arXiv:2603.01399v1 [cs.DC] 2 Mar 2026

Quasar：通过内存高效验证实现用于快速推理的量化自推测加速

Guang Huang [1] **Zeyi Wen** [1 2]

摘要

推测解码（SD）已成为加速大语言模型（LLM）推理的一项前沿技术，它将词元生成拆分为快速起草和并行验证。尽管近期自推测和前瞻解码方面的进展成功将起草开销降至最低，但主要性能瓶颈也因此转移到了验证阶段。由于验证需要对目标模型进行一次完整的前向传播，因此仍然严格受内存带宽限制，从根本上限制了可实现的最大加速比。本文提出**Quasar**（**量**化**自**推测**加**速**快**速推理），这是一种无需训练的新型框架，专门针对验证阶段采用低比特量化，以突破这道“内存墙”。我们的实证分析表明，尽管激进的结构化剪枝会显著降低验证准确率，基于量化的验证却能高保真地保留逻辑值分布，同时有效地将内存流量减半。在 OpenPangu 和 Qwen3 等先进模型上进行的大量实验表明，Quasar 在实现端到端吞吐量提升 的同时，其推测接受长度与全精度方法相当。Quasar 与现有起草策略正交，为加速推测执行中的验证环节提供了一条通用且高效的途径。代码见 https://github.com/Tom-HG/Quasar。

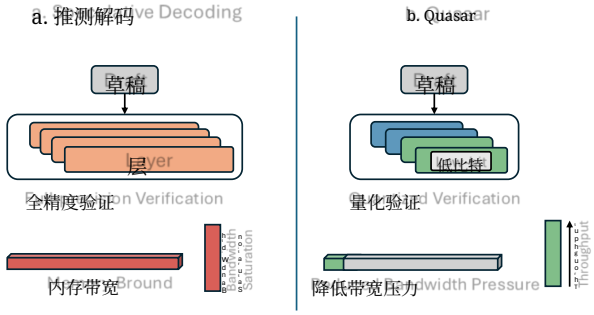


图*1*。**自推测解码与 Quasar 的验证瓶颈对比**。左图展示了标准的全精度验证过程，该过程受内存限制，并使内存带宽总线达到饱和，从而导致显著延迟。右图展示了 Quasar 使用低比特量化验证的方法，该方法有效降低了带宽压力，并实现了高吞吐量验证。

1. 引言

大语言模型（LLM）的迅速普及彻底改变了自然语言处理，使其能够实现前所未有的推理和生成能力（Brown 等人，2020；Touvron 等人，2023年）。然而，在实际应用中部署这些模型面临着一项重大挑战：推理成本高得令人望而却步。基于 Transformer 的模型采用标准的自回归解码过程，每次生成一个词元。这种序列式特性使该过程受内存带宽限制，而非受计算能力限制，导致延迟较高，且现代硬件加速器的利用率不尽理想（Shazeer，1911）。因此，在不影响生成质量的前提下加速大语言模型推理，已成为提升人工智能服务可扩展性的必要条件（Huang 等人，2025年）。

推测解码（Leviathan 等人，2023年；Chen 等人，2023年）已成为解决这一瓶颈的一种有前景的方案。该范式利用轻量级“草稿模型”预测一小段未来词元序列，从而将词元生成与验证解耦。这些词元随后由目标大语言模型并行验证。近来，自推测解码（Fu等人，2024年；Elhouderi等人，2024年；Luo等人，2024年）受到关注，其草稿由目标模型自身生成（例如通过跳层）。该方法巧妙地解决了寻找兼容草稿模型的问题。

[1]香港科技大学（广州），中国广州 [2]香港科技大学，中国香港。通讯作者：Zeyi Wen <wenzeyi@hkust- gz.edu.cn>。

预印本。审稿中。

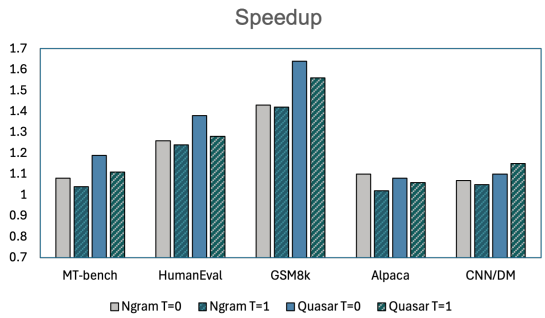


图 *2*。**各项基准测试的端到端加速比对比**。我们比较了 Quasar 在两种采样设置下相较于 N元语法基线所实现的加速效果：贪心解码（$T = 0$）和随机采样（$T = 1$）。在 MT-bench、HumanEval、GSM8K、Alpaca 和 C NN/DM 上，Quasar 始终优于基线；在 GSM8K 等推理密集型任务上，加速比最高可达 $1.6\times$ 。

尽管自推测解码有效加速了起草阶段，但它暴露出一个新的瓶颈：验证延迟（如图1所示）。虽然验证是并行进行的，但它需要对候选序列执行目标模型的完整前向传播。由于大语言模型推理主要受内存带宽限制，为验证加载全精度权重会消耗大量内存带宽。随着草稿长度增加以最大化潜在加速，该验证步骤的成本变得不可忽略，往往会削弱整体延迟收益。具体而言，在带宽受限的环境中，验证词元所需的时间可能与推测式起草所节省的时间相当。因此，一个关键的研究问题随之出现：如何在不显著降低生成质量的前提下，加速自推测解码中的并行验证阶段？

