通过黑箱在线调优提升大语言模型性能：将系统规格纳入可信人工智能事实表的理由

1st YonasAtinafu滑铁卢大学大卫·R·切里顿计算机科学学院加拿大滑铁卢
yatinafu@uwaterloo.ca

2nd HenryLin滑铁卢大学大卫·R·切里顿计算机科学学院加拿大滑铁卢
h293lin@uwaterloo.ca

3rd RobinCohen滑铁卢大学大卫·R·切里顿计算机科学学院加拿大滑铁卢
rcohen@uwaterloo.ca

arXiv:2603.11340v1 [cs.AI] 11 Mar 2026

摘要——**本文提出了一种新颖的黑箱在线控制器方法，该方法仅利用短时间段内的端到端测量（无需内部仪器监测），并采用爬山法来最大化有效吞吐量（即满足服务级别目标的请求吞吐量）。我们提供了经验证据，表明这一设计有充分依据。以这一大语言模型服务领域的具体进展为鲜明案例，我们进一步讨论将系统性能和可持续性指标纳入 Factsheets 对于采用人工智能系统的组织的重要性。**

索引词——**可信人工智能、大语言模型系统性能、实证评估、仿真**

I. 引言

本文提出了部署环境中大语言模型吞吐量和面向用户的性能所面临的一项挑战：交互式服务受尾延迟主导 [1], , 而实际工作点则由少数几个服务栈设置决定（客户端并发度、批处理限制和推测解码参数）。由于这些设置之间的相互作用取决于工作负载和硬件，默认配置可能导致昂贵的 GPU 利用不足，也可能使系统承受过大的排队压力，造成 p99 延迟飙升，少数用户遭遇极端延迟。随后，我们提出一种解决这一问题的新方法：SLO-Tuner[1]，，一种黑盒在线控制器，仅使用短时段内的端到端测量结果（无需内部监测工具），并通过一种考虑 p99 值的爬山算法（该值表示负载下可能失败的服务请求比例达到百分之一的情形），最大化有效吞吐量（满足服务级别目标的请求吞吐量）。这里的 SLO（服务级别目标）是明确的尾延迟目标，例如 p99 $\leq$ 1.2 秒。我们提供了经验证据，表明这一设计有充分依据：在通过 vLLM 提供服务的 TinyLlama 上，SLO-Tuner 将类似默认的配置的 p99 延迟从 $\approx$1.36 秒降至 $\approx$0.70 秒，同时在 1.2 秒 SLO 下将有效吞吐量从 $\approx$8 提升至 $\approx$15 请求/秒。为支持高效探索和压力测试，我们还引入了一种轻量级离散事件模拟器，可捕捉主要的排队与批处理动态，并复现与在线系统相同的定性趋势。基于对大语言模型性能相关问题的这些认识，我们探讨了负责任人工智能的重要性，并指出用户事实表有必要且切实可行地纳入更多性能指标（包括尾延迟服务级别目标合规性），以支持可信人工智能。我们认为，部署的现实情况会影响可靠性、尾部公平性和实际采用，同时也倡导纳入可持续性指标。

贡献。 **[1] SLO 优先的目标：** 我们将大语言模型服务的在线调优表述为：在明确的 p99 SLO 下最大化有效吞吐量 （满足 SLO 的吞吐量），而非优化平均吞吐量/延迟。**[2] 将推测执行作为运行时控制：** 我们将推测解码参数视为可调的运行时设置，其满足 SLO 的取值取决于工作负载，而且可能不利于 p99。 **[3] 可移植的逻辑调节参数：** 我们引入一组面向运维人员的逻辑调节参数 （排队压力、批次形成、推测积极度），并配备一个轻量适配器，将它们映射到特定服务栈的标志参数，从而支持黑盒部署。**[4] 模拟器与在线系统的对齐：** 我们提供了一种离散事件模拟器，可捕捉主要动态，并在定性上与在线 vLLM 的行为相符，支持模拟器引导搜索和压力测试。

II. 调优大语言模型服务器以提升吞吐量和公平性

我们研究配置大语言模型服务栈这一实际问题：一种在实时约束下接收用户提示词并返回生成文本的系统。在部署中，

[1]代码和相关材料：https://github.com/Seoulsim2/SLO-Tuner。

运维人员面临一个常见的两难：追求更高的 GPU 利用率（通过增加并行请求和扩大批次）可能会显著增加部分用户的排队延迟。由于用户受到最严重的延迟影响尤为明显，我们重点关注尾延迟（第99百分位），并将满足第99百分位服务级别目标（SLO）视为首要约束，同时最大化吞吐量。

背景：[1] 服务级别目标（SLO）与有效吞吐量。交互式大语言模型服务的评判标准是尾延迟，因此我们以第99百分位服务等级目标（SLO）为优化目标，例如 $p99 \leq 1.2$ 秒（99% 的请求在 1.2 秒内完成） [2]。第99百分位 SLO 是一项公平性约束：它可避免少数用户遭受极端延迟，而平均值看起来却一切正常。由于即使许多请求违反 SLO，原始吞吐量仍可能显得很高，因此我们优化有效吞吐量：即满足SLO 的请求速率（违反目标的请求不计入有效吞吐量）。 **[2] 服务器调节参数与"拐点"**。有效吞吐量和第99百分位延迟受少数服务器调节参数影响：（i）客户端并发度（正在处理中的请求数）；（ii）批处理（将多个序列分组以供 GPU 执行，例如 max_num_seqs）；以及（iii）推测解码参数。提高并发度或批大小可以提升 GPU 利用率，但也会加剧排队，常常导致一个第99百分位延迟急剧飙升的"拐点"。推测解码 [3] 使用较小的草稿模型来提出令牌，再由较大的目标模型进行验证；它可以降低平均延迟，但被拒绝的草稿会增加验证工作量和延迟波动，从而损害第99百分位延迟。目前的技术栈将推测解码设置作为运行时标志开放，但不会针对特定的第99百分位 SLO 自动调优 [4]。

A. 相关工作

尾延迟约束下的黑盒调优。现有系统将配置视为黑盒优化问题（例如 Otter- Tune [5], CherryPick [6]）。我们的场景在安全要求和优化重点上有所不同：大语言模型服务的主要瓶颈是尾延迟 [1], ，因此我们仅使用端到端测量结果（不进行插桩），在明确的 p99 服务等级目标下优化有效吞吐量，并在接近服务等级目标边界时在线 进行调优。**面向服务等级目标的推理引擎调优：** SCOOT [7] 通过贝叶斯优化，在压力测试下优化推理引擎参数以满足大语言模型服务的服务等级目标，并利用随机森林学习隐藏的可行性约束。与 SCOOT（不包括推测解码）不同，我们将推测解码视为一项一等运行时调节参数，并采用确定性的邻域爬山法，以及协同设计的离散事件模拟器，以安全地进行部署前探索并验证趋势。**服务质量与副本内控制：** Quasar [8]和 Sinan [9] 等服务质量感知系统主要通过集群调度/部署发挥作用。我们则针对副本内调节参数（并发度、批处理和推测解码）；即使集群级决策良好，这些参数仍可能导致 p99 不达标。我们的贡献是提出一种可移植、面向运维人员的本地控制问题表述（在 p99 约束下优化有效吞吐量），并提供一种黑盒控制器，可与现有集群调度器和自动伸缩器协同部署，而非取而代之。**将推测执行作为运行时调节参数：** 推测解码能够提高平均解码速度 [3] ，并有许多算法变体（例如，

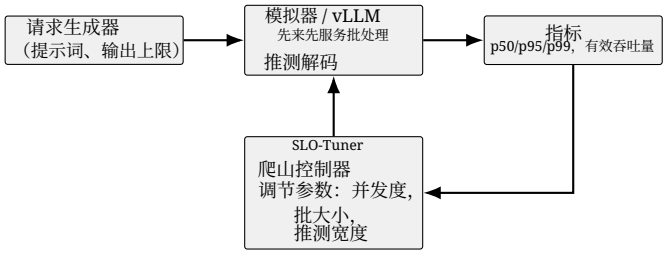


图 1. SLO-Tuner 将服务器视为黑盒，并根据尾延迟和有效吞吐量调节并发度、批大小和推测宽度。

Medusa/Hydra [10], [11]）；我们将推测执行视为一个运行时控制面，其最佳设置取决于工作负载和服务等级目标（如 vLLM [4]等服务栈所体现的那样）。**模拟器引导、测量校准的调优：** 我们使用轻量级离散事件模拟器，以低成本进行压力测试并提供趋势指导；最终决策则基于目标硬件上的在线测量结果。这种分工——模拟器提供指导，在线测量确保准确性——既支持实际部署，也保留了跨服务栈和加速器的可移植性。

B. SLO-Tuner 系统设计

我们将依次介绍概述、控制器算法、实现细节、模拟器、集成和惩罚项。**概述**。 图 1 展示了该架构。请求生成器会发出提示词，并指定最大输出长度。请求由以下任一方式处理：(i) 离散事件模拟器，或 (ii) 在线 vLLM 服务器。指标模块根据逐请求跟踪记录计算延迟百分位数（p50/p95/p99）和有效吞吐量。调优器执行爬山循环：运行一个短测量阶段，为配置评分，评估邻近配置（邻居），仅在评分有所提升时才进行调整。

设计目标： (1) 黑盒：仅使用标准 API（例如，兼容 OpenAI 的端点）和公开标志，从而确保可移植到不同服务栈。(2) *SLO* 优先：优先满足第99百分位服务等级目标，而非最大化原始吞吐量。(3) 预算感知：在有限数量的短阶段内完成，适用于真实部署。

与以往针对平均吞吐量或需要内部监测工具的调优工作不同，SLO-Tuner 根据黑盒测量结果优化满足服务等级目标的 吞吐量，并将推测解码视为一项一等运行时调节参数。概念上，SLO-Tuner 的做法是"尝试几种相近的旋钮设置，保留能提升满足服务等级目标的吞吐量的设置，并在第99百分位变差时回退。"

对于我们的**控制器算法：** 我们调整一个调节参数向量 $K = \{\text{conc}, \text{max seqs}, \text{spec tokens}, \text{spec on/off}\}$，其中 conc 表示客户端并发度，max_seqs 表示最大 _ 序列 _ 数，而 spec_tokens/spec_on/off 用于控制推测解码。控制器围绕一组小型逻辑调节参数 定义，用以体现与运维人员相关的权衡：排队压力（并发度）、批次形成（最大序列数）以及推测积极度。模拟器以因子化形式公开推测执行——草稿宽度 W

和验证器节奏 k——因此我们可以明确建模接受/验证动态，并压力测试推测执行如何增加尾延迟 [3]。实际服务栈并不总是公开相同的参数。例如，vLLM 提供 --speculative-model 和 --num-speculative-tokens 等公开标志，但并未提供一个与模拟器的验证器节奏 k [4]完全对应的独立调节参数。

SLO-Tuner 从设计上处理这一不匹配：爬山逻辑保持不变，只有一个轻量的 适配器 将逻辑旋钮向量映射到特定技术栈所提供的参数。在我们的实时 vLLM 实验中，我们因此调节一个受限但可部署的子集（并发度、max_num_seqs 和推测令牌宽度），无需依赖特权监测或内部调度器钩子。控制器从基线配置开始，反复评估当前设置以及附近旋钮设置组成的小范围邻域。对于每个候选项 K，我们测量端到端延迟百分位数（p50/p95/p99），并计算目标 SLO 下的有效吞吐量。

在我们的实现中，每次评估都会先进行短暂预热，随后进入固定的测量窗口；失败或无效的请求（例如超时或连接错误）会从百分位数计算中排除。

对于持续时间为 T 、每个请求延迟为 $\{\ell_i\}_{i=1}^{N}$ 的片段，我们定义

$$\text{goodput}(K) = \frac{1}{T}\sum_{i=1}^{N} \mathbf{1}\{\ell_i \le \text{SLO}\}, \tag{1}$$

并将 $p_{99}(K)$ 记作 $\{\ell_i\}$的经验第 99 百分位数。

由于某些旋钮的增大（例如提高并发度或增大批次）也会增加资源压力，因此我们纳入一个简单的硬件强度代理指标：

$$\begin{aligned}\text{hw_cost}(K) = {} & w_{\text{conc}}\,\text{concurrency} + w_{\max}\,\text{max_num_seqs} \\ & + w_{\text{spec}}\,\text{num_spec_tokens},\end{aligned} \tag{2}$$

在我们的 vLLM 运行中使用 w_{conc}=$w_{\max}$=0.01和 w_{spec}=0.02。选择这些权重，是为了让第99百分位服务等级目标违规相较于小幅利用率增益对得分的影响更大；所有实验中我们都保持这些权重不变，以确保比较一致。

vLLM 控制器最大化得分

$$\begin{aligned}S(K) = {} & \text{goodput}(K) - \lambda \max\big(0,\ p_{99}(K) - \text{SLO}\big) \\ & - \text{hw_cost}(K),\end{aligned} \tag{3}$$

其中， λ 是 SLO 惩罚权重（5.0）。我们设置 λ ，使得即使是轻微的第99百分位违规，其影响也超过相当程度的吞吐量增益；并且在所有运行中保持 λ 不变。这样可以直接抑制违反 SLO 的配置（这对用户不利），即使它们能提高原始利用率。

邻域生成是确定性的，步长根据旋钮范围确定：并发度 $\pm$2在 [2,16]以内；最大_序列数_在 $\pm$3 以内；推测宽度 $\pm$4在 [0,16]以内；此外还包括一个推测执行开启/关闭切换。所有候选值都会被限制在边界内。每一步都会评估当前点及其所有邻居；如果得分至少提升 δ=0.02，或者当前第99百分位违反服务等级目标且任一邻居能改善得分，控制器就会移动到最佳邻居。运行

算法 1SLO-Tuner 控制循环

要求： 步数预算 T，SLO 目标 SLO，阈值 δ

初始化 $K \leftarrow K_0$ （并发度 8，max_num_seqs 8，宽度 8，推测开启）

$K_{\text{best}} \leftarrow K$, $S_{\text{best}} \leftarrow -\infty$

适用于 $t = 1$ 至 T **执行**

在 K 运行一次预热+测量阶段

通过公式 (1) 计算 $p_{50}(K)$、 $p_{95}(K)$、 $p_{99}(K)$、有效吞吐量(K)

通过 (3) 计算 $S(K)$

如果 $S(K) > S_{\text{best}}$ **则**

$S_{\text{best}} \leftarrow S(K)$； $K_{\text{best}} \leftarrow K$

结束如果

$\mathcal{N}(K) \leftarrow K$ 的邻居（步长 $\pm$2, $\pm$3, $\pm$4，推测开/关）

对于所有 $K' \in \mathcal{N}(K)$ **执行**

计算 $S(K')$

结束循环

$K^\star \leftarrow$ 取最大值$_{K' \in \mathcal{N}(K)}\, S(K')$

如果 $S(K^\star) - S(K) \ge \delta$ **或** ($p_{99}(K)$ > 服务等级目标 **且** $S(K^\star)$ > S（K）） **然后**

$K \leftarrow K^\star$

结束条件判断

结束循环

在 K_{best} 重新运行一个片段并报告最终指标

在固定迭代次数后停止（vLLM 设置中为 8 次），不进行随机重启。

噪声处理依赖于可重复的片段时长、丢弃预热流量，以及每个步长评估所有邻居，而非采用随机采样。我们不会对各步长的指标取平均；相反，控制器会保留迄今为止表现最佳的配置，并在最后重新测量该配置以生成摘要。此外，模拟器会持续运行，直到同时达到最短时长和请求数，以平滑高方差片段；同时还会使用 p

99 $p_{99}^{(t)} =$ 的指 $\beta p_{99}^{(t)} + \ 1-\beta\ p_{99}^{(t-1)}$ $\beta \in \ 0,1$

数加权移动平均（EMA）来抑制短期尖峰：ˆ ( )ˆ，其中 ( )。

模拟器。我们实现了一个离散事件模拟器，用于探索趋势，并在调优在线服务器之前，以低成本对控制器进行压力测试。请求以稳态（泊松）或突发式开关负载模式到达；到达的请求会进入先到先服务队列；服务器空闲时，会组成一个最多包含 B 个请求的批次（也可选择最多等待 max_ 毫秒）。

每个批次包含两个阶段。预填充时间由批次中最长的提示词（以及经过校准的令牌速率）决定。解码时间取决于活跃序列的数量和推测解码行为。推测执行通过草稿宽度 W 和验证器节奏 k进行建模：接受率较高时，更高的草稿激进度可以减少验证器步数；但接受率较低时，也可能因额外的验证工作量而增加方差，从而推高第99百分位。

计时参数从根据测量结果校准的配置文件中加载，并加入少量噪声以体现变化。与在线系统一样，我们通过公式 (1) 报告 p50/p95/p99 和有效吞吐量。该模拟器并非用于精确预测内核级计时；

它用于捕捉排队与批处理动态，并引导搜索趋向满足服务等级目标的区域。

vLLM 集成。控制器使用选定的调节参数构造 vLLM 命令行，启动服务器，探测 /v1/models，并运行异步客户端以调用 /v1/chat/completions [12]。先预热 10 秒，再进行 30 秒的测量窗口；预热期间的响应会被丢弃。每个分段都会分别设置客户端并发度、服务器的 --max-num-seqs 以及 --num-speculative-tokens /--speculative-model [4], [13]。每个分段都会启动并停止服务器，且实现仅与 vLLM 的公共 API 交互。

成本和服务等级目标惩罚项。 服务等级目标惩罚权重 λ和硬件成本权重（并发度、max_num_seqs、推测宽度）用于权衡延迟违规与资源强度。两个控制器采用相同的有效吞吐量减惩罚结构，但模拟器还加入了草稿模型/验证器特定成本和更严厉的违规惩罚（将违规项乘以10λ），因此当第99百分位超过目标时会积极回退。

C. 结果

我们从三个层面呈现结果。首先，模拟器绘制了有效吞吐量与第99百分位的权衡，并展示批处理和推测设置如何影响稳态及突发负载下的 SLO曲面。其次，通过对 vLLM 进行在线调优，我们证明相同的爬山逻辑在真实服务栈中也能改善第99百分位和有效吞吐量。第三，通过有针对性的消融实验，我们分别分析并发度、批大小和推测宽度如何在明确的 SLO 下影响第99百分位。

实验设置：

模型和工作负载。我们使用 TinyLlama/TinyLlama-1.1B-Chat-v1.0，这是一款拥有 11 亿参数的聊天模型 [14]，，通过 vLLM [15] 提供服务。客户端重放单个提示词（“您是一位系统工程专家.....”），将响应限制为 64 个令牌，并将到达过程固定为采用闭环并发度的稳定负载。这种合成的单提示词工作负载有意排除了提示词组合变异性，使我们能够先单独分析服务参数对第99百分位和有效吞吐量的影响，再处理更复杂的轨迹。在模拟器中，提示词和输出长度遵循对数正态分布，以模拟“大提示词、适中输出”的场景。

指标与目标。 延迟百分位数按每个测量窗口计算；对于 vLLM，采用客户端观测到的端到端延迟。对于 vLLM 运行，我们将第99百分位服务等级目标设为 1.2 秒；在模拟器中，也使用相同的 1.2 秒第99百分位服务等级目标，以便在受控工作负载下对批处理和推测解码决策进行压力测试。我们沿用控制器设计中的评分函数，并将超参数与实现相匹配；只有达到服务等级目标时，有效吞吐量才会成为评分的主要决定因素。

模拟器配置。模拟器的搜索空间涵盖草稿宽度 $W \in [1,4]$、验证器宽度 $k \in [2,16]$、批大小 $B \in [1,32]$，以及可选的排队延迟 max_wait $\in [0,50]$ ms。每个片段会运行，直到生成完成数至少达到 1,500，且实际运行时间达到 90 s，同时满足 1.2 s 的第99百分位服务等级目标。我们将 W 限制在较小的值，因为对于 1.1B 模型，较宽的草稿会增加验证器和计算成本，收益却有限。预填充/解码令牌速率和噪声参数遵循配置的计时模块，并驱动离散事件模拟器。模拟器上的调优实验采用与 vLLM 相同的爬山逻辑，但由于片段开销较低，可以执行更多步长。

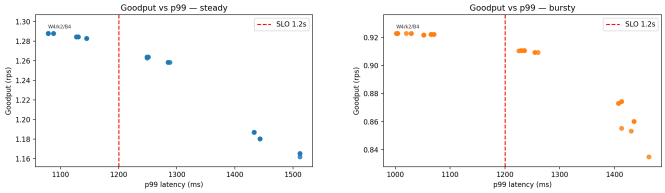
vLLM 配置。我们使用第 2 节所述的控制器（旋钮范围、相邻步长和评分方式均相同）。每次控制迭代先预热 10 秒（不计入测量），再测量 30 秒；控制器运行 8 次迭代，以限制调优时间。

硬件与软件。 vLLM 实验通过 SLURM 申请单个 GPU 节点，配备一块 NVIDIA L40S GPU（48 GB 显存）、 [16], 8 个 CPU 核心和 48 GB 随机存取存储器。客户端与服务器进程在同一节点上运行；每个阶段都会重启 vLLM，以确保旋钮变更能够正常生效。由于在我们的配置中无法安全地热切换这些标志，因此每个阶段结束后都会重启 vLLM；未来将致力于消除重启需求。我们使用 vLLM 0.6.1、FP16 权重和张量并行度 1，并在较新的 CUDA 与驱动程序栈上运行。

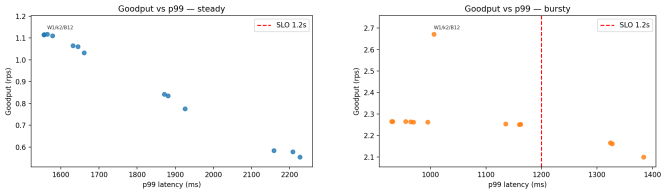
模拟器实验。我们首先使用离散事件模拟器绘制配置空间并在受控环境中对控制器进行压力测试，然后再接触真实服务器。所有模拟器运行均采用 config/defaults.yaml 中相同的时序和工作负载参数、1.2 秒的 p99 服务等级目标，以及配置的稳态或突发到达模式。

基线。基线网格对 $W \in \{1,2,4\}$、 $k \in \{2,4,8,12\}$ 和 $B \in \{4,8,16\}$ 进行扫描，涵盖稳态和突发到达。在稳态下得到的帕累托前沿（图 2a）表明，批处理至关重要：$B{=}1$ 会使 p99 达到 4.98 秒，而 $B{\geq}2$ 则可使 p99 接近 1.2 秒的边界（有时略高于该边界），具体取决于 W 和 k；接近边界的点可达到 $\approx$1.29 rps 的有效吞吐量。这促使我们采用感知服务等级目标的调优，以进入服务等级目标可行区域（$p99 \leq 1.2$ 秒），而不是依赖单一固定的批次设置。不过，对于较小的 W 和较保守的 k，适中的批次既能实现较高的有效吞吐量，又能将 p99 控制在 1.2 秒以内；批次过大最终会因等待时间而加剧长尾延迟。在突发模式下（图 2b），总体规律相同，但随着突发流量加剧排队，各点更接近服务等级目标边界，安全区域也缩小至较小的 W 和适中的 B；激进的 k 或极端的 B 值在两种模式下都远离前沿。

调优轨迹。 爬山控制器在每个状态阶段运行六个步长，从 $W{=}2$, $k{=}12$, $B{=}16$ 开始，并在 W, k, B 中评估单跳邻居以及可选的排队延迟。在稳态和突发状态阶段，控制器会迅速转向更小的 k 和略微降低的 B，最终收敛到 $W{=}1$, $k{=}2$, $B{=}12$ ，同时使 EMA p99 保持在服务等级目标附近。图 3a 展示了稳态阶段的调优轨迹：随着 k 降低，EMA p99 在最初几个步长中急剧下降，随后趋于稳定；与此同时，B 从 16 调整至 12。当连续移动违反服务等级目标时，控制器会回退，从而避免振荡。突发状态阶段的轨迹（图 3b）噪声更大，但遵循相同的定性路径：控制器首先摆脱高-k

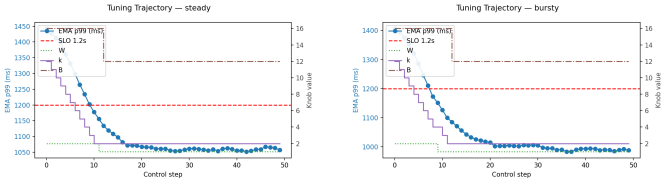


(a) 基线（稳态）。 (b) 基线（突发）。



(c) 压力测试（稳态）。 (d) 压力测试（突发）。

图2. 模拟器有效吞吐量–第99百分位权衡（稳态与突发）。



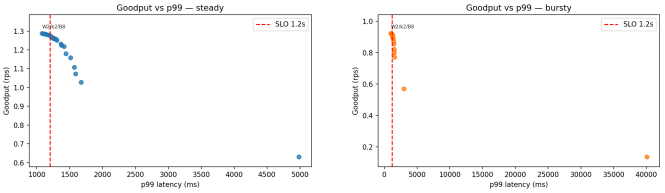
(a) 稳态到达下的调优。 （b）突发到达下的调优。

图 3. 模拟器爬山轨迹。

配置（这些配置会浪费验证器工作量），然后稳定在较小的-W、中等的-B 工作点上，使 p99 在突发情况下仍低于 1.2s 的目标。

压力测试。我们通过更长的提示词（平均长度按 1.5× 倍缩放）和更突发的通断到达过程对模拟器进行压力测试，同时将固定基线与调优后的配置进行比较。在这些场景下，在稳定负载下看似有吸引力的激进固定设置——较大的 B、较高的 k 或更宽的草稿——往往会使第99百分位接近或超过服务等级目标边界，而有效吞吐量仅略有提升。相比之下，调优后的配置通常会降低推测宽度，并避免极端批次，在维持与最佳固定配置相当的有效吞吐量的同时，使第99百分位保持在 1.2 秒目标附近（稳定负载见图 2c，突发负载见图 2d）。这进一步说明，相较于针对单一温和工作负载进行一次性的手动选择，有必要采用感知服务等级目标的调优。

消融实验。最后，模拟器消融实验围绕中间点配置，每次只调整一个参数：固定 $W{=}2$ and$B{=}8$，同时调整 k；固定 $k{=}8$and $B{=}8$，同时调整W；固定 $W{=}2$，$k{=}8$ ，同时将 B 从 1 扫描至 32。由此得到的帕累托图（稳态见图 4a，突发见图 4b）分别揭示了各项权衡：达到非常稀疏的验证后，增加 k 会使第99百分位持续上升，而有效吞吐量几乎没有收益；将草稿宽度扩展到超过 $W{=}2$ 几乎不会提升有效吞吐量，却可能在突发负载下推高第99百分位；而批次过小会导致服务器利用不足，批次过大则会



（a）消融实验（稳定负载）。 （b）消融实验（突发负载）。

图4. 针对推测宽度、验证器频率和批大小进行的模拟器消融实验。

延长排队时间并恶化尾部延迟。结合基线和调优运行结果，这些消融实验支持一条简单的经验法则：较小的推测宽度和适中的批次，对于在稳态和突发性工作负载下满足尾部延迟服务等级目标而言，都是最稳妥的选择。

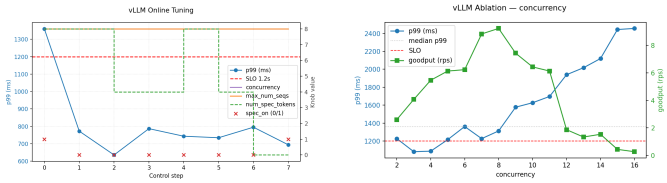
现在我们将讨论 vLLM 在线调优和消融实验，接着比较模拟器与 vLLM 的一致性，最后进行可移植性检查。

vLLM 在线调优和消融实验。在 vLLM（TinyLlama）上，控制器从类似默认配置的基线开始（并发度 8、max_num_seqs 8、推测宽度 8、开启推测执行），其第99百分位为 1.36 s，有效吞吐量为 8.13 每秒请求数。最初几个爬山法步长会尝试切换推测执行并缩小草稿宽度，同时保持并发度和批大小不变。降低推测宽度可显著提高有效吞吐量并降低第99百分位；中间宽度的结果参差不齐，有些区间略有改善，另一些则显示出 SLO 曲面的噪声。在 vLLM 中，将推测宽度设为 0 即可有效禁用推测执行（即使功能标志仍处于启用状态）；最佳点（宽度为 0，并发度和批大小不变）在 1.2 s 的 SLO 下达到 p99 ≈0.7 s 和 ≈15每秒请求数的有效吞吐量。图 5a 展示了调优轨迹和参数轨迹；并发度和批大小保持不变，而推测执行参数发生变化，这证实仅通过对推测宽度进行简单的逐坐标搜索，就足以摆脱默认配置。

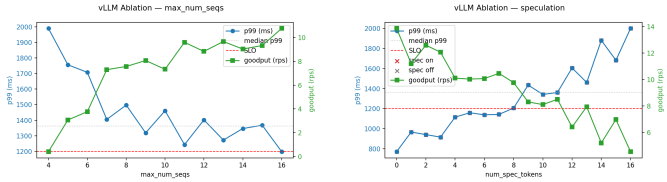
调优成本可控：进行 8 次迭代 × （当前 + 最多 7 个邻居），可得到约 64 个测量片段，每段 40 秒（含预热），此外还需对最佳点进行一次最终重新评估。在 L40S 上，完整运行约需 40–45 分钟的单 GPU 时间，主要耗时在于每个片段都要启动 vLLM。这一预算足够小，可以在部署或配置变更期间运行，远低于针对相同参数范围进行完整网格搜索的成本。

并发度呈现出明显的拐点：线程数从 2 增至 8 时，有效吞吐量从每秒 2.6 个请求升至每秒 9.2 个请求，而第99百分位从 1.23 秒升至 1.31 秒；但线程数超过 10 后，第99百分位就会超过 1.6 秒，且有效吞吐量在 16 个线程时骤降至每秒 0.27 个请求（图 5b）。原始吞吐量仍随并发度略有上升，但大多数请求都违反了 1.2 秒的服务等级目标，因此不会贡献任何有效吞吐量。这说明，在 SLO约束服务中，仅优化吞吐量会产生误导。

批大小（max_num_seqs）在一定程度上受益于更大的队列：从 4 提高到 16 可将有效吞吐量从



(a) 在线调优轨迹。 (b) 并发消融。



(c) 批大小消融。 (d) 推测宽度消融。

图 5. vLLM 实验：SLO-Tuner 调优轨迹及并发度、批大小和推测宽度的消融实验。

表 I vLLM 主要结果（在 vLLM 上运行 TinyLlama，p99 服务等级目标为 1.2 秒）。

旋钮设置	第99百分位（秒）	有效吞吐量（每秒请求数）
基线：推测宽度 8，推测开启	≈1.36	≈8.1
调优后：推测宽度 0，推测开启	≈0.70	≈15.0

0.40 每秒请求数提升至 10.77 每秒请求数，并将第99百分位从 2.00 秒降至 1.20 秒；排队拐点出现在约 11–13 个序列处，此时第99百分位越过 1.2 秒边界，有效吞吐量开始趋于平稳或下降（图 5c）。对运维人员而言，这表明中等批次已足够：将批大小推至硬件极限仅能带来边际有效吞吐量增益，同时若工作负载发生漂移，还可能面临更高的尾延迟风险。

对于此服务等级目标，推测宽度呈单调有害：关闭推测（宽度为 0）可达到 13.87 每秒请求数，第99百分位为 0.77 秒；宽度为 8 时降至 9.77 每秒请求数，第99百分位为 1.21 秒；宽度为 16 时进一步降至 4.53 每秒请求数，第99百分位为 2.00 秒（图 5d）。禁用或缩小草稿是在 TinyLlama 上满足 1.2 秒第99百分位目标的唯一方法。这强调了推测宽度应被视为一个可调旋钮，其最优设置取决于服务等级目标和工作负载，而不是一种始终有益的优化。

模拟器与 vLLM 的一致性。 模拟器与 vLLM 在推测宽度上呈现出相同的排序：较小的宽度（或关闭）可保持较低的第99百分位和较高的有效吞吐量，而更宽的草稿会提高第99百分位，却没有相应的有效吞吐量增益。对于可比的宽度，模拟的第99百分位为 1.25–1.36 秒，而 vLLM 上为 0.77–1.21 秒；数值大小不同，但随着宽度增加，有效吞吐量和第99百分位的趋势方向相同。模拟器将宽度上限设为 4，因此低估了 vLLM 中 16 个推测令牌处出现的急剧性能下降；使极端宽度行为对齐是未来的工作。

可移植性检查：Apple Silicon 上的 MLX。 为对 NVIDIA GPU 之外的可移植性进行合理性验证，我们使用 MLX [17] 在 Apple Silicon 上开展了一项小型补充评估，采用 Qwen-4B 目标模型和 Qwen-0.6B 草稿模型。我们扫描了三个面向操作员的调节参数——客户端并发度、推测执行的 lookahead 0，以及草稿宽度（由 top- 代理）——并将第99百分位趋势与模拟器进行比较（图 6）。

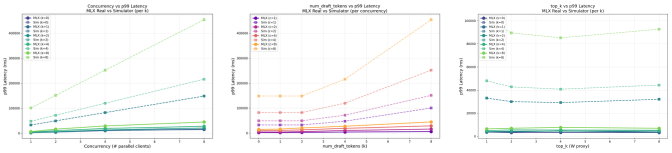


图 6. MLX 与模拟器：并发度、前瞻 k 和 top-k 宽度。虚线=模拟器，实线=MLX。

虽然模拟器在绝对延迟方面表现出明显的校准偏差，但它能够匹配调节参数变化所带来的方向性响应：提高并发度会使第99百分位延迟恶化，而更激进的推测执行可能会增加尾延迟。这正是我们在工作流中对模拟器的要求：它能够以较低成本进行压力测试，并为满足服务等级目标的区域提供趋势正确的指导；与此同时，在线控制器对于适应静态计时模型无法预测的硬件特定的饱和点仍然不可或缺。

III. 讨论与局限

讨论。 推测宽度并非“越宽越好”：在 TinyLlama 上，较宽的草稿会推高第99百分位延迟并降低有效吞吐量，因此，尽管推测执行有望提升吞吐量，控制器仍经常缩小宽度或禁用推测执行。这补充了以往侧重吞吐量和成本的研究，并强调在服务等级目标严格时，应将推测解码与批大小和并发度联合调优。对于实践者而言，关键启示是：即使平均延迟和吞吐量看起来很有吸引力，启用宽度较大的推测解码也可能在不知不觉中违反服务等级目标。

黑盒方法在 vLLM 上实现了端到端运行，且无需更改代码，这表明它可以迁移到其他技术栈（如 TGI、Triton、MLX） [17]–[19] ，且集成工作量极小。由于 SLO-Tuner 仅通过标准 API 和命令行标志进行交互，因此可以与现有监控和自动扩缩容基础设施协同部署，而非取而代之。在生产部署中，可以在非高峰时段或金丝雀副本上运行 SLO-Tuner，然后将最佳配置推广到其余实例。

局限性与有效性威胁。 我们的工作尚处于初步阶段。我们在单个 GPU 节点上，使用合成单提示词工作负载和稳态闭环到达，对 3 个相对较小的模型（TinyLlama-1.1B、Qwen-0.6B、Qwen-4B）进行了评估。实际部署中，提示词和输出长度各不相同，请求类型多样，到达模式也随时间变化（包括昼夜周期和突发性）。虽然我们的模拟器近似模拟了部分变化，但并未建模多租户干扰、多 GPU 调度效应或集群级准入控制，而这些因素都可能显著影响尾延迟。

我们的搜索方法是一阶爬山法（固定 8 个步长），而非多臂老虎机算法或贝叶斯优化。这使调优开销可预测，但如果性能曲面呈非凸形，搜索可能会陷入局部最优。我们还使用固定的评分函数；其他服务级别目标、硬件或工作负载

可能需要重新调优这些超参数。更具自适应性的评分和其他搜索策略仍是重要的研究方向。

在实验上，我们报告的是单条 vLLM 调优轨迹，而不是在不同随机种子或起点下多次运行所形成的分布，这限制了我们刻画方差和鲁棒性的能力。我们同样没有在相同的分段预算下纳入随机搜索或其他无梯度优化器；此类比较会加强评估，但需要额外的 GPU 时间。

最后，对于非平稳或多租户工作负载，若 SLO曲面的变化速度快于控制器的反应，或者上游机制（例如准入控制、限流、缓存）与调优后的调节参数相互作用，SLO-Tuner 可能会表现不佳。严格的运营预算也可能使反复的服务器重启变得不可行，从而促使人们设计无需重启即可在线更新调节参数的方案。

结论。SLO-Tuner 是一种实用的黑盒控制器，用于结合推测解码的 p99感知的大语言模型服务。它将离散事件模拟器与实时 vLLM 集成相结合，通过公开API调优并发度、批处理和推测宽度。在 TinyLlama 上，在 1.2 秒的 SLO 下，它将 p99延迟减半（1.36 秒 $\rightarrow$ 0.7 秒），同时使有效吞吐量近乎翻倍（8 每秒请求数 $\rightarrow$ 15每秒请求数）。展望未来，加强模拟器与系统的校准、扩展到更大模型与更真实的工作负载、处理多租户场景，以及在更多服务栈和 SLO目标上进行验证，是关键的后续步骤。

IV. 负责任人工智能、信任与可持续性

负责任人工智能研究了各种考量因素，以判断大语言模型是否适合决定采用它们的组织。常见考量包括透明度、准确性、公平性和偏见 [20], [21], ，这些因素旨在提升可信性。除了通过新颖的技术方案改善人机关系 [22],，人们也提倡采用更有效的沟通方式。一些非常有意思的研究提出引入事实说明书，以推动认证特定人工智能系统是否能有效服务其用户，且不会产生意料之外的负面结果 [23]。传统上，关于事实说明书的提案并未将大语言模型的系统性能纳入考量，但我们在此提出，这一因素理应纳入其中。我们的工作探索了改进运行大语言模型的系统的方法。由此产生的一个问题是，如果这些性能没有提升，是否会引发任何重大担忧？如果事实说明书以及对人工智能系统能否被视为负责任的相关洞见未能对系统性能作出评判，那么可能会出现哪些具体负面影响，进而开始影响负责任人工智能的一些常见关注事项?

一个核心问题是，我们构建的人工智能系统是否值得信任。大语言模型尤其容易产生幻觉，或是用于训练它们的数据可能存在隐性偏见，从而损害其在影响个人生活的决策中可能发挥的作用 [24]。随着系统性能下降，我们发现这可能为组织调整参数打开方便之门，使系统能够完成任务并给出结果。然而，这样做可能会减少所使用的数据集；如果不谨慎采取这一步骤，还可能引发更多偏见和公平性方面的问题。另一种选择是搁置对透明度和解释的要求，因为推动系统更快地产生结果，可能会以牺牲这些方面为代价 [25]。我们认为这些问题非常重要，与立场论文 [26] 中提出的观点一致；这些论文阐明了当系统决定关键结果时，稳健系统设计的重要性。负责任人工智能还涉及另一个重要考量，即可持续性；我们注意到，这一主题已纳入近期一项重要的负责任人工智能工作 [27]。鉴于当今组织对人工智能系统的使用日益增加，也建议进一步关注这一问题。像我们这样的努力旨在提高大语言模型的运行效率，有助于实现这一目标。

近期，COMPASS 等平台上具有重要价值、同时促进包容性的研究成果，也进一步凸显了可持续人工智能研究的普遍价值 [28]。本文的研究成果之一，或许是让人们重新关注这一要素，并设法将其纳入负责任人工智能事实表。归根结底，系统性能至关重要：首先，它有助于在组织内继续推动大语言模型的有益采用；其次，当人们可能采取捷径来提升性能时，它有助于减轻由此产生的额外危害。我们注意到，已有研究考察了人工智能系统生命周期各阶段的可信性 [29] ，因此，将系统性能问题纳入考量具有重要意义。本文的研究进一步支持这样一种观点：让用户更加重视大语言模型的系统性能，不仅重要，而且可行。

在我们的研究中，从业者可以根据所选的大语言模型和参数设置，了解当前实现的系统吞吐量表现；表 5 展示了一个示例。一些关于可持续性的当前研究 [28] 也向用户提供了量化数据，展示结果随弱势群体占比变化而出现的退化。我们希望未来能让如今采用大语言模型的人们持续了解这些值得关注的问题。

人工智能生成内容声明

生成式人工智能工具仅用于文稿部分内容的校对和轻微语法修改，包括部分表格文本。研究内容与发现完全为作者本人的工作。

参考文献

[1] J. Dean 等，《规模下的长尾》，《*ACM*通讯》，第 56 卷，第 2 期，第 74–80 页，2013 年。[2] 谷歌，《SLO的艺术》，2025 年，谷歌 SRE工作手册。访问日期：2025-12-18。[3] Y. Leviathan 等，《通过推测解码实现 Transformer 的快速推理》，载于*ICML*，2023 年。[4] vLLM Project，《推测解码（vLLM文档）》，2025 年，文档。访问日期：2025-12-18。

[5] S. Chen 等，《Ottertune：数据库管理系统自动调优》，载于 SIGMOD，2019 年。[6] O. Alipourfard 等，《CherryPick：自适应发掘大数据分析的最佳云配置》，载于 NSDI，2017 年。[7] K. Cheng 等，《SCOOT：面向大语言模型推理引擎的服务级别目标性能调优》，载于《2025年ACM Web会议论文集》（WWW '25）。ACM，2025 年，第 829–839 页。[8] C. Delimitrou 等，《Quasar：资源高效且服务质量感知的集群管理》，载于 ASPLOS，2014 年。[9] Y. Zhang 等，《Sinan：面向云微服务的基于机器学习且服务质量感知的资源管理》，载于 ASPLOS，2021 年。[10] T. Cai 等，《Medusa：采用多个解码头的简单大语言模型推理加速框架》，载于 ICML，2024 年。[11] Z. Ankner 等，《Hydra：用于 Medusa解码的序列依赖草稿头》，载于 COLM，2024 年。[12] vLLM Project，《兼容OpenAI的服务器（vLLM文档）》，2025 年，文档。访问日期：2025-12-18。[13] ——，《服务（vLLM文档）》，2025 年，文档。访问日期：2025-12- 18。[14] P. Zhang 等，《TinyLlama：开源小型语言模型》，arXiv，第 abs/2401.02385 卷，2024 年。[15] W. Kwon 等，《利用分页注意力实现大语言模型服务的高效内存管理》，载于 SOSP，2023 年。[16] PNY Technologies，《NVIDIA L40S产品简介》，PNY Technologies，技术报告，2023 年，产品简介。访问日期：2025-12-18。[17] A. Hannun 等，《MLX：在 Apple Silicon 上实现高效灵活的机器学习》，2023 年，软件。[18] Hugging Face，《文本生成推理》，Hugging Face，技术报告，2023 年，软件。[19] NVIDIA，《Triton推理服务器》，NVIDIA，技术报告，2022 年，软-件。[20] M. Benk 等，《弥合知识差距：理解用户对可信大语言模型标准的期望》，载于 AAAI，2025 年。[21] R. Cohen 等，《可信人工智能与信任建模对多智能体系统的贡献》，载于 AAMAS（蓝天），2019 年。[22] L. Cheng 等，《具有社会责任感的人工智能算法：问题、目的与挑战》，《人工智能研究期刊》，2021 年。[23] M. Arnold 等，《Factsheets：通过供应商符合性声明提升对人工智能服务的信任》，《IBM研究与开发杂志》，第 63 卷，第 4/5 期，第 6:1–6:13 页，2019 年。[24] L. Weidinger 等，《语言模型所带来的风险分类体系》，载于 FAccT，2022 年。[25] S. Kambhampati 等，《以人为本的人工智能系统面临的挑战：AAAI主席演讲》，《人工智能杂志》，第 41 卷，第 3 期，第 3–17 页，2020 年。[26] K. R. Varshney 等，《人民的数据科学：为人民、由人民：关于新兴两极分化的一种观点》，D4GX，2015 年。[27] 蒙特利尔大学，《蒙特利尔人工智能负责任发展宣言》，2018 年，网站。访问日期：2025- 12-14。[28] Y. Atinafu 等，《PixLift：通过人工智能超分辨率加速网页浏览》，载于 COMPASS，2025 年。[29] E. Toreini 等，《对人工智能的信任与可信机器学习技术之间的关系》，载于 FAccT，2020 年。