arXiv:2601.01712v1 [cs.DC] 5 Jan 2026

RelayGR：通过跨阶段接力赛推理扩展长序列生成式推荐

Jiarui Wang*、Huichao Chai*、Yuanhang Zhang、ZongjinZhou、WeiGuo、XingkunYang、Qiang Tang、Bo Pan、Jiawei Zhu、Ke Cheng†、Yuting Yan†、Shulan Wang†、YingjieZhu†、Zhengfan Yuan、Jiaqi Huang、Yuhan Zhang、XiaosongSun、Zhinan Zhang、Hong Zhu、Yongsheng Zhang、Tiantian Dong、Zhong Xiao、DeliangLiu、Chengzhou Lu、Yuan Sun、Zhiyuan Chen、Xinming Han、Zaizhu Liu、YaoyuanWang、Ziyang Zhang、Yong Liu、Jinxin Xu、Yajing Sun、Zhoujun Yu、Wenting Zhou、Qidong Zhang、Zhengyong Zhang、Zhonghai Gu、Yibo Jin✉、Yongxiang Feng✉、Pengfei Zuo✉

华为技术有限公司

摘要

实时推荐系统在严格的尾延迟服务等级目标约束下执行多阶段级联（召回、预处理、细粒度排序），留给排序的时间仅有数十毫秒。生成式推荐（GR）模型能够通过处理较长的用户行为序列来提升质量，但在生产环境中，其在线序列长度受到排序阶段 P99 预算的严格限制。我们发现，大多数生成式推荐令牌编码的是与候选物品无关的用户行为，这表明可以预先推理一次用户行为前缀，并在排序过程中复用，而无需在关键路径上重新计算。要在工业级规模下实现这一构想并非易事：在最终排序实例确定之前，前缀缓存必须跨多个流水线阶段保留；用户群体规模意味着缓存占用远超单台设备的承载能力；而不加选择地进行预推理，则会在高QPS下使共享的CPU/PCIe/NPU资源不堪重负。

我们介绍 RelayGR，这是一种支持生成式推荐（GR）进行*HBM*内接力赛推理的生产系统。RelayGR 有选择地对长期用户前缀进行预推理，在请求生命周期内将其逐层 KV 缓存驻留于 HBM，并确保后续排序阶段无需远程获取即可使用这些缓存。RelayGR 结合了三项技术：（1）序列感知触发器，在有界缓存占用和预推理负载下，仅接纳存在风险的请求；（2）亲和性感知路由器，将辅助预推理信号和排序请求路由至同一实例，使缓存生产与消费在该实例协同进行；（3）内存感知扩展器，利用服务器本地 DRAM 捕获短期跨请求复用，同时避免冗余重新加载。我们在华为昇腾神经网络处理器上实现了 RelayGR，并在生产镜像环境中使用真实查询对其进行评估。在固定的 P99 服务等级目标下，RelayGR 支持长达 1.5× 倍的序列，并将符合服务等级目标的吞吐量最高提升至 3.6× 倍。

*共同贡献。
†在华为实习期间完成。✉通讯作者：{jinyibo1, fengyongxiang1, pengfei.zuo}@huawei.com。

目录

摘要.1 目录.2 1 引言.3 2 背景与动机.5

2.1 实时推荐系统流水线.5 2.2 严格 P99 时延下的生成式推荐.5 2.3 机遇：预推理用户行为前缀.6 2.4 挑战：如何在工业级规模下实现前缀缓存.7

3 RelayGR 设计.8

3.1 系统概述.8 3.2 序列感知触发器.10 3.3 亲和性感知路由器.11 3.4 内存感知扩展器.13

4 性能　　评估.14

4.1 原型与实验设置.14 4.2 Q1：RelayGR 的有效性.16 4.3 Q2：RelayGR 在扩展序列上的表现.17 4.4 Q3：RelayGR 的扩展.18

5 相关工作.20 6 结论.21 参考文献.21

1 引言

现代工业推荐系统 [1, 14, 22] 每天处理数百亿个请求，采用多阶段级联架构，包括召回、预处理（也称为粗排）和细粒度排序。由于用户参与度对延迟高度敏感，整个流程必须在几百毫秒内完成；否则，超时会直接导致收入损失。细粒度排序阶段是瓶颈：在第99百分位（P99）下，它通常只有数十毫秒的时间，需使用高容量模型为数百个候选项评分。

生成式推荐（GR）模型 [2, 3, 6, 10, 16, 21, 28, 40, 41, 45, 46, 48] 正日益被采用为下一代排序模型。与传统深度学习推荐系统（深度学习推荐模型） [5, 15, 34, 35, 49–51],相比，生成式推荐模型能够更好地捕捉长序列行为，并且随着序列更长、模型容量更大，通常展现出更优的扩展效益。离线训练和评估通常会为每位用户使用数千个行为令牌；然而，在在线服务中，增加序列长度（或特征维度）会迅速推高推理延迟，并违反排序阶段 P99 服务等级目标。以往工作要么追求统一召回与排序的端到端生成式推荐 [6, 21, 48] ，但可能需要削弱或移除候选项特定交叉特征 [10],；要么将生成式推荐部署在召回/排序 [10, 29, 46] 中，并将交叉特征作为补偿。在所有情况下，生产部署仍受同一项一阶瓶颈的限制：排序阶段 P99 限制了可用的序列长度，进而限制了生成式推荐在线扩展的效益（图 1）。

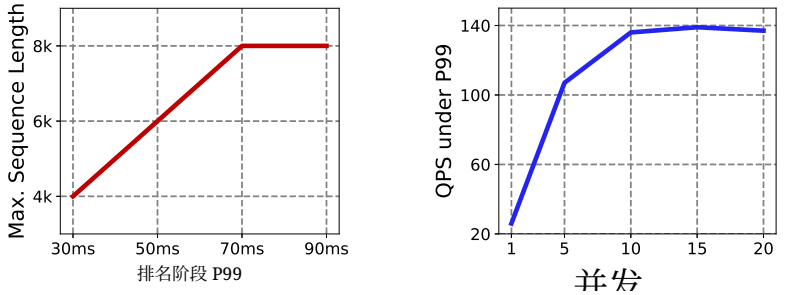
我们的关键观察是，生成式推荐的输入具有一种关键结构特性，即大多数词元编码的是用户行为 [11, 43, 45] ，而非候选项。长期行为（例如数周或数月内的点击和浏览）自然构成一个前缀，其变化速度比短期信号以及后续注入的候选项相关交叉特征更慢。这带来了一个简单的机会：预推理长期用户前缀，缓存其中间状态（逐层 KV），并在细粒度排序中为候选项评分时复用这些状态。这与大语言模型服务中的前缀缓存 [8, 27], 类似，但推荐场景本质上更具挑战性：前缀是用户专属的（不会在不同查询间共享），而且在执行排序之前，缓存必须跨越多个流水线阶段。

要将上述思路转化为生产环境优化，需要解决三个系统挑战。**（1）在延迟绑定的情况下跨流水线衔接缓存**。前缀缓存可能在召回期间生成，但只有在预处理完成后的排序阶段才能使用。创建缓存时，最终的排序实例可能尚未确定；如果生成和使用发生在不同实例上，就必须从远端获取缓存——这对于耗时仅几十毫秒的排序而言，代价往往过高。即使在同一台服务器上，设备内存溢出也会迫使系统刷新并重新加载缓存，从而与排序服务等级目标争用资源。

(2) 用户群体规模使“缓存一切”不可行。工业推荐系统服务数十亿用户 [7, 10, 33]。即使是规模适中的生成式推荐骨干网络 [10, 45] ，在处理数千个词元时，也会为每位用户生成数兆字节的 KV 状态，汇总起来可达拍字节级（数兆字节乘以所有用户数）。朴素的分布式缓存池会频繁触发远程获取，其延迟可能接近甚至超过排序预算，从而使整个方案失效。

(3) 高 QPS 使无条件预推理不安全。在生产环境规模下，每个加速器实例每秒可能要处理数百个查询 [4, 31]。经过特征处理和嵌入查找后，每个查询可能携带数十兆字节的嵌入 [19, 42]。在并发情况下，主机到设备传输和加速器执行会争用共享的 CPU/PCIe/NPU 资源 [52]。如果对每个请求都预推理前缀，就可能使维持排序延迟在 P99 范围内所需的资源过载。因此，预推理必须具备选择性并感知负载。

为应对这些挑战，我们提出了 RelayGR，这是一套支持生成式推荐进行*HBM*内接力赛推理的生产系统。RelayGR 仅在有益时才对用户长期前缀进行预推理，在请求生命周期内将生成的逐层KV缓存保留在设备内存中，并确保



(a) 受限序列长度　　(b) 受限吞吐量

图 1. 排序阶段 P99 限制了性能。

后续排序阶段能够在本地使用缓存，无需远程获取。更广泛地说，RelayGR 解决了一个独特的系统问题：延迟绑定放置下的生命周期缓存，即缓存生产与消费之间隔着多个流水线阶段，且只有在中间过滤之后才能确定消费者。

RelayGR 围绕三项技术构建，针对实时推荐系统共同设计了何时进行预推理、在何处放置缓存，以及如何安全地扩展复用：

序列感知触发器。 在召回阶段，RelayGR 会检查轻量级的用户行为元数据（例如，前缀长度和/或维度），并估计该请求在完整推理下是否有高风险 违反排序阶段 P99。只有高风险请求才会获准进行前缀预推理，从而限制开销，防止其成为新的瓶颈。

亲和性感知路由器。 对于获准的请求，RelayGR 强制执行一项路由契约：后续排序请求必须到达生成缓存的同一实例。该设计利用了请求生命周期仅持续几百毫秒这一事实，使 HBM 能够充当滑动窗口缓存，将每用户前缀保留足够长的时间以供使用（例如启用接力赛），同时消除排序关键路径上的跨服务器获取。

内存感知扩展器。RelayGR 进一步利用服务器本地 DRAM，在不引入远程流量的情况下，捕获同一用户重复请求之间的短期复用（例如快速刷新）。HBM 可保证单个生命周期内的缓存可用性，而 DRAM 则充当受控的补偿层，以有界的主机到设备传输成本将复用范围扩展到 HBM 窗口之外。

我们在华为昇腾神经网络处理器上实现了 RelayGR，并在生产环境约束下使用真实查询对其进行评估。在固定的排序阶段 P99 服务等级目标（也是整个推荐系统的服务等级目标）下，RelayGR 支持最长 1.5× 倍的输入序列，同时将系统吞吐量最高提升 3.6× 倍。

总而言之，本文作出了三项贡献：

- 问题与洞见。 我们发现，排序阶段 P99 是长序列生成式推荐在线部署的首要障碍，并观察到大部分生成式推荐计算都集中在与物品无关的用户行为前缀中。
- 系统设计。 我们提出了 RelayGR，这是一套生产系统，通过序列感知触发器、亲和性感知路由和 DRAM 支持的扩展器，在延迟绑定放置下实现 HBM 内接力赛推理生命周期缓存。
- 生产环境实现与评估。我们在昇腾 NPU 上实现了 RelayGR，并证明在固定的 P99 服务等级目标下，支持的序列长度最多可提升 1.5× 倍，符合服务等级目标的吞吐量最高可提升 3.6× 倍。

