arXiv:2611.28502v1 [cs.DB] 21 Nov 2026

Shoonya：面向图数据库的零套接字进程内小语言模型：采用模式纪元键值缓存、实时目录对数几率屏蔽和验证器优势蒸馏

Madhulatha Mandarapu* Sandeep Kunkunuru†

VaidhyaMegha Private Limited,
印度https://samyama.ai/

2026年11月

摘要

将属性图数据库与外部语言模型服务器连接，会带来三重套接字开销：序列化开销远超微秒级图读取的耗时、冗长模式需要反复预填充，以及盲目解码与实时目录和多版本并发控制（MVCC）引擎相脱节。我们推出 Shoonya：从零开始打造、商业用途合规（非商业或相同方式共享数据占比为 0.0%）的三模态（Python、Rust、OpenCypher）小语言模型（参数量从 16.8M 到 1.07B），并与 shoonya-infer 协同设计。shoonya-infer 是嵌入 samyama-graph-enterprise 地址空间的零依赖纯 Rust 运行时。直接使用 GraphStore引用，解锁了三项数据库原生功能：（1）基于写时复制 PagedKvArena 的 SchemaPrefixCache，在轮次失效后可将预热 CPU 首 token 时间加速 $4.91\times$–$13.68\times$，且 DDL 失效耗时为 O；（2） $36\,\mu$s 列式目录和源跨度对数几率屏蔽，确保本体标识符有效且跨度原文不变；（3）亚 $65\,\mu$s 的只读 MVCC 门控，在 389.8 MiB INT8 常驻集大小（59.8–67.9 令牌/秒）下提供 Tier-0 查询服务，幻觉泄漏率为 0.0%。秩分解绑定嵌入（词汇表权重减少 -74.5%）、原子缩进 token（缩进错误减少 $0/421$ ）、4:1 分组查询注意力与 2:1 跨层 KV 共享（INT8 KV 内存减少 -93.75%）、预热稳定衰减训练、KnOTS 奇异值分解合并、Step-GRPO，以及 Rust 雅可比奇异值分解治理，共同确保 100% 的层保持重尾分布（$\alpha \in [1.88, 2.20]$）。最后，我们总结了 v0.1–v1.0 阶段的九条经验，通过教师模型扩展和零显存验证器优势门控知识蒸馏（AdvKD）解决反向 token 密度蒸馏陷阱，恢复 $+13.61$ 个 MBPP 得分点（33.07%），并在增强、基于依据的抽取和工具规划任务上达到 100% 的有效率。

1 引言

现代属性图数据库日益将语言驱动工作流与声明式查询执行相结合：把自然语言问题转换为 OpenCypher [Francis 等人， 2018],合成多步智能体工具计划，并从非结构化数据中提取类型化实体关系三元组

临床或企业文本，并丰富领域本体 [Mandarapu andKunkunuru，2026b]。针对临床问答的受控实验表明，知识图谱 grounding 只有在关键机构事实不在预训练语料库

*madhulatha@samyama.ai
†sandeep@samyama.ai

[Mandarapu and Kunkunuru，2026a]。然而，在私有子图无法离开本地部署边界的受监管部署中，即使通过独立的本地主机 HTTP 或 gRPC 服务器自行托管开放权重模型，也会产生根本性的**套接字开销**：

1. **进程间通信与序列化不对称性**。内存列式属性图可在 15–65 μ 秒内解析单跳邻接关系并确定 MVCC 可见性，而回环 JSON 序列化、操作系统套接字调度和 Python 运行时调度每次调用都会耗费数毫秒，这使得无法对每个词元或候选项进行紧密的数据库验证。

2. **冗余模式预填充**。企业图模式包含数百至数千个词元。无状态套接字调用每轮都会重新预填充模式提示，导致 CPU 首令牌时间（TTFT）居高不下；而外部 KV 前缀缓存 [Zheng 等人， 2024] 在数据库 DDL 语句变更标签或属性时，无法保证事务一致性。