为解决这一瓶颈，我们提出了一种新颖的量化验证框架——**Quasar**（**Qua**ntized **S**elf-speculative **A**cceleration for **R**apid Inference，即量化自推测加速快速推理）。我们的核心洞见是，通过降低模型的权重精度，可以大幅减轻验证期间的内存带宽压力。我们不使用全精度模型进行验证，而是采用目标模型的量化版本作为验证器。通过压缩权重，我们显著减少了并行前向传播期间的数据传输需求，从而加快验证步骤。我们认为，现代训练后量化技术（Frantar 与 Alistarh，2023年；Lin 等人，2024年）已足够成熟，使得量化分布与全精度分布之间的差异微乎其微，因此量化验证是一种稳健且高吞吐量的替代方案。实验表明，与标准验证基线相比，我们的方法实现了更高的端到端吞吐量，同时保持了生成质量。如图所示在图2中，Quasar 在各类基准测试中均实现了稳定的加速。

主要贡献。本文提出了一个打破推测解码中验证瓶颈的框架。我们的主要贡献总结如下：

- 我们发现，由于内存带宽限制，验证步骤是现有自推测解码系统中的主要瓶颈。
- 我们提出了 Quasar，它采用低比特权重表示来加速并行草稿词元验证。
- 实验表明，与标准验证基线相比，Quasar 实现了更高的端到端吞吐量，同时保持了生成质量。

2. 相关工作

2.1. 推测解码范式

推测解码（SD）从根本上改变了自回归生成过程，打破了词元生成的严格顺序依赖。该方法首次由（Leviathan 等人，2023年）和（Chen 等人，2023年）提出，SD 采用预测器-验证器架构：一种计算高效的草稿模型提出一系列候选词元（即草稿），随后由目标模型并行验证。此后已建立理论框架，以确保推测分布通过拒绝采样或最优传输方案与目标分布严格对齐（Sun 等人，2023年；Miao 等人，2024年；Kim 等人，2023年）。尽管早期迭代依赖于独立且更小的草稿模型（例如，为 70B 目标模型使用 7B 起草模型），但这会带来显著的运维复杂性，包括需要管理双份 KV 缓存，并在分布式环境中协调不同的模型权重。

2.2. 高效草拟的进展

为减轻维护独立草稿模型所带来的开销，近期研究已转向自我推测，即将草拟机制整合到目标模型生态系统中。这些方法大致分为三类：

检索与模式匹配。受自然语言固有的局部性和重复性启发，提示查找解码（PLD）（Somasundaram 等人，2025年）和 REST（He 等人，2024年）利用 n-gram 与预生成上下文或数据存储库进行匹配，以构建草稿。这些无需训练的方法在重复率高的任务（例如代码编辑）中非常有效，但在上下文复用极少的开放式生成任务中表现不佳。

并行不动点迭代。另一类研究将解码重新表述为一个非线性方程组。前瞻解码（Fu等人，2024年）和基于雅可比迭代的方法（Santilli 等人，2023年）对目标模型的生成轨迹应用并行不动点迭代（雅可比迭代）。这些方法利用模型内部的注意力机制同时优化多个词元，无需显式的草稿权重，但与基于模型的草稿生成器相比，接受长度通常较短。

辅助验证结构。受分块并行解码（Stern 等人，2018年）的启发，Medusa（Cai 等人，2024年）通过为冻结的目标模型增添多个轻量级解码头（多层感知机），使其能够在单次前向传播中预测未来词元。EAGLE（Li 等人，2024a）、EAGLE-2（Li 等人，2024b）和 EAGLE-3（Li 等人，2025年）等最先进的后续方法进一步改进了这一方案，它们引入特征复用，使起草过程能够基于更丰富的语义历史。

验证瓶颈。虽然上述方法成功优化了起草阶段，却没有触及验证阶段。随着起草变得更加激进（生成更长的链或更深的树），验证步骤在端到端延迟中所占的比重日益增加。这主要是因为验证需要对高精度目标模型进行完整的前向传播。在内存受限场景中，这次验证前向传播会成为新的瓶颈，严格限制加速潜力。

2.3. 量化与推测

模型量化通过降低权重和激活值的精度（例如 W8A8 或 W4A4），缓解内存带宽限制。Xiao 等人，2023年提出的 SmoothQuant 和 AWQ（Lin 等人，2024年）等技术，在保持模型保真度的同时实现了高效的低精度推理。近年来，量化与推测的结合受到广泛关注。推测量化方法（Kim 等人，2023年；Zhang 等人，2024年；Svirschevski 等人，2024年）通常将目标模型的高度量化版本（例如 INT4）专门用作草稿模型，同时保留全精度（FP16/BF16）模型用于验证，以确保质量不受影响。

与我们工作的区别。我们的方法指出了现有文献中的一个关键空白：即假设验证器必须保持高精度。不同于以往通过量化来构建较弱起草模型的工作，我们提出了量化验证。我们证明，W8A8 量化模型保留了足够的保真度，可以直接充当验证器。该方法直接解决验证阶段的内存带宽瓶颈，并且与起草策略的选择正交，因此能够为自推测框架提供通用的加速提升。

3. 方法论

本节介绍我们提出的框架，该框架旨在加速推测解码的验证阶段。我们首先建立标准推测解码的数学表述，随后详细说明所提出的量化验证机制。为解决大语言模型（LLM）固有的激活值离群问题，我们采用 SmoothQuant 的增强变体，以实现稳健的 W8A8（8 位权重和激活值）推理。最后，我们进行理论分析，说明该方法如何缓解内存带宽瓶颈。