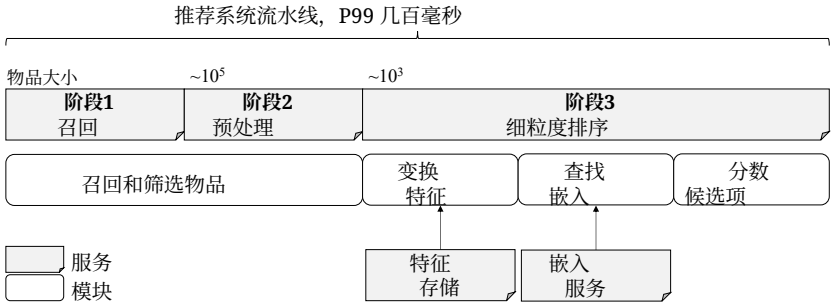


图 2. 多阶段推荐器。

2 背景与动机

实时推荐系统实际上会在严格的尾延迟服务等级目标下执行多阶段级联——召回、预处理和细粒度排序。与此同时，生成式推荐（GR）模型受益于较长的用户行为序列，但生产部署会限制在线序列长度（以及特征维度），以满足排序阶段 P99 预算。本节首先回顾实时流水线（§2.1）和 GR 推理的结构（§2.2），然后探讨预推理用户行为前缀的机会（§2.3），并说明为何将这一机会转化为工业规模优化并非易事（§2.4）。

2.1 实时推荐系统流水线

延迟预算与尾延迟放大。 工业推荐系统处理海量流量，必须将 P99 端到端延迟控制在几百毫秒以内，以避免超时导致的故障和收入损失 [37, 38]。关键在于，延迟预算会在各个流水线阶段之间进行分配，因此留给细粒度排序的时间只有数十毫秒。由于排序以高并发运行，生产部署还会避免在接近峰值利用率的状态下运行：CPU、PCIe 或加速器一旦饱和，就会放大排队延迟并恶化尾延迟，使 P99 合规变得脆弱。

三阶段级联。 典型的推荐系统流水线包括：(i) 召回，用于选取大型候选池；(ii) 预处理/粗排，执行特征变换并筛除候选项；以及 (iii) 细粒度排序，使用重量级模型为剩余候选项打分 [1, 14, 22]。图 2 展示了这一级联流程。候选集大小随着阶段推进而逐步缩小，但排序仍然对延迟至关重要，因为它必须在严格的 P99 时间窗口内为数百个物品打分，同时还要与流水线中的其他工作竞争共享的 CPU/PCIe/加速器资源。

2.2 P99 严格约束下的生成式推荐

生成式推荐为何具有吸引力。生成式推荐模型通过建模用户的长序列行为来提升推荐质量，并且随着序列变长、模型容量增大，表现出良好的扩展性 [2, 3, 6, 10, 16, 21, 28, 40, 41, 45, 46, 48]。在离线训练和评估中，生成式推荐模型可处理每位用户数千个行为令牌，从而学习用户行为轨迹与随时间呈现的内容之间丰富的依赖关系。

生成式推荐有两种常见的部署模式。在判别式模式下，模型会为候选项排序输出分数/逻辑值，实际上是在物品之间建立决策边界 [2, 3, 10]。在生成式模式下，模型逐个令牌生成目标项，或生成供下游塔式网络用于召回或排序的表示 [29, 45, 46]。尽管近期的端到端生成式推荐方案旨在统一召回与排序 [6, 21, 48], , 但在实践中，候选项相关交叉

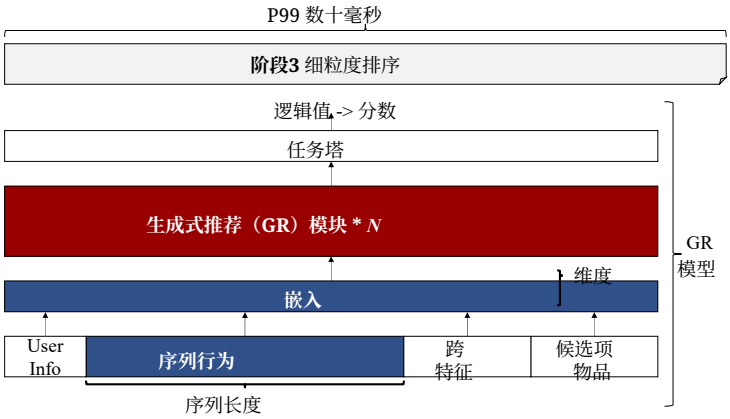


图3. 受限序列（长度或维度）。

特征仍然是质量提升的重要来源 [10]。因此，即便采用 GR 骨干网络，许多部署仍会（显式或隐式地）依赖这些特征。

为什么生成式推荐在在线场景中受限。在线精细排序的 P99 预算只有数十毫秒。增加生成式推荐的序列长度或特征维度会直接提高推理成本，使尾延迟难以维持。因此，生产系统会限制在线序列长度，导致离线训练（数千个行为）与在线推理（通常少得多）之间出现差距 [3, 6, 10, 45, 46]。现有应对措施大多是缩减计算量以适应固定预算，例如压缩词元或表示 [2] （图3）。然而，对于工业部署而言，要实现生成式推荐的扩展效益，还需要具备互补能力：提高在线序列长度上限，同时仍满足相同的尾延迟服务等级目标。

2.3 机遇：预推理用户行为前缀

长期行为构成一个庞大且稳定的前缀。生成式推荐的输入交错包含用户侧上下文和物品目标项。实际中，用户行为按时间排序；长期行为（数月或数年前至数日前）出现在前部，构成一个较长的前缀 [6, 10, 46]。其余词元通常包括短期行为（数小时或数日前）和交叉特征，后面则是待评分的候选项。重要的是，长期行为占据了大部分词元数量（通常有数千个），且变化缓慢；而短期信号和嵌入会频繁刷新，并与紧凑的更新流水线（例如每小时刷新）紧密关联。因此，本文聚焦于预推理仅长期前缀：它涵盖了主要计算量，同时避免与快速的模型/嵌入刷新紧密耦合，并保持生产环境鲁棒性 [17, 19, 36, 39]。

前缀复用：与大语言模型类似，但复用语义有所不同。前缀缓存广泛用于大语言模型推理，以便在请求之间复用计算 [8, 9, 27, 44]。在大语言模型中，前缀通常是共享的（常见的提示词/模板），因此复用发生在跨请求和跨用户之间。在生成式推荐中，前缀主要是特定于用户的：两个用户很少拥有相同的行为历史。因此，最可靠的复用机会是在同一请求生命周期的不同阶段同一用户内复用（也可视情况在同一用户的重复请求之间复用），而非广泛进行跨用户缓存。

排序前预推理。 长期前缀在召回开始时即可获得，且在很大程度上与候选筛选相互独立。这使得“抢先执行”成为可能：在召回期间计算前缀，缓存其中间状态，并在之后对候选项进行排序时复用这些状态以计算分数

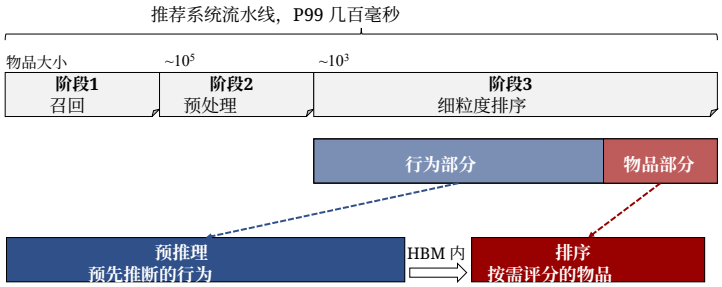


图 4. 预先推断的行为，并在排序时复用。

（图 4）。如果将耗时的长前缀计算从排序关键路径中移除，排序延迟对序列总长度的敏感度就会大幅降低；它主要取决于增量令牌（短期/交叉特征）和物品批次。我们将这一思路形式化为

$$\psi \leftarrow f([\mathcal{U}, \mathcal{S}_l, \emptyset, \emptyset], \emptyset), \quad \left| f([\mathcal{U}, \mathcal{S}_l, \widetilde{\mathcal{S}}_l, \mathcal{I}], \emptyset) - f([\emptyset, \emptyset, \widetilde{\mathcal{S}}_l, \mathcal{I}], \psi) \right| \le \varepsilon,$$

其中，f 是生成式推荐排序模型（生成式骨干网络加下游任务塔）。~输入序列包含用户信息 U、长期行为 S_l、非长期词元 S_l（短期行为 S_s 和交叉特征 C；两者均可能为 $\emptyset$），以及候选项 I。ψ 是通过预推理长期前缀而生成的逐层缓存键值状态。界限 ε 表示使用 ψ 可保留排序分数（仅有微小偏差），与完整推理相比。

2.4 挑战：在工业级规模下实现前缀缓存

预推理长期前缀很有吸引力，但要使其在线有效运行，还需要解决三个系统挑战。

（*1*）延迟绑定放置下的跨阶段缓存存续性。在大语言模型服务中，预填充产生的缓存会立即被解码使用，系统可以在明确的阶段边界 [26, 47]处将其交接。在推荐系统流水线中，生成式推荐前缀缓存可能在召回或预处理期间完成，但只有在细粒度排序开始时才会被使用。在生产环境运行时，最终的排序实例通常是在预处理之后才选定，因此缓存生产方和消费者不仅在时间上分离，在路由上也彼此分离。如果消费者与生产方不同，就需要远程获取；如果缓存溢出到主机内存，设备到主机／主机到设备传输会增加延迟并引发资源争用。无论哪种路径，都可能消耗数十毫秒排序预算中的相当一部分。因此，有效的解决方案必须确保缓存能够跨阶段存续，并在本地被使用。

（*2*）十亿用户规模下的存储占用使“缓存一切”成为不可能。加速器的高带宽内存仅有几十 GB。即使对于相对小型的 GR 模型，当输入包含几千个令牌时，每用户前缀缓存也可能达到 MB 级；在十亿用户规模下，汇总起来必然达到 TB 至 PB 级。这排除了任何假设前缀能在设备内存中为每个用户持久驻留的设计，也使朴素的分布式键值池难以行得通：远程获取延迟可能达到甚至超过排序阶段 P99。因此，在线解决方案必须避免在排序关键路径上进行远程缓存获取，并将缓存视为一种生命周期机制，而非全局持久化存储。

（*3*）高*QPS*和共享资源争用下的运行安全性。生产流量每天达到数百亿次请求，每个实例通常持续承载数百QPS。在这种并发情况下，CPU 特征处理、主机到设备传输（每个请求传输数十 MB 的嵌入）和加速器执行会争用共享资源。对每个请求都无条件进行预推理