3. **目录盲子词解码**。进程外解码器在无法实时访问图目录的情况下采样词元，因而会臆造不存在的边标签（例如， `:TREATS` 而非 `:TARGETS`）、颠倒边箭头，或编造没有依据的实体字符串。外部语法编译器[Willard 和 Louf， 2023] 在模式演化时需要耗费数十至数百毫秒来重建静态自动机。

正如 DuckDB 通过将向量化 OLAP 嵌入宿主进程，消除了分析型 SQL 的客户端-服务器线路开销 [Raasveldt 和 Mühleisen， 2019],我们将这一模式反转，应用于图原生 AI：将零依赖的 *Rust* 语言模型运行时直接嵌入图数据库自身的 *Rust* 地址空间，并共享对其实时目录和 *MVCC* 存储引擎的零拷贝引用。

要实现这一架构，既不依赖数万亿 token 的不透明网络爬取数据，也不受著作权传染或非商业训练数据污染，必须从零开始构建 **Shoonya** ，并历经十二次迭代发布（`v0.1` → `v0.9.5-AdvKD` 和 `Stage-3 1.3B-v1.0`）。Shoonya 将 PyTorch/CUDA 预训练与强化学习技术栈，与一个**由八个 crate 组成、无外部依赖的 Rust 平面**（`shoonya-core`、 `shoonya-tokenizer`、 `shoonya-telemetry`、 `shoonya-eval`、 `shoonya-infer`、`shoonya-search`、 `shoonya-ffi`和 `shoonya-cli`）相结合，最终实现了在 `samyama-graph-enterprise` （`v1.15.0`） [Mandarapu and Kunkunuru、2026b] 中无需套接字、直接在进程内执行，且未修改核心数据库代码库中的任何代码行。

贡献。 本文详细介绍了 Shoonya 的完整架构、数据治理、训练目标、长期学习成果以及企业评估：

1. 零套接字进程内图 SLM 引擎（第 2 节）。将纯 Rust `shoonya-infer`（0 个外部 Crate， `FP32` 奇偶一致性 $\leq 1.29 \times 10^{-5}$）直接连接到不可变的 `&GraphStore` 引用，解锁了：(i) 固定于 DDL 轮次的 `SchemaPrefixCache` ，基于写时复制的 `PagedKvArena` （4.91×–13.68× 预热 CPU 首令牌生成时间加速比，且 O（1）次 DDL 失效）；(ii) 29.8–56.4 μ 秒的实时列式目录、已转义的 `ToolPlan`以及源跨度 FSM 对数几率掩码；(iii) 19.6–66.2 μ 秒的只读多版本并发控制执行和覆盖率门控，在 389.8 MiB 常驻集大小（`350M INT8`，59.8–67.9 令牌/秒）和 1,162.9 MiB 常驻集大小（`1.3B INT8`，22.0–25.0 令牌/秒）的条件下，实现了 0.0% 的假阳性幻觉泄漏。

2. **参数密集型三模态主干网络与硬件对齐消融实验（第3节）**。协同设计一个 50,264词元 `CodeTokenizer` ，采用互不重叠的填充中间（FIM）哨兵标记和原子缩进 token（37.9% 缩进压缩； 0/421 `IndentationError` 在 MBPP + HumanEval 上），秩r 分解绑定嵌入（−74.5% 至 −75.0% 词汇表权重），4:1分组查询注意力，采用2:1

跨层 KV 共享（-87.5% BF16/ -93.75% INT8KV 内存）、 8K YaRN以及 Warmup-Stable-Decay（WSD）调度，可实现 30,068 令牌/秒的 SRAM 融合训练和 120,051 令牌/秒的 4K预填充——避免了我们在稀疏掩码和循环式替代方案上测得的 $28\times$–$35\times$ HBM 瓶颈。