3.1. 预备知识：推测解码

令 $\mathcal{M}$ 表示目标大语言模型，该模型对条件概率分布 $p(x_t|x_{<t})$进行建模。目标是生成序列 x_1、……、x_T。推测解码利用草稿模型 $\mathcal{M}_d$（或草稿策略），以显著更低的延迟近似$\mathcal{M}$。解码过程分为两个阶段迭代进行：起草和验证。

起草。在每次迭代中，草稿模型自回归地生成一系列 γ 候选词元：

$$\tilde{x}_1, \ldots, \tilde{x}_\gamma \sim q(x|x_{<t}) \tag{1}$$

其中，q 表示草稿模型的概率分布。

验证。目标模型 $\mathcal{M}$ 对序列 $[x_{<t}, \tilde{x}_1, \ldots, \tilde{x}_\gamma]$ 执行一次并行前向传播，以计算目标概率 $p(x)$。为确保最终输出分布与 $p(x)$保持一致（无损保证），采用拒绝采样标准。第 i个候选词元 $\tilde{x}_i$ 在满足以下条件时被接受：

$$r < \min\left(1, \frac{p(\tilde{x}_i|x_{<t}, \tilde{x}_{<i})}{q(\tilde{x}_i|x_{<t}, \tilde{x}_{<i})}\right), \quad r \sim U[0,1] \tag{2}$$

拒绝 $\tilde{x}_i$ 后，将从调整后的分布中重新采样一个纠正词元：

$$x_i' \sim \text{norm}(\max(0, p(x) - q(x))) \tag{3}$$

当前解码步骤至此结束。在标准实现中，计算 $p(\cdot)$需要加载全精度权重（通常为 BF16 或 BF16）。随着草稿长度 γ 增加，该验证步骤的计算与内存比仍然较低，使整个过程完全**受内存带宽限制**。

3.2。基于增强型 Smooth 的 W8A8 验证 量化

为缓解带宽瓶颈，我们用采用 W8A8 量化的量化验证器 $\mathcal{M}_q$替代全精度目标模型 $\mathcal{M}$ 。然而，对大语言模型中的激活值进行朴素的逐张量或逐词元量化，通常会因特定激活通道中存在系统性离群值而导致性能显著下降。

为克服这一问题，我们采用了一种基于 SmoothQuant 的增强算法。该方法通过平滑离群通道，从数学上降低了激活值的量化难度，并有效地将量化误差敏感性从激活值转移到权重。

平滑变换。形式上，考虑一个权重为$\mathbf{W} \in \mathbb{R}^{d_{out}\times d_{in}}$、输入激活值为$\mathbf{X} \in \mathbb{R}^{d_{in}\times b}$的线性层。标准计算方式为 $\mathbf{Y} = \mathbf{W}\mathbf{X}$。m2 算法引入一个逐通道的平滑因子向量$\mathbf{s} \in \mathbb{R}^{d_{in}}$，用于缩放激活值并对权重进行反向缩放：

$$\mathbf{Y} = \mathbf{W}\mathbf{X} = (\mathbf{W}\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{X}) = \tilde{\mathbf{W}}\tilde{\mathbf{X}} \quad (4)$$

这里，$\tilde{\mathbf{W}}$ 和$\tilde{\mathbf{X}}$ 分别表示平滑后的权重和激活值。平滑因子$\mathbf{s}$ 通过离线校准，使量化难度趋于平衡。具体而言，对于第 j 个输入通道，s_j 由激活统计信息推导得出：

$$s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha} \quad (5)$$

其中 α 是控制迁移强度的超参数。我们增强的 m2 方法优化了这一校准，以确保稳健的离群值抑制。

量化推理。 在平滑 transfor-mation 之后，$\tilde{\mathbf{W}}$ 和$\tilde{\mathbf{X}}$ 均落入适合 8 位表示的数值范围。我们采用对称均匀量化：

$$\hat{\mathbf{W}}_{\text{INT8}} = Q(\tilde{\mathbf{W}}, \Delta_w) \quad (6)$$

$$\hat{\mathbf{X}}_{\text{INT8}} = Q(\tilde{\mathbf{X}}, \Delta_x) \quad (7)$$

其中 $Q(\cdot)$ 是量化算子，Δ 表示步长。随后使用 INT8 张量核心执行矩阵乘法：

$$\mathbf{Y} \approx \Delta_w \Delta_x (\hat{\mathbf{W}}_{\text{INT8}} \times \hat{\mathbf{X}}_{\text{INT8}}) \quad (8)$$

该形式将内存访问量减少 50%（加载 INT8 权重相对于 BF16），同时保留拒绝采样所需的精度。

3.3. 量化验证的执行流水线

将W8A8验证器集成到推测解码框架中，需要精心设计的执行流水线以最小化在线开销。一批草稿令牌 $[\tilde{x}_1, \ldots, \tilde{x}_\gamma]$ 的验证过程如下：

离线权重准备。 在推理之前，我们应用 m2 算法校准平滑因子$\mathbf{s}$。随后对目标模型的权重进行离线平滑与量化：$\hat{\mathbf{W}}_{\text{INT8}} = Q(\mathbf{W} \cdot \text{diag}(\mathbf{s})^{-1})$。这些量化权重存储在 GPU显存中，与 BF16 相比，内存占用大约减少 $2\times$。