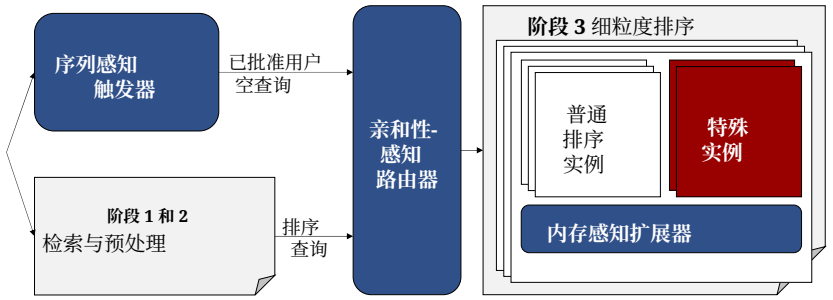


图 5. RelayGR 系统概述。

不可行：这会增加 PCIe/神经网络处理器的压力，并可能加剧排序尾延迟。因此，系统必须（i）仅接纳存在风险的请求，即那些原本会导致排序阶段 P99 超标的请求；（ii）针对具体服务限制接纳的预推理速率和实时缓存占用，在召回阶段可用的余量与排序阶段附近增加的负载之间进行权衡。

启示。 有效的设计必须有选择地对长期前缀进行预推理，确保缓存生产与消费之间的亲和性以消除远程获取，并在高QPS下严格控制额外的计算能力消耗和内存压力。这正是第 §3 节 RelayGR 设计的动机。

3 RelayGR 设计

3.1 系统概述

RelayGR 解决了长序列生成式推荐服务面临的产业瓶颈：排序必须满足数十毫秒的 P99 预算，但大部分生成式推荐计算都集中在用户行为前缀上，该前缀可提前获取，且独立于候选项。RelayGR 引入了一条接力赛旁路，在推荐系统流水线（召回 → 预处理 → 排序）中计算并复用前缀状态，同时不会在排序关键路径上增加远程缓存获取。

抽象：延迟绑定放置下的生命周期缓存。我们将前缀复用建模为生命周期缓存。对于与用户键 u 关联的请求，系统可以提前生成前缀缓存 $\psi(u)$，并在稍后使用：

- **缓存对象**。 $\psi(u)$ 是在 u 的长期行为前缀上计算得到的生成式推荐骨干网络的逐层 *KV*缓存。重要的是， $\psi(u)$ 是前缀令牌和模型权重的确定性函数，不依赖于候选项。
- **生命周期窗口**。令 $T\text{life}(u)$ 表示发出辅助预推理的时间到相应排序使用 $\psi(u)$ 的时间间隔。在生产环境中， $T\text{life}$ 受流水线尾部（召回 + 预处理 +排序）的限制，通常为几百毫秒。
- **延迟绑定放置**。 最终的排序实例是在预处理之后选定的。因此，在生成 $\psi(u)$时，消费端位置尚未确定，除非系统强制执行放置契约。

系统的目标是在 $T_{\text{life}}(u)$ 期间始终提供 $\psi(u)$，并使其在高QPS和设备内存受限的情况下，可在排序时于本地使用。

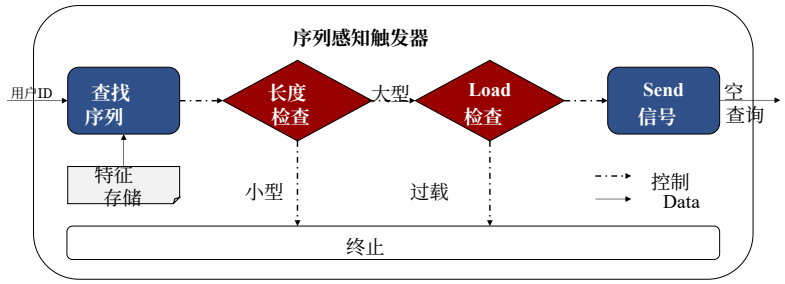


图6. 序列感知触发器的工作流程。

系统不变量。RelayGR 旨在为每个已接纳的长序列请求强制满足两个不变量：

- **（I1）排序关键路径上不进行远程获取。** 排序要么在本地使用 $\psi(u)$（高带宽内存或服务器本地 DRAM），要么安全回退到基线推理；它绝不会因跨服务器缓存获取而阻塞。
- **（I2）在有界占用和负载下保障可存续性。**系统限制 (a) 高带宽内存中的实时缓存占用，以及 (b) 获准的预推理速率，从而确保缓存至少存续 Tlife，同时维持 P99 服务级目标。

分解：准入、放置和本地容量扩展。 如图5所示，RelayGR 将生命周期缓存分解为三项职责：

（1）**准入**（序列感知触发器）。 确定哪些请求应创建 $\psi(u)$（仅限“存在风险”的长序列请求），以及有多少请求可以获准，使其缓存能在高带宽内存和速率预算限制下存续 T_{life}（落实 I2）。（2）**放置**（亲和性感知路由器）。将延迟绑定路由转换为早绑定放置契约：确保同一 u对应的辅助预推理和后续排序请求落在同一个特殊实例上，使排序能够复用 $\psi(u)$，而无需远程获取（落实 I1）。（3）**本地容量扩展**（内存感知扩展器）。利用服务器本地 DRAM，机会性地扩展同一用户重复请求间的复用，同时限制 DRAM→高带宽内存重载开销，并避免并发情况下的冗余重新加载（预先满足 I1 并稳定尾延迟）。

端到端流程。 如果触发器接纳用户 u，它会向特殊的排序服务发出无响应预推理信号，使选定的实例计算能力 $\psi(u)$，并将其驻留于高带宽内存 $T_{\text{life}}(u)$。随后，路由器确保将针对 u的后续排序请求路由至同一实例，从而使排序能够在本地复用 $\psi(u)$。使用后，扩展器可以将$\psi(u)$溢出至服务器本地 DRAM，以加速短期重复请求（例如快速刷新），同时限制重载并发度并采用保持正确性的顺序控制。

总而言之，本文的设计目标是为 GR 前缀复用提供一套系统契约：准入的缓存能够跨越流水线阶段保留，并在排序时严格遵循约束，在本地使用。RelayGR 将计算什么（准入）与在哪里使用（放置）以及如何在本地扩展复用（动态随机存取存储器层）分离开来，将前缀缓存转变为一种可控机制，在保证正确性的同时避免远程获取和尾延迟激增。

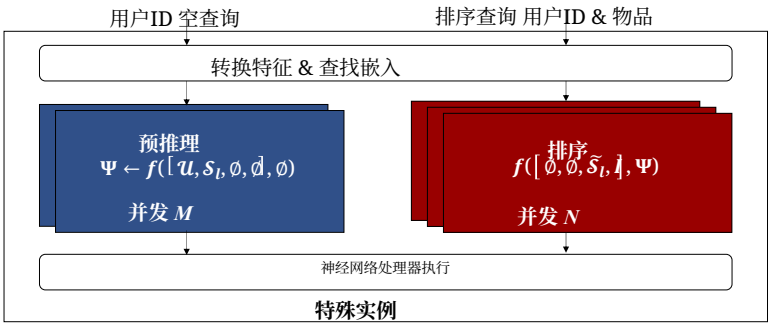


图 7. “特殊”实例内的推理。

3.2 序列感知触发器

设计目标。仅准入有风险的长序列请求进行前缀预推理，并确保其缓存 ψ 能够保留至排序阶段使用，同时将高带宽内存占用和额外预推理负载控制在有限范围内。

使用元数据进行旁路风险测试。 图 6 展示了触发器工作流。触发器与召回并行运行，只获取轻量级行为元数据（例如前缀长度和/或特征维度）。这样可以避免传输完整的行为序列，并保持触发器的快速响应。如果元数据表明完整排序推理不太可能违反排序阶段 P99，触发器便会立即终止，且不会 增加任何额外工作。只有被判定为存在风险的请求（例如前缀较长或特征维度较高）才会成为准入候选项。

通过生命周期窗口可存活性进行准入控制。 只有当排序到达时 ψ 仍然可用，预推理才有益。设 Q_{admit} 为给定特殊实例的准入预推理速率（查询/秒）。在长度为 T_{life}的生命周期窗口内，每个特殊实例中同时存活的 缓存数量上限为

$$L \triangleq Q_{\text{admit}} \cdot T_{\text{life}}. \tag{1}$$

以 kv_{p99} 表示 ψ 的 P99 占用（每个准入用户），并预留 HBM 的 r_1 用于活动缓存，则要确保每个特殊实例都能持续运行，必须满足以下不等式:

$$L \cdot kv_{p99} \le r_1 \cdot HBM. \tag{2}$$

限制预推理负载。设 Q_m 为特殊实例上每个模型槽位可持续承载的预推理吞吐量（查询/秒）， M 为并发模型槽位数。触发器通过以下方式限制准入量:

$$Q_{\text{admit}} \le Q_m \cdot M, \qquad Q_{\max} \le (Q_m \cdot M) \cdot (r_2 \cdot N), \tag{3}$$

其中， N 为排序实例总数， r_2N 个被指定为特殊实例。第一个不等式用于防止单个实例过载；第二个不等式将系统范围内准入的长序列流量限制在特殊实例池的总容量以内（即， r_2 是一个控制变量）。

示例（合理性检查）。如果预推理耗时为 35ms，则每个模型槽位可处理 $Q_m \approx \lfloor 1000/35 \rfloor \approx 30$ 次查询/秒。在 M=5、 kv_{p99}≈0.1GB、 HBM=32GB 和 r_1=0.5 条件下，根据公式 2，每个特殊实例最多可容纳 $L \le 16/0.1 = 160$ 个活动缓存，根据公式 1，对应于 $Q_{\text{admit}} \le 160/T_{\text{life}}$。同时，公式 3 对计算能力设定了上限：每个特殊实例为 $Q_{\text{admit}} \le 30 \cdot 5 = 150$ QPS。在 N=100 和 r_2=0.1 条件下，该资源池可支持 $Q_{\max} \le 1500$ QPS 的准入长序列流量。

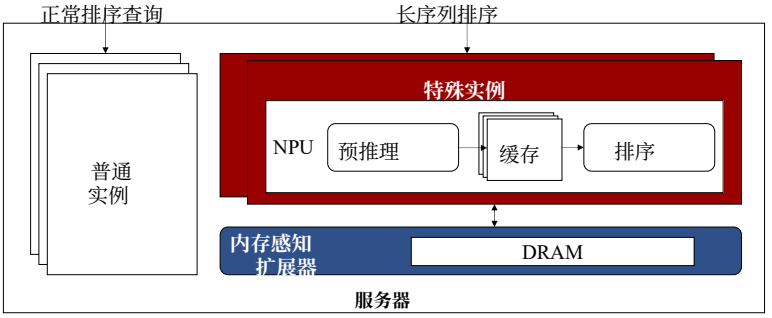


图8. 实例部署。

辅助预推理请求作为控制信号。对于准入请求，触发器会发送一个不带响应的预推理信号，其中携带用户键控一致性哈希：

请求头 { 一致性哈希键: 用户ID, ... } 请求体 { 用户_ID:
用户ID,阶段: 预推理 }

该信号要求特殊实例获取用户的长期行为，执行预推理，并将 ψ 填充到高带宽内存中。由于该信号在召回期间发出，因此不包含候选项（遵循与排序查询相同的格式）。

特殊实例内的并发。 特殊实例会处理来自不同用户的辅助预推理请求和排序请求（图7）。增加 M 可以提升重叠度，但也会加剧计算能力和内存传输引擎上的资源争用；因此，M 是一个可调参数，需与 r_2 和准入阈值联合配置。请求到达时，实例会检查阶段：如果是预推理阶段，就计算并存储 ψ；否则，按用户ID查找 ψ，并在缓存前缀存在时使用该前缀进行排序。</output>