3. 商业合规数据管线、闭环谱控制与三模态 RLVR（第 4 节）。在 886.47M 个经 SPDX 验证的商业合规令牌中（0.0% NC-SA，在 593 个留出任务中检测到 0 个 SHA- 256 泄漏），我们结合在线随机化/循环 Jacobi 奇异值分解谱正则化（$\alpha \in [1.85, 3.5]$）、 KnOTS 奇异值分解适配器合并与正交零空间清理（OrthoKnOTS）、稠密符号 Step-GRPO以及 FLARE 熵门控主动 KV 落地——在 4.0T–5.5T 个令牌的外部基线陷入 $\alpha < 1.85$ 相关性陷阱时，保持 $20/20$（350M）和 $24/24$ （1.3B）层不发生坍缩。

4. 纵向陷阱（v0.1–v0.9.4）、反向令牌密度陷阱与 AdvKD （第 5 节）。我们汇总了在先前版本中发现的九种具体数学与训练失效模式，最终归结为反向 token 密度蒸馏陷阱：当更宽的教师模型的每参数令牌数远低于更小的学生模型时（$\rho_T = 0.059 \ll \rho_S = 2.55$），无门控正向 KL 能够传递结构化 JSON/Cypher 格式（$0\% \to 66.7\%$），但会使学生模型的MBPP 下降 -12.06 个百分点，并使 Rustc-Exec-64 下降 -15.62 个百分点。增加教师模型合规令牌 $6.53\times$（411.04M），并通过零显存 LoRA 切换式验证器优势门控知识蒸馏（AdvKD）对教师模型 KL 进行门控（55.76% 的路由位置），可恢复 $+13.61$ MBPP 个 TTC 点（33.07%）和 $3.33\times$Rust 执行，同时相较于 CodeGen-350M-mono和 SmolLM2-360M-Instruct （第 6 节）实现 $420\times$–$5,019\times$ 倍更高的样本效率。

本方案不是什么，以及产物范围。 Shoonya 是一款嵌入式 Tier-0 结构引擎和零泄漏混合式守门器，并非 可在开放域聊天或复杂多跳 OpenCypher 查询（OPTIONAL MATCH 反连接和 WITH ... HAVING聚合）中取代 70B+ 前沿模型的独立替代品。在这些场景下，我们的亚 $65\,\mu$ 秒 MVCC 门控会将 100% 未验证的查询升级至前沿 API（泄漏率为 0.0%）。开放模型权重（Apache-2.0）、分片溯源清单、基准测试评分卡以及完整深度的 Rust 奇异值分解谱证书均已公开，见 https://huggingface.co/VaidhyaMegha/Shoonya；训练技术栈和 Rust 数据库集成仍由 VaidhyaMegha Private Limited / Samyama.ai 内部维护。

2 零套接字进程内 Graph-LLM 架构 e

图 1 展示了 samyama-graph-enterprise(v1.15.0) 与 shoonya-infer之间的单地址空间集成。进程内 Rust 桥接层不会修改数据库引擎，也无需跨越 IPC套接字，而是与 samyama-graph-enterprise 并行 shoonya-infer链接在同一个操作系统进程内，并将不可变的 &GraphStore 引用直接传递给 KV缓存查找、令牌级logit掩码和 MVCC候选验证。

2.1 零依赖纯 Rust shoonya-infer运行时

为在企业级Rust DBMS二进制文件中实现无缝嵌入，避免 C++/CUDA ABI 冲突和供应链臃肿，shoonya-infer 仅依赖 Rust标准库（std）以及同级零依赖工作区crate（shoonya-core和 shoonya-tokenizer）。它实现了原生 .safetensors 二进制解析（F32、 BF16、 F16）、逐行对称 INT8 权重量化（$w_{o,i} \approx s_o q_{o,i}$）、 RMSNorm、半拆分 RoPE/YaRN、 SwiGLU、4:1 分组查询注意力与 2:1 跨层 KV 共享、秩-r 因子化绑定嵌入，以及迭代级 ContinuousBatchScheduler。