在线激活平滑与量化。 在验证前向传播期间，输入激活$\mathbf{X}$（对应序列 $[x_{<t}, \tilde{x}_1, \ldots, \tilde{x}_\gamma]$）保持高精度（BF16/BF16）。为利用 INT8 张量核心，我们会即时执行以下变换：

$$\hat{\mathbf{X}}_{\text{INT8}} = Q(\mathbf{X} \odot \mathbf{s}, \Delta_x) \quad (9)$$

此处，逐元素乘法 $\odot\mathbf{s}$ 会应用 m2平滑，动态抑制离群值。此步骤与量化核融合，以尽量减少内存访问开销。

INT8 GEMM 与反量化。 核心计算通过整数矩阵乘法执行：$\mathbf{Y}_{\text{INT32}} = \hat{\mathbf{W}}_{\text{INT8}} \times \hat{\mathbf{X}}_{\text{INT8}}$。结果以 INT32 累加，随后反量化为高精度（BF16），供后续非线性层（例如层归一化、GeLU）和最终的 Softmax 运算使用：

$$\mathbf{Y}_{\text{BF16}} \approx \mathbf{Y}_{\text{INT32}} \cdot (\Delta_w \cdot \Delta_x) \quad (10)$$

无损拒绝采样。 最后，将恢复出的概率分布 $p(x)$ 与草稿分布 $q(x)$ 通过式 2进行比较。至关重要的是，由于反量化将对数几率恢复为高精度，拒绝采样仍然稳健，从而确保最终生成质量与目标模型一致。

3.4. 理论加速分析

草稿长度为 γ 的单个投机步的总延迟建模为 $T = T_{\text{draft}} + T_{\text{verify}}$。验证延迟主要由内存读取操作决定。设 M 表示模型规模（参数量），BW 表示内存带宽。全精度验证的延迟近似为：

$$T_{\text{verify}}^{\text{BF16}} \approx \frac{M \cdot 2\text{B}}{BW} + T_{\text{compute}}^{\text{BF16}} \quad (11)$$

通过采用 W8A8方案，内存负载实际上减半：

$$T_{\text{verify}}^{\text{INT8}} \approx \frac{M \cdot 1\text{B}}{BW} + T_{\text{compute}}^{\text{INT8}} \quad (12)$$

鉴于 $T_{\text{INT8计算}} < T_{\text{compute}}^{\text{BF16}}$ 且该过程受内存限制，我们有 $T_{\text{INT8验证}} \ll T_{\text{verify}}^{\text{BF16}}$。m2 算法确保量化分布与原始分布之间的 KL 散度可以忽略不计，从而保持较高的接受率 α 并最大限度地提高整体系统吞吐量 S:

$$S = \frac{\gamma \cdot \alpha + 1}{T_{\text{draft}} + T_{\text{verify}}^{\text{INT8}}} \tag{13}$$

4. 实验

在本节中，我们通过实证评估量化验证框架（命名为 **Quasar**）的有效性。我们旨在回答三个主要研究问题:

- **有效性（RQ1）：** 将 BF16 验证器替换为 W8A8 量化验证器，能否在内存受限场景中显著提升墙钟时间加速比？我们评估了相较于基线方法，本方法带来的主要加速效果和平均接受长度。
- **鲁棒性（RQ2）：** 本方法能否在不同采样温度下保持稳定的墙钟时间加速比？我们研究了面对不同采样分布时，效率增益的稳定性。
- **敏感性（RQ3）：** 关键超参数，尤其是草稿长度，会如何影响整体性能？我们分析了草稿长度与验证延迟之间的权衡，以确定最优配置。
- **准确率（RQ4）：** 量化验证能否保持原始模型在下游任务上的性能？我们在多个不同的基准测试中，将 Quasar 的准确率与全精度基线进行比较。

4.1. 实验设置

模型与数据集。遵循 EAGLE（Li 等人，2024a）和 Spec-Bench（Xia 等人，2024），我们使用 Qwen3-8B（Yang 等人，2025）和 OpenPangu-7B（Chen 等人，2025）模型，在五项常见任务上评估我们的方法。针对多轮对话、代码生成、数学推理、指令遵循和摘要任务，我们分别选用 MT-bench（Zheng 等人，2023）、HumanEval（Chen，2021）、GSM8K（Cobbe 等人，2021）、Alpaca（Taori 等人，2023）和 CNN/Daily Mail（Nallapati 等人，2016）数据集。

基线。我们将本方法与以下方案进行比较：**自回归（原始版本）：** 标准 vLLM 生成（Kwon 等人，2023年）。**自推测（N元语法）（Somasundaram 等人，2025年）：** 使用提示查找解码方法进行起草，但依赖标准 BF16 验证。

实现细节。我们使用 NCLL 实现了 W8A8 验证，并将其集成到 vLLM-Ascend 推理引擎中。所有实验均在单个昇腾 910B2（64GB）神经网络处理器上进行。提示查找长度会动态调整，最大值为 4，最小值为 1。

4.2. 主要结果（RQ1）

端到端加速比。表 1展示了 OpenPangu 和 Qwen3 基准测试中全面的性能对比。总体而言，Quasar 在不同模型和采样温度下始终优于自回归（原始版本）和 N元语法基线。具体来说，在 Qwen3（$T = 0$）上，Quasar 实现了显著的总体加速比 1。28×，超过了基于 BF16 的 N元语法基线（1。18×）。在 OpenPangu（$T = 0$）上也呈现出相同趋势，Quasar 的加速比达到 1。13×，而 N元语法为 1。08×。值得注意的是，在 GSM8K 等内存带宽密集型任务中，Quasar 展现出最高效率：在 Qwen3 上实现 1。64× 的加速比，在 OpenPangu 上则达到 1。26× 。这些结果有力地验证了我们的假设：W8A8 量化验证能够有效缓解内存带宽瓶颈，直接显著降低实际运行延迟。