为何有效。 触发器将无界优化转化为有界优化：仅准入那些原本会导致排序阶段 P99 超标的请求，而公式1–3确保准入的缓存能够保留 T_{life}，同时不会使高带宽内存或预推理容量过载（不变量I2）。</output>

3.3 亲和性感知路由器

设计目标。 保证放置：对于每个获准用户，辅助预推理请求及其后续排序请求都路由到同一个特殊实例，以便排序能够复用 ψ，而无需跨服务器获取（不变量I1）。</output>

特殊实例池与干扰控制。如图8所示，RelayGR 在各服务器上部署普通排序实例和特殊排序实例，并通过服务名称加以区分。为了限制共享服务器资源上的干扰，我们限制每台服务器上的特殊实例数量（通常为一个或两个）。这很重要，因为预推理会给以下资源带来较大压力：(i) CPU（用于处理行为以构建模型输入）；(ii) PCIe/H2D 带宽（用于将嵌入从主机DRAM传输到设备HBM）。即使服务器被划分为多个实例，PCIe 仍是共享瓶颈；因此，限制每台服务器上的特殊实例密度，可以降低最坏情况下的资源争用。

独立请求与耦合请求之间的负载均衡。现有负载均衡器和网关会将长序列流量分发到专用实例（图9）。标准策略

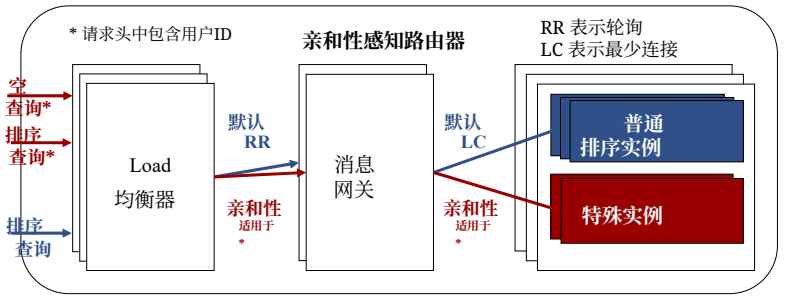


图 9. 通过负载均衡器和网关进行路由。

（轮询、最少连接数）会对独立请求进行负载均衡。然而，RelayGR 会为同一用户引入耦合请求：一个辅助预推理请求和一个后续排序请求。为了复用 ψ，这两个请求必须汇聚到同一个缓存所在实例。

将请求亲和性路由到确切的缓存所在实例。对于长序列用户，排序请求的请求头中包含一个由 ID 派生的一致性哈希键：

请求头 { consistency-hash-key: userID, ... } 请求体 {
user_id: userID,items: ... }

请求体包含排序所需的用户 ID 和筛选后的项目。在预处理期间，系统会确定请求应由普通实例还是专用实例处理：短序列请求使用普通服务，而长序列请求则发送至专用服务。对于专用实例，用户 ID 也用作驻留于 HBM 的前缀缓存的查找键；在请求头和请求体中都包含该 ID 可提高稳健性（例如，对于需要额外处理的缓存未命中回退路径）。

辅助预推理请求和长序列排序请求在请求头中的一致性哈希键字段携带相同的键。负载均衡器和网关分别依据该键进行一致性哈希，以选择网关和最终实例，确保这两个相关请求被路由到同一接收方。如果亲和性遭到破坏（例如部署变动或网络变化），导致排序请求到达不同的实例，该实例只需回退到不使用缓存的推理，从而保证正确性，但会失去优化带来的收益。对于不包含该键的普通排序请求，系统继续采用标准负载均衡策略（例如轮询和最少连接数）。

HBM 作为生命周期窗口。 根据触发器的约束，前缀缓存占用的 HBM 总量有上限。因此，HBM 起到滑动窗口缓存的作用（图 10）：预推理插入每用户前缀缓存，排序请求使用这些缓存，随后随着新准入用户的到来，这些缓存会被驱逐。尽管 HBM 无法容纳所有用户的缓存，但准入控制可确保它能够容纳在单个推荐生命周期内必须保留的缓存——通常为几百毫秒。由于缓存生命周期很短，即使存在模型并发，HBM 容量也足以覆盖所需的窗口。

最后，专用实例占全部排序实例的比例决定了长序列负载的分布方式：提高这一比例可以减轻单实例压力（CPU/PCIe/高带宽内存），并扩大有效窗口；降低这一比例则会将长序列流量集中到更少的实例上。实践中，我们将该比例设得较小（通常为 $< 10\%$），大致与长序列用户所占比例相当，并依靠自动扩缩容在流量增长时维持单实例负载稳定。

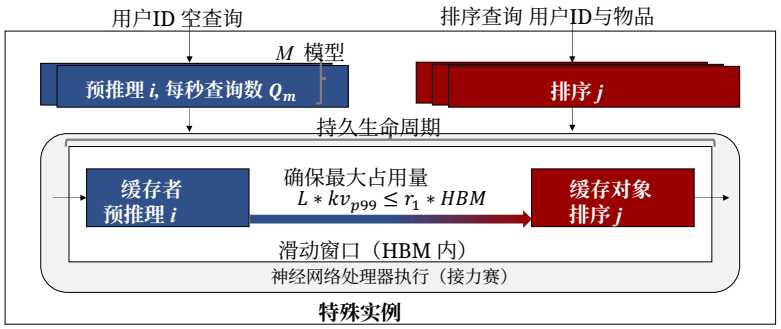


图 10。高带宽内存形成滑动窗口。

其有效原因。 一致性哈希将延迟绑定放置转化为稳定的亲和性契约：生产者和消费者无需协调，便能在同一实例上会合，从而消除排序关键路径上的远程获取（不变量I1）。如果亲和性遭到破坏（例如发生实例变动），系统会安全地回退到基线推理，确保正确性。

3.4 内存感知扩展器

设计目标。 利用服务器本地 DRAM，将来自同一用户的重复请求中 ψ 的复用范围扩展，同时限制 DRAM→高带宽内存重载开销，并防止并发及乱序到达时发生冗余重载。

将动态随机存取存储器用作服务器本地复用层。 高带宽内存中的驻留足以在单个推荐生命周期内衔接缓存：一旦前缀缓存被接纳，就会一直保留到相应的排序使用它为止。然而，同一用户可能会在短时间内触发多次推荐试次（例如快速刷新），而仅靠高带宽内存无法在服务众多其他用户的同时，跨这些试次保留缓存。相比之下，每台服务器的主机DRAM提供数百GB的容量。因此，RelayGR将动态随机存取存储器用作服务器本地复用层：缓存使用完毕后，可以将其溢出到动态随机存取存储器，并在之后重新加载，以加速后续请求，同时避免跨服务器获取。

特殊实例收到排序请求后，RelayGR以用户ID为键执行两级查找：首先检查高带宽内存；若未命中，则查询本地扩展器，以确认是否命中动态随机存取存储器。如果缓存存在于动态随机存取存储器中，该实例便将其重新加载到高带宽内存（产生主机到设备传输），然后继续排序。为防止扩展器成为新的瓶颈，从动态随机存取存储器到高带宽内存的重新加载会受到限速，并以有界并发方式调度。

否则，同一用户的并发请求可能会触发重复的从动态随机存取存储器到高带宽内存的重新加载以及重复的缓存状态转换（溢出/重新加载）。扩展器通过按用户串行化来确保这一点：它维护按用户队列（或锁），并确保每个用户在任何时刻最多只有一个影响缓存的操作正在执行。同一用户的后续请求会等待第一个操作完成，然后复用高带宽内存中的缓存。

处理乱序到达。 实际上，同一??用户的辅助预推理和排序请求可能会乱序到达（图）。例如，如果行为处理较慢，预推理请求可能会延迟，而一个或多个排序请求（候选集各不相同）则会更早到达。简单的设计会导致每个排序请求都独立地从动态随机存取存储器重新加载缓存，从而造成冗余的主机到设备传输，并增加尾延迟。

为处理乱序到达并消除冗余重新加载，RelayGR 会在每个排序请求之前插入一个轻量级的伪预推理 步骤。概念上，每个用户的队列变为：伪预推理 → 排序 → 伪预推理 → 排序 → . . . 伪步骤执行与真实预推理相同的缓存检查：先探测高带宽内存，未命中时再检查动态随机存取存储器。如果缓存位于动态随机存取存储器中，则只有第一个 伪步骤会触发从动态随机存取存储器到高带宽内存的重新加载；之后的所有伪步骤都会命中高带宽内存，并直接进行排序。这样，即使真实的预推理请求迟到（例如，在多个排序请求之后才到达），系统也能确保每个用户最多只重新加载一次，避免冗余传输，同时保证正确性。

原理。 服务器本地 DRAM 在不违反远程获取约束（I1）的情况下，提升了有效缓存容量。每个用户的单航班串行化和幂等伪预推理确保每个用户每次突发请求至多进行一次 DRAM→HBM 重载，从而消除冗余传输，并在并发情况下稳定尾延迟。

4 性能评估

我们评估　在生产镜像推荐系统环境中对 RelayGR 进行评估，以回答三个问题：

- **Q1（有效性）**。 HBM内接力赛推理是否能提升满足尾延迟要求的容量，即最大支持的序列长度和满足服务级目标的吞吐量?
- **Q2（扩展性）**。 随着序列长度增加，RelayGR 是否能够平稳退化，并在相同的 P99 约束下继续提供具有实际意义的吞吐量?
- **Q3（可扩展性）**。扩展 GR 模型（例如，增加嵌入维度和加深骨干网络）以及调整部署参数（例如，候选集大小和并发）时，RelayGR 是否仍然有效?