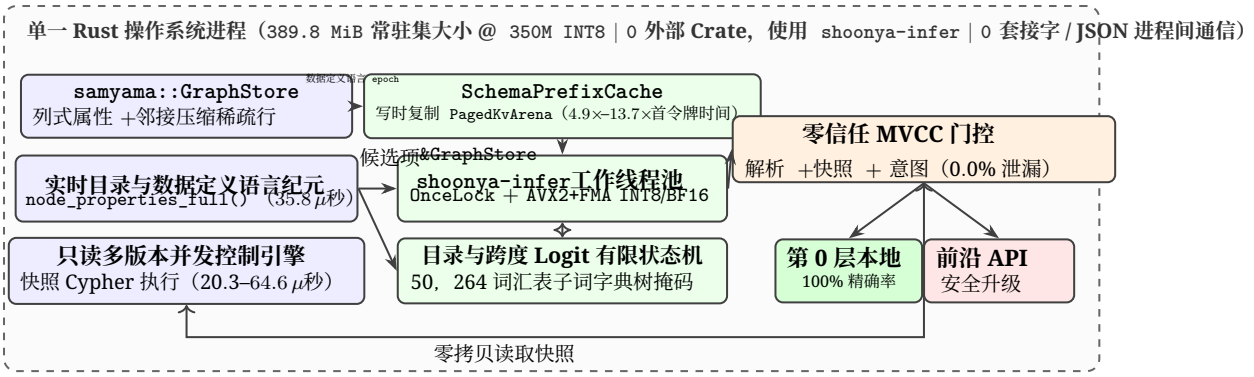

图 1：在同一个 Rust 地址空间中，将 samyama-graph-enterprise(v1.15.0) 与shoonya-infer 耦合的零套接字进程内架构。直接 &GraphStore 引用支持 DDL 世代 KV 前缀缓存、 36 μ的列式目录，以及针对 50, 264 个词汇表令牌的源跨度 Logit 遮罩，并可在不到65 μ秒内完成只读多版本并发控制验证。

在全部三个稠密模型层级中——Small （16.8M， $L = 4$、 $d = 256$）、 Stage-1 （142.5M， L =12、 $d = 768$）和 Stage-2（317.3M， $L = 20$、 $d = 1024$、 $r = 256$）——shoonya-infer 复现了 PyTorch FP32 的 logits，最大绝对误差分别为 5.72×10^{-6}、 5.72×10^{-6} 和 1.29×10^{-5}，且贪心 top-1 词元一致性达到 100%。两项 CPU 执行优化提升了交互性能：

- **运行时 AVX2+FMA SIMD 内核。** 投影循环在运行时探测 avx2 和 fma CPU 标志（ is_x86_feature_detected!），使用融合乘加将量化的 INT8 行扩展到 256 位 SIMD 通道，无需指定针对主机的编译器标志。
- **持久化 OnceLock 工作线程池。** 在里程碑 M1 (v0.9.2) 中，每次调用层时都通过std::thread::scope 生成的操作系统线程并行执行投影，使 350M INT8 CPU 解码速度限制在 28.7 令牌/秒（热启动首令牌时间为 920.8 毫秒）。在 M2 (v0.9.3) 中，以持久化的 OnceLock 先自旋后条件变量工作线程池取代作用域线程，消除了内核线程创建开销，使独立 INT8 吞吐量提升至 73.8 令牌/秒（热启动首令牌时间为 154.4 毫秒），进程内实时 samyama-graph-enterprise 解码速度达到 67.9 **令牌/秒（400.4 毫秒热启动首令牌时间）**，同时常驻集大小为 389.5–389.8 **MiB 常驻集大小** （ 1.3B INT8为 22.0–25.0 令牌/秒时，常驻集大小为 1,162.9 MiB）。

