器大语言模型服务，这些平台使用软件加密弹跳缓冲区进行中央处理器–图形处理器传输，包括 NVIDIA Hopper 和 Blackwell（NVIDIA，2025年）。其仅依赖软件的设计支持在无需修改硬件或驱动程序的情况下完成部署。尽管未来的硬件辅助 TEE-I/O 机制（例如 PCIe 完整性与数据加密（IDE）和 TEE 设备接口 Secu-

rity Protocol（TDISP）能够消除弹跳缓冲区并支持快速多GPU服务，但它们需要额外的端到端硬件和软件支持（Zhao et al.，2025年；NVIDIA，2025年；PCI-SIG，2022）。因此，我们聚焦于现有的基于弹跳缓冲区的机密计算部署，将 TEE-I/O 和多GPU服务留在研究范围之外。

2 背景

2.1 分层 KV 缓存

服务系统会保留键值块，以便在不同请求和对话轮次之间复用（Kwon等人，2023年；Zheng等人，2024年；Gao等人，2024年）。分层缓存通过主机内存和其他存储层扩展了超出图形处理器容量的复用范围（Gao等人，2024年；Liu等人，2025年）。我们重点关注主机层：通过备份将键值从图形处理器复制到主机（设备到主机），而恢复则将其传回（主机到设备）。由于键值块可能散布在图形处理器键值池中，内核会将其收集到图形处理器暂存区，以便批量进行设备到主机复制。恢复时，先进行主机到设备复制，再进行散射（Liu等人，2025年）。

这两条路径都可以与独立计算重叠（Gao等人，2024年；Liu等人，2025年）。除非另有说明，我们采用写穿透式备份，即在图形处理器上缓存块时启动备份（Xie，2025年）。备份完成且不再使用后，无需再次进行设备到主机复制即可回收这些块。对于恢复操作，调度器可以在执行就绪请求的同时推迟处理那些正在等待键值的请求，从而使传输与计算重叠（Ozeri 和 Harnik，2026年）。

2.2 GPU机密计算

NVIDIA GPU机密计算将保护服务运行时和主机私有内存的机密虚拟机（CVM），与保护设备执行和内存的GPU可信执行环境相结合（Apsey等人，2023年；Gu等人，2026年）。

在本文研究的受保护CPU–GPU传输路径中，GPU无法直接通过直接内存访问（DMA）写入CVM私有内存。因此，传输会经过弹跳缓冲区，这些缓冲区位于CVM私有内存保护范围之外的共享主机内存中，并承载经认证的密文（图1）。对于H2D，驱动程序在中央处理器上使用AES-GCM对数据进行加密和认证，GPU复制引擎则对其进行验证和解密。D2H则反转这一过程（Apsey等人，2023年；Gu等人，2026年；Tan等人，2025年）。

威胁模型。 我们沿用 GPU 机密计算平台的威胁模型（Gu等人，2026年），信任 CPU/GPU可信执行环境以及 CVM 内的软件，同时将主机操作系统、虚拟机监控程序和 CPU–GPU互连视为不可信。 *HiPerfCC-* 键值缓存 使用现有的受保护传输接口，不改变这些信任假设。

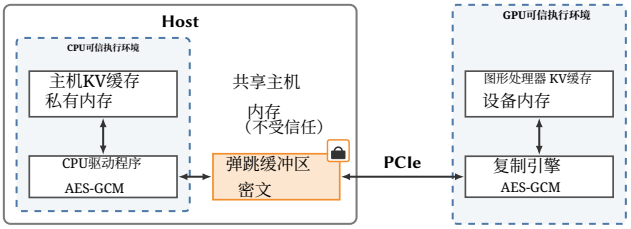


图 *1.* 中央处理器和图形处理器可信执行环境之间的键值移动。可信执行环境之外的有效载荷均经过软件加密和认证。

2.3 CUDA 执行与主机内存

CUDA上下文 负责管理设备资源和执行状态。在一个上下文中，流 对已提交的复制操作和内核进行排序，而事件用于跟踪完成情况并表示流之间的依赖关系（NVIDIA，2026年）。不同流中的独立操作可以并发执行。

这些操作使用的主机缓冲区可通过不同接口分配。*CUDA* 主机内存分配，通过 `cudaHostAlloc` 等接口获得，通常会提供页锁定（固定）内存，而普通主机内存通过 `malloc` 分配后可分页（NVIDIA，2026年）。这些应用程序缓冲区不同于承载密文的共享主机弹跳缓冲区。

主机内存类型和传输方向会影响 `cudaMemcpyAsync` (NVIDIA，2026年) 等异步复制 API 的主机侧行为。尽管这些调用可能在设备传输完成前就已返回，但机密计算中的中央处理器侧加密或解密仍可能使调用线程一直处于忙碌状态（Tan等人，2025年；Yin& Wang，2026年）。因此，仅将拷贝提交到另一条流，并不能将这部分中央处理器侧工作从调用线程转移出去。

3 现有方法的局限

主机 KV 缓存可避免重复进行预填充计算，但在 GPU 机密计算下，受保护的数据传输可能抵消这些收益。我们将考察现有传输配置和运行时优化措施如何恢复这一优势，重点关注相较于关闭 CC 时采用主机缓存的情况，输出令牌吞吐量和平均首令牌生成时间（TTFT）的表现。

我们在单个 H200 图形处理器上运行 Qwen2.5-14B，使用多轮工作负载。可复用的 KV 状态超过图形处理器内存容量，但可存放在主机内存中。默认情况下，中央处理器端的批次准备和输出处理与图形处理器执行重叠。完整设置见附录 A。图 2 展示了不同 SGLang 配置（包括关闭 CC 的配置）下的吞吐量和 TTFT。

传输配置。 我们使用带有 Hi- Cache（Xie，2025年）的 SGLang 作为主要平台。HiCache

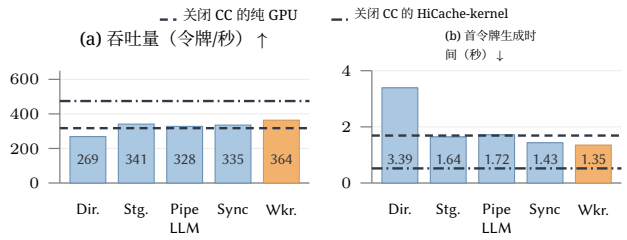


图 *2*。SGLang 中的输出令牌吞吐量和平均首令牌时间。柱状图：CC开启；折线图：关闭 CC。目录/分阶段/工作线程：HiCache-直接/分阶段/工作线程。

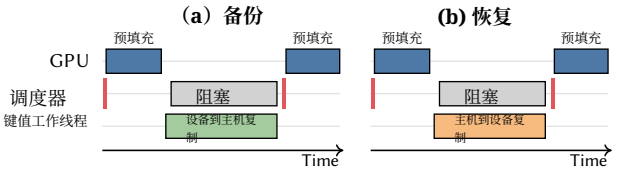


图 *3.* CC开启时 HiCache-Workers 中服务停滞的示意图。备份和恢复使用不同的工作线程，二者均标记为“KV 工作线程”。红色标记表示批量提交。

提供了一种通过 GPU 内核访问主机端 KV 的内核后端，以及一种使用显式`cudaMemcpyAsync`调用的直接后端（SGLang 团队，2026年）。在禁用机密计算的情况下，内核后端的平均首令牌时间低于直接后端，因此我们将其作为主机缓存的参考。在机密计算环境下，两个原生后端相较于仅 GPU 服务均失去了首令牌生成时间优势，不过直接后端的性能优于内核后端。

因此，我们将 GPU 暂存（第2.1节）应用于直接后端，将大量细小的键值复制合并为较大的传输，以进行备份和恢复。由此得到的*HiCache* 分阶段模式与 CC开启的直接后端相比，平均首令牌时间大约减半，但吞吐量仅达到关闭 CC 的 HiCache-kernel 参考实现的 72%，且其平均首令牌时间为 3.2×。附录5 中的表A 报告了详细结果。

使用 PipeLLM 进行推测式加密。PipeLLM 利用规律性的内存交换模式预测即将发生的主机到设备传输，并将其加密与图形处理器计算重叠（Tan等人，2025年）。对于前缀缓存，恢复顺序取决于后续请求的到达情况和图形处理器缓存的驻留状态，因此，仅凭备份顺序无法确定接下来要恢复哪些键值。我们在 SGLang 中将 PipeLLM 与 HiCache 分阶段模式集成。在我们的工作负载下，恢复的键值字节中仅有 0.016% 使用了预加密密文；与 HiCache 分阶段模式相比，吞吐量下降，平均首令牌时间上升。覆盖率如此有限，增加的推测式工作可能抵消恢复路径上的收益。

同步调度。遵循 Serialized Bridge（Yin &Wang，2026年），我们在 HiCache 分阶段模式中禁用重叠调度，得到 *HiCache* 同步模式。调度器会等待 GPU 执行和输出回读完成，然后再处理结果并准备下一批次。这样可以避免延迟输出处理，但会牺牲批次间的 CPU–GPU 重叠。与 HiCache 分阶段模式相比，平均首令牌时间减少约 13%，但吞吐量下降 1.6%。

KV 传输工作线程。继承先前的工作（Yin & Wang，2026年；Asad& Grunseid，2026年），我们的重叠调度配置将阻塞式令牌回读调用卸载到 CPU 响应工作线程。我们将这一方法扩展到键值备份和恢复，在服务上下文中构建了包含独立 CPU 工作线程和 CUDA 流的*HiCache-Workers*。它保留了 HiCache 分阶段模式的图形处理器暂存和重叠调度。调度器会推迟等待恢复键值的请求，以便就绪请求继续处理。

与 HiCache 分阶段模式相比，HiCache-Workers 的吞吐量提升了 6.7%，平均首令牌时间降低了约 18%，在评估的 CC开启 SGLang 配置中实现了最高吞吐量和最低平均首令牌时间。因此，我们在后续分析和评估中将其作为基线。然而，其吞吐量仅达到关闭 CC 的 HiCache-kernel 参考值的 77%，且 2.6× 其平均首令牌时间。

4 揭秘 KV 传输期间的停滞

我们考察了 HiCache-Workers 在 CC开启时为何仍存在显著的首令牌生成时间差距。我们的分析表明，KV 传输期间的服务停滞与 CUDA 驱动程序内部的争用有关。

观察到的服务停滞。图3 总结了 CC开启时 H200 上 HiCache-Workers 服务跟踪记录中的两种停滞模式。在 (a) 中，调度器在备份工作线程执行设备到主机复制期间发生停滞，使计算流在下一次预填充之前处于空闲状态。在 (b) 中，恢复工作线程中的主机到设备复制延迟了另一个请求的预填充提交，尽管该请求的 KV 已经驻留在图形处理器上。这些停滞中断了就绪工作，尽管 KV 复制运行在不同的工作线程和流上。

驱动程序锁竞争。为确定这些停滞期间阻塞调度器的因素，我们在机密计算环境下对 H200 和 RTX PRO 6000 使用了有针对性的 CUDA API 探针和驱动锁跟踪。批量`cudaMemcpyAsync` 和`cudaMemcpyBatchAsync` 调用会获取三类用户空间驱动锁。两个平台都呈现出相同的锁作用域和持有模式。前台 API 探针的详细信息见附录B。

上下文级资源管理（*C*）。在整个主机侧传输处理过程中，副本始终以读模式（C_{R}）持有上下文级读写锁。需要写入访问权限的资源管理操-

表 *1.* 机密计算下前台 CUDA API 类别观察到的 C/S/T 获取。

前台 API 类别	获取的锁
事件/流 创建与销毁	C_{W}
CUDA 设备/主机内存分配与释放	$C_{\mathrm{R}}+S$
内核/图启动、事件记录	$C_{\mathrm{R}}+S+T$
页锁定主机到设备传输和设备到主机复制	

作（C_{W}）（例如事件创建与销毁）无论选择何种流，都会等待。

流提交（*S*）。复制操作会独占其所在流的提交互斥锁，直到主机 API 调用即将返回时才释放。该流上的内核启动、事件记录和后续复制操作都要等待 S，而另一个流上的操作则使用不同的实例。

同一上下文内的同方向传输（*T*）。同方向的页锁定复制操作会跨流共享一个上下文级互斥锁。反方向传输和不同上下文使用不同的 T 实例。单缓冲区调用在整个主机端处理期间都会持有 T。批处理调用会在处理各缓冲区之间释放并重新获取 T，但等待中的前台复制操作可能会在多次获取期间持续阻塞。在未插桩探测中，同一上下文内其他流上的大多数同方向前台复制操作，只有在批次返回后才会完成。

表1 总结了前台操作的锁获取情况。独立流使用不同的 S 实例，但共享 C 和同方向的 T。禁用机密计算时，相同的固定内存复制探针不会获取 T，并会在传输提交后不久释放 C 和 S。在机密计算下，中央处理器加密或解密会延长主机调用及其持锁时间，从而增加冲突操作的等待时间。

对*HiCache-Workers* 进行诊断重放，可以将前台锁等待与批量复制锁持有的重叠，对应到图 3所示的停滞模式。在备份期间，调度器的事件创建与销毁需要 C_{W}，并在备份的 C_{R} 持有期间之后等待。在恢复期间，一次 8 字节的页锁定元数据上传，在另一条流上一次 384 MiB 的主机到设备复制之后，于 T 上等待了 73.7 毫秒。尽管该批次的键值已经位于图形处理器上，这仍会延迟批次启动。

这些锁作用域和持有模式引出了三个运行时选择：何时创建和回收资源、如何限制和排序复制提交，以及哪些传输共享一个上下文。第5 节将这些选择应用于直写备份和按需恢复。

5 *HiPerfCC-KVCache*设计

HiPerfCC-KVCache 组织 KV 传输，以减少对就绪请求的干扰，并加速等待主机缓存的 KV 的请求的恢复。它基于 *HiCache-*工作线程 （第3节），保留重叠调度，并延迟等待恢复的 KV 的请求（ Ozeri 和 Harnik，2026 年）。

图4 概述了*HiPerfCC-KVCache*，其结合三种机制（P1–P3），以协调键值传输与正在进行的服务。在两条传输路径中，*HiPerfCC-KVCache*管理资源生命周期，以减少调度器侧停顿（P1）。对于直写备份，它限制提交数量，并优先处理已就绪的令牌读回（P2）。对于会阻塞请求准入的恢复，它使用辅助上下文来减少驱动程序锁干扰并实现复制并行化（P3）。

5.1 管理资源生命周期（P1）

批量复制会以读模式持有 C，因此需要写访问的事件生命周期操作和主机内存分配可能会使调度器停滞。*HiPerfCC-KVCache* 通过在批次间复用资源并控制其生命周期，减少了这些冲突。

为实现传输同步，*HiPerfCC-KVCache* 会预先创建就绪事件。调度器在 GPU 计算准备好传输的源数据后记录一个事件，复制工作线程则在复制前等待该事件。记录已有事件只需对 C 进行兼容的读访问，而使用单独的流可避免与复制操作争用 S 实例。相应工作线程的等待完成后，该事件便可复用，从而避免在调度器路径上执行创建与销毁操作。

同样的复用策略也适用于每批元数据，例如序列长度。调度器从预先分配的页锁定主机缓冲区上传这些元数据；上传完成后，这些缓冲区即可复用，从而避免重复进行主机内存分配。

分配器回收临时 GPU 数组时，也可能间接触发事件操作。收集和散射操作使用 GPU 索引数组来定位 KV 页，而 HiCache-Workers 会通过分配器流跟踪来防止存储过早复用。后台工作线程会一直持有这些数组的引用，直到使用它们的操作完成。*HiPerfCC-KVCache* 利用这些引用保护数组的生命周期，并省去冗余的跟踪，从而消除可能需要等待 C 或 S 的清理事件操作。

5.2 调度 KV 备份（P2）

令牌回读和 KV 备份在服务上下文中获取同一个 D2H 传输锁 T，即使它们使用的是独立流。在结果处理期间，调度器会等待令牌回读完成。因此，即使图形处理器已经生成了输出令牌，一次耗时较长的备份调用仍可能导致后续批处理调度停滞。*HiPerfCC-KVCache*控制每次备份提交的大小，以及备份工作线程和响应工作线程提交其复制操作的顺序。

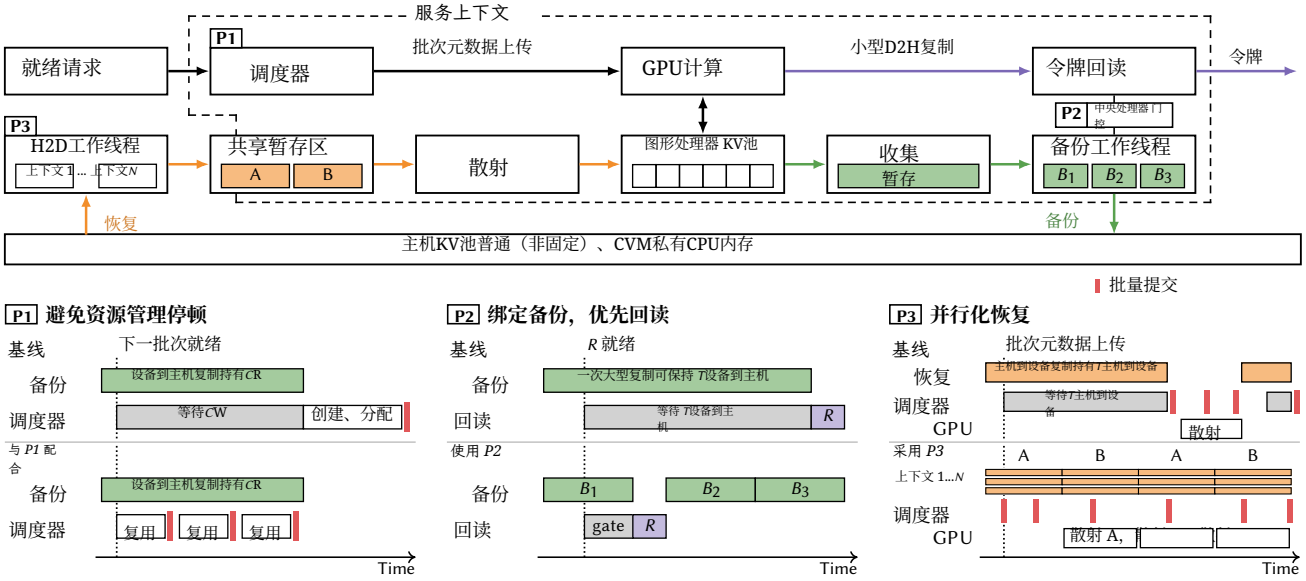


图 4. *HiPerfCC-KVCache* 概览及基线/P1–P3 时间线。灰色：等待。 C_R/C_W：上下文锁读写访问。 T：每个上下文、每个方向的传输锁。A/B：暂存区域。 B_i：备份组。 R：令牌回读。

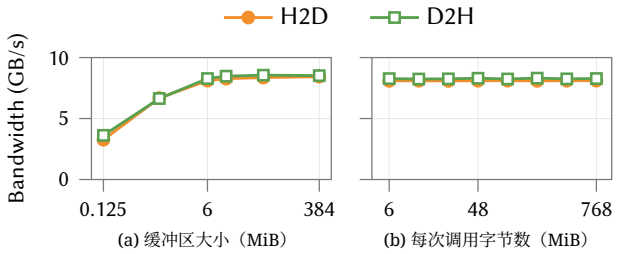


图 5. 在机密计算（CC）环境下，H200 上使用批处理 API 和固定主机缓冲区时的传输带宽。(a) 每次调用 768 MiB，改变连续缓冲区大小。(b) 缓冲区大小为 6 MiB，改变每次调用的字节数。

限制备份提交。 为了限制一次性接纳的备份工作量，同时保留传输带宽，我们将连续缓冲区大小与每次 API 调用提交的总字节数区分开来。对更少的完整缓冲区进行分组，可以在不改变各缓冲区大小的情况下减小提交大小。我们使用 `cudaMemcpyBatchAsync` 在采用机密计算和固定主机缓冲区的 H200 上独立改变这两个维度（附录 C）。

每次调用 768 MiB 时，将缓冲区大小从 128 KiB 增至 6 MiB，可使两个方向的带宽都提升一倍以上（图5(a)）。将缓冲区保持在 6 MiB，同时将每次调用字节数从 768 MiB 降至 6 MiB，可保持约 8 GB/s 的带宽，但会将主机调用时长的中位数从 98–99 ms 降至 0.75–0.78 ms（图 5(b)）。因此，较小的子任务可让运行时更频繁地重新评估备份准入，同时保留带宽。

HiPerfCC-KVCache 将 KV 页收集到 GPU 暂存区，并分组复制整页；字节预算限制每次提交的页面数，同时保留 KV 页的大小和布局。

优先处理就绪的回读。较小的提交会创造更多回读机会，但仅释放 T 并不会让响应工作线程优先于下一次备份提交。在一次连续执行 6 MiB 设备到主机（D2H）子任务的探测中，单次提交的完成时间中位数为 0.70 ms，但回读完成延迟的第 90 百分位数达到 106.4 ms。回读仍处于等待状态时，已有多个备份组完成。

HiPerfCC-KVCache 因此通过由备份工作线程和响应工作线程共享的 CPU 门控来协调提交顺序。只有在就绪事件确认图形处理器（GPU）已生成输出令牌后，响应工作线程才会请求优先级。在此之前，备份可以继续进行。每个工作线程都会持有该门，直到其复制完成，从而将正在执行的备份限制为一个备份组。门释放后，等待中的回读会先于下一备份组执行。

5.3 并行 KV 恢复（P3）

等待主机缓存的 KV 的请求只有在恢复完成后才符合准入条件。服务上下文中的大量主机到设备复制也可能延迟其他就绪批次所需的元数据上传。 *HiPerfCC-KVCache* 采用辅助上下文并行执行恢复，并将恢复操作的 C 和 T 锁与前台操作隔离；这一设计建立在用于模型权重加载的上下文并行机制之上（Yin& Wang，2026年）。

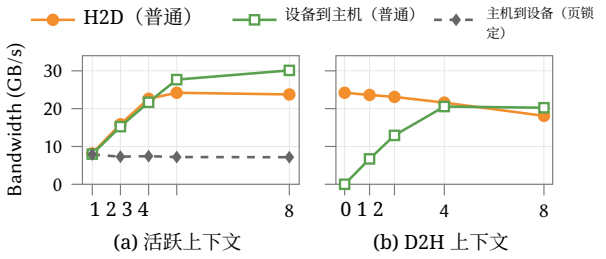


图 *6.* 机密计算环境下 H200 上的上下文并行。(a) 单独执行各个方向。(b) 使用四个主机到设备上下文，并改变设备到主机的并发度。

为了在服务期间进行在线恢复和直写备份，*HiPerfCC-KVCache* 会协调主机分配所有权与传输容量。各上下文共享 GPU 暂存区，使服务上下文能够直接散射恢复后的 KV，从而避免中间的组装复制。

启用并发复制。 共享主机 KV 池的内存分配也会影响复制并行度。在机密计算环境下的 H200 上，来自服务上下文中共享 `cudaHostAlloc`池的 6 MiB 主机到设备复制，在各复制上下文之间仍会串行执行（图 6(a)）。使用普通 CPU 内存时，上下文数量从一个增加到八个，主机到设备带宽便从 8.18 GB/s 提升至 23.76 GB/s。*HiPerfCC-KVCache*因此将主机 KV 池分配在普通 CPU 内存中，并让每个工作线程通过持久辅助上下文和流执行显式受保护的主机到设备复制。使用普通主机内存进行的测量也证实，有界的设备到主机提交能够保持带宽，而提高回读优先级可减少前台延迟（附录 C）。

分配复制容量。并发备份仍会与恢复争夺传输容量。使用四个主机到设备上下文时，将设备到主机上下文从一个增加到八个，虽能提高设备到主机带宽，却会使主机到设备带宽降低 23%（图6(b)）。恢复会限制请求准入，而直写备份可以在必须回收图形处理器页面之前开始。*HiPerfCC-KVCache*因此在服务上下文中使用多个恢复上下文和单个备份工作线程。将备份也置于此上下文中，还能避免占用本可用于存放 KV 状态的额外图形处理器内存。

共享暂存区与流水线恢复。恢复工作线程将主机缓存的 KV 复制到持久的图形处理器暂存分配区，并通过 CUDA 虚拟内存接口将其映射到所有复制上下文和服务上下文。工作线程填充互不重叠的范围，而服务上下文中的散射操作会将 KV 从这些范围直接移入图形处理器 KV 池，从而避免额外的组装复制。

为使主机到设备复制与散射重叠，*HiPerfCC-KVCache* 将这块内存分配划分为两个区域（图4中的 A/B）。工作线程填充一个区域，同时服务上下文对另一个区域进行散射。可重复使用的完成事件可防止某个区域在散射完成前被覆盖。

在每个区域内，散射操作都要等所有主机到设备复制完成，因此工作线程完成时间不均可能导致流水线停滞。为平衡工作量，*HiPerfCC-KVCache* 将连续的主机 KV 范围划分为有界的拷贝单元，并放入共享队列。工作线程依次领取可用单元，使速度更快的工作线程能够处理更多单元，以适应拷贝服务时间的变化。最后一次散射完成后，依赖请求即可获得准入资格。

6 评估

在本节中，我们量化*HiPerfCC-KVCache*在机密计算（CC）下端到端吞吐量和首令牌生成时间（TTFT）的提升（第6.2节），并解释其在不同工作负载和模型上的吞吐量提升及剩余差距（第6.3节）。随后，我们单独分析各项优化的贡献（第6.4节），并评估其在不同机密计算图形处理器平台上的通用性（第6.5节）。

6.1 评估设置

实现与测试平台。我们在 SGLang 0.5.18 中集成 HiCache 实现*HiPerfCC-KVCache*，新增或修改了约 3,800 行运行时代码（不包括测试和评估脚本）。H200 实验在一台 AMD SEV-SNP 机密虚拟机中进行，该虚拟机配备 16 个虚拟 CPU，并通过 PCIe 5.0 连接一块 NVIDIA H200 NVL 图形处理器。开启和关闭机密计算的运行使用相同的软件栈：NVIDIA 驱动程序 610.43.02、CUDA 13.0 和 PyTorch 2.13.0。我们将静态图形处理器内存占比设为 0.9，用于模型权重和键值池；启用主机缓存时，其容量为 500 GB。

模型。我们评估了三种总参数量为 14B–106B 的模型：Qwen2.5-14B-Instruct（Qwen 团队，2024年）、Qwen3.8-27B（Qwen 团队，2026年）和 GLM-4.5-Air（Z.ai，2025年）。它们分别涵盖稠密全注意力、混合注意力和混合专家架构。Qwen 模型使用 BF16 权重，而 GLM 使用 4 位 AWQ 检查点（cyankiwi，2025年）。所有模型均使用 BF16 键值缓存和 131,072 令牌的上下文窗口。

服务配置。 仅 *GPU*保留 GPU 前缀缓存，但禁用主机缓存。*HiCache-Workers* 使用第 3 节所述的分阶段 KV 传输工作线程和延迟恢复准入。 *HiPerfCC-KVCache* 在 HiCache-Workers 的基础上加入 P1–P3。关闭 CC 的配置指仅 GPU 配置，以及采用内核传输和延迟恢复准入的原生 HiCache 配置。所有 CC开启配置均启用后台响应工作线程，所有主机缓存配置均使用写通备份。 *HiPerfCC-KVCache* 使用三个辅助恢复上下文和 32 MiB 拷贝组预算。

表 2。 三个模型共有的工作负载。令牌数表示跨模型的每个请求平均值范围。SWE-bench 大小指完整轨迹。

工作负载	Size	输入令牌	输出令牌
LooGLE	105 篇文档 1,951 个请求	22,083–22 ,914	14–15
NarrativeQA	40 份文档 1,187 个请求	52,797–54,618	6
SWE-bench	665 个会话 23,356 轮	25,185–27,510	212

工作负载。 遵循先前工作（Xie 等人，2026年；Xia 等人，2026年），我们评估文档前缀共享和多轮会话复用。*LooGLE* （Li 等人，2024年）和 *Narra-tiveQA* （Ko˘cisk´y 等人，2018年）在共享文档上提出相互独立的问题，采用泊松到达，且在途请求最多为 128 个。对于 *SWE-bench*，我们重放 LMCache 智能体轨迹（LMCache，2026年），使用固定数量的活跃会话池。每个后续轮次均在上一次响应完成后，并经过记录的轮间延迟后到达。已完成的会话会被替换，以维持会话池规模。表 2总结了三个模型共用的预先准备好的工作负载。

编译预热后，LooGLE 和 SWE-bench 从空缓存开始。NarrativeQA 从空的 GPU 缓存开始；若启用主机缓存，则该缓存会预先填充文档 KV。这突出了恢复过程，与 LooGLE 的初始预填充和缓存填充形成互补。预填充不计入测量。

指标。我们报告输出令牌吞吐量，以及从客户端发出请求起算的平均首令牌时间（TTFT）。文档工作负载在每次负载测试中重放完整请求集一次，包括最后的排空阶段。SWE-bench 运行 1,200 秒。吞吐量统计第 600–1,200 秒内收到的输出令牌，TTFT 则涵盖在该时间段内到达的请求。令牌间延迟（ITL）结果见附录D.1。

6.2 Q1：在机密计算下，*HiPerfCC-KVCache* 能将服务性能提升多少?

在九组模型–工作负载组合中，*HiPerfCC-KVCache*相较于 HiCache-Workers，将吞吐量提高了几何平均值 $1.74\times$ ，并且在每组的峰值负载下，平均 TTFT 降低了 36%；峰值负载是三种主机缓存配置均测得的最高负载。图 7 展示了请求速率或活跃会话数增加时的吞吐量和平均 TTFT。

文档工作负载。 NarrativeQA 的提升最大，因为预先填充的主机缓存使恢复在测量工作中占据显著比重：峰值负载下，吞吐量最高可达 $2.7\times$ HiCache-Workers 的水平，平均首令牌时间缩短了超过一半，且在所有三种模型上都是如此。对于 LooGLE，初始预填充和缓存填充在测量工作中占主导，因此提升较小，但所有模型的表现都有改善。仅 GPU 模式下，没有主机缓存来复用被逐出的键值，在两种工作负载的较低请求速率下都远远落后。

智能体会话。 在 SWE-bench 上，随着会话池扩大，性能提升也随之增加。会话池较小时，几乎所有提示词令牌都命中 GPU 缓存，三种配置的表现几乎相同。随着会话池扩大，仅 GPU 配置的性能急剧下降，HiCache-Workers 的性能也有所下降，而 *HiPerfCC-KVCache* 则能保持更高的吞吐量；在 64 个会话时，其吞吐量达到 HiCache-Workers 的 1.15 倍，且平均首令牌时间更短。

与关闭 CC 的配置比较。 在峰值负载下，*HiPerfCC-KVCache*在 LooGLE 和 SWE-bench 上的吞吐量与关闭 CC 的 HiCache 相差不到 10%。NarrativeQA 是个例外，其吞吐量仅为关闭 CC 配置的 65–73%：它在性能提升方面超过 HiCache-Workers 的幅度最大，同时剩余差距也最大。第6.3 节将解释这两点。

6.3 理解性能提升与剩余差距

我们结合实测服务行为与一个简单的容量模型，解释 *HiPerfCC-KVCache*在哪些方面最有帮助，以及在恢复饱和 和恢复非饱和 场景下，哪些因素仍在限制性能。

分析框架。 该模型估算*HiPerfCC-KVCache*相对于关闭 CC 的吞吐量，并区分初始缓存填充与后续复用。对于每个阶段p，令 c_p 和 ρ_p 分别表示经估算并按该阶段关闭 CC 时长归一化的计算时间和恢复时间。计算项包括 CC 下额外的预填充工作，恢复项则包括复制时间和恢复之间的等待时间。若需求低于 1，则可在参考时长内完成。由于计算和恢复可能重叠，我们用两者中较大的资源需求来近似各阶段的减速幅度。根据各阶段占关闭 CC 时间的比例 f_p 加权后，得到

$$\widehat{R} = \left[\sum_p f_p \max\{1,\, c_p,\, \rho_p\}\right]^{-1}. \qquad (1)$$

当 恢复饱和 时， $\rho_p > \max\{1, c_p\}$，因此恢复会限制吞吐量。否则，该阶段为恢复未饱和。在 63 组配对配置中，平均绝对误差为 1.51 个百分点。校准和支持性测量见附录D.2。

恢复饱和状态。 NarrativeQA 预先填充的主机缓存使恢复在测量工作中占据突出地位，因此，更快的恢复相比 HiCache-Workers 有很大的改进空间。尽管 *HiPerfCC-KVCache*提升了恢复带宽，但所需的键值数据量相对于有效的受保护传输带宽而言仍然很大。在高负载下，由此产生的复制服务时间和

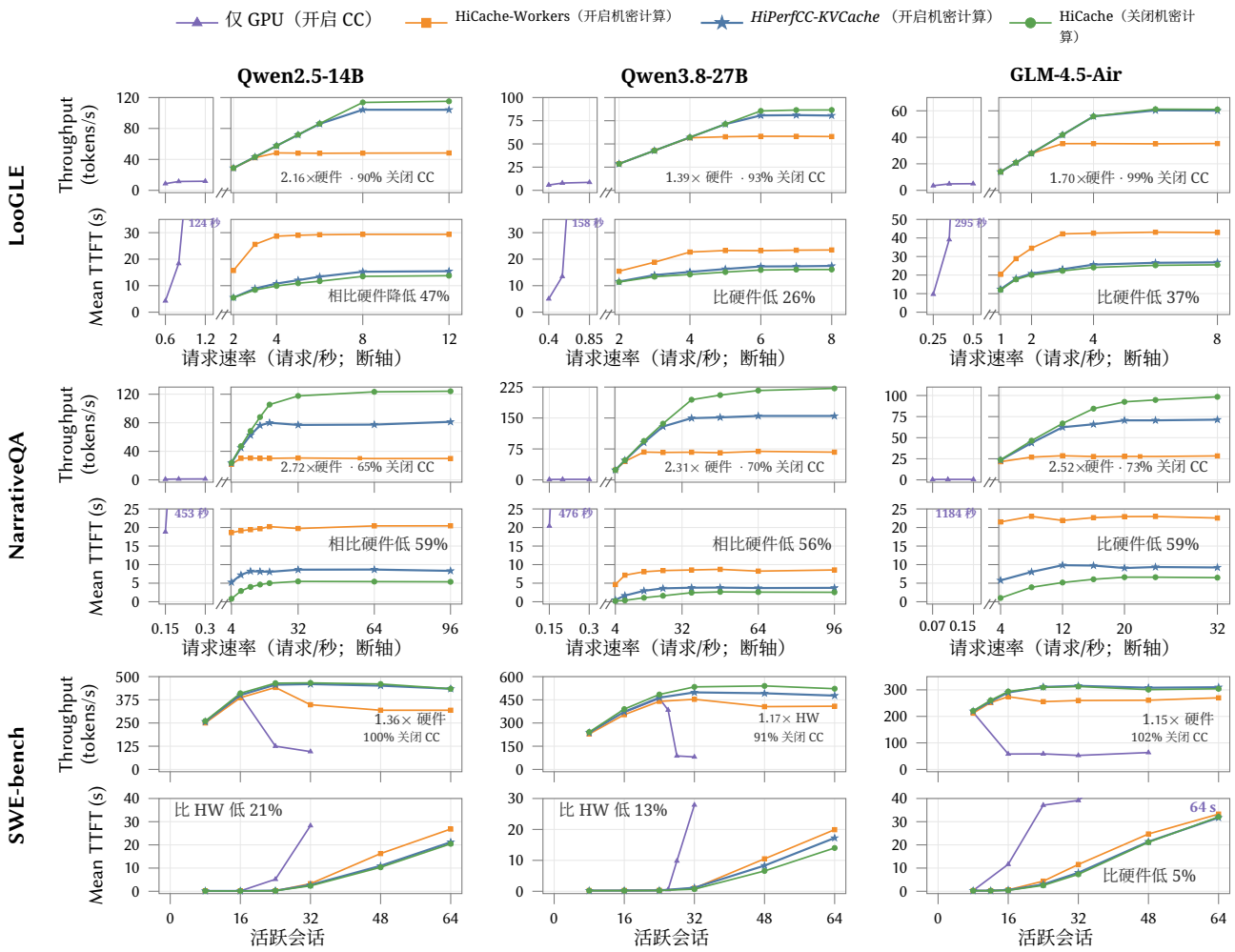


图 7. H200 上的服务性能。标注显示 *HiPerfCC-KVCache* 相较于 HiCache-Workers（硬件）的性能提升，以及峰值负载下相对于关闭 CC 的 HiCache 的吞吐量。紫色标注报告了 GPU-only 在超出图表范围的最高测量负载下的首令牌生成时间。

准入等待时间两者相加后超过了关闭 CC 的重放持续时间（图8（a））。因此，恢复仍然限制着吞吐量，这解释了为何相比 HiCache-Workers 能够取得显著提升，但与关闭 CC 之间仍存在较大差距。

恢复未饱和状态。 SWE-bench 每轮生成约 212 个令牌，而文档工作负载的每个请求生成 6–15 个令牌。这些较长的生成阶段将恢复工作分散到更多服务时间中，并使其能够与正在进行的计算重叠。在测得的 SWE-bench 负载下，估计的恢复需求低于 0.41，这有助于解释为何 *HiPerfCC-KVCache* 的吞吐量接近关闭 CC 时的水平。

在恢复低于容量时，模型将剩余差距归因于计算开销（图8（b））。对于混合式27B模型，仅前向探测显示，短时预填充执行会分别启动许多小型内核，因此对机密计算更高的单次启动开销较为敏感。在 64 个 SWE-bench 会话下，其估计的额外预填充工作量相当于关闭 CC 时间的 5.7%，即使恢复需求保持在容量以下，也会造成剩余差距。

阶段组成的影响。 LooGLE 包含初始预填充和缓存填充，因此其性能提升不仅取决于在备份期间保护前台执行，也取决于加快恢复速度。在峰值负载下，初始阶段占关闭 CC 时长的 74–83%，估算增加的预填充工作仅使该阶段耗时增加 1–4%。因此，后续复用期间的恢复压力只影响重放过程中的较小部分，从而限制了整体差距。

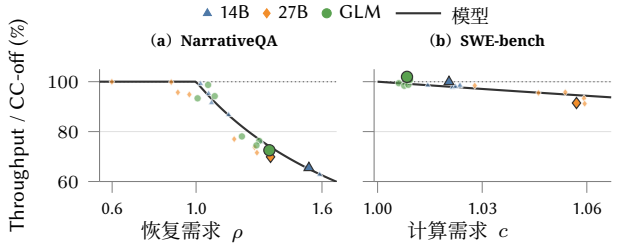


图*8*。 NarrativeQA 和 SWE- bench 上相对于关闭 CC 的吞吐量。曲线显示模型预测值：(a) 中为 $1/\max(1,\ \rho)$，(b) 中为 $1/c$。轮廓线标示每个模型经测试的最高负载。

表 *3*。在 H200 上使用 Qwen2.5-14B、针对 LooGLE（12 请求/秒）进行的传输路径消融。加速比相对于 HiCache-Workers 计算。完整P3 为 *HiPerfCC-KVCache*。

配置	吞吐量（令牌/秒）↑	加速比 ↑	平均首令牌时间（秒）↓
HiCache-Workers	48.3	1.00×	29.4
+ P1	63.3	1.31×	25.7
+ P2	64.8	1.34×	25.1
+ P3 基础版	99.0	2.05×	16.3
+ 仅负载均衡	101.5	2.10×	15.8
+ 仅重叠	101.8	2.11×	15.9
+ 完整P3	**105.3**	**2.18×**	**15.2**
HiCache（机密计算关闭）	115.1	2.38×	13.8

P1/P2 为累积式优化。P3 变体保留 P1+P2，并使用包含三个恢复上下文、普通主机内存和通过 VMM 共享的 GPU 暂存的独立匹配批次，无需组装复制。P3 基础版禁用负载均衡和主机到设备传输与散射重叠。

6.4 问题 2：哪些优化措施带来了收益?

我们在 LooGLE 上以 12 req/s 的负载，评估 Qwen2.5-14B 的 P1–P3 累计增益及各个 P3 组件；此时 *HiPerfCC-KVCache* 的吞吐量已达到饱和（表 3）。

保护前台执行（P1 和 P2）。P1 减少了资源管理停顿，使吞吐量较 HiCache-Workers 提升 31%。P2 限制备份提交，并优先处理就绪的令牌回读，进一步提升 2%。

加速恢复（P3）。P3 基础版将三个辅助恢复上下文与第 5.3 节所述的主机内存和共享暂存组织相结合，使吞吐量较 P1+P2 提升 53%。工作负载均衡和主机到设备传输与散射重叠共同带来额外 6.4% 的提升。其机制在于带宽：恢复字节数除以恢复持续时间总和后，吞吐率从不使用 P3 时的 6.8 GB/s 提升至完整 P3 时的 20.0 GB/s。

6.5 跨CCGPU平台的通用性

跨平台配置。 我们在一块 96 GB 的 NVIDIA RTX PRO 6000 Blackwell Server Edition 图形处理器上重复评估。两台主机的中央处理器均支持 AVX-512；RTX PRO 6000 主机的 Zen 5 核心提供 512 位向量数据通路，

表 *4*。在 RTX PRO 6000 Blackwell 上提供服务。硬件：HiCache- 工作线程；本文方法：*HiPerfCC-KVCache*；关闭 CC：不使用 CC 的 HiCache。

		吞吐量（令牌/秒）↑			平均首令牌时间（秒）↓		
模型	工作负载	HW	Ours	关闭 CC	HW	Ours	关闭 CC
14B	LooGLE	16.2	**21.2**	21.0	100.8	**78.0**	77.6
	NarrativeQA	45.0	**99.1**	110.9	13.8	**5.8**	5.0
	SWE-bench	197.2	**210.6**	211.2	52.2	**48.0**	46.8
27B	LooGLE	45.3	**50.0**	51.3	36.3	**33.1**	32.0
	NarrativeQA	74.9	**83.4**	107.7	8.7	**7.9**	5.8
	SWE-bench	205.2	**203.8**	212.7	47.3	**49.6**	48.3
GLM	LooGLE	13.6	**17.5**	17.1	125.8	**98.0**	99.7
	NarrativeQA	19.7	**35.6**	63.5	36.1	**19.1**	10.8
	SWE-bench	151.7	**158.7**	162.5	72.4	**71.4**	69.6

而 H200 主机的 Zen 4 核心则分两步执行 512 位运算，每步 256 位（AMD，2023年；2025年）。我们沿用第 6.1 节中的配置，使用 30 篇 NarrativeQA 文档以适配主机缓存。各平台的具体设置见附录 D。

服务性能。 在表4中的九组模型–工作负载组合中，*HiPerfCC-KVCache* 实现了几何平均吞吐量加速比 1.28×，比 HiCache-Workers 高，平均首令牌时间也降低了 19%。由于受保护传输路径速度更快，性能增益小于 H200：Zen 5 更宽的向量数据通路可能有助于加快 CPU 端加密；受保护的主机到设备带宽也从 H200 上约 8 GB/s 提高到 17 GB/s，接近翻倍，从而降低了 HiCache-Workers 的恢复开销，留给 *HiPerfCC-KVCache* 可挽回的空间也就更小。

驱动锁作用域与第4节相同，第6.3 节的两种工作状态也同样适用。NarrativeQA 仍处于恢复饱和状态，相较于 HiCache-Workers 的加速比最高，达到 2.2×，与关闭 CC 的差距也最大。SWE-bench 仍处于恢复非饱和状态，吞吐量与关闭 CC 相差不超过 5%。唯一没有提升的组合是 SWE-bench 上的 27B，其性能与 HiCache-Workers 相差不到 1%：平台的单上下文带宽已能满足其恢复需求，因此辅助上下文对吞吐量的提升有限，却会占用 6.3% 的 GPU 键值缓存容量。

7 相关工作

机密 GPU 服务。先前研究指出，机密 GPU 推理中存在传输和框架开销（Park 等人，2026年；Asad &Grunseid，2026年）。后台输出工作线程恢复了令牌读回与计算之间的重叠（Asad &Grunseid，2026年；Yin &Wang，2026年）。Serialized Bridge 评估了针对键值缓存周转的同步调度，并利用上下文并行加速模型加载（Yin & Wang，2026年）。*HiPerfCC-* 键值缓存指出了键值传输驱动锁和内存分配

约束，随后协调资源生命周期、备份和并行恢复，同时保留重叠调度。

降低受保护传输的开销。 PipeLLM 会预测传输内容和初始化向量，以便预先加密数据（Tan等人，2025年）。LÆGIS 提出 GPU 扩展，以灵活管理初始化向量，从而在统一内存迁移期间预先加密页面（Yang 与 Jog，2026年）。Fastrack 将客户端—GPU 安全通道、并行认证和流水线加密相结合（Wang等人，2024年）。*HiPerfCC-KVCache* 通过运行时协调现有 CUDA API，减少受保护的键值传输与前台服务之间的干扰。

8 结论

在 GPU 机密计算环境下，如果受保护的传输使独立的服务工作停滞，分层键值缓存可能无法发挥作用。我们找出了导致这些停滞的驱动锁冲突，以及传输并行性受到的内存分配约束。基于这些发现，*HiPerfCC-KVCache* 利用现有的受保护 CUDA 接口协调资源生命周期、备份提交和并行恢复。在 SGLang 中实现后，*HiPerfCC-KVCache* 在 H200 满负载下相较于 HiCache-Workers 实现了 $1.74\times$ 的几何平均吞吐量加速比，在 Blackwell 上也观察到了性能提升。

参考文献

AMD。第 4 代 AMD EPYC 处理器架构。白皮书，第三版，2023 年 9 月。网址 https://www.amd.com/content/dam/amd/en/documents/products/epyc/4th-gen-epyc-processor-architecture-white-paper.pdf。

AMD。第 5 代 AMD EPYC 处理器架构。白皮书，第一版，2025 年 3 月。网址 https://docs.amd.com/api/khub/documents/UIqhAbjRhgnzgzzdVU4pUw/content。

Apsey、E.、Rogers、P.、O'Connor、M. 和 Nertney、R. 面向安全可信人工智能的 NVIDIA H100 GPU 机密计算。NVIDIA 技术博客，2023 年 8 月。网址 https://developer.nvidia.com/blog/confidential-computing-on-h100-gpus-for-secure-and-trustworthy-ai/。

Asad、A. 和 Grunseid、A. NVIDIA Blackwell GPU 上机密计算性能的基准测试。arXiv 预印本 arXiv:2608.26575，2026年。网址 https://arxiv.org/abs/2608.26575v2。

cyankiwi。GLM-4.5-Air-AWQ-4bit，2025年。URL https://huggingface.co/cyankiwi/GLM-4.5-Air-AWQ-4bit。

Gao、B.、He、Z.、Sharma、P.、Kang、Q.、Jevdjic、D.、Deng、J.、Yang、X.、Yu、Z. 和 Zuo、P。使用 CachedAttention 实现多轮对话的经济高效大语言模型服务。载于 *2024* 年 *USENIX* 年度技术会议（*USENIX ATC 24*），第 111–126 页。USENIX 协会，2024年。URL https://www.usenix.org/conference/atc24/presentation/gao-bin-cost。

Gu, Z., Valdez, E., Ahmed, S., Stephen, J. J., Le, M., Jamjoom, H., Zhao, S. 和 Lin, Z. Blueprint、bootstrap 与 bridge：对 NVIDIA GPU机密计算的安全审视。载于 机器学习与系统会议论文集，第 8 卷，第 1546–1568 页。MLSys，2026年。网址：https://proceedings.mlsys.org/paper_files/paper/2026/hash/906419cd502575b617cc489a1a696a67-Abstract-Conference.html。

Ko˘cisk´y, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G. 和 Grefenstette, E. NarrativeQA 阅读理解挑战。计算语言学协会会刊，6:317–328，2018年。doi：10.1162/tacl a 00023。网址：https://aclanthology.org/Q18-1023/。

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H. 和 Stoica, I. 使用 PagedAttention 为大语言模型服务实现高效内存管理。载于 第 *29* 届操作系统原理研讨会论文集，第 611–626 页。计算机协会，2023年。doi：10.1145/3600006.3613165。URLhttps://doi.org/10.1145/3600006.3613165。

Li, J., Wang, M., Zheng, Z. 和 Zhang, M. LooGLE：长上下文语言模型能否理解长上下文？ 载于 计算语言学协会第 *62* 届年会论文集（第 *1* 卷：长篇论文），第 16304–16333 页。计算语言学协会，2024年。doi：10.18653/v1/2024.acl-long.859。URL https://aclanthology.org/2024.acl-long.859/。

Liu, Y., Cheng, Y., Yao, J., An, Y., Chen, X., Feng, S., Huang, Y., Shen, S., Zhang, R., Du, K., and Jiang, J. LMCache：一种用于企业级大语言模型推理的高效键值缓存层。arXiv 预印本 arXiv:2510.09665，2025年。网址https://arxiv.org/abs/2510.09665v2。

LMCache。LMCache智能体数据集：用于键值缓存基准测试的多轮大语言模型智能体会话，2026年。网址 https://huggingface.co/datasets/sammshen/lmcache-agentic-traces。

Mohan、A.、Ye、M.、Franke、H.、Srivatsa、M.、Liu、Z. 和 Gonzalez、N. M. 云中的人工智能推理安全：CPU-GPU机密计算是否已准备就绪？ 载于*2024*年第*17*届*IEEE*云计算国际会议 *(CLOUD)*，第164–175页。IEEE，2024年。doi: 10.1109/CLOUD62652.2024.00028。网址 https://research.ibm.com/publications/securing-ai-inference-in-the-cloud-is-cpu-gpu-confidential-computing-ready。

Nertney, R. NVIDIA Hopper H100 上的机密计算。白皮书 WP-11459-001，NVIDIA，2023年7月。网址 https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/HCC-Whitepaper-v1.0.pdf。

NVIDIA。《NVIDIA Blackwell和Hopper GPU安全人工智能》白皮书 WP-12554-001，NVIDIA，2025年8月。网址 https://docs.nvidia.com/nvidia-secure-ai-with-blackwell-and-hopper-gpus-whitepaper.pdf。1.3版。NVIDIA。*CUDA*编程指南，2026年。网址 https://docs.nvidia.com/cuda/cuda-programming-guide/。访问于2026年9月。Ozeri, O. 和 Harnik, D. 深入了解 vLLM 全新的 KV 卸载连接器：通过更智能的内存传输最大化推理吞吐量。vLLM博客，2026年1月。网址https://vllm.ai/blog/2026-01-08-kv-offloading-connector。Park, E.、Thans, T.、Kalidasan, V. K.、Hu, Q. 和 Xiong, W. 机密计算GPU上大语言模型服务的实证研究。载于*2026*年*IEEE*系统与软件性能分析国际研讨会（*ISPASS*），第311–322页。IEEE，2026年。doi: 10.1109/ISPASS69572.2026.00038。网址 https://doi.org/10.1109/ISPASS69572.2026.00038。PCI-SIG。TEE设备接口安全协议（TDISP）。工程变更通知（ECN），2022年8月。网址 https://pcisig.com/PCI%20Express/ECN/Base/TEEDeviceInterfaceSecurityProtocol。影响PCIe基础规范5.0版和6.0版。Pethe, S.、Krishnan, V.、Manjunatha, A.、Raza, M. 和 Li, J. 硬件根信任人工智能安全，不会拖慢你的速度。NVIDIA 技术博客，2026年7月。网址 https://developer.nvidia.com/blog/hardware-rooted-ai-security-that-wont-slow-you-down/。Qwen 团队。Qwen2.5技术报告。 *arXiv* 预印本 *arXiv:2412.15115*，2024年。网址 https://arxiv.org/abs/2412.15115。

Qwen 团队。Qwen3.8-27B，2026年。网址 https://huggingface.co/Qwen/Qwen3.8-27B。SGLang 团队。*HiCache* 系统设计与优化，2026年。网址https://docs.sglang.io/docs/advanced_features/hicache_design。访问于2026年9月。

Tan, Y., Tan, C., Mi, Z., 和 Chen, H. PipeLLM：采用推测式流水线加密，实现快速、机密的大语言模型服务。载于 第*30*届 *ACM* 国际编程语言与操作系统架构支持会议论文集，第*1*卷，第843–857页。计算机协会，2025年。doi：10.1145/3669940.3707224。网址 https://doi.org/10.1145/3669940.3707224。

Wang, J.、Han, J.、Wei, X.、Shen, S.、Zhang, D.、Fang, C.、Chen, R.、Yu, W. 和 Chen, H. 大型云服务提供商实际环境中的 KVCache 缓存：KVCache 缓存的特征分析与优化。载于 *2025*年 *USENIX* 年度技术会议（*USENIX ATC 25*），第 465–482 页。USENIX 协会，2025年。网址：https://www.usenix.org/conference/atc25/presentation/wang-jiahao。Wang, Y.、Rajat, R.、Lee, J.、Tang, T. 和 Annavaram, M. Fastrack：使用 GPU 可信执行环境实现安全机器学习的快速 IO。arXiv 预印本 arXiv:2410.15240，2024年。网址：https://arxiv.org/abs/2410.15240。Xia, T.、Li, H.、Li, Z.、Chen, X.、Kang, H.、Qiao, Y.、Xu, Y. 和 Stoica, I. 空闲具有相对性：利用 MORI 在智能体系统中工具调用的空闲窗口进行卸载。*arXiv* 预印本 *arXiv:2606.00866*，2026年。网址：https://arxiv.org/abs/2606.00866。Xie, Z. SGLang HiCache：借助您喜爱的存储后端实现快速分层 KV 缓存。LMSYS 组织博客，2025年9月。网址：https://www.lmsys.org/blog/2025-09-10-sglang-hicache/。Xie, Z.、Xu, Z.、Zhao, M.、An, Y.、Mailthody, V. S.、Mahlke, S.、Garland, M. 和 Kozyrakis, C. Strata：面向长上下文语言模型服务的分层上下文缓存。载于 第 *20* 届 *USENIX* 操作系统设计与实现研讨会（*OSDI 26*），第 1–16 页。USENIX 协会，2026年。网址：https://www.usenix.org/conference/osdi26/presentation/xie-zhiqiang。Yang, Y. 和 Jog, A. LÆGIS：精准定位并解决基于 GPU 的机密计算性能开销。载于 *2026*年 *ACM/IEEE* 第 *53* 届年度国际计算机体系结构研讨会（*ISCA*），第 465–481 页。IEEE，2026年。doi：10.1109/ISCA66397.2026.00046。网址：https://doi.org/10.1109/ISCA66397.2026.00046。

Yin, H. 和 Wang, K. Serialized Bridge：理解并恢复 Blackwell GPU 机密计算环境下的大语言模型服务性能。arXiv 预印本 arXiv:2606.23969，2026年。网址：https://arxiv.org/abs/2606.23969v2。

Z.ai. GLM-4.5-Air，2025年。网址 https://huggingface.co/zai-org/GLM-4.5-Air。

Zhao, S., Gu, Z., Ahmed, S., Valdez, E., Jamjoom, H., and Lin, Z. GPU Travelling：借助TEE赋能的GPU实现高效的机密协作训练。载于 *2025*年*ACM SIGSAC*计算机与通信安全会议论文集（*CCS '25*），第2653–2667页。ACM，2025年。doi: 10.1145/3719027.3765029。网址 https://doi.org/10.1145/3719027.3765029。

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Bar- rett, C., and Sheng, Y. SGLang：高效执行结构化语言模型程序。载于 神经信息处理系统进展，第37卷，第62557–62583页。Curran Associates, Inc.，2024年。doi: 10.52202/079017-2000。网址https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html。

AI 使用披露

Claude Code 和 Codex 在作者的监督下用于协助润色稿件语言、制作图表，以及完成部分代码实现。作者审阅并编辑了 AI 协助生成的文本和代码，并对论文和代码承担全部责任。

表 *5*。H200 动机工作负载下仅 GPU 服务和原生 HiCache 后端的输出令牌吞吐量与平均首令牌时间。

配置	吞吐量（令牌/秒）↑		平均首令牌时间（秒）↓	
	机密计算关闭	机密计算开启	机密计算关闭	机密计算开启
仅 GPU	317	314	1.69	1.71
HiCache-kernel	474	152	0.52	9.15
HiCache 直接模式	470	269	0.61	3.39

A 动机实验详情

服务配置。 第 3 节中的 SGLang 实验 采用与主要评估相同的软件栈。在各种机密计算模式下，只要启用了主机缓存，所有配置均使用容量为 163,840 个令牌的图形处理器键值缓存，以及容量为 500 GB 的主机缓存。在机密计算模式下，主机缓存使用 PyTorch 的固定内存分配器进行 CUDA 主机内存分配，以取代不受支持的 `cudaHostRegister`路径（Nertney，2023年）。

工作负载与测量。工作负载包含 40 个对话会话，每个会话有八轮，初始段落取自 LooGLE。每个会话以 8,192 个输入令牌开始。后续请求保留生成的对话历史，并增加 256 个输入令牌；每个响应生成 256 个令牌。客户端以闭环方式维持八个并发请求，不增加思考时间。在每次测量运行前，我们先发送八个预热请求，然后清空 GPU 缓存以及任何已启用的主机缓存。表格5 报告了各项配置和机密计算模式下的结果。

B 驱动锁测量

锁探测。在 H200 和 RTX PRO 6000 的机密计算模式下，后台线程通过 cudaMemcpyAsync 或 cudaMemcpyBatchAsync 发起页锁定主机到设备传输或设备到主机复制。前台操作在复制流、同一上下文中的另一个流，或单独的预热上下文中运行。我们在每个进程内按锁地址匹配前台等待与后台持有情况，从而识别这两种拷贝 API 使用的相同 C、S 和 T 实例。延迟测量使用单独的未插桩运行和空闲对照。

单缓冲区探测使用 1 GiB 后台复制，并在 10 ms 后调用表6中的前台 API。事件查询/等待探测使用已完成的事件，异步内存分配/释放则使用预热池。批次探测测试事件创建、事件记录、内核启动和小规模复制。关闭 CC 的对照测试涵盖两款图形处理器的单缓冲区调用，以及 H200 上的批次调用。

表 *6*。在 H200 和 RTX PRO 6000 上开启 CC 时，相对于批量 `cudaMemcpyAsync` 测试的前台 API 变体。锁定标记沿用表 1。除非另有说明，复制大小均为 4 KiB。“无”表示未观察到 C、S 或 T 的获取。固定源和目标内存分配后，可分页内存 H2D 阈值仍然存在，表明该阈值取决于复制长度。

API 变体	结果
cudaEventCreateWithFlags, cudaEventDestroy, cudaStreamCreateWithFlags, cudaStreamDestroy, cudaHostAlloc，cudaFreeHost，cudaMalloc、cudaFree	C_{W}
内核启动，cudaGraphLaunch, cudaEventRecord	$C_{\mathrm{R}} + S$
cudaMemcpyAsync：页锁定主机到设备传输，页锁定/-可分页设备到主机传输	$C_{\mathrm{R}} + S + T$
cudaMemcpyAsync：可分页主机到设备传输，复制大小 $\leq$ 64KiB	$C_{\mathrm{R}} + S$
cudaMemcpyAsync：可分页主机到设备传输，复制大小 $>$ 64KiB	$C_{\mathrm{R}} + S + T$
cudaEventQuery, cudaEventSynchronize, cudaStreamQuery, cudaStreamSynchronize, cudaStreamWaitEvent，cudaMemGetInfo，cudaPointerGetAttributes	None
cudaMallocAsync、cudaFreeAsync	None
cudaHostRegister, cudaHostUnregister	不支持

一个批次内的前台进度。 对于表7中的配置，前台拷贝会在后台调用开始后，以均匀采样的偏移量到达。所有拷贝均在后台调用返回前开始。每种配置均跨越两个全新进程。完成时间包含前台流同步。尽管 T 会在缓冲区之间被释放，但在批次返回前完成的前台拷贝仅占 2.5–22.5%。

C 设计实验详情

以下所有测量均使用启用机密计算（CC）的 H200。

拷贝粒度测量。 在图 5中，带宽包含最终的流同步。主机调用时长仅测量拷贝 API 中的时间。每种配置每次运行包含五个测量样本。

前台读回探测。 备份工作线程通过连续的 6 MiB 设备到主机提交传输 1.5 GiB，并在每次提交后进行同步。备份开始后，另一个线程会发出一个 32 字节的

表 7. 机密计算下 H200 上，批处理 API 返回前的前台完成情况。一个 32 字节的页锁定拷贝与同一上下文中另一条流上的 6 MiB 批处理缓冲区沿同一方向竞争。条目报告了 40 次未插桩试验中的计数和百分比。

每批字节数	H2D	D2H
24 MiB	2/40 (5%)	1/40 (2.5%)
768 MiB	5/40 (12.5%)	9/40 (22.5%)

页锁定设备到主机回读请求，位置为随机偏移，使用同一上下文中的另一条流。完成延迟包括流同步。我们汇总了来自三个全新进程的 90 次试验。

普通内存验证。 我们使用普通主机内存作为大块复制的目标位置，重复这些设备到主机探测，同时保持回读缓冲区处于页锁定状态。对于 6 MiB 缓冲区，将每次调用字节数从 768 降至 6 MiB，可在将主机调用时长中位数从约 99 毫秒降至 0.74 毫秒的同时，维持约 8.2 GB/s 的带宽。每种条件进行 90 次试验时，优先级门控可将第 90 百分位的回读延迟从约 129 毫秒降至 0.78 毫秒，其中包括门控等待和流同步。

上下文并行度测量。 每个工作线程都使用单独的 CPU 核心、CUDA 上下文和非阻塞流。图形处理器缓冲区通过 CUDA 的虚拟内存接口共享，并使用互不重叠的主机到设备和设备到主机区域。固定主机内存池使用默认的 `cudaHostAlloc` 标志。图6报告的带宽，是在预热后通过统计共同的 20 秒时间窗口内完成的字节数得出的。

D 评估详情

Blackwell 配置。 第6.5 节中的 RTX PRO 6000 实验沿用了第6.1节的实现、服务配置和测量规则，使用 Flash-Infer 注意力和 330 GB 主机缓存。NarrativeQA 会预先将 30 份文档的键值载入主机缓存，以满足其容量限制。

D.1 令牌间延迟

图9 报告了 H200 评估中客户端观测到的平均令牌间延迟（ITL）。我们汇总第6.1节中用于测量首令牌生成时间（TTFT）的相同请求所产生的连续令牌间隔。

与 HiCache-Workers 的比较。在每个模型–工作负载组合的峰值负载下，与 HiCache-Workers 相比，平均 ITL 的相对降幅在九个组合中平均为 55%，最高可达 95%。这些收益与驱动程序锁停顿的减少一致（第4节）。

与关闭 CC 的比较。在相同的峰值负载下，与原生关闭 CC 的 HiCache 相比，平均 ITL 的相对变化范围为增加 12% 至减少 51%。对文档运行中的服务计数器进行分析后发现，在 ITL 较低的情况下，较慢的受保护恢复会拉开请求准入的间隔，从而产生更小的解码批次和更少的交错预填充。

D.2 吞吐量模型

阶段核算。 当每份文档都生成了首个响应令牌后，我们将每次 LooGLE 运行划分为初始填充和复用。关闭 CC 阶段定义了时长T_p 和权重 $f_p = T_p / \sum_q T_q$，对应于公式 1。相应的 CC 开启阶段决定分配给各阶段的恢复工作。NarrativeQA 和 SWE-bench 均使用一个阶段，涵盖第6.1节中定义的评估区间。

计算需求。 对于模型 m，令 u 作为基线计算需求的代理量，δ_m 作为每个预填充批次额外的 CC时间， n_p 作为阶段 p 中关闭 CC 的预填充批次速率。我们估算 $c_p = u + \delta_m n_p$。对于文档工作负载， u为关闭 CC 时的请求吞吐量除以同一模型–工作负载曲线上的最大值。对于 SWE-bench， $u = 1$近似于一个繁忙的闭环服务器。我们根据八个 SWE-bench 会话下开启/关闭 CC 时每请求平均预填充前向延迟的差值，对 δ_m 进行校准：对于 14B、27B 和 GLM，分别为 10.50、23.46 和 5.86 毫秒。由于其中 97–98% 的批次仅包含一个请求，我们将该差值近似为每个预填充批次的额外时间。

对于 27B，在不进行 KV 传输的情况下，长度为 64 个令牌的预填充会在 CUDA图之外启动约 1,000 个线性注意力内核。未经性能分析的短预填充探测显示，额外 CC延迟约为 17 毫秒，这支持了第 6.3节中关于启动成本的解释。

恢复和准入等待。我们根据主机命中令牌和主机命中请求，分别按比例将测得的 CC开启全程字节数和恢复次数分配到各阶段。将每个阶段的总量乘以其关闭 CC/CC开启请求数比率，再除以 T_p，即可得到恢复字节速率 D_p 和恢复操作速率 r_p。字节数包括混合模型的循环状态。

完成恢复后，在所有者接纳之前，其槽位仍会被占用，因此在复制完成后还会产生额外的平均等待时间 g。我们使用在主动恢复时间内测得的 $B = 18.9$ GB/s来估算 $\rho_p = D_p/B + r_p g$。在假设存在周期性接纳机会且完成时间均匀分布在各阶段的情况下，我们将 g近似为配对CC关闭中位ITL的一半。

比较。 图 10 将模型根据测量输入进行的重构与全部63种配对配置进行了比较。在 B、每个模型的 δ_m以及半ITL规则在各负载下均固定的情况下，

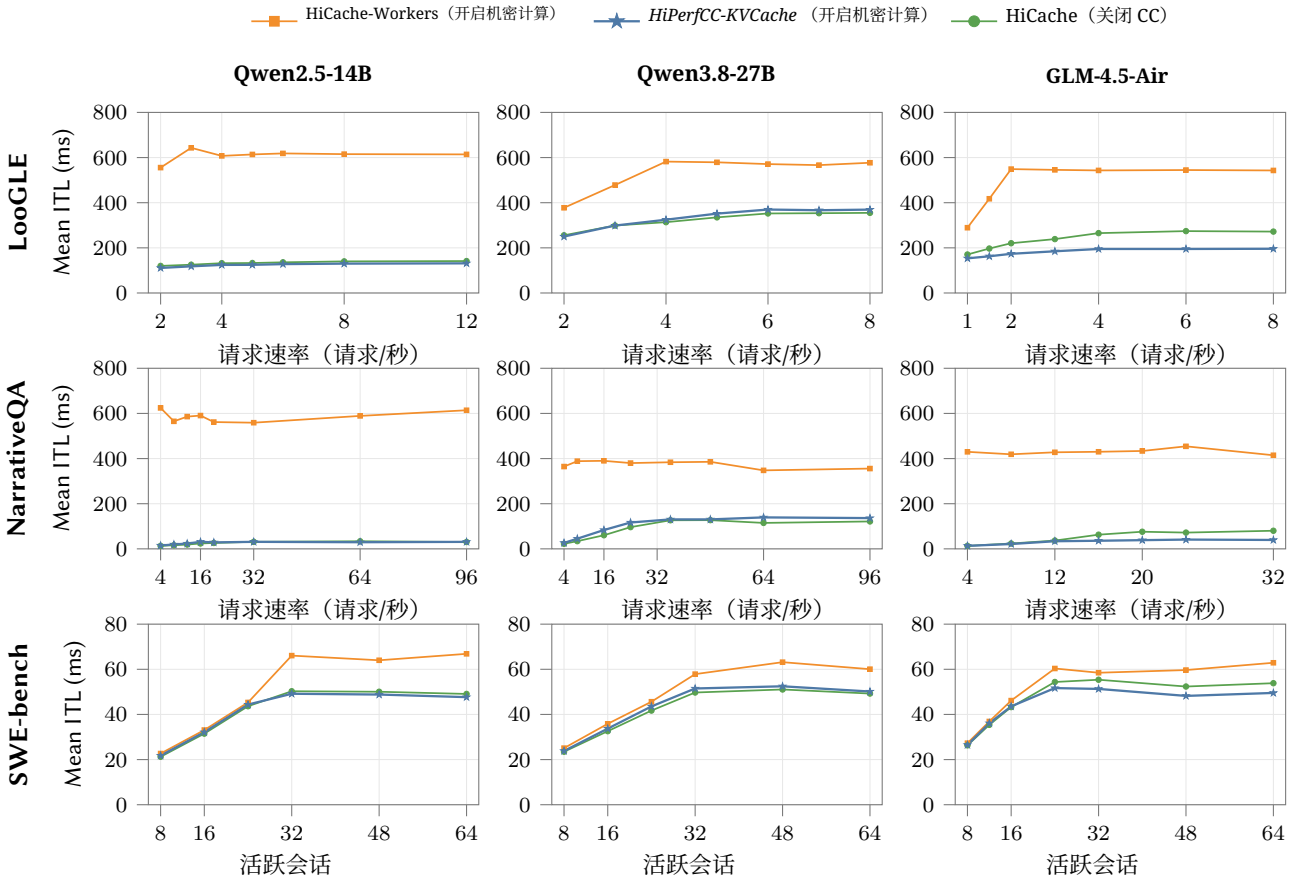


图 *9.* H200 上的平均词元间延迟（越低越好）。曲线展示了三种主机缓存配置下测得的所有负载。

平均绝对误差为1.51个百分点，最大误差为7.63个百分点。该比较使用了测量得到的CC开启恢复工作量，不构成留出验证。

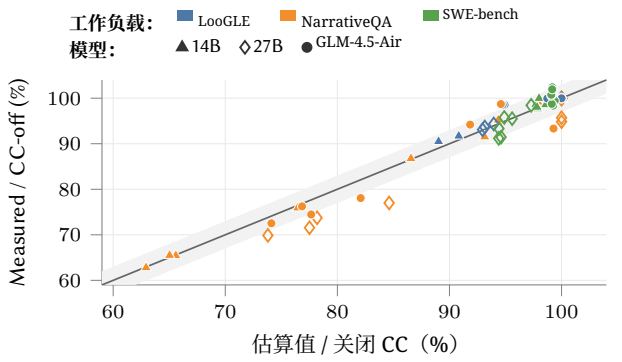


图*10*。所有63种 H200 配置相对于关闭 CC 的输出令牌吞吐量。颜色表示工作负载，形状表示模型。直线表示相等，阴影表示偏差达到 ±3 个百分点。