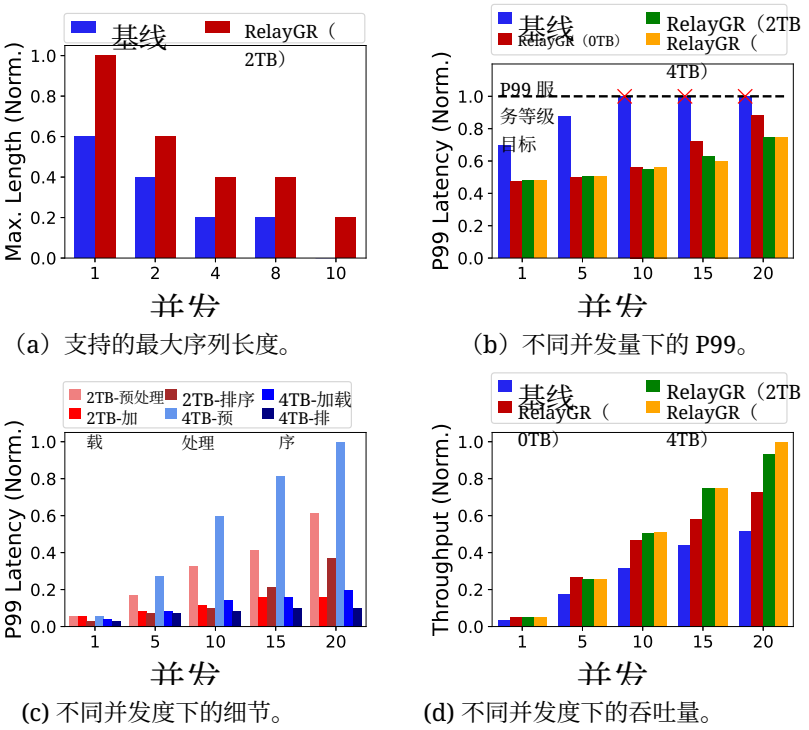
4.1 原型与实验设置

生产镜像环境中的原型。我们将 RelayGR 部署为一组兼容生产环境的服务，在镜像环境中使用真实查询进行评估。推荐系统流水线遵循标准级联流程：召回 → 预处理（粗排） → 细粒度排序。每个阶段通常耗时数十毫秒；流水线级尾延迟服务等级目标为 P99≤ 135 毫秒，而细粒度排序是限制最严格的阶段（例如，P99 时的预算为 ~50 毫秒）。细粒度排序包括：（i）特征处理，（ii）通过外部嵌入服务进行嵌入查找，以及（iii）在神经网络处理器上进行 GR 推理。

部署与路由基础设施。请求通过与生产环境相同的负载均衡器和消息网关转发；我们对其进行扩展，以支持通过用户键控一致性哈希键实现亲和性感知路由。网络按租户隔离，而负载均衡器和网关由多个服务共享。我们部署多个负载均衡器和网关实例，以匹配生产环境中的扇出和争用模式。

排序实例与“专用”实例。细粒度排序使用两种服务名称：普通实例和专用实例（用于处理超长序列，例如长度超过 4K 等配置阈值的序列）。每个排序实例（普通或专用）都映射到一台配备数十 GB 高带宽内存的昇腾 910C 神经网络处理器。实例通过结合 CPU 侧并行处理（特征/序列处理）与神经网络处理器侧模型并发（多个模型槽位），并发处理多个请求。序列感知触发器控制准入的预推理负载，从而决定发送到专用实例的有效每秒查询数。

模型与工作负载。我们评估了多种实际应用中的生成式推荐（GR）架构及其变体（例如 HSTU [45] 及其修订版，以及我们的内部 GRblocks）。在各项实验中，序列



（a）支持的最大序列长度。 （b）不同并发量下的 P99。

(c) 不同并发度下的细节。 (d) 不同并发度下的吞吐量。

图11. RelayGR 有效性的评估。

长度从 1K 到数万词元不等；嵌入维度从 128 到 1024 不等；生成式骨干网络包含数层至数十层。这些设置在长序列场景下因浮点运算次数和内存开销高昂，对排序关键路径构成了压力。用户行为长度的分布显示，大多数用户的历史记录较短，只有不到 6% 的用户拥有超过 2K 词元的长序列。即使只缓存这些长序列用户的数据，也需要数十太字节的存储空间，远超任何单台设备的容量。

缓存语义。缓存对象 ψ 是由生成式骨干网络根据长期行为前缀计算得到的逐层 KV 缓存（例如，在 HSTU 中，针对包含 8 层、256 维嵌入且采用 fp32 格式的模型，处理 2K 令牌的请求需要 ~32MB）。RelayGR 在请求生命周期窗口内将 ψ 保留在高带宽内存中，并可选择将其溢出到服务器本地 DRAM，以支持短期跨请求复用。

指标与测量协议。我们报告：(1) 最大支持序列长度，定义为满足流水线 SLO（P99 ≤ 135 ms）且在所有评估查询中**成功率**达到≥ 99.9%的最大序列长度；(2)符合服务等级目标的吞吐量（每秒查询数），包括每个特殊实例和系统级别的吞吐量，测量时采用相同的 P99 约束；以及 (3) 预处理（预推理）、加载（从 DRAM→加载缓存至 HBM）和排序（使用缓存的排序）的组件延迟（P99）。我们区分每秒查询数（每秒完成的查询数）与并发度（同时处于执行中的查询数），因为后者是资源竞争和尾延迟的关键驱动因素。

基线与变体。基线是不采用接力赛的生产配置：细粒度排序在排序阶段 P99 的约束下内联执行完整的 GR 推理。*RelayGR* 表示不进行 DRAM 复用的 HBM 内接力赛推理（DRAM命中率为 0%）。*RelayGR +x%* 表示增强后的 RelayGR

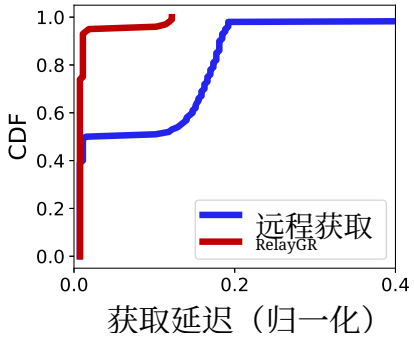


图 12. 本地获取（RelayGR）与远程获取的比较。

，采用服务器本地 DRAM 复用，其中 x%为测得的 DRAM 命中率（由用于溢出的可用 DRAM 预算控制，通常为 500GB，最高为 4TB）。

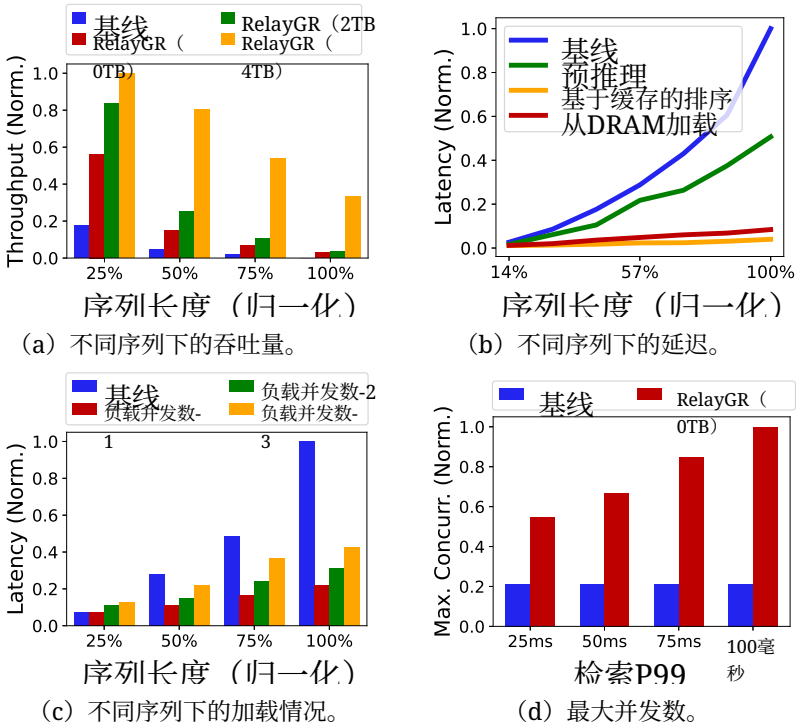
4.2 Q1：RelayGR 的有效性

最大支持序列长度。图 11a 显示，与基线相比，RelayGR 提高了满足 P99 服务等级目标的最大序列长度。通过服务器本地 DRAM 重用，避免了来自同一用户的请求突发期间重复进行预推理，从而进一步延长了最大支持长度。在我们的设置中，使用所有可用的 DRAM（~500GB）可实现约 10% 的 DRAM 命中率，并使最大支持序列长度较基线最多增加 **1**。**5**×。（我们将通过额外层级（例如 SSD）实现更高命中率作为扩展方向进行讨论；本文的核心结果聚焦于 HBM/DRAM，即 2TB 和 4TB 分别达到 ~50% 和 ~100% 的命中率。）

并发下的尾延迟。 图 11b 报告了固定序列长度下不同并发级别对应的端到端 P99。随着并发增加，基线很快违反流水线SLO，因为完整的 GR 推理仍处于排序关键路径上。RelayGR 将代价高昂的长前缀计算转移到中继竞速路径，从而在我们的实验中使在不超过 P99 服务等级目标的情况下可支持的在途请求数增加约 **2**×（在生产环境中也得到了确认）。进一步提高动态随机存取存储器命中率，可减少专用实例上的计算量，从而在相同并发度下提升尾延迟裕量。

节省来自何处。 图 11c 将 P99 延迟分解为预推理、加载和排序三个部分。由于需要处理长期前缀，预推理成本会随着序列长度快速增长；这正是必须将其移出排序关键路径的原因。相比之下，加载和排序的增长速度要慢得多：缓存加载与缓存大小呈线性关系，而基于缓存的排序只处理增量令牌和候选项。因此，即使并发增加，RelayGR 也能通过将高成本组件限制在中继竞速路径上，使排序尾延迟保持低于基线。使用半精度（即 FP16 格式）或计算能力更高的神经网络处理器类型后，排序延迟还会进一步降低。

符合服务等级目标的吞吐量。图11d显示，在P99约束下，RelayGR提升了吞吐量（每秒查询数）。借助动态随机存取存储器复用，更多请求可以跳过预推理，直接复用缓存的ψ，从而进一步提升性能。在我们的设置中，使用所有可用的动态随机存取存储器，吞吐量最高可比基线提升**3.6**×。这些结果表明，中继竞速推理是让长序列生成式推荐在生产环境中实现成本效益，同时维持严格P99服务级别目标的一条切实可行的路径。



（a）不同序列下的吞吐量。 （b）不同序列下的延迟。

（c）不同序列下的加载情况。 （d）最大并发数。

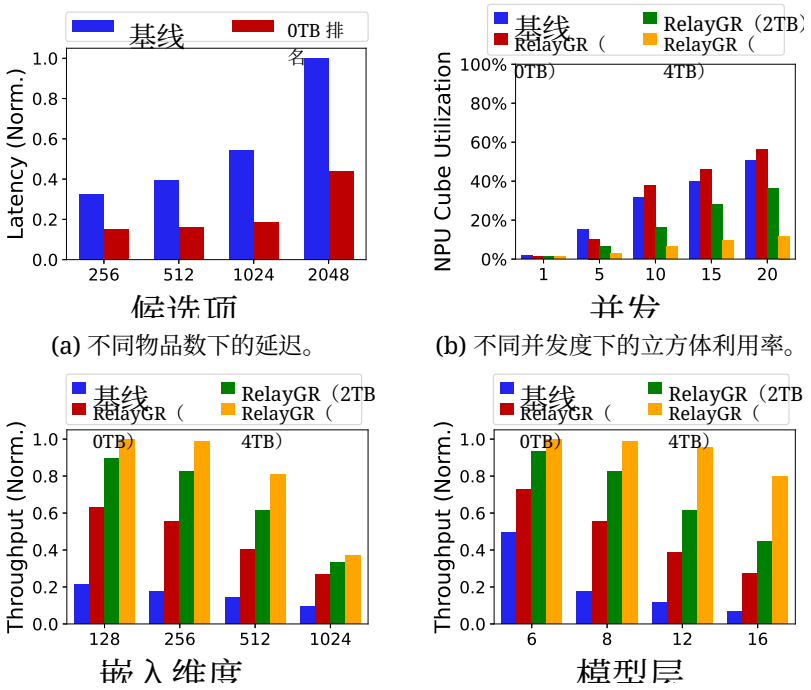
图13. RelayGR 对扩展序列的评估。

亲和性不可或缺。如果将前缀缓存放入分布式池而不设置亲和性（即不使用RelayGR），排序可能需要跨服务器远程获取缓存。如图12所示，远程获取延迟（蓝线）可能比本地缓存访问（RelayGR，红线）高出数百倍，而且很容易超过请求生命周期窗口。。这些结果凸显了RelayGR的关键不变量：排序关键路径上不发生远程获取。

4.3 Q2：RelayGR 支持扩展序列

随着序列变长，吞吐量平缓下降。图13a显示，随着序列长度增加，所有方法的吞吐量都会下降，但 RelayGR 的下降速度远慢于基线。在相同的服务级目标下，当词元数超过 ~6K 时，基线的吞吐量会骤降至每秒仅数个查询（远不能满足吞吐量需求）；而即使 DRAM 命中率为 0%（纯 HBM 内接力赛），RelayGR 在词元数超过 6K 时仍能保持每秒数十个查询（生产环境中每个实例的吞吐量约为每秒数十或数百个查询）。当命中率更高时（例如，DRAM 命中率为 100%），即使序列长度超过 8K，RelayGR 仍能维持每秒数百个查询。总的来说，RelayGR 扩大了可行运行区域，使长序列生成式推荐能够在严格的尾延迟要求下运行。

更长序列下的延迟构成。图13b报告了随着序列长度增加而变化的组件延迟。预推理延迟随序列长度增加而上升，但由于它只处理长期行为（不包括候选项-项目词元），因此小于基线完整推理。即使在序列较长时，加载和排序的延迟仍保持在数十毫秒以内，适用于排序阶段 P99 预算。在我们的设置中，RelayGR 支持长度达到



(a) 不同物品数下的延迟。 (b) 不同并发度下的立方体利用率。

(c) 不同维度下的吞吐量。(d) 不同模型层数下的吞吐量。

图 14. RelayGR 扩展方案的评估。

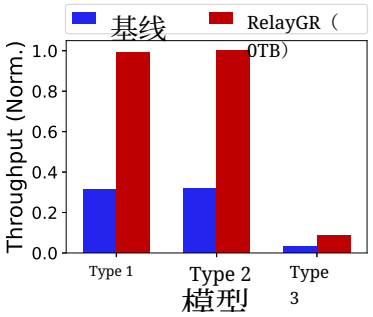
~15K 个词元的序列，加载（无并发）延迟低于 20 毫秒，排序延迟低于 10 毫秒（符合排序阶段 P99 服务等级目标）。

序列增长和并发度变化时的缓存加载。 图 13c 显示，动态随机存取存储器→高带宽内存的加载延迟会随着序列长度和并发度的增加而上升，但仍远低于基线完整推理。这与成本模型一致：注意力计算随序列长度呈超线性增长，而缓存加载则近似随缓存大小线性增长（每位用户约几十 MB）。因此，对于长序列，使用存储的 ψ 来避免重新计算越来越有优势，尤其是在中等并发度和严格的服务等级目标约束下。

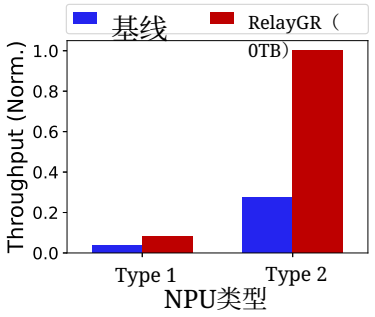
检索松弛量的影响。 图 13d 研究了检索阶段尾延迟预算对最大支持并发度的影响。基线不受影响，因为所有计算都发生在细粒度排序阶段。相比之下，RelayGR 可以利用额外的检索松弛量执行更多接力赛预推理，从而增加可同时支持的查询数量。在检索 P99 为 100毫秒时，我们的实验中最大支持并发度约为基线的 5× 倍（检索期间会触发更多预推理）。这表明，在突发场景中，RelayGR 可以用检索松弛量换取排序尾延迟余量。

4.4 Q3：RelayGR 的扩展

对候选集大小的敏感性。图 14a 显示，RelayGR 的排序延迟主要由增量输入（短期词元和候选项）决定，显著低于长前缀计算仍处于关键路径上的基线。在生产环境中，



（a）不同模型



（b）不同神经网络处理器

图 15. 不同模型和神经网络处理器的通用性。

表 1. 默认设置下的 KV 缓存。

模型	Seq.	层	格式	Dim.	Size
Type 1/2/3	2K	8	fp32	256	32MB

每个查询的候选集大小通常约为 512；RelayGR 能将基于缓存的排序延迟控制在较低水平（例如，在我们的设置中，即使每次排序推理包含 2048 个物品，延迟也低于 10 毫秒）。这表明，接力赛设计将长前缀计算成本与每个查询的物品评分工作负载解耦。

并发下的利用率。图 14b 展示了不同并发级别下的神经网络处理器（立方体）利用率。GR 推理属于计算密集型任务；并发度提高时，利用率也会提升，但资源争用也会增加尾延迟。DRAM 命中率为 0% 的 RelayGR 可以提升利用率，因为它引入了额外的预推理工作；较高的 DRAM 命中率会减少这部分额外工作，从而降低利用率。重要的是，RelayGR 可以在保持排序阶段 P99 不变的同时，调整这种利用率与延迟之间的权衡。

扩展至更大的嵌入维度。图 14c 显示，随着嵌入维度增大，基线和 RelayGR 的吞吐量都会下降，这反映了浮点运算次数的增加。然而，RelayGR 始终能提供更高的符合服务等级目标的吞吐量：在 1024 维时（其参数量是原始 HSTU 的数倍），在我们的设置中，基线的每秒查询数降至 50 以下，而 RelayGR 可达到 $\geq 2\times$ 吞吐量；DRAM 命中率为 100% 的 RelayGR 则可达到 $\sim 3\times$。这表明，RelayGR 不仅适用于序列扩展，也能推广至“宽度”扩展。

扩展到更深的模型。图 14d 显示，吞吐量会随着模型深度的增加而下降。层数翻倍时，基线吞吐量下降约 50%；而得益于缓存前缀复用，RelayGR 的性能下降更平缓；在 100% 命中率下，层数翻倍仅使吞吐量下降约 14%。当骨干网络有 16 层时，RelayGR 的吞吐量比基线高出 $\geq 4\times$，这表明中继竞速推理对更深的 GR 模块仍然有益。

模型与神经网络处理器的普适性。最后，我们在多个生成式推荐模型（图 15a）和不同类型的昇腾神经网络处理器（例如昇腾 310（Type 1）和 910C（Type 2），图 15b）上评估了 RelayGR。尽管不同模型的绝对延迟和吞吐量差异显著——在默认设置下有时相差近一个数量级——但相对趋势一致：RelayGR 扩大了可服务的最大序列长度，并在严格的尾延迟约束下提高了符合服务等级目标的吞吐量。模型规模也存在显著差异。组合后的 Longer [2] + RankMixer [51]模型（Type 3）明显大于 HSTU（Type 1）及其变体（Type 2）。

在默认设置下，表 1 报告的每个请求的 KV 缓存占用为 32,MB。请注意，Type 2 与 Type 1 的区别仅在于注意力计算；对于 Type 3，我们只缓存 Longer 组件；RankMixer 则作为下游深度学习推荐模型，在细粒度排序阶段实现。最后，即使输入长度为 2K 个令牌，Type 1 基线的 P99 延迟也可能超过预算，使得不使用缓存的长序列生成式推荐无法实现——这进一步凸显了 RelayGR 的实际价值。

5 相关工作

我们讨论与 RelayGR 最相关的三类工作：（i）在不同阶段管理 KV 缓存的大语言模型服务架构；（ii）推动长序列推理发展的生成式推荐（GR）模型；以及（iii）生产级推荐系统优化。

大语言模型服务：预填充–解码放置与 *KV* 缓存管理。大量大语言模型服务研究探讨如何放置预填充和解码阶段，以及如何管理 KV 缓存，从而在服务级目标下最大化有效吞吐量。代表性系统将预填充与解码解耦并加以协调（例如 Splitwise [26]、DistServe [47]、TetriInfer [13]、P/D-Serve [18]），在 GPU/NPU 集群中扩展部署规模（例如 Mooncake [27]及相关生产规模部署），动态平衡预填充与解码之间的资源（例如 DynaServe [30]、Adrenaline [20]），并使用分布式 KV 缓存池或缓存中心化设计（例如 MemServe [12]、CachedAttention [8]、TaiChi [32]）。

这些系统大多采用两阶段推理图：预填充生成 KVCache，供解码立即使用。因此，生产者在交接时知道（或可以决定）消费者，缓存迁移是直接的阶段边界操作。RelayGR 面向不同的场景：在多阶段推荐系统流水线中，GR 缓存是一个生命周期 问题。前缀缓存较早生成（在召回期间），在经过预处理后较晚使用（在排序期间）；而最终的排序实例只有在中间过滤之后才确定。这种场景带来了 (i) 不容忽视的存活窗口，(ii) 延迟绑定路由下的放置问题，以及 (iii) 高QPS下严格的 P99 约束。RelayGR 通过联合实施准入控制（限制实时缓存占用）和亲和路由约定（排序关键路径上不进行远程获取）来应对这些挑战。因此，我们研究的是一个独特的系统问题：延迟绑定放置下的生命周期缓存。

生成式推荐与长序列建模。传统推荐模型（例如深度学习推荐模型及其变体 [5, 15, 34, 35, 49–51]）推动了大量关于高效嵌入和排序的系统研究。近来，生成式推荐模型通过建模长序列行为提升推荐质量，并且随着序列变长、模型容量增大，展现出良好的扩展性。代表性的生成式推荐模型包括 HSTU [45],LONGER [2],MTGR [10],HLLM [3],TIGER [28], LUM [41], Climber [40], GenRank [16], OneRec [6], OneRec-v2 [48], OneRec- Think [21], andOneTrans [46]。这些工作涵盖了用于评分/排序的判别式用法（例如 LONGER/MTGR/HLLM），以及生成式用法（例如生成嵌入或逐 token 目标，通常还会配合额外的任务塔）。

我们的工作与之互补：我们并未提出新的 GR 模型，而是着力解决阻碍生成式推荐实现其离线扩展潜力的在线系统瓶颈。生产环境排序必须满足数十毫秒级的尾延迟要求；增加序列长度或特征维度会推高推理成本，并很快导致 P99 超标，迫使系统限制在线序列的长度。RelayGR 通过预推理用户长期行为前缀并缓存逐层 KV，实现长前缀复用，从而将耗费巨大的长前缀计算移出排序关键路径。与模型侧压缩（例如令牌压缩）相比，RelayGR 在保留长序列语义的同时，通过准入控制和放置来满足尾延迟约束。

生产级推荐系统优化。工业推荐系统技术栈通过内核融合、混合精度、嵌入缓存和高效的特征/嵌入服务来优化端到端流水线。NVIDIA Merlin [24] 和面向推荐系统的生成式推荐技术栈 [23] 等工具体现了高效训练与推理的最佳实践。其他生产环境实践则着重于整体流水线优化（例如，对整个推荐器进行协同设计） [25, 36, 42]。

这些系统主要优化阶段内 执行（例如，更快的嵌入查找、融合算子或效率更高的模型内核）。此类优化能够改善常数项，但无法从根本上解决长序列生成式推荐推理的跨阶段特性：昂贵的前缀计算与排序阶段之间隔着多个中间流水线阶段。RelayGR 与这些优化互为正交：它引入了一条跨阶段执行路径，将长前缀计算提前，并在生命周期窗口和严格约束下使其能够在后续阶段复用。具体而言，RelayGR 明确处理了以下问题：(i) 前缀 KV 缓存在流水线各阶段之间的存续能力；(ii) 将路由延迟绑定到最终的排序实例；以及 (iii) 高 QPS 争用。这些并非阶段内内核和嵌入优化的主要关注点。

6 结论

长序列生成式推荐目前受排序阶段 P99 的制约：随着序列长度增长，完整的 GR 推理很快便无法在在线关键路径上执行，导致生产系统无法实现离线观察到的扩展收益。本文表明，关键机遇在于预推理 用户的长期行为前缀，并在多阶段推荐流水线中复用其逐层 KV 缓存。

我们介绍了 RelayGR，这是一套生产系统，可实现 *HBM*内接力赛推理——一种在延迟绑定放置下的生命周期缓存形式——让前缀缓存得以提前生成、在中间阶段持续保留，并由排序阶段使用，无需远程获取。RelayGR 结合了：(i) 序列感知触发器，仅在有界缓存占用和加载条件下接纳高风险请求；(ii) 亲和性感知路由器，即使路由决策采用延迟绑定，也能将缓存生产与消费部署在一起；(iii) 内存感知扩展器，利用服务器本地动态随机存取存储器，在避免冗余重新加载的同时，让重复请求得以复用。

我们在昇腾神经网络处理器上实现了 RelayGR，并在生产环境镜像环境中使用真实查询对其进行了评估。在工业级 P99 约束下，RelayGR 将支持的最大输入序列长度提升了最高 1.5× ，并将符合服务等级目标的吞吐量提升了最高 3.6×，这表明接力赛推理是将长序列生成式推荐引入实时推荐系统的一条有效系统路径。

参考文献

[1] Robin D. Burke、Alexander Felfernig 和 Mehmet H. Göker。2011年 32, 3 2011年 13 18。《推荐系统：概述》。*AI Mag.* ( )，–。[2] Zheng Chai、Qin Ren、Xijun Xiao、Huizhi Yang、Bo Han、Sijun Zhang、Di Chen、Hui Lu、Wenlin Zhao、Lele Yu、Xionghang Xie、Shiru Ren、Xiang Sun、Yaocheng Tan、Peng Xu、Yuchao Zheng 和 Di Wu。2025年。LONGER：在工业推荐系统中扩展长序列建模。*arXiv preprint arXiv:2505.04421* (2025年)。[3] Junyi Chen、Lu Chi、Bingyue Peng 和 Zehuan Yuan。2024年。HLLM：通过用于物品与用户建模的分层大语言模型增强序列推荐。*arXiv preprint arXiv:2409.12740* (2024年)。[4] Minmin Chen、Bo Chang、Can Xu 和 Ed H. Chi。2021年。用于改进 REINFORCE 推荐系统的用户响应模型。收录于 *ACM WSDM*。121–129。[5] Qiwei Chen、Changhua Pei、Shanshan Lv、Chao Li、Junfeng Ge 和 Wenwu Ou。2021年。点击率预测模型中的端到端用户行为检索。*arXiv preprint arXiv:2108.04468* (2021年)。[6] Jiaxin Deng、Shiyao Wang、Kuo Cai、Lejian Ren、Qigen Hu、Weifeng Ding、Qiang Luo 和 Guorui Zhou。2025年。OneRec：通过生成式推荐系统和迭代式偏好对齐统一检索与排序。*arXiv preprint arXiv:2502.18965* (2025年)。

[7] Chantat Eksombatchai、Pranav Jindal、Jerry Zitao Liu、Yuchen Liu、Rahul Sharma、Charles Sugnet、Mark Ulrich 和 Jure Leskovec。2018年。Pixie：一个实时向 200+ 百万用户推荐 3+ 十亿物品的系统。收录于 *ACM WWW*。1775–1784。[8] Bin Gao、Zhuomin He、Puru Sharma、Qingxuan Kang、Djordje Jevdjic、Junbo Deng、Xingkun Yang、Zhou Yu 和 Pengfei Zuo。2024年。利用 CachedAttention 为多轮对话提供高性价比的大语言模型服务。收录于 *USENIX ATC*。111–126。[9] In Gim、Guojun Chen、Seung seob Lee、Nikhil Sarda、Anurag Khandelwal 和 Lin Zhong。2024年。Prompt Cache：面向低延迟推理的模块化注意力复用。*arXiv preprint arXiv:2311.04934* (2024年)。[10] Ruidong Han、Bin Yin、Shangyu Chen、He Jiang、Fei Jiang、Xiang Li、Chi Ma、Mincong Huang、Xiaoguang Li、Chunzhen Jing、Yueming Han、MengLei Zhou、Lei Yu、Chuan Liu 和 Wei Lin。2025年。MTGR：美团工业级生成式推荐框架。收录于 *ACM CIKM*。[11] Zhicheng He、Weiwen Liu、Wei Guo、Jiarui Qin、Yingxue Zhang、Yaochen Hu 和 Ruiming Tang。2023年。推荐系统中的用户行为建模综述。收录于 *IJCAI*。6656–6664。[12] Cunchen Hu、Heyang Huang、Junhao Hu、Jiang Xu、Xusheng Chen、Tao Xie、Chenxi Wang、Sa Wang、Yungang Bao、Ninghui Sun 和 Yizhou Shan。2024年。MemServe：利用弹性内存池为解耦式大语言模型服务提供上下文缓存。*arXivpreprintarXiv:2406.17565* (2024年)。[13] Cunchen Hu、Heyang Huang、Liangliang Xu、Xusheng Chen、Jiang Xu、Shuang Chen、Hao Feng、Chenxi Wang、Sa Wang、Yungang Bao、Ninghui Sun 和 Yizhou Shan。2024年。无干扰推理：面向混合下游工作负载的解耦式大语言模型推理。*arXiv preprint arXiv:2401.11181* (2024年)。[14] Junjie Huang、Jizheng Chen、Jianghao Lin、Jiarui Qin、Ziming Feng、Weinan Zhang 和 Yong Yu。2024年。推荐系统检索方法全面综述。*arXiv preprint arXiv:2407.21022* (2024年)。[15] Tongwen Huang、Zhiqi Zhang 和 Junlin Zhang。2019年。FiBiNET：结合特征重要性与双线性特征交互的点击率预测。收录于 *ACM RecSys*。169–177。[16] Yanhua Huang、Yuqi Chen、Xiong Cao、Rui Yang、Mingliang Qi、Yinghao Zhu、Qingchang Han、Yaowei Liu、Zhaoyu Liu、Xuefeng Yao、Yuting Jia、Leilei Ma、Yinqi Zhang、Taoyu Zhu、Liujie Zhang、Lei Chen、Weihang Chen、Min Zhu、Ruiwen Xu 和 Lei Zhang。2025年。迈向大规模生成式排序。*arXiv preprint arXiv:2505.04180* (2025年)。[17] Yibo Huang、Yiming Qiu、Zhenning Yang、Yi Dai、Dingming Wu、Fan Lai、Jiarong Xing 和 Ang Chen。2025年。迈向完全解耦式推荐模型服务。收录于 *ACM SIGOPS APSys*。38–45。[18] Yibo Jin、Tao Wang、Huimin Lin、Mingyang Song、Peiyang Li、Yipeng Ma、Yicheng Shan、Zhengfan Yuan、Cailong Li、Yajing Sun、Tiandeng Wu、Xing Chu、Ruizhi Huan、Li Ma、Xiao You、Wenting Zhou、Yunpeng Ye、Wen Liu、Xiangkun Xu、Yongsheng Zhang、Tiantian Dong、Jiawei Zhu、Zhe Wang、Xijian Ju、Jianxun Song、Haoliang Cheng、Xiaojing Li、Jiandong Ding、Hefei Guo 和 Zhengyong Zhang。2024年。P/D-Serve：大规模解耦式大语言模型服务。*arXiv preprint arXiv:2408.08147* (2024年)。[19] Fan Lai、Wei Zhang、Rui Liu、William Tsai、Xiaohan Wei、Yuxi Hu、Sabin Devkota、Jianyu Huang、Jongsoo Park、Xing Liu、Zeliang Chen、Ellie Wen、Paul Rivera、Jie You、Chun-cheng Jason Chen 和 Mosharaf Chowdhury。2023年。AdaEmbed：面向大规模推荐模型的自适应嵌入。收录于 *USENIX OSDI*。817–831。[20] Yunkai Liang、Zhangyu Chen、Pengfei Zuo、Zhi Zhou、Xu Chen 和 Zhou Yu。2025年。为大语言模型服务注入肾上腺素：通过注意力解耦提升资源利用率和吞吐量。*arXiv preprint arXiv:2503.20552* (2025年)。[21] Zhanyu Liu、Shiyao Wang、Xingmei Wang、Rongzhou Zhang、Jiaxin Deng、Honghui Bao、Jinghao Zhang、Wuchao Li、Pengfei Zheng、Xiangyu Wu、Yifei Hu、Qigen Hu、Xinchen Luo、Lejian Ren、Zixing Zhang、Qianqian Wang、Kuo Cai、Yunfan Wu、Hongtao Cheng、Zexuan Cheng、Lu Ren、Huanjie Wang、Yi Su、Ruiming Tang、Kun Gai 和 Guorui Zhou。2025年。OneRec-Think：生成式推荐的文本内推理。*arXiv preprint arXiv:2510.11639* (2025年)。[22] Maxim Naumov、Dheevatsa Mudigere、Hao-Jun Michael Shi、Jianyu Huang、Narayanan Sundaraman、Jongsoo Park、Xiaodong Wang、Udit Gupta、Carole-Jean Wu、Alisson G. Azzolini、Dmytro Dzhulgakov、Andrey Mallevich、Ilia Cherniavskii、Yinghai Lu、Raghuraman Krishnamoorthi、Ansha Yu、Volodymyr Kondratenko、Stephanie Pereira、Xianjie Chen、Wenlin Chen、Vijay Rao、Bill Jia、Liang Xiong 和 Misha Smelyanskiy。2019年。用于个性化与推荐系统的深度学习推荐模型。*arXiv preprint arXiv:1906.00091* (2019年)。[23] 英伟达。2025年。NVIDIA RecSys 示例。https://github.com/NVIDIA/recsys-examples。[24] NVIDIA 开发者。2025年。NVIDIA-Merlin。https://github.com/NVIDIA-Merlin/Merlin。[25] NVIDIA 技术博客。2023年。从离线到在线：使用 NVIDIA Merlin 为实时推荐系统提供特征存储。https://developer.nvidia.com/blog/offline-to-online-feature-storage-for-real-time-recommendation-systems-with-nvidia-merlin/。[26] Pratyush Patel、Esha Choukse、Chaojie Zhang、Aashaka Shah、Íñigo Goiri、Saeed Maleki 和 Ricardo Bianchini。2024年。Splitwise：通过阶段拆分实现高效的生成式大语言模型推理。*arXiv preprint arXiv:2311.18677* (2024年)。[27] Ruoyu Qin、Zheming Li、Weiran He、Jialei Cui、Feng Ren、Mingxing Zhang、Yongwei Wu、Weimin Zheng 和 Xinran Xu。2025年。Mooncake：以更多存储换取更少计算——一种以 KVCache 为中心的大语言模型服务架构