2.2 数据定义语言轮次 SchemaPrefixCache与写时复制 PagedKvArena

受 PagedAttention [Kwon 等人、 2023] 和 RadixAttention [Zheng 等人的启发， 2024], shoonya-infer 将 KV 状态存储在采用引用计数的 16 词元页面中，位于 PagedKvArena 支持 FP32 和按头划分的 INT8 页面，并通过 $O(1)$ 写时复制页面拆分（fork_sequence）支持多候选测试时计算。与仅按词元 ID 为缓存前缀建立索引的外部服务器不同， SchemaPrefixCache 会将每个缓存的 KV 句柄绑定到数据库的事务模式世代：

$$k_{\text{cache}} = (\texttt{graph_id},\ \texttt{schema_epoch},\ \text{FNV1a}(\mathbf{x}_{\text{schema}})), \tag{1}$$

其中 $\mathbf{x}_{\text{schema}}$ 是一个紧凑的列式标头，由 &GraphStore合成（例如，(:Trial {enrollment, nct_id, phase, status, title})-[:TARGETS]->(:Disease {category, disease_id, name})，将散文 db.schema.forLLM()输出压缩 3.4×）。

在稳定状态的轮次 e，查询完全绕过模式预填充，使 CPU首令牌生成时间减少 13.68× **达BF16 (7**，543.3 → 551.5 毫秒)，在 192 个令牌的模式上减少 11.66× **达 INT8 (4**， 910.1 → 421.0 毫秒)，并且

4.91×–6.04× 对于紧凑型临床企业级头部（6， 917.9 → 1， 193。1 毫秒，位于 1.3B INT8；图 2a）且0。0 logit/令牌漂移。在数据定义语言事务改变本体（添加 :Biomarker 和:HAS_BIOMARKER 在我们的实时演化检查中）时，将 schema_epoch 推进至 $e+1$ 会在第 1 轮确定性地使过期的 KV 页失效，并在第 $e+1$ 轮预热 2+。

2.3 Sub-40 μ 微秒实时列式目录与源跨度 Logit 遮罩

与 GraphStore 共享地址空间后，推理桥接器可直接检查 node_properties_full()，仅需 29.8–37.5 μs（若同时为实时 PropertyValue 列字典编制索引，则需 56.4 μs），即可在四种企业级表面上，为包含 50,264 个词元的词汇表构建子词字典树掩码：

- **表面 1（NLQPipeline）与表面 2（AgentPlanner ToolPlan）**。词法状态跟踪器会限制词元：在（var: 之后只能使用实时节点标签 $\mathcal{L}(G)$；在 [: 之后只能使用符合端点拓扑的实时关系类型 $\mathcal{R}(G)$；在 var. 之后只能使用 var绑定标签 ℓ的属性键 $\mathcal{P}_G(\ell)$。通过扩展该自动机以跟踪 JSON 转义的内层 Cypher（M5.A），无需重新训练，即可将 350M-v0.9.5 上的实时多版本并发控制执行率从 0.0% 提升至 33.3%。

- **表面 3（ExtractionAgent）与表面 4（Enrichment）**。对于从文档 d中进行实体关系 JSON 抽取，动态构建子串索引仅需 $< 20\,\mu$s，即可将 "name" 字符串值限制为 d中的原样字节跨度，同时将 "label"和 "type"字段限制为 $\mathcal{L}(G) \cup \mathcal{R}(G)$，从结构上消除幻觉实体和模式外谓词。

2.4 亚65 μs 零信任 MVCC 执行与覆盖率门控

每个候选查询在本地交付前都要通过两阶段零信任 MVCC 门控：(1) 针对进程内 GraphStore MVCC 快照执行只读操作（平均延迟为 19.6–66.2 μs），并拒绝任何解析、规划或类型错误；(2) 执行无需预言机的结构化意图检查，以验证投影非空、最高级提示中的 ORDER BY/LIMIT 对齐、CONTAINS/ OPTIONAL MATCH 关键词对齐，以及不存在无依据的训练集字面量。通过验证的查询会在 Tier-0 本地返回；被拒绝的查询则升级至前沿 API。在所有基准测试中（表 3），将实时目录有限状态机与 MVCC 门控（Arm 4）结合使用，可将假阳性泄漏维持在 0.0%（已服务查询的精确率为 100.0%）；而停用目录有限状态机（Arm3，针对 v0.9.3 BF16）则会导致 8.33% 的泄漏（1/12）。