对接受长度的影响。量化推测解码中的一个关键问题是，降低精度是否会损害接受率（α）或平均接受长度（L）。与对量化噪声的担忧相反，表 1 中的结果表明，Quasar 保持并常常超过全精度 N 元语法基线的接受长度。例如，在 Qwen3（$T = 0$）上，Quasar 取得了更优的平均接受长度 $L = 1.40$，而 N元语法为 $L = 1.33$。即使在更高的温度（$T = 1$）下，Quasar 仍保持其优势（例如，$L = 1.36$对比 $L = 1.31$ 总体）。这些证据表明，W8A8 量化保留了足够的表示精度，能够准确区分正确词元，从而确保所观察到的加速来自高效验证，而非生成质量下降。

4.3. 跨温度的稳健性（RQ2）

为评估我们方法的稳健性，我们在从 $T = 0$ 到$T = 1$ 的温度范围内评估其性能。更高的温度会向采样分布引入更大的随机性，这通常会挑战推测解码的接受率。

如表2下半部分所示，Quasar 对这些变化表现出很强的韧性。在 Qwen3 基准测试中，即使在 $T = 1$的情况下，我们的方法仍保持着具有竞争力的端到端加速比 1。23× ，平均接受长度为 $L = 1.36$。尽管加速比自然会

表*1.* 效率结果。**速度**：推理吞吐量的提升。 L：平均接受长度（质量指标）。**T=0/ T=1**：采样温度。结果表明，Quasar实现了最佳的加速与质量的权衡。

Task		MT-bench		HumanEval		GSM8k		Alpaca		CNN/DM		总体	
OpenPangu													
	方法	速度	L	速度	L	速度	L	速度	L	速度	L	速度	L
T=0	原始版本	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00
	N元语法	1.04×	1.21	1.13×	1.36	1.23×	1.44	1.01×	1.16	1.03×	1.19	1.08×	1.27
	Quasar	**1.11×**	**1.25**	**1.22×**	**1.41**	**1.26×**	**1.46**	**1.06×**	**1.17**	1.02×	1.15	**1.13×**	**1.29**
T=1	原始版本	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00
	N元语法	1.03×	1.13	1.13×	1.36	1.18×	1.41	1.03×	1.18	1.02×	1.16	1.07×	1.24
	Quasar	**1.07×**	**1.17**	**1.22×**	**1.41**	**1.21×**	**1.43**	**1.08×**	**1.19**	1.01×	1.13	**1.12×**	**1.27**
Qwen3													
	方法	速度	L	速度	L	速度	L	速度	L	速度	L	速度	L
T=0	原始版本	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00
	N元语法	1.08×	1.25	1.26×	1.45	1.43×	1.47	1.10×	1.19	1.07×	1.29	1.18×	1.33
	Quasar	**1.19×**	**1.37**	**1.38×**	**1.47**	**1.64×**	**1.66**	1.08×	1.18	**1.10×**	**1.32**	**1.28×**	**1.40**
T=1	原始版本	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00	1.00×	1.00
	N元语法	1.04×	1.24	1.24×	1.41	1.43×	1.51	1.02×	1.15	1.05×	1.28	1.15×	1.31
	Quasar	**1.11×**	**1.29**	**1.28×**	**1.48**	**1.56×**	**1.59**	**1.06×**	**1.17**	**1.15×**	**1.31**	**1.23×**	**1.36**

表 *2* $T \in$ 。不同 $[0, 1]$ **3** 采样温度（）下的鲁棒性分析。结果为 **Qwen** 在所有任务上的平均值。**速度**：相对于原始版本的端到端推理加速比。**L**：平均接受长度。随着随机性增加，Quasar 仍能保持较高的加速比，展现出更强的稳定性。

温度	**N元语法（BF16）**		**Quasar（W8A8）**	
	速度	L	速度	L
$T = 0.0$	1.18×	1.33	**1.28×**	**1.40**
$T = 0.2$	1.17×	1.31	**1.27×**	**1.39**
$T = 0.4$	1.16×	1.30	**1.26×**	**1.38**
$T = 0.6$	1.15×	1.29	**1.25×**	**1.36**
$T = 0.8$	1.14×	1.28	**1.24×**	**1.35**
$T = 1.0$	1.15×	1.31	**1.23×**	**1.36**
平均下降	*-2.5%*	*-1.5%*	***-3.9%***	***-2.8%***

略低于贪心设置（$T = 0$，1.28×），这是由于目标分布的熵有所增加，但 Quasar 始终优于 N元语法基线（在 $T = 1$时仅达到 1.15× ）。这表明，即使生成过程变得高度随机，Quasar 的量化验证依然有效且稳定。这种稳定性凸显了 Quasar 对各种部署场景的适用性，涵盖从确定性的逻辑推理到多样化的创意生成任务。

表*3*。 Quasar 与 N元语法的详细敏感性分析。 K de- 表示提示查找范围， γ 表示推测令牌的数量。对于每种方法，我们均报告端到端**加速** 和**平均接受长度**（L）。

K	**方法**	**指标**	**推测令牌（γ）**			
			3	**5**	**7**	**9**
(1, 3)	N元语法	速度	1.27×	1.28×	1.23×	1.23×
		L	1.35	1.43	1.49	1.52
	Quasar	速度	1.34×	**1.47×**	1.45×	1.42×
		L	1.44	1.46	1.49	1.59
(2, 4)	N元语法	速度	1.14×	1.21×	1.28×	1.29×
		L	1.25	1.29	1.31	1.33
	Quasar	速度	1.30×	1.32×	**1.35×**	1.34×
		L	1.18	1.41	1.44	1.45
(3, 5)	N元语法	速度	1.11×	1.17×	1.12×	1.15×
		L	1.16	1.19	1.21	1.22
	Quasar	速度	1.24×	**1.26×**	1.22×	1.21×
		L	1.28	1.32	1.33	1.34

4.4. 敏感性分析 (RQ3)

最后，我们研究结构超参数对 Quasar 性能的影响，具体包括推测令牌的数量 (γ) 和提示查找的搜索范围 (K_{min}， K_{max})。

如表 3所示，我们使用 Qwen3 作为基础模型，在 HumanEval 基准测试上评估了加速和平均接受长度（L）。结果表明，端到端

表4。不同基准测试中的得分比较。对于每个模型，我们将原始 BF16 基线与 Quasar（W8A8）进行比较。结果表明，Quasar 的性能与基线几乎完全一致。

基准测试（准确率 ↑）	OpenPangu-7B		Qwen3-8B	
	得分	Δ	得分	Δ
MMLU-pro	75.5	2.4%	63.1	2.5%
CEval	84.9	3.9%	83.4	1.3%
GPQA-Diamond	73.2	2.5%	62.0	3.1%
MATH-500	97.0	3.6%	87.4	2.1%
AIME24	79.3	3.4%	76.0	4.5%
AIME25	70.0	2.1%	67.3	3.1%
LiveCodeBench	58.2	3.1%	57.5	3.7%
MBPP+	76.0	3.4%	69.8	3.2%
平均值	**76.7**	**3.1%**	**70.1**	**2.9%**

加速对 γ 的选择高度敏感。尽管在所有配置中，L会随着更大的 γ 而单调增长，但实际运行时间加速却呈现出明显的非单调趋势。例如，在 $K = (1,\ 3)$ 下，将 γ 从 3 增加到 5 会使性能达到 1.47× 的峰值，但进一步扩展至 $\gamma = 9$ 后，性能反而下降至 1.42×。这是因为验证阶段增加的计算开销超过了接受长度带来的边际收益。

关于提示查找范围，我们观察到，较窄的窗口（例如 $K = (1,3)$）始终优于较宽的范围。当 K 从 (1,3) 变为 (3,5) 时，峰值加速从 1.47× 降至 1.26×，这表明较宽的搜索范围会引入质量较低的候选项，增加搜索延迟，却无法相应提升 L。这些发现表明，尽管 Quasar 对细微变化具有鲁棒性，但只有当 γ 与模型的预测能力精准匹配，且查找窗口保持局部化时，才能实现最高效率。

4.5. 准确性评估（RQ4）

虽然 Quasar 主要针对推理加速，但保持基础模型的预测性能至关重要。本节评估 W8A8 量化验证是否会导致模型在各种下游任务上的准确率显著下降。

与全精度基线的比较。 如表4所示，Quasar 的 W8A8 量化变体与其原始 BF16 版本在所有已评估基准测试上的性能差异均可忽略不计。具体而言，Quasar 实现了近乎无损的压缩，OpenPangu-7B 的平均差异为 3.1%，Qwen3-8B 仅为 2.9%。这表明量化噪声不会损害模型的推理能力，甚至可能起到

表5. Qwen3 的**结构化剪枝** 与**Quasar** 对比。 α 表示令牌接受率， L 表示平均接受长度。结果表明，免训练剪枝无法保持有效推测解码所需的分布对齐。

方法	保留率 / 精度	L	加速
原始版本（完整模型）	100% 层 / BF16	1.00	1.00×
结构化剪枝			
剪枝至 90%	90% 层 / BF16	1.62	0.80×
剪枝-75%	75% 层 / BF16	1.27	0.68×
剪枝-50%	50% 层 / BF16	1.03	0.62×
Ours			
Quasar	100% 层 /**W8A8**	**1.40**	**1.28**×

正则化作用。表中的分数结果来自原始技术报告（Chen 等人，2025年；Yang 等人，2025年），而差异 Δ 为我们所测得，来自语言模型评估工具包（Gao 等人，2024年）。这些结果共同表明，Quasar 的压缩验证器有效保留了原始大规模模型的核心推理能力与广泛知识库。

关于量化鲁棒性的讨论。预测准确率仅有轻微下降，这可归因于验证器在推测解码框架中的特定作用。在 Quasar 中，W8A8 模型充当最终裁判，负责评估 Ngram 起草器提出的候选结果。

从理论上讲，推测系统的最终输出与单独运行验证器的结果完全相同。我们的实证结果表明，W8A8 量化能够极好地保留相对 logit 排名。只要量化过程不会改变 top-1 预测，验证逻辑就能确保生成质量与全精度对应模型毫无差异。这一点在 CEval 等由逻辑推理驱动的基准测试中尤为明显：尽管位宽大幅缩减，准确率仍保持稳定。这些发现证实，Quasar 带来了“免费午餐”：实际运行时间大幅加速，而模型智能或任务特定精度几乎没有任何损失。

5. 讨论

虽然 Quasar 通过量化验证实现了稳健的性能，但我们进一步研究了验证器加速的其他策略，以论证我们的设计选择。具体而言，我们探索了将**结构化剪枝**（层丢弃）作为构建轻量级代理验证器的潜在方案。

免训练剪枝的失败。 一种自然的假设是，大语言模型中固有的深度冗余使得

结构化剪枝模型（例如保留前 75% 的层）可以充当高效的起草模型。这种方法从理论上避免了硬件级量化的复杂性，同时减少了起草阶段所需的浮点运算次数。然而，表5 揭示了免训练剪枝中一个关键的效率失配。尽管保守剪枝（保留 90% 和 75%）能维持相对较高的平均接受长度（L），但这些“重量级”起草模型的计算开销超过了其预测收益，导致总体速度变慢（0。80× 和 0。68×）。相反，虽然激进剪枝（保留 50%）显著降低了每个词元的延迟，却造成了灾难性的分布偏移。最终的接受长度 $L \approx 1.03$表明，起草模型的输出几乎总是被拒绝，使推测过程徒劳无功。根据推测解码理论，只有当起草模型在较低的每个词元延迟与较高的目标模型对齐度之间达到最佳平衡时，才能实现加速。剪枝后的验证器无法满足这一标准：它们要么计算开销过高，无法提供加速空间；要么准确性不足，无法实现有意义的推测收益。相比之下，Quasar 通过量化实现了所需的分布对齐，在保持完整模型深度的同时，将每个词元的成本降至最低，实现了 1。28×倍加速。

量化为何更胜一筹。 这种偏差揭示了一个根本性见解：对于验证而言，保持网络的拓扑完整性比维持较高的数值精度更为关键。 W8A8 量化会在整个参数空间中引入均匀的近似噪声，但保留了原始模型的完整深度和非线性残差流。相比之下，剪枝会移除整个功能变换，造成无法弥补的“特征缺口”，进而导致严重的 logit 偏差。因此，要在无需昂贵的重新训练或蒸馏的情况下实现即插即用的加速，量化验证显然是更稳健、更有效的范式。

6. 结论与未来工作

本文介绍了**Quasar**，这是一种全新的框架，通过量化执行加速验证，从而突破推测解码中的内存墙。传统的推测解码研究主要聚焦于降低起草成本，而我们的实证分析表明，并行验证阶段已成为一个关键的次级瓶颈，受限于高精度模型权重对内存带宽的需求。我们的核心贡献是专门将 W8A8 量化用于验证草稿令牌，在保留 logit分布保真度的同时有效地将内存流量减半。我们在 OpenPangu 和 Qwen3 等模型上进行的大量实验表明，端到端吞吐量提升了 $1.28\times$，且生成质量毫不逊色。

尽管 Quasar 提升了性能，但其效果与量化验证模型和原始高精度分布之间的对齐程度密切相关。在复杂推理任务中，即使是轻微的量化误差也可能导致接受率略微下降，从而削弱减少带宽带来的加速效果。此外，当前实现依赖经过优化的 INT8内核，这可能限制其在缺乏专用量化支持的硬件平台上的可移植性。

展望未来，仍有若干值得探索的方向：

- **超低比特验证：** 除 W8A8 外，探索更低比特的表示形式（例如 4 比特或 2 比特权重量化）或可进一步缓解带宽压力。我们计划研究精度阈值：当精度下降导致的验证准确率损失超过延迟收益时，该阈值即为所求。
- **动态精度缩放：** 一种潜在扩展是根据草稿模型的置信度动态调整验证精度。例如，对低置信度词元采用高精度验证，对高度可预测的序列采用低比特执行，可进一步优化速度与精度的权衡。
- **硬件感知优化：** Quasar 的未来迭代将针对特定硬件特性进行定制，例如现代 GPU 上的 INT8/INT4 张量核心或专用 NPU 加速器（如昇腾 910C），以在验证前向传播期间最大化计算吞吐量。
- **与基于树的推测的集成：** 我们计划评估 Quasar 与更复杂的草拟机制（如基于树的推测，例如 EAGLE、Medusa）的协同效应，以确定其在非线性验证场景中的可扩展性。

参考文献

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. 等。语言模型是少样本学习者。神经信息处理系统进展，33：1877–1901，2020。

Cai, T.、Li, Y.、Geng, Z.、Peng, H.、Lee, J. D.、Chen, D.和 Dao, T. Medusa：简单的大语言模型推理加速框架，具有多个解码头。*arXiv*预印本 *arXiv:2401.10774*，2024年。

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L. 和 Jumper, J. 加速大语言模型

通过投机采样进行解码。*arXiv* 预印本 *arXiv:2302.01318*，2023年。Chen, H., Wang, Y., Han, K., Li, D., Li, L., Bi, Z., Li, J., Wang, H., Mi, F., Zhu, M. 等人。盘古嵌入式：一种具备元认知能力的高效双系统大语言模型推理器。*arXiv* 预印本 *arXiv:2505.22375*，2025年。Chen, M. 评估基于代码训练的大语言模型。*arXiv* 预印本 *arXiv:2107.03374*，2021年。

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R. 等。训练验证器以解决数学应用题。*arXiv* 预印本 *arXiv:2110.14168*，2021年。

Elhoueder, M., Ivkin, M., Xu, T., Shmakov, A., Pehlevan, C. 和 Canis, A. LayerSkip：实现推理提前退出和自推测解码。*arXiv* 预印本 *arXiv:2404.16710*，2024年。

Frantar, E. 和 Alistarh, D. SparseGPT：大规模语言模型可以通过一次性剪枝实现高精度。发表于国际机器学习会议，第 10323–10337 页，2023年。

Fu, Y., Bailis, P., Stoica, I. 和 Zhang, H. 使用前瞻解码打破大语言模型推理的顺序依赖。*arXiv* 预印本 *arXiv:2402.02057*，2024年。

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K. 和 Zou, A. 语言模型评估工具包，2024年7月。网址https://zenodo.org/records/12608602。

He, Z., Zhong, Z., Cai, T., Lee, J., and He, D. Rest：基于检索的推测解码。载于《*2024*年计算语言学协会北美分会会议论文集：人类语言技术》（第*1*卷：长篇论文），第 1582–1595 页，2024年。

Huang, G., Xiao, Y., Jiang, L., Yin, M., and Wang, P. 超越提示工程：一种面向大语言模型的强化令牌级输入优化。载于《*AAAI*人工智能会议论文集》，第 39 卷，第 24113–24121 页，2025年。

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. 使用大小解码器进行推测解码。《神经信息处理系统进展》，36:39236–39256，2023年。

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., 和 Stoica, I. 高效面向大语言模型服务的内存管理：PagedAttention。在操作系统原理研讨会第*29*届会议论文集，第611–626页，2023年。Langley, P. 撰写机器学习论文。载于 Langley, P.（编），第*17*届国际机器学习会议论文集（*ICML 2000*），第1207–1216页，美国加利福尼亚州斯坦福，2000年。Morgan Kaufmann。Leviathan, Y., Kalman, M., 和 Matias, Y. 通过推测解码实现 Transformer 的快速推理。载于国际机器学习会议，第19274–19286页。PMLR，2023年。

Li, Y., Wei, F., Zhang, C., 和 Zhang, H. Eagle：投机采样要求重新思考特征不确定性。*arXiv* 预印本 *arXiv:2401.15077*，2024a。

Li, Y., Wei, F., Zhang, C., 和 Zhang, H. Eagle-2：利用动态草稿树加速语言模型推理。*arXiv* 预印本 *arXiv:2406.16858*，2024b。

Li, Y., Wei, F., Zhang, C., 和 Zhang, H. Eagle-3：通过训练时测试扩展大语言模型的推理加速。*arXiv* 预印本 *arXiv:2503.01840*，2025年。

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., 和 Han, S. AWQ：用于端侧大语言模型压缩与加速的激活感知权重量化。机器学习与系统会议论文集，6：87–100，2024年。

Luo, X., Wang, Y., Zhu, Q., Zhang, Z., Zhang, X., Yang, Q. 和 Xu, D. 化废为宝：通过令牌回收加速大语言模型推理。*arXiv* 预印本 *arXiv:2408.08696*，2024年。

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X. 等。Specinfer：通过基于树的推测推理和验证加速大语言模型服务。在第*29*届 *ACM* 编程语言与操作系统架构支持国际会议论文集，第*3*卷，第932–949页，2024年。

Nallapati, R., Zhou, B., Dos Santos, C., Gulc¸ehre, C¸. 和 Xiang, B. 使用序列到序列 RNN 及其后续方法进行抽象式文本摘要。在第*20*届 *SIGNLL*计算自然语言学习会议论文集，第280–290页，2016年。

Santilli, A., Severino, S., Postolache, E., Maiorca, V., Mancusi, M., Marin, R., 和 Rodol`a, E. 通过并行解码加速用于机器翻译的 Transformer 推理。*arXiv* 预印本 *arXiv:2305.10427*，2023年。

Shazeer, N. 快速 Transformer 解码：只需一个写入头，2019年。*URL https://arxiv.org/abs*，1911年。

Somasundaram, S., Phukan, A., and Saxena, A. 提示查找解码+：利用语言模型工件加速大语言模型推理。发表于计算语言学协会年会研究成果：*NAACL 2025*，第 6075–6089 页，2025年。

Stern, M., Shazeer, N., and Uszkoreit, J. 面向深度自回归模型的分块并行解码。神经信息处理系统进展，第 31 卷，2018年。

Sun, Z., Suresh, A. T., Ro, J. H., Beirami, A., Jain, H., and Yu, F. Spectr：通过最优传输实现快速推测解码。神经信息处理系统进展，36:30222–30242，2023年。Svirschevski, R., May, A., Chen, Z., Chen, B., Jia, Z., and Ryabinin, M. Specexec：面向消费级设备上交互式大语言模型推理的大规模并行推测解码。神经信息处理系统进展，37:16342–16368，2024年。Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca：一种强大且可复现的指令遵循模型。斯坦福基础模型研究中心。*https://crfm. stanford. edu/2023/03/13/alpaca. html*，3(6):7，2023年。Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozi` ere, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama：开放且高效的基础语言模型。*arXiv* 预印本 *arXiv:2302.13971*，2023年。Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. 释放大语言模型推理的效率：推测解码综合综述。*arXiv* 预印本 *arXiv:2401.07851*，2024年。Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant：面向大语言模型的精准高效训练后量化。发表于机器学习国际会议，第 38087–38099 页。PMLR，2023年。Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*，2025年。Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft& verify：通过自推测解码实现无损大语言模型加速。发表于计算语言学协会第 *62* 届年会论文集（第 *1* 卷：长篇论文），第 11263–11282 页，2024年。

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., 等. 评判使用 MT-bench 和 Chatbot Arena 进行基于大语言模型的评审。神经信息处理系统进展，36: 46595–46623，2023年。