聊天机器人。在第 23 届 USENIX 文件与存储技术会议（FAST 25）中。[28] Shashank Rajput、Nikhil Mehta、Anima Singh、Raghunandan Hulikal Keshavan、Trung Vu、Lukasz Heldt、Lichan Hong、Yi Tay、Vinh Q. Tran、Jonah Samost、Maciej Kula、Ed H. Chi 和 Mahesh Sathiamoorthy。2023 年。基于生成式检索的推荐系统。在 NeurIPS。[29] Shashank Rajput、Nikhil Mehta、Anima Singh、Raghunandan H. Keshavan、Trung Vu、Lukasz Heldt、Lichan Hong、Yi Tay、Vinh Q. Tran、Jonah Samost、Maciej Kula、Ed H. Chi 和 Maheswaran Sathiamoorthy。2023 年。基于生成式检索的推荐系统。arXiv 预印本 arXiv:2305.05065（2023 年）。[30] Chaoyi Ruan、Yinhe Chen、Dongqi Tian、Yandong Shi、Yongji Wu、Jialin Li 和 Cheng Li。2025 年。DynaServe：面向动态解耦式大语言模型服务的统一弹性执行。arXiv 预印本 arXiv:2504.09285（2025 年）。[31] Blaz Skrlj、Yonatan Karni、Grega Gaspersic、Blaz Mramor、Yulia Stolin、Martin Jakomin、Jasna Urbancic、Yuval Dishi、Natalia Silberstein、Ophir Friedler 和 Assaf Klein。2025 年。DCNˆ2：大规模推荐中隐式碰撞权重与显式交叉层的交互。arXiv 预印本 arXiv:2506.21624（2025 年）。[32] Chao Wang、Pengfei Zuo、Zhangyu Chen、Yunkai Liang、Zhou Yu 和 Ming-Chang Yang。2025 年。预填充-解码聚合还是解耦？统一两者以实现吞吐效率优化的大语言模型服务。arXiv 预印本 arXiv:2508.01989（2025 年）。[33] Jianling Wang、Haokai Lu、Yifan Liu、He Ma、Yueqi Wang、Yang Gu、Shuzhou Zhang、Ningren Han、Shuchao Bi、Lexi Baugher、Ed H. Chi 和 Minmin Chen。2024 年。用于大规模推荐系统中用户兴趣探索的大语言模型。在 ACM RecSys。872–877。[34] Ruoxi Wang、Bin Fu、Gang Fu 和 Mingliang Wang。2017 年。用于广告点击预测的深度交叉网络。在 ACM ADKDD。12:1–12:7。[35] Ruoxi Wang、Rakesh Shivanna、Derek Cheng、Sagar Jain、Dong Lin、Lichan Hong 和 Ed Chi。2021 年。DCN V2：改进的深度交叉网络及互联网规模学习排序系统的实践经验。在 ACM WWW。1785–1797。[36] Yingcan Wei、Matthias Langer、Fan Yu、Minseok Lee、Jie Liu、Ji Shi 和 Zehuan Wang。2022 年。面向大规模深度推荐模型的 GPU 专用推理参数服务器。在 ACM RecSys。408–419。[37] Yunjia Xi、Weiwen Liu、Yang Wang、Ruiming Tang、Weinan Zhang、Yue Zhu、Rui Zhang 和 Yong Yu。2023 年。基于异构行为建模的端侧集成式重排序。在 ACM SIGKDD。5225–5236。[38] Yunjia Xi、Muyan Weng、Wen Chen、Chao Yi、Dian Chen、Gaoyang Guo、Mao Zhang、Jian Wu、Yuning Jiang、Qingwen Liu、Yong Yu 和 Weinan Zhang。2025 年。打破信息茧房：利用对齐的大语言模型增强惊喜式推荐。arXiv 预印本 arXiv:2502.13539（2025 年）。[39] Minhui Xie、Youyou Lu、Jiazhen Lin、Qing Wang、Jian Gao、Kai Ren 和 Jiwu Shu。2022 年。Fleche：用于个性化推荐的高效 GPU 嵌入缓存。在 ACM EuroSys。402–416。[40] Songpei Xu、Shijia Wang、Da Guo、Xianwen Guo、Qiang Xiao、Bin Huang、Guanlin Wu 和 Chuanjiang Luo。2025 年。Climber：迈向高效的大型推荐模型缩放定律。在 ACM CIKM。6193–6200。[41] Bencheng Yan、Shilei Liu、Zhiyuan Zeng、Zihao Wang、Yizhen Zhang、Yujin Yuan、Langming Liu、Jiaqi Liu、Di Wang、Wenbo Su、Wang Pengjie、Jian Xu 和 Bo Zheng。2025 年。利用基于三步范式的大型用户模型解锁工业推荐系统中的缩放定律。arXiv 预印本 arXiv:2502.08309（2025 年）。[42] Lingyun Yang、Yongchen Wang、Yinghao Yu、Qizhen Weng、Jianbo Dong、Kan Liu、Chi Zhang、Yanyi Zi、Hao Li、Zechao Zhang、Nan Wang、Yu Dong、Menglei Zheng、Lanlan Xi、Xiaowei Lu、Liang Ye、Guodong Yang、Binzhang Fu、Tao Lan、Liping Zhang、Lin Qu 和 Wei Wang。2025 年。面向大规模深度学习推荐模型的 GPU 解耦式服务。在 USENIX NSDI。847–863。[43] Zhen Yang、Haitao Lin、Jiawei xue 和 Ziji Zhang。2025 年。GR-LLMs：基于大语言模型的生成式推荐最新进展。arXiv 预印本 arXiv:2507.06507（2025 年）。[44] Lu Ye、Ze Tao、Yong Huang 和 Yang Li。2024 年。ChunkAttention：结合前缀感知 KV 缓存和两阶段分区的高效自注意力。arXiv 预印本 arXiv:2402.15220（2024 年）。[45] Jiaqi Zhai、Lucy Liao、Xing Liu、Yueming Wang、Rui Li、Xuan Cao、Leon Gao、Zhaojie Gong、Fangda Gu、Michael He、Yinghai Lu 和 Yu Shi。2024 年。行动胜于言语：用于生成式推荐的万亿参数序列转换器。arXiv 预印本 arXiv:2402.17152（2024 年）。[46] Zhaoqi Zhang、Haolei Pei、Jun Guo、Tianyu Wang、Yufei Feng、Hui Sun、Shaowei Liu 和 Aixin Sun。2025 年。OneTrans：在工业推荐器中使用单个 Transformer 统一特征交互与序列建模。arXiv 预印本 arXiv:2510.26104（2025 年）。[47] Yinmin Zhong、Shengyu Liu、Junda Chen、Jianbo Hu、Yibo Zhu、Xuanzhe Liu、Xin Jin 和 Hao Zhang。2024 年。Dist-Serve：为实现吞吐效率优化的大语言模型服务而解耦预填充与解码。arXiv 预印本 arXiv:2401.09670（2024 年）。[48] Guorui Zhou、Hengrui Hu、Hongtao Cheng、Huanjie Wang、Jiaxin Deng、Jinghao Zhang、Kuo Cai、Lejian Ren、Lu Ren、Liao Yu、Pengfei Zheng、Qiang Luo、Qianqian Wang、Qigen Hu、Rui Huang、Ruiming Tang、Shiyao Wang、Shujie Yang、Tao Wu、Wuchao Li、Xinchen Luo、Xingmei Wang、Yi Su、Yunfan Wu、Zexuan Cheng、Zhanyu Liu、Zixing Zhang、Bin Zhang、Boxuan Wang、Chaoyi Ma、Chengru Song、Chenhui Wang、Chenglong Chu、Di Wang、Dongxue Meng、

Dunju Zang、Fan Yang、Fangyu Zhang、Feng Jiang、Fuxing Zhang、Gang Wang、Guowang Zhang、Han Li、Honghui Bao、Hongyang Cao、Jiaming Huang、Jiapeng Chen、Jiaqiang Liu、Jinghui Jia、Kun Gai、Lantao Hu、Liang Zeng、Qiang Wang、Qidong Zhou、Rongzhou Zhang、Shengzhe Wang、Shihui He、Shuang Yang、Siyang Mao、Sui Huang、Tiantian He、Tingting Gao、Wei Yuan、Xiao Liang、Xiaoxiao Xu、Xugang Liu、Yan Wang、Yang Zhou、Yi Wang、Yiwu Liu、Yue Song、Yufei Zhang、Yunfeng Zhao、Zhixin Ling 和 Ziming Li。2025 年。OneRec-V2 技术报告。*arXiv* 预印本 *arXiv:2508.20900*（2025 年）。[49] Guorui Zhou、Na Mou、Ying Fan、Qi Pi、Weijie Bian、Chang Zhou、Xiaoqiang Zhu 和 Kun Gai。2019 年。用于点击率预测的深度兴趣演化网络。在 *AAAI*。5941–5948。[50] Guorui Zhou、Xiaoqiang Zhu、Chengru Song、Ying Fan、Han Zhu、Xiao Ma、Yanghui Yan、Junqi Jin、Han Li 和 Kun Gai。2018 年。用于点击率预测的深度兴趣网络。在 *ACM SIGKDD*。1059–1068。[51] Jie Zhu、Zhifang Fan、Xiaoxie Zhu、Yuchen Jiang、Hangyu Wang、Xintian Han、Haoran Ding、Xinmin Wang、Wenlin Zhao、Zhen Gong、Huizhi Yang、Zheng Chai、Zhe Chen、Yuchao Zheng、Qiwei Chen、Feng Zhang、Xun Zhou、Peng Xu、Xiao Yang、Di Wu 和 Zuotao Liu。2025 年。RankMixer：扩展工业推荐系统中的排序模型。*arXiv* 预印本 *arXiv:2507.15551*（2025 年）。[52] Pengfei Zuo、Huimin Lin、Junbo Deng、Nan Zou、Xingkun Yang、Yingyu Diao、Weifeng Gao、Ke Xu、Zhangyu Chen、Shirui Lu、Zhao Qiu、Peiyang Li、Xianyu Chang、Zhengzhong Yu、Fangzheng Miao、Jia Zheng、Ying Li、Yuan Feng、Bei Wang、Zaijian Zong、Mosong Zhou、Wenli Zhou、Houjiang Chen、Xingyu Liao、Yipeng Li、Wenxiao Zhang、Ping Zhu、Yinggang Wang、Chuanjie Xiao、Depeng Liang、Dong Cao、Juncheng Liu、Yongqiang Yang、Xiaolong Bai、Yi Li、Huaguo Xie、Huatao Wu、Zhibin Yu、Lv Chen、Hu Liu、Yujun Ding、Haipei Zhu、Jing Xia、Yi Xiong、Zhou Yu 和 Heng Liao。2025 年。在华为 CloudMatrix384 上提供大语言模型服务。*arXiv* 预印本 *arXiv:2506.12708*（2025 年）。