3 三模态骨干网络与硬件-物理设计

3.1 硬件-物理语言拆分、分词与节省KV的几何结构

Shoonya 严格遵循硬件-物理边界： $> 98.5\%$ 的预训练浮点运算量仍由 PyTorch/CUDA 承担，使用 BF16 Tensor Core、原生 16:4GQA、FlashAttention-2 和缩放点积注意力 [Dao，2024]（通过消除手动 4× repeat_interleave 复制和 D2H .item() 同步，将显存占用控制在 11.0 GiB 以下 fullgraph=True torch.compile）；而受全局解释器锁限制的分词、最佳适应递减 AST 序列打包、沙箱执行和谱奇异值分解，则在我们的零依赖 Rust 工作区中运行。四项架构选择最大限度地提升了每兆字节的推理能力：

- 50，264**词元 CodeTokenizer 采用原子缩进**。将50个257词元 GPT-2 BPE词表扩展，加入四个互不重叠的中间填充（FIM） [Bavarian等人， 2022]以及填充哨兵（50257..50260），另加入三个原子缩进 token（<|indent_4|>， <|indent_8|>，

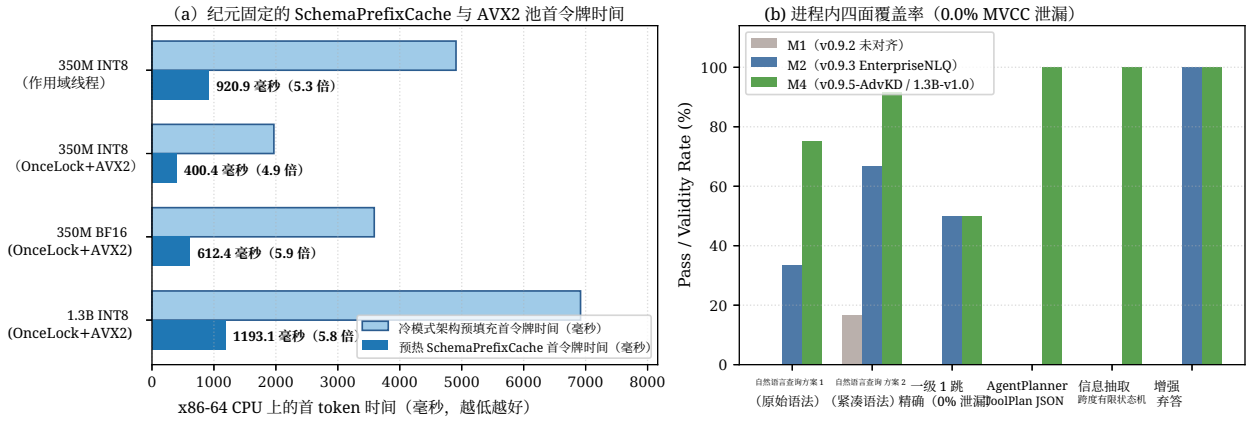

图 2：实时零套接字 `samyama-graph-enterprise v1.15.0` CPU 评估。**(a)** 作用域线程（`v0.9.2`）与持久化`OnceLock` + `AVX2+FMA` 工作线程池（`v0.9.3/1.3B-v1.0`）中的冷模式架构预填充与热模式 `SchemaPrefixCache` 首令牌时间（TTFT）对比。**(b)** 在 0.0% 多版本并发控制（MVCC）幻觉泄漏条件下，四面准确率从未对齐状态 `M1` （`v0.9.2`）到 `M4` （`v0.9.5-AdvKD` / `1.3B-v1.0`）的变化。

`<|indent_12|>`），可将缩进的 Python/Rust 代码块压缩37。9%，将空白符意外度降低至 3.1×10^{-4} 纳特 [Mandarapu and Kunkunuