计算机科学
COMPUTER SCIENCE

大语言模型服务系统服务级目标和系统级指标优化

王智彬, 李世鹏, 周宇航, 李雪, 张中辉, 蒋智威, 顾荣, 田臣, 陈贵海, 仲盛

引用本文

王智彬, 李世鹏, 周宇航, 李雪, 张中辉, 蒋智威, 顾荣, 田臣, 陈贵海, 仲盛. 大语言模型服务系统服务级目标和系统级指标优化[J]. 计算机科学, 2026, 53(3): 23-32.

WANG Zhibin, LI Shipeng, ZHOU Yuhang, LI Xue, ZHANG Zhonghui, JIANG Zhiwei, GU Rong, TIAN Chen, CHEN Guihai, ZHONG Sheng. Optimization of Service Level Objectives and System Level Metrics in Large Language ModelServing System [J]. Computer Science, 2026, 53(3): 23-32.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250900173

大语言模型服务系统服务级目标和系统级指标优化

王智彬[1] 李世鹏[1,2] 周宇航[1] 李 雪[2] 张中辉[1] 蒋智威[1] 顾 荣[1] 田 臣[1] 陈贵海[1] 仲 盛[1]
1 计算机软件新技术全国重点实验室(南京大学) 南京 210023
2 阿里巴巴集团 杭州 310000
(wzbwangzhibin@gmail.com)

摘 要 在大语言模型服务系统中,用户体验是一个关键考量因素。服务级目标和系统级指标是两种关键的性能衡量标准,前者关注单个请求的体验,后者关注系统的整体性能。然而,现有的度量标准存在两个与直觉相悖的问题:1)通过刻意延迟部分词元的交付可以提升服务级目标指标;2)主动丢弃不满足服务级目标的请求可以改善系统级指标。为解决上述问题,重新分析了大语言模型服务中的服务级目标和系统级指标,并提出了一种与用户体验更一致的新型服务级目标。基于此服务级目标,提出了一种名为"平滑有效吞吐量"的综合度量框架,其通过整合服务级目标和系统级指标来反映大语言模型服务中用户体验的本质。利用该统一框架,对不同大语言模型服务系统在多种工作负载下的性能进行了重新评估。评估结果表明,所提出的度量框架能够对词元交付和请求处理提供更全面的评估维度,并有效地捕捉在不同服务策略下用户体验与系统性能的最优点。

关键词:大语言模型;推理服务系统;服务级目标;调度
中图分类号 TP319

Optimization of Service Level Objectives and System Level Metrics in Large Language Model Serving System

WANG Zhibin[1],LI Shipeng[1,2],ZHOU Yuhang[1],LI Xue[2],ZHANG Zhonghui[1],JIANG Zhiwei[1],GU Rong[1],TIAN Chen[1],CHEN Guihai[1] and ZHONG Sheng[1]
1 State Key Laboratory of Novel Software Technology,Nanjing University,Nanjing 210023,China
2 Alibaba Group,Hangzhou 310000,China

Abstract In Large Language Model(LLM) serving systems,user experience is a critical consideration. Service-Level Objectives (SLOs) and System-Level Metrics(SLMs) are two key performance measures:the former focuses on the experience of individual requests,while the latter reflects the overall performance of the system. However,existing metrics exhibit two counterintuitive issues:1) manually delaying the delivery of some tokens can improve SLOs;2) actively abandoning requests that do not meet SLOs can improve SLMs. To address these issues,the definitions of SLOs and SLMs in LLM serving are revisited and a new type of SLO is proposed that aligns more closely with actual user experience. Based on this SLO,a comprehensive metric framework called smooth goodput is developed,which integrates SLOs and SLMs to reflect the nature of user experience in LLM serving. Through this unified framework,the performance of different LLM serving systems under multiple workloads is reassessed. Evaluation results show that the proposed metric framework provides a more comprehensive view of token delivery and request processing,and effectively captures the optimal point of user experience and system performance with different serving strategies.

Keywords Large language model,Inference serving system,Service level objectives,Scheduling

1 引言

当前大型语言模型(Large Language Model,LLM)[1-4]在聊天机器人[5-7]、虚拟助手[8-9]等各类任务中展现出卓越性能,服务提供商部署 LLM,以向用户提供服务,已成为普遍趋势。随着 LLM 服务需求的不断增长,LLM 服务系统的性能受到广泛关注。起初,LLM 服务系统的核心目标是最大化吞吐量[10-14],这一目标虽能显著提升资源利用率并降低成本,但

到稿日期:2025-09-29 返修日期:2025-11-29
基金项目:南京"U35 强基项目"(U(2024)001);国家自然科学基金(61872176,62272215,62325205,62172204);江苏省自然科学基金领先技术计划(BK20202001);国家重点研发计划(2020YFB1005900);江苏省自然科学基金重点计划(BK20243053)
This work was supported by the Nanjing "U35" Talent Cultivation Program(U(2024)001),the National Natural Science Foundation of China (61872176,62272215,62325205,62172204),Leading-edge Technology Program of Jiangsu Natural Science Foundation(BK20202001),National Key R&D Program of China(2020YFB1005900) and Key Program of the Natural Science Foundation of Jiangsu Province(BK20243053).
通信作者:顾荣(gurong@nju.edu.cn)

往往会导致请求延迟增加——因为系统需处理更高的每秒请求数(Query Per Second,QPS),进而影响到用户与系统间的实时交互体验[8-9,15-17]。

近年来,服务提供商开始将用户体验置于 LLM 服务系统的优先位置,为此引入了服务级目标(Service Level Objectives,SLO)与系统级指标(System Level Metrics,SLM)来评估 LLM 服务系统的性能[18-22]。简而言之,SLO 定义为对单个请求性能的约束(如输出词元的延迟),而 SLM 定义为整个系统的性能表现(如有效吞吐量,即满足 SLO 的请求的处理吞吐量)。

为评估 LLM 服务系统中的用户体验,研究者采用了与用户体验对齐的服务级目标[18-19,21-22],例如首词元生成时间(Time to First Token,TTFT)、词元间隔时间(Time Between Tokens,TBT)和每词元输出时间(Time Per Output Token,TPOT)。由于 LLM 具有自回归特性,因此首词元生成(即预填充阶段)[20,23]的成本高于后续词元生成(即解码阶段):预填充阶段,模型需处理整个提示词(prompt),而解码阶段仅需处理当前词元。综合考虑用户体验与预填充阶段特性,TTFT 被用于衡量生成首词元所需时间,其数值通常远大于 TBT 或 TPOT。TPOT 衡量单个请求中词元间的平均时间间隔,但该指标对用户体验的反映过于宽松——请求过程中出现的长时间停滞可能被其他词元间的短间隔平均抵消,而这种停滞实际上会损害用户体验。为此,Sarathi-Serve[19]引入 TBT 指标,以约束两个连续词元间的时间间隔。

为进一步评估 LLM 服务系统在满足 SLO 约束下的性能,研究者提出了系统级指标,用于衡量系统中每个请求的性能表现,如 SLO 达成率和有效吞吐量[19-20]。其中,SLO 达成率衡量满足 SLO 的请求占比,可视为服务系统的约束条件;有效吞吐量衡量每秒内完成且满足 SLO 的请求数量,可视为服务系统的性能表现。

同时还注意到,已有研究提出了多种系统与优化策略,以在 SLO 约束下提升系统级指标[18-22]。然而,我们发现这些指标(即 SLO 与 SLM)无法捕捉用户体验的本质。实际上,实时 LLM 服务是一种高动态交互行为,与网页浏览类似[24-25]。用户不会将其感知为一系列独立词元(如 TBT 所建模的),而是将其视为连续的信息流(用户需在接收下一个词元前处理已接收的词元)。在流式 LLM 服务中,若忽略用户体验的固有属性,会导致评估偏差,甚至使基于这些指标的优化方向偏离最优路径。将现有指标的局限性归纳为:

1)现有 SLO 中,TBT 对整体用户体验约束过于严格,而 TPOT 与端到端(End to End,E2E)延迟约束过于宽松。

TBT 衡量单个请求内每个词元间的时间间隔,TPOT 则反映词元间的平均间隔。正如文献[26]所指出的,流式服务中的用户体验受“无信息可处理的等待时间”影响。如图 1 所示,若用户有足够信息可处理,偶尔的停滞(即高 TBT)可能不会损害体验。例如,若系统在第 1 秒内快速生成 10 个词元,随后停滞 1 秒,以每秒 4 个词元的速度阅读的用户仍会获得良好体验,尽管此时 TBT 高达 1 秒。反之,若在 1 秒停滞前仅交付 2 个词元,用户会因等待而产生负面体验,尽管此时 TPOT 仅为 0.1 秒。换言之,高延迟迭代的代价会由之前的迭代共同承担。

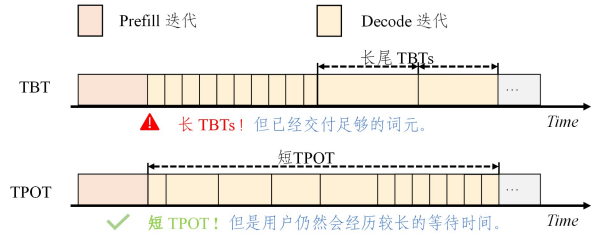


图 1 TBT 和 TPOT 局限性的示例

Fig. 1 Examples illustrating the limitations of TBT and TPOT

示例 1 为体现现有 SLO 的局限性,考虑一种“输出延迟”策略:手动将所有词元的交付延迟至 TBT 阈值时刻。例如,若 TBT 阈值为 200 毫秒,系统会在距离上一个词元交付满 200 毫秒后,才交付当前词元。在图 1 的第一个示例中,假设 TBT 阈值为 200 毫秒,系统在第 1 秒内快速生成的 10 个词元不会立即交付给用户,而是被手动延迟至间隔达到 200 毫秒时再交付。这种方式为后续较慢的词元生成提供了充足的缓冲时间,可显著提升 TBT 指标。然而,该技巧实际上延迟了所有词元的交付时间——对用户而言,每个词元的到达时间不会早于“生成后立即交付”策略下的到达时间。显然,这会导致用户体验恶化,但在现有 SLO 框架下却表现为性能提升,这一结果违背直觉。

2)现有 SLM 中,有效吞吐量与 SLO 达成率无法反映未满足 SLO 请求的价值。

有效吞吐量是一种系统级指标,反映每秒内完成且满足 SLO 的请求数量;SLO 达成率则反映满足 SLO 的请求占比。然而,现有指标定义忽略了未满足 SLO 请求的贡献。因此,看似最优的策略是丢弃或拒绝已无法满足 SLO 的请求,但这对用户而言显然并非理想选择。我们认为,未满足 SLO 要求的请求仍具有价值,需充分考虑所有请求的效益。

示例 2 为体现现有系统级指标的局限性,考虑一种“主动丢弃”策略:该策略主动丢弃已无法满足 SLO 的请求。由于这些请求对有效吞吐量的贡献始终为 0,丢弃它们不会导致有效吞吐量损失,同时还能通过减少资源竞争提升剩余运行请求的处理速率,进而提高吞吐量。然而,在 LLM 服务场景中,仅基于这些指标制定策略会给用户带来不可接受的结果。尽管延迟确实会损害用户体验,但完全丢弃请求的负面影响更为严重,而这一点在现有系统级指标中未被考虑。

结合 SLO 与 SLM,图 2 展示了流式词元交付如何影响用户体验,以及单个请求对平滑有效吞吐量的贡献如何由词元交付时间线决定。其中,蓝色线条代表词元生成时间线,每个圆点表示一个词元。在时间线上,初始阶段用户需等待首词元;随后词元快速交付,不会导致用户体验下降;时间线中期($t=10$ 至 $t=15$)词元交付缓慢,会严重损害用户体验——因为用户无信息可处理,只能等待下一个词元。一旦这种负面影响产生,后续词元的快速交付将无法弥补已造成的等待损失。因此,请求对平滑有效吞吐量的贡献由总等待时间决定(总等待时间可通过与交付时间线相切的参考线表示,参考

线斜率反映阅读速度，本示例中为 4.5 词元/秒）。参考线所在的颜色区域标识了该请求对平滑有效吞吐量的实际贡献，绿色（左上区域）表示贡献较高，红色（右下区域）表示贡献较低。同时，从交付时间线可观察到：偶尔的停滞会导致长尾 TBT，但只要用户有足够词元可阅读，就不会影响用户体验。具体而言，尽管 S_0 处的 TBT 较大，但用户在 S_0 前仍在阅读已交付的信息，因此可能不会察觉该停滞；反之，在 S_1 处，由于用户已阅读完 S_1 前交付的词元，会因等待而产生负面体验。

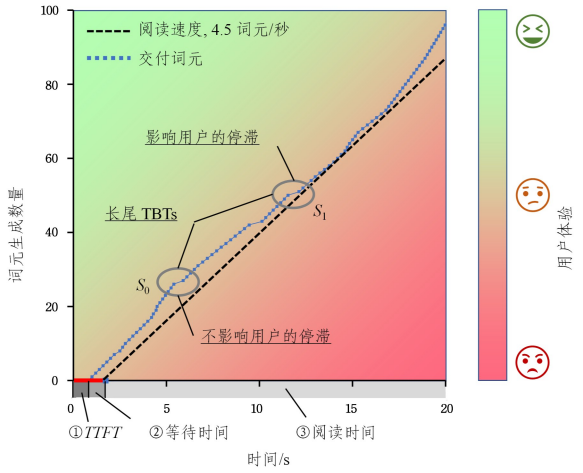


图 2 LLM 服务系统中的词元生成时间线及其对用户体验的影响（电子版为彩图）

Fig. 2 Token generation timeline in LLM serving systems and its impact on user experience

本文重新审视了 LLM 服务系统中的系统级指标与 SLO，指出它们在建模用户体验方面的局限性。为解决这一问题，本文提出一种重新设计的 SLO 指标：将词元截止时间定义为相对于请求提交时间，而非上一个词元的交付时间。基于该 SLO，进一步引入“平滑有效吞吐量”这一新型性能指标，用于实现在词元生成效益与词元交付过程中用户空闲时间惩罚之间的平衡。通过这一统一框架，在多种工作负载下重新评估了不同 LLM 服务系统的性能，旨在为以用户体验优化为核心的 LLM 服务研究统一发展方向。

本文的主要贡献包括：

1）分析探讨了 LLM 服务系统中的现有 SLO 与 SLM，通过两个反直觉示例指出它们在建模用户体验方面的局限性（见第 3、第 4 章）。

2）提出一种新型 SLO 指标，将每个词元的截止时间定义为相对于请求提交时间（而非上一个词元的交付时间），更贴合用户处理信息的实际方式（见第 3 章）。

3）基于新型 SLO 指标，引入平滑有效吞吐量指标，用于评估 LLM 服务系统性能。该指标结合词元生成对用户体验的贡献与系统吞吐量，综合考虑所有生成词元的效益（见第 4 章）。

4）基于所提框架开展大量实验，评估不同 LLM 服务系统的性能，解决现有 SLO 与 SLM 在用户体验和系统性能评估中产生的反直觉问题。实验结果表明，该框架能更全面地呈现词元交付与请求处理过程（见第 5 章）。

2 背景

2.1 LLM 推理

LLM 通过自回归推理，基于输入提示词和已生成词元生成输出词元。具体而言，长度为 k 的提示词可表示为词元序列 $(t_1, t_2, \cdots, t_k)$，LLM 生成的输出同样为词元序列，长度为 n，记为 $(t_{k+1}, t_{k+2}, \cdots, t_{k+n})$。整个推理过程包含 n 轮迭代，每轮迭代生成一个词元，即自回归过程。由于 LLM 的自回归特性，生成每个后续词元都需用到所有先前词元的键值（KV）状态，因此需缓存先前的 KV 状态以供复用，这一机制被称为 KV 缓存。

根据计算与内存访问特性，这些迭代可分为两个阶段：预填充（Prefill）阶段与解码（Decode）阶段。如图 3 所示，在预填充阶段，LLM 需在单轮迭代（A^P）内处理整个提示词，该阶段耗时通常长于解码阶段，且需为每个词元计算 KV 状态。预填充阶段包含高度可并行化的矩阵乘法运算，因此属于计算密集型阶段[19]。后续的解码阶段包含多轮迭代（A_1^D，$A_2^D, \cdots, A_n^D$），直至生成序列结束（EOS）词元 A_{EOS}^D。在解码阶段的每轮迭代中，输入为提示词与先前迭代生成的词元的拼接结果，其 KV 状态已缓存至内存中，无需重新计算，但需更多内存访问操作。解码阶段的算术强度较低，因此属于内存密集型阶段，计算过程主要受内存访问速度限制[19]。为减少这两个计算特性差异显著的阶段之间的干扰，近期研究[18,20,27]提出将预填充与解码阶段解耦，部署在不同实例上。尽管这种方式会增加额外的通信开销，但为针对不同阶段的特性定制资源配置提供了可能，可显著提升每个阶段的效率。

2.2 LLM 服务

2.2.1 LLM 服务任务

在实际应用中，LLM 通常以服务形式部署，为用户提供推理能力。根据与用户的交互模式，这些服务可大致分为两类：在线服务与离线服务。

在线 LLM 服务通常为用户提供实时服务[19,21,28-31]，是一种与网页浏览类似的高动态交互行为[24-25]。用户在与 LLM 交互时，期望系统快速响应并提供即时反馈，同时以连续信息流的形式消费信息。在这一连续信息流过程中，用户既不希望等待首响应词元的时间过长，也不希望等待后续词元的时间过长[26]。已有研究[32]对文本阅读与处理速度进行了分析，成年人的平均阅读速度约为每秒 34 个单词。结合不同语言的词元化（Tokenization）粒度，可大致估算词元生成速度的目标值。

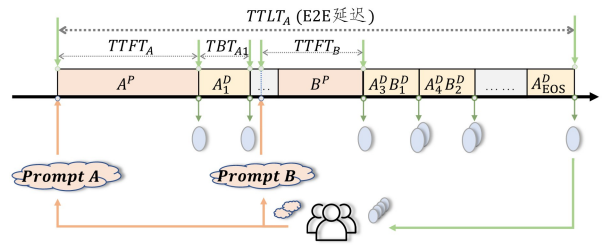


图 3 LLM 服务的现有 SLO

Fig. 3 Existing SLOs of LLM Serving

离线 LLM 服务提供非流式服务，其用户体验要求不如

在线场景严格[33-35]。用户通常更关注批量离线任务的端到端指标，对 TBT，TPOT 等流式专属指标无特殊要求。本文主要关注在线 LLM 服务，在这类场景中，用户体验至关重要。

2.2.2 LLM 服务指标

为确保始终如一的高质量用户体验与高效的资源利用率，服务提供商需通过多种指标监控和评估 LLM 服务系统的性能。基于评估目标，这些指标可分为服务级目标和系统级指标两类。

服务级目标(SLO)(见第 3 章)是直接反映单个请求词元交付状态的指标，包括首词元生成时间(TTFT)、词元间隔时间(TBT)、每词元输出时间(TPOT)等响应时间指标，用于评估请求是否能在预期时间内交付响应。通过定义 SLO，服务提供商可设定可观测的阈值，主动评估系统是否符合用户期望，这在动态变化的在线服务环境中至关重要。如图 3 所示，为确保用户获得预期服务质量，这些 SLO 主要关注 TTFT，TBT，TPOT 等方面，是评估系统响应速度、确保用户在与 LLM 交互过程中获得及时反馈的关键指标。需要注意的是，该图忽略了 LLM 生成词元与将其交付给用户之间的差异。

系统级指标(SLM)(见第 4 章)是在 SLO 约束下评估基础设施性能与运行效率的指标，包括系统吞吐量、资源利用率及用户体验相关评估等，具体如 SLO 达成率[36]、有效吞吐量[20]、容量[19]等。这类指标是反映系统性能有效性、确保系统同时满足成本与体验需求的关键。

通过结合 SLO 与 SLM，服务提供商可全面掌握系统性能。因此，重新审视这些指标对于理解 LLM 服务系统的性能及其对用户体验的影响至关重要。

3 服务级目标

本章聚焦单个请求的用户体验，即 SLO。首先介绍 SLO 的通用框架，该框架可根据不同工作负载需求进行定制，并重新审视近期 LLM 服务研究中提出的现有 SLO；随后通过“输出延迟技巧”证明现有 SLO 存在反直觉问题；最后提出一种更符合用户体验的新型 SLO，该 SLO 重点关注用户信息处理与服务信息交付之间的关系。

3.1 SLO 框架

回顾 LLM 服务中的生成过程，单个请求的输出以词元序列形式呈现，每个词元的交付时间对用户体验至关重要。为此，提出一种统一的 SLO 框架，该框架可通过定制满足不同工作负载的需求。无论采用何种 SLO，请求级别的核心目标都是衡量词元生成时间是否符合阈值要求，因此统一将 SLO 定义为每个词元的截止时间。

3.1.1 框架定义

对于输出长度为 n 的请求 r，设 i 为输出词元的索引(取值范围为 1 至 n)。定义请求 r 的第 i 个输出词元的截止时间为 d_i，第 i 个输出词元的实际生成时间为 t_i，则 SLO 约束可表示为：

$$\forall i, t_i \leqslant d_i \tag{1}$$

3.1.2 现有 SLO

通过调整每个词元的截止时间，可基于上述框架定制不同研究中提出的 SLO，具体如下：

1)首词元生成时间(TTFT)与词元间隔时间(TBT)：TTFT 反映生成首个输出词元所需时间，TBT 反映单个请求中两个相邻词元间的细粒度时间间隔，二者均聚焦于词元生成的逐轮过程。基于上述框架，TTFT 与 TBT 的 SLO 可定义为：

$$d_i = \begin{cases} TTFT_\theta, & i=1 \\ t_{i-1} + TBT_\theta, & i>1 \end{cases} \tag{2}$$

其中，$TTFT_\theta$ 与 TBT_θ 分别为 TTFT 与 TBT 的阈值[19,37]。若首词元的生成时间小于 $TTFT_\theta$，且相邻词元的生成时间间隔小于 TBT_θ，则该请求满足 SLO。需注意的是，第 i 个词元的截止时间由上一个词元的生成时间决定，后续将证明这种定义方式与用户体验并不贴合。

2)首词元生成时间(TTFT)与每词元输出时间(TPOT)：TPOT 反映生成单个词元的平均时间(不包含首词元)。基于上述框架，TTFT 与 TPOT 的 SLO 可定义为：

$$d_i = \begin{cases} TTFT_\theta, & i=1 \\ t_1 + (n-1) * TPOT_\theta, & i>1 \end{cases} \tag{3}$$

其中，$TPOT_\theta$ 为 TPOT 的阈值[18,20,38]。若首词元的生成时间小于 $TTFT_\theta$，且请求中所有词元的生成时间小于 t_1 与 $(n-1)\times TPOT_\theta$ 之和(即 $t_1+(n-1)\times TPOT_\theta$)，则该请求满足 SLO。换言之，仅首词元与末词元需满足时序约束。

3)端到端(E2E)延迟：E2E 延迟反映单个请求(或一批请求)从用户提交到处理完成的总时间。基于上述框架，E2E 延迟的 SLO 可定义为：

$$d_i = E2E_\theta \tag{4}$$

其中，$E2E_\theta$ 为 E2E 延迟的阈值。显然，若末词元的生成时间小于 E2E 延迟阈值，则该请求满足 SLO。如前所述，E2E 延迟与 TPOT 的约束过于宽松，无法始终贴合用户体验需求。

显然，TPOT 可视为 E2E 延迟的推广形式，因此下文不再单独讨论 E2E 延迟。

3.2 现有 SLO 的反直觉示例

本文从“输出延迟技巧”这一反直觉示例入手，分析现有 SLO 的问题。输出延迟技巧是指不立即交付生成的词元，而是等待至 TBT 截止时间再交付。具体而言，在 TBT SLO 下，延迟交付第 i 个词元可为第 $i+1$ 个词元的生成时间提供更宽松的约束——第 $i+1$ 个词元的生成时间约束可从 TBT_θ 放宽至 $TBT_\theta + t_{\text{delay}}$(其中 t_{delay} 为第 i 个词元的延迟交付时间)。在推理引擎与客户端之间添加中间缓冲层，即可轻松实现输出延迟。

然而，从用户体验角度来看，这种技巧并不合理：延迟词元交付会导致用户接收词元的时间晚于词元实际生成时间，进而损害用户体验。本质上，这是因为词元的过早交付会无意中为后续词元增加额外的延迟约束。因此，迫切需要设计一种新型 SLO——既能保障用户体验，又不会因词元过早交付而对其进行惩罚。

下文将介绍一种受 TBT SLO 误导的优化策略——分块预填充(Chunked Prefills)，并提出一种简单的模拟策略——解码提前(decode prepone)。该策略仅通过调度与输出延迟技巧，即可在 TBT 指标上实现与分块预填充相近的效果。

如图 4 所示，以 Sarathi-Serve 框架[19]（一种典型的分块预填充策略）为例，展示两个请求 A 与 B 的生成过程。在 vLLM 中，为提升吞吐量，采用“预填充优先”原则，因此请求 A 后续词元的解码阶段需暂停，直至请求 B 的预填充阶段完成，这会导致生成停滞，即 A_5^D 与 A_6^D 之间出现较大 TBT。为此，Sarathi-Serve 将请求 B 的预填充阶段拆分为多个块（B_1^P，B_2^P，B_3^P），并将这些块与请求 A 的解码阶段融合到同一批次中。具体而言，请求 B 的一个预填充块会与请求 A 的一个词元解码过程绑定，如 $A_6^D B_1^P$，$A_7^D B_2^P$，$A_8^D B_3^P$。假设请求 B 的预填充阶段被拆分为 n_c 个块，则请求 A 的停滞时间可缩短至原有时间的约 $\frac{1}{n_c}$。通过这种方式，停滞时间被平滑化，从而降低 TBT。然而，我们观察到：请求 B 的解码词元（B_1^D，B_2^D，…）的绝对延迟并未从该优化中获益。此外，这种拆分方式需频繁访问 KV 缓存，可能会无意中增加整体延迟而非降低延迟。

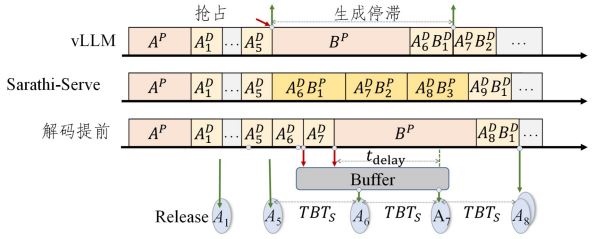


图 4 迭代调度策略示例

Fig. 4 Illustration of iteration scheduling strategies

综上，分块预填充通过拆分预填充阶段并将其与其他请求的解码阶段融合，实现了 TBT 的平滑化。这引发一个思考：能否不通过拆分，而是通过手动调度预填充与解码阶段，实现更优性能？

本文提出一种简单的模拟策略——解码提前。该策略无需拆分，仅通过调度即可在 TBT 指标上实现与分块预填充相近的效果。如图 4 所示，将请求 A 的后续 n 个解码词元（A_6^D 与 A_7^D）提前至请求 B 的预填充阶段开始前生成。同时，不直接输出这些词元（若直接输出，A_7^D（第 n 个词元）与 A_8^D（第 $n+1$ 个词元）之间会出现较大 TBT），而是在请求 B 的预填充阶段内平滑输出这些词元。

为实现平滑输出，采用一种直观方法——为每个提前生成的词元的输出时间分配一个延迟 t_{delay}。如图 4 所示，尽管 A_6^D 与 A_7^D 已完成解码，但会在延迟 t_{delay} 后按顺序释放，且确保其输出时间不超过 B 的预填充阶段完成时间。该策略在保持整体延迟不变、缓解 TBT 过大问题的同时，实现了输出流的平滑化。此外，通过延迟首词元交付，该策略还可在 TTFT 与 TBT 或 TPOT 之间进行权衡。

输出延迟技巧反映了现有 SLO 的不合理性，而解码提前策略进一步表明，以这些不合理 SLO 为目标的优化，未必能提升用户体验。

3.3 新型 SLO 定义

3.3.1 设计思路

实际上，用户在生成过程中通常不会注意到最后一个单词的延迟。只要词元交付速度与用户阅读速度对齐，生成停滞未必会损害用户体验。鉴于 TBT 在约束相邻词元时间间隔方面的局限性，本文将 SLO 的关注点转向实际用户体验。例如，根据用户可容忍的响应延迟与输出信息处理速度（如阅读聊天机器人输出、理解长文本摘要、倾听等），为每个请求设定约束。

3.3.2 具体定义

将新型 SLO 纳入上述框架，可表示为：

$$d_i = V \times i \tag{5}$$

其中，V 为用户的输出信息处理速度；i 为输出词元的索引；d 为第 i 个词元的截止时间，超过该时间，用户会感知到输出流的停滞。

4 系统级指标

基于请求级别的 SLO，研究者提出了系统级指标，以衡量服务性能。本章首先重新审视现有系统级指标（包括 SLO 达成率与有效吞吐量），分析它们在 LLM 服务中的不足；随后提出一种名为“平滑有效吞吐量”的新型指标，用于衡量 LLM 服务性能。

4.1 SLM 的公式表示

SLO 仅关注请求级别的用户体验，而从系统视角来看，服务提供商更关注服务的整体性能。

在介绍平滑有效吞吐量之前，首先重新探讨现有系统级指标（SLO 达成率与有效吞吐量），并分析它们在 LLM 服务中的不足。

对于 LLM 服务，设请求集合为 R，$|R|$ 为请求总数。则现有系统级指标可表示为：

1）SLO 达成率

SLO 达成率用于描述满足 SLO 的请求占比，可定义为：

$$SLO_{\text{attainment}} = \frac{\sum_{r \in R} \mathbb{I}(\forall i, t_i \leqslant d_i)}{|R|} \tag{6}$$

其中，$\mathbb{I}(\cdot)$ 为指示函数，若请求满足 SLO，则返回 1，否则返回 0。基于 SLO 达成率，可定义“容量”：在特定 SLO 达成率约束下，系统能处理的最大请求速率。

2）有效吞吐量

有效吞吐量为服务中每秒内完成且满足 SLO 的请求数量，综合考虑了资源利用率与用户体验的权衡关系，可定义为：

$$Goodput = \frac{\sum_{r \in R} \mathbb{I}(\forall i, t_i \leqslant d_i) \cdot n_r}{T} \tag{7}$$

其中，T 为处理请求集合 R 的时间，n_r 为请求 r 生成的词元数量。

4.2 现有 SLM 的反直觉示例

本文观察到：若一个请求未满足 SLO，则其对 SLO 达成率与有效吞吐量的贡献均为 0。在以系统级指标为优化目标时，这种特性往往会导致系统丢弃无法满足 SLO 的请求[38]。具体而言，对于在第 i 个词元生成过程中已无法满足 SLO 的请求 r，有：

$$\mathbb{I}(\forall i, t_i \leqslant d_i) = 0 \tag{8}$$

即该请求对有效吞吐量与 SLO 达成率的贡献均为 0。从有效吞吐量最优调度策略来看，应终止该请求，并优先处理可满足 SLO 的下一个请求，否则资源会被浪费在无法满足 SLO 的请求上。

尽管延迟确实会损害用户体验，但完全丢弃请求的负面影响更为严重。试想：当你正在阅读某个问题的答案时，若系统中途停止生成词元，相比多等待几秒以获取完整答案，这种情况会让你更加沮丧。此外，用户可能会重新提交请求（因为他们需要答案），这会进一步增加系统负载，导致资源浪费与用户等待时间延长。

因此，本文认为现有有效吞吐量与 SLO 达成率指标不适合作为 LLM 服务系统的优化目标。主动丢弃未满足 SLO 的请求既违背直觉，又会导致用户体验恶化。4.3 节将提出一种名为“平滑有效吞吐量”的新型指标，以解决这一问题。

4.3 平滑有效吞吐量

现有有效吞吐量指标存在明显不足，因此新型指标需充分考虑每个请求的贡献——即使该请求略微超出 SLO 要求。在这种情况下，用户在读完所有已交付词元后，需等待下一个词元生成。

4.3.1 流式服务与用户体验

与单次前向推理模型不同，由于 LLM 的自回归特性，交互式 LLM 应用通常以流式服务形式部署。针对网页流式服务的研究[26]表明，用户等待时间是影响用户体验的关键因素。

为此，引入“用户等待时间”（即用户空闲延迟）的概念，用于衡量用户体验。用户空闲延迟指的是由于词元生成速度过慢，用户处于空闲状态并等待新词元生成的累积时长。具体而言，请求 r 的用户空闲延迟 l_r 可定义为：

$$l_r=\max_{i=1}^{n}(t_i-d_i) \tag{9}$$

其中，t_i 为第 i 个词元的生成时间，d_i 为第 i 个词元交付给用户的截止时间，n 为请求 r 的输出词元数量。在实际场景中，用户等待延迟是请求生成过程中用户无词元可阅读、只能等待下一个词元生成的总时长。用户空闲延迟越大，用户体验越差。

4.3.2 平滑有效吞吐量的定义

平滑有效吞吐量定义为单位时间内的服务效益。单个请求的效益由请求生成的词元数量与请求的用户空闲延迟两个因素决定。直观来看，词元生成时间线代表服务提供商的“收益流”，而用户空闲时间则代表用户体验的“损失”。吞吐量越高、空闲时间越短，服务的潜在效益越大。具体而言，请求 r 的效益可定义为：

$$benefit(r)=n_r-\alpha\cdot f(l_r) \tag{10}$$

其中，n_r 为请求 r 生成的词元数量，l_r 是式(9)定义的用户等待延迟，$f(\cdot)$ 为映射函数——将用户等待延迟转换为请求效益的损失比例，α 为权重。在实现过程中，$f(\cdot)$ 是一个线性函数，但用户也可以修改成平方或者指数函数，来表示更大时延会有更多的惩罚。对于延迟要求严格的交互式应用，应选择更大的 α 值，以确保等待延迟最小化。在实际部署中，可利用历史工作负载数据（包括请求延迟指标与用户行为数据，如请求取消、投诉等）校准效益函数的参数，使服务特性与效益计算更贴合实际需求。

平滑有效吞吐量可定义为：

$$smooth\_goodput=\frac{\sum_{r\in R}benifit(r)}{T} \tag{11}$$

其中，T 为处理请求集合 R 的时间。Andes[36]也考虑了未满足 SLO 请求的效益，但其关注的是词元相对于 SLO 截止时间的平均延迟，而本文关注的是词元的最大延迟（即用户空闲延迟）。在实际场景中，一旦延迟发生，后续的“追赶”无法弥补用户已感知的延迟——最大延迟反映了整个请求过程中词元生成时间与截止时间的最大偏差，对应用户等待词元生成的总时长。因此，在这一场景下，平滑有效吞吐量更为合理。

5 实验及其结果分析

本章首先基于所提出的统一指标框架，重新评估不同调度策略的性能；随后分析评估结果，总结 LLM 服务面临的挑战；最后通过与现有指标对比，验证平滑有效吞吐量的优势。

5.1 实验设置

5.1.1 硬件与软件环境

实验在配备 NVIDIA A100-SXM4-80 GB GPU 的服务器上进行，操作系统为 Debian GNU/Linux 12，CUDA 版本为 12.2。实验以 LLaMA-3.1-8B-instruct[1]与 Qwen2.5-14B[39]为基础模型，所有代码基于 vLLM 0.6.3 开发，所需依赖包版本均与 vLLM 要求一致。

5.1.2 工作负载

工作负载方面，采用 ShareGPT[40]模拟聊天机器人对话场景，采用 LooGLE[41]模拟长对话场景。参考之前文献的设置[11]，请求到达时间服从泊松分布，或采用真实世界轨迹数据（平均到达率作为参数）模拟请求到达过程。此外，通过真实世界轨迹实验，评估系统在实际场景中的性能。

5.1.3 评估指标

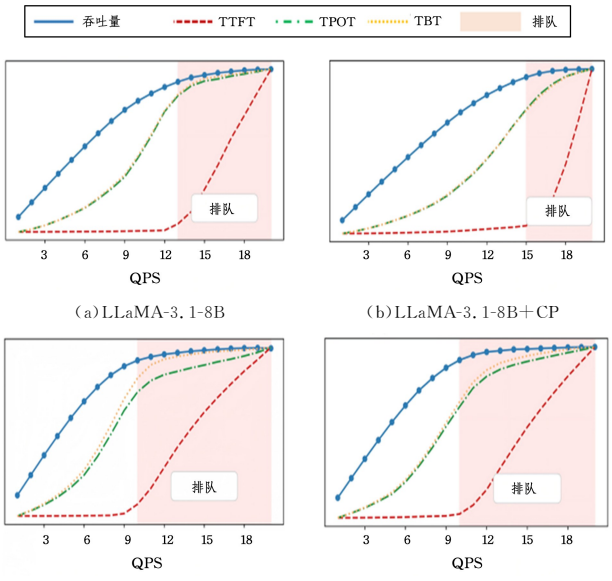
采用平滑有效吞吐量评估 LLM 服务性能，作为对比，同时采用现有 SLO 与系统级指标进行评估。

5.2 基于现有指标与平滑有效吞吐量的分析

首先通过现有指标分析不同策略的性能，揭示 vLLM 在不同请求速率下的统计规律并探究其根本原因；随后在相同调度策略下引入平滑有效吞吐量，挖掘现有指标无法捕捉的新洞察，更全面地理解系统性能与用户体验的权衡关系。

5.2.1 基于现有指标的分析

图 5 展示了在 ShareGPT 数据集（提示词与响应均较短）上，vLLM 在不同请求速率下的性能，其中 CP 表示采用分块预填充。在系统未饱和阶段，随着请求速率增加，系统资源利用率提升，并行处理能力增强，吞吐稳步上升；同时，批次大小增加，导致批次处理时间延长，进而使中位 TBT、TPOT 等指标数值升高。当系统达到容量上限后，继续提高请求速率会导致更多请求进入队列，使平均 TTFT 显著增加。该分析表明：现有指标虽能全面呈现服务性能，但更侧重于吞吐量与硬件效率，而非实际用户体验；无法明确识别出同时兼顾原始吞吐量与用户感知等待时间的平衡工作点。



(a)LLaMA-3.1-8B (b)LLaMA-3.1-8B+CP

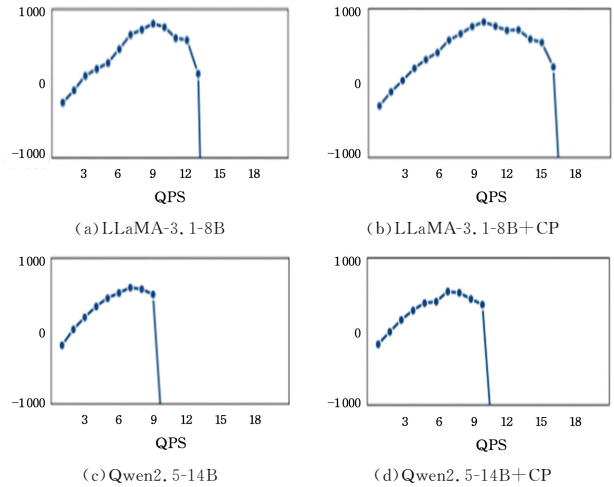
(c)Qwen2.5-14B (d)Qwen2.5-14B+CP

图 5 不同 QPS 下 vLLM 的性能

Fig. 5 Performance of vLLM under different QPS

5.2.2 基于平滑有效吞吐量的分析

与之相反，平滑有效吞吐量通过综合考虑吞吐量与用户空闲时间，量化了服务为用户带来的效益。实验中设置信息消费速度为 5 词元/秒，$\alpha=10$。如图 6 所示，在系统未饱和阶段，随着请求速率增加，平滑有效吞吐量上升——因为新增吞吐量带来的效益超过了用户空闲时间增加带来的损失。然而，当系统负载继续增加，用户空闲时间成为主导因素，单个请求的效益开始下降，导致平滑有效吞吐量降低。值得注意的是，采用分块预填充的系统比标准 vLLM 系统在更高请求速率下达到了平滑有效吞吐量峰值。这是因为分块预填充通过融合预填充与解码阶段，最大化 GPU 并行性，使系统在队列延迟显著影响性能前，能处理更多请求。综上，该扩展分析强调了在 LLM 服务系统中同时考虑吞吐量与用户体验的重要性，并证明平滑有效吞吐量相比传统指标能更均衡、更深入地反映系统性能。



(a)LLaMA-3.1-8B (b)LLaMA-3.1-8B+CP

(c)Qwen2.5-14B (d)Qwen2.5-14B+CP

图 6 不同 QPS 下 vLLM 的平滑有效吞吐量

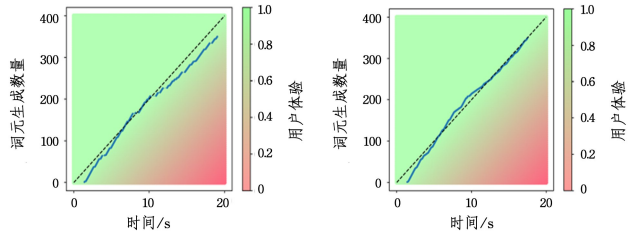
Fig. 6 Smooth goodput of vLLM under different QPS

5.3 基于 SLO 的分析

通过实验系统验证新型 SLO 能否准确衡量单个请求的效益。实验聚焦于平均长度约为 1600 词元的提示词，从两种策略的服务日志中筛选出在相同条件下处理的相同请求，进行直接对比。

5.3.1 请求级性能分析

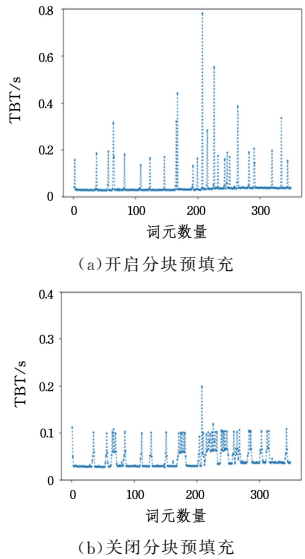
图 7 和图 8 展示了相同请求在“采用分块预填充”与“不采用分块预填充”两种模式下的词元生成过程。采用分块预填充后，词元生成过程的停滞显著减少，这得益于预填充与解码阶段的更深度融合，使词元交付时间线更平滑。然而，数据显示，部分由预填充抢占导致的生成停滞并未被用户感知——只要已交付的词元数量足够，则后续停滞对用户感知的响应速度影响较小。



(a)开启分块预填充 (b)关闭分块预填充

图 7 vLLM 的词元交付时间线

Fig. 7 Token delivery timeline of vLLM



(a)开启分块预填充

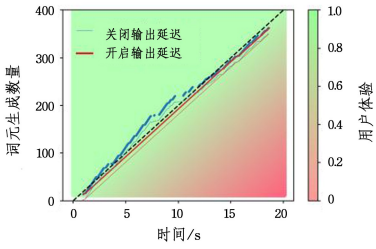
(b)关闭分块预填充

图 8 vLLM 的 TBT 指标

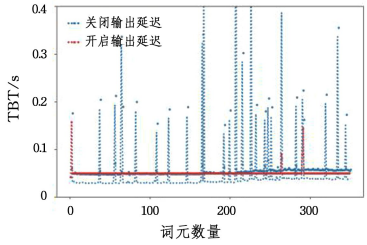
Fig. 8 TBT of vLLM

5.3.2 输出延迟技巧的验证

为进一步验证对 SLO 的讨论结果，引入输出延迟技巧。如图 9(a)所示，该技巧通过缓冲词元并以刻意降低的速率释放词元实现，完全独立于框架底层调度策略——可在服务器端或客户端同等实现。与无延迟场景相比，输出延迟技巧能有效降低长尾 TBT(见图 9(b))，同时不影响服务整体吞吐量。本质上，尽管该技巧使 TBT 几乎保持恒定(与用户典型信息消费速度对齐)，但并未缩短用户实际空闲时间。因此，尽管在传统评估指标中表现更优，但用户实际体验到的总空闲时间基本不变。这一观察进一步证明，传统指标无法准确捕捉真实用户体验，亟需平滑有效吞吐量这类更全面的衡量指标。



(a)词元交付时间线



(b)TBT 指标

图 9 输出延迟技巧示例

Fig. 9 Illustration of output delay trick

5.4 扩展至预填充-解码解耦架构

本节展示所提框架在预填充-解码解耦 LLM 服务中的可扩展性。在解耦架构中，预填充与解码阶段在不同实例上执行，这为解码实例实现更大批次大小、减少生成停滞提供了可能。因此，本节聚焦于解码阶段，通过现有指标与所提框架评估其性能。

尽管解耦架构减少了预填充阶段导致的生成停滞，但对于输出长度较长的请求，解码阶段的持续时间相对较长，TBT 会随输出长度增加而升高。当请求最终违反 SLO 时，可能会触发迁移或抢占等操作，导致显著开销。

图 10 展示了解耦 LLM 服务中解码阶段的词元交付时间线。由于解码阶段在独立实例上执行，可观察到，解码阶段不会被预填充阶段抢占，词元交付时间线更稳定。然而，解码阶段仍受输出长度与批次大小影响——随着交付速度降低，在红色点 a 处长尾 TBT 增加。但由于初始阶段已交付足够词元，用户的实际体验并未受到显著影响。这一观察与平滑有效吞吐量的评估结果一致：平滑有效吞吐量在紫色点 b 处触发迁移(紫色点 b 表示用户无词元可阅读，将面临空闲时间)。得益于对整个时间线的综合考虑，平滑有效吞吐量触发的迁移与抢占事件更少，表明系统能更高效地处理请求，避免不必要的操作。

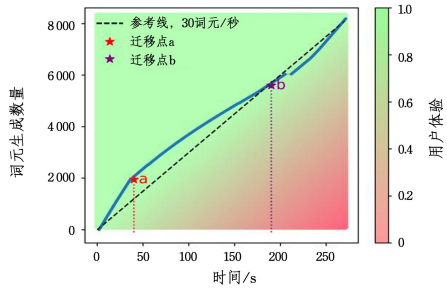


图 10 预填充-解码解耦架构中的词元交付时间线

Fig. 10 Token delivery timeline in prefill-decode disaggregated architecture

5.5 参数敏感性分析

α 用于衡量用户等待时延对用户体验的影响大小，更大的 α 对应更严格的时延要求。α 参数的不同设置将影响平滑有效吞吐量的变化趋势，从而影响最大值点对应的 QPS。在完全相同的实验场景下，设置了不同的 α 值，以展现 α 参数对最大值点对应的 QPS 的影响。结果如图 11 所示，随着 α 的增加，QPS 逐渐降低。

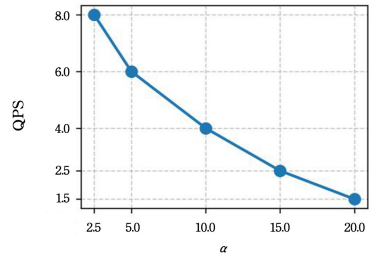


图 11 α 的设置对 QPS 的影响

Fig. 11 Impact of α on QPS

6 讨论

本章讨论以现有指标为目标的系统优化方向，以及平滑有效吞吐量对这些优化设计的影响。

6.1 基于现有指标的优化

6.1.1 以吞吐量为目标的优化

Orca[10]引入连续批处理(Continuous Batching)技术，动态构建并处理批次，充分利用 GPU 并行性。在此基础上，vLLM[11]进一步集成页式注意力(Paged Attention)技术，显著提升计算吞吐量并降低运营成本，已成为推理服务领域广泛采用的最先进(SOTA)框架。这类以吞吐量为核心目标的研究未明确定义 SLO，因此无法将其性能与以 SLO 达成率、有效吞吐量为目标的研究直接对比。

6.1.2 以 SLO 达成率为目标的优化

Splitwise[18]与 TetriInfer[27]基于预填充与解码阶段在计算和内存访问特性上的差异，提出将二者拆分到不同设备上执行。Sarathi-Serve[19]引入分块预填充与无停滞批处理(Stall-free Batching)技术，以缓解生成停滞问题。SCOOT[21]提出一种自动参数调优系统，为系统寻找满足 SLO 的最优配置。这些研究通过优化不同指标(如 TTFT，TBT，TPOT)的 SLO 达成率，使更多请求能在 SLO 约束下得到处理。QLM[22]强调等待时间与执行时间对实现 E2E 延迟型 SLO 的影响。对于这类研究，SLO 基于不同指标定义，在相同 SLO 要求下，平滑有效吞吐量可作为统一框架，结合效率与用户体验，对比不同研究的性能。

6.1.3 以提升有效吞吐量为目标的优化

DistServe[20]通过避免预填充与解码阶段的干扰，在 TTFT 与 TPOT 型 SLO 约束下实现了更高的有效吞吐量，即每秒内能处理更多满足 SLO 的请求。对于这类以提升有效吞吐量为设计目标的研究，平滑有效吞吐量能在细粒度层面为所有请求分配效益，更贴合用户体验需求。

综上，尽管现有研究采用的指标各异，但 Splitwise，DistServe，TetriInfer 等已发现相似的优化方向。然而，无法在不同衡量体系下直接对比这些研究的优化效果，为优化策略选择带来了挑战。

6.2 未来研究方向

最新的“慢思考”模型[15-16,42]在向用户交付词元前，需经历漫长的思考过程——这表明需要更精细的、“语义感知”的SLO[43]。这一观察启发了探索 SLO 设计的创新方向：根据输入请求的语义内容与复杂度，调整 SLO 要求的粒度。例如，对于包含更复杂或更高价值信息的请求，可设置更宽松的SLO，为模型提供更多时间处理并生成更丰富、更详细的响应，同时不损害整体用户满意度。

此外，随着模型在规模与推理能力上的不断发展，输出吞吐量与输出质量之间的相互作用愈发显著。未来研究可探索“自适应 SLO 框架”——基于输入复杂度、系统负载与预期输出质量的实时评估，动态调整性能要求。这种语义感知的方法不仅能更有效地平衡吞吐量与用户体验，还能为更具韧性、更贴合上下文的优化策略奠定基础。

同时，探索这类自适应 SLO 有助于构建更能适应现代语言模型多样化运行模式的系统，即从快速推理到慢思考。这将使系统能在延迟与输出质量之间实现更精细的权衡，确保即使在处理需求较高的场景下，整体用户体验仍能保持稳定与良好。

这些方向为未来研究开辟了广阔前景，最终目标是构建一个统一框架，在 LLM 服务中实现吞吐量、响应质量与语义深度的协同优化。

结束语 本文提出一种用于评估 LLM 服务性能的指标框架。研究表明，现有指标无法准确捕捉用户体验，并揭示了流式 LLM 服务中用户体验与输出交付速度的关联。为此，本文引入“平滑有效吞吐量”指标，用于衡量单位时间内的服务效益，综合考虑服务效率与用户体验。通过在多种工作负载与模型上的实验，验证了该框架能更全面地呈现 LLM 服务的词元交付与请求处理过程，有效捕捉不同服务策略下用户体验与系统性能的最优平衡点，为 LLM 服务性能评估提供更贴合实际需求的统一视角。期望该框架能为 LLM 服务性能评估提供统一标准，并推动 LLM 服务优化领域的研究发展。

未来研究中，一方面可进一步探索“语义感知型 SLO”设计，结合输入请求的内容复杂度（如专业领域文本、长文档摘要等）动态调整 SLO 约束粒度，为高价值、高复杂度请求预留更充足的处理时间，以保障输出质量；另一方面，可针对 LLM 服务的动态场景（如潮汐式请求负载、模型推理精度与速度的权衡需求），研究自适应平滑有效吞吐量优化策略，通过实时感知系统资源状态与用户体验反馈，动态调整 α 权重与用户信息处理速度参数，使指标框架更适配真实服务环境的动态变化。此外，对于预填充-解码解耦等新型架构，还可深入分析不同硬件部署模式（如边缘节点与云端协同）对平滑有效吞吐量的影响，为 LLM 服务的架构选型与资源调度提供更具体的指导。

参 考 文 献

[1] TOUVRON H, LAVRIL T, IZACARD G, et al. Llama: Open and efficient foundation language models[J]. arXiv: 2302.13971, 2023.

[2] WAKE A, CHEN B, LYU C X, et al. Yi-Lightning Technical Report[J]. arXiv: 2412.01253, 2024.

[3] Google-DeepMind. Gemini 2.0[EB/OL]. [2025-10-15]. https://deepmind.google/technologies/gemini/.

[4] xAI. Bringing Grok to Everyone[EB/OL]. [2025-10-15]. https://x.ai/.

[5] OpenAI. ChatGPT [EB/OL]. [2025-10-15]. https://chat.openai.com.

[6] ZHENG L, CHIANG W L, SHENG Y, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena[C]// Advances in Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc., 2023: 46595-46623.

[7] MONTAGNA S, FERRETTI S, KLOPFENSTEIN L C, et al. Data decentralisation of LLM-based chatbot systems in chronic disease self-management[C]// Proceedings of the 2023 ACM Conference on Information Technology for Social Good. New York: ACM, 2023: 205-212.

[8] VU M D, WANG H, LI Z, et al. GPTVoiceTasker: LLM-powered virtual assistant for smartphone[J]. arXiv: 2401.14268, 2024.

[9] DONG X L, MOON S, XU Y E, et al. Towards next-generation intelligent assistants leveraging llm techniques[C]// Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York: ACM, 2023: 5792-5793.

[10] YU G I, JEONG J S, KIM G W, et al. Orca: A distributed serving system for {Transformer-Based} generative models[C]// 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22). Carlsbad, CA: USENIX Association, 2022: 521-538.

[11] KWON W, LI Z, ZHUANG S, et al. Efficient Memory Management for Large Language Model Serving with PagedAttention[C]// Proceedings of the 29th Symposium on Operating Systems Principles. New York: ACM, 2023: 611-626.

[12] CHENG K, HU W, WANG Z, et al. Enabling efficient batch serving for lmaas via generation length prediction[C]// 2024 IEEE International Conference on Web Services (ICWS). IEEE, 2024: 853-864.

[13] ZHANG P, SU L, YANG J, et al. Topology-aware Preemptive Scheduling for Co-located LLM Workloads[J]. arXiv: 2411.11560, 2024.

[14] ZHU K, ZHAO Y, ZHAO L, et al. NanoFlow: Towards Optimal Large Language Model Serving Throughput[J]. arXiv: 2408.12757, 2024.

[15] GUO D, YANG D, ZHANG H, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning[J]. arXiv: 2501.12948, 2025.

[16] JAECH A, KALAI A, LERER A, et al. OpenAI o1 System Card[J]. arXiv: 2412.16720, 2024.

[17] ONG I, ALMAHAIRI A, WU V, et al. RouteLLM: Learning to Route LLMs with Preference Data[J]. arXiv: 2406.18665, 2024.

[18] PATEL P, CHOUKSE E, ZHANG C, et al. Splitwise: Efficient Generative LLM Inference Using Phase Splitting[C]// 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). IEEE, 2024: 118-132.

[19] AGRAWAL A, KEDIA N, PANWAR A, et al. Taming {Throu-

ghput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}[C]// 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24). Santa Clara, CA: USENIX Association, 2024: 117-134.
[20] ZHONG Y, LIU S, CHEN J, et al. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving[J]. arXiv: 2401.09670, 2024.
[21] CHENG K, WANG Z, HU W, et al. SCOOT: SLO-Oriented Performance Tuning for LLM Inference Engines[J]. arXiv: 2408.04323, 2024.
[22] PATKE A, REDDY D, JHA S, et al. One Queue Is All You Need: Resolving Head-of-Line Blocking in Large Language Model Serving[J]. arXiv: 2407.00047, 2024.
[23] VASWANI A, SHAZEER N, PARMAR N, et al. Attention Is All You Need[J]. arXiv: 1706.03762, 2017.
[24] WEINREICH H, OBENDORF H, HERDER E, et al. Not quite the average: An empirical study of Web use[J]. ACM Transactions on the Web, 2008, 2(1): 1-31.
[25] SKADBERG Y X, KIMMEL J R. Visitors' flow experience while browsing a Web site: its measurement, contributing factors and consequences[J]. Computers in Human Behavior, 2004, 20(3): 403-422.
[26] EGGER S, HOSSFELD T, SCHATZ R, et al. Waiting times in quality of experience for web based services[C]// 2012 Fourth International Workshop on Quality of Multimedia Experience. IEEE, 2012: 86-96.
[27] HU C, HUANG H, XU L, et al. Inference without interference: Disaggregate llm inference for mixed downstream workloads[J]. arXiv: 2401.11181, 2024.
[28] SUN B, HUANG Z, ZHAO H, et al. Llumnix: Dynamic Scheduling for Large Language Model Serving[C]// 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24). Santa Clara, CA: USENIX Association, 2024: 173-191.
[29] KOSSMANN F, FONTAINE B, KHUDIA D, et al. Is the GPU Half-Empty or Half-Full? Practical Scheduling Techniques for LLMs[J]. arXiv: 2410.17840, 2024.
[30] WU B, ZHONG Y, ZHANG Z, et al. Fast Distributed Inference Serving for Large Language Models[J]. arXiv: 2305.05920, 2023.
[31] JIANG X, ZHOU Y, CAO S, et al. NEO: Saving GPU Memory Crisis with CPU Offloading for Online LLM Inference[J]. arXiv: 2411.01142, 2024.
[32] BRYSBAERT M. How many words do we read per minute? A review and meta-analysis of reading rate[J]. Journal of Memory and Language, 2019, 109: 104047.
[33] QIAO Y, ANZAI S, YU S, et al. ConServe: Harvesting GPUs for Low-Latency and High-Throughput Large Language Model Serving[J]. arXiv: 2410.01228, 2024.
[34] WANG Z, LI S, LI X, et al. Echo: Efficient Co-Scheduling of Hybrid Online-Offline Tasks for Large Language Model Serving[J]. arXiv: 2504.03651, 2025.
[35] ZHAO Y, YANG S, ZHU K, et al. BlendServe: Optimizing Offline Inference for Auto-regressive Large Models with Resource-aware Batching[J]. arXiv: 2411.16102, 2024.
[36] LIU J, WU Z, CHUNG J W, et al. Andes: Defining and Enhancing Quality-of-Experience in LLM-Based Text Streaming Services[J]. arXiv: 2404.16283, 2024.
[37] AGRAWAL A, PANWAR A, MOHAN J, et al. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills[J]. arXiv: 2308.16369, 2023.
[38] QIN R, LI Z, HE W, et al. Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving[J]. arXiv: 2407.00079, 2024.
[39] YANG A, YANG B S, ZHANG B C, et al. Qwen2.5 Technical Report[J]. arXiv: 2412.15115, 2024.
[40] ECCLESTON D. ShareGPT[DB/OL]. [2025-10-15]. https://github.com/domeccleston/sharegpt.
[41] LI J, WANG M, ZHENG Z, et al. LooGLE: Can Long-Context Language Models Understand Long Contexts? [J]. arXiv: 2311.04939, 2023.
[42] ZHAO Y S, WANG Y D, JI M Y. Overview of reasoning with large language models based on thought chain prompts[J]. Journal of Harbin Vocational and Technical College, 2025(4): 5-7.
[43] SHEN Y, ZHANG J, HUANG J, et al. DAST: Difficulty-Adaptive Slow-Thinking for Large Reasoning Models[J]. arXiv: 2503.04472, 2025.

WANG Zhibin, born in 1996, Ph.D, is a member of CCF (No. 62267M). His main research interests include graph computing, mining learning, and machine learning system.

GU Rong, born in 1988, Ph.D, associate professor, Ph.D supervisor, is a member of CCF (No. 32327S). His main research interests include cloud and big data computing systems, distributed AI training and inference system, intelligent data management system.

(责任编辑:喻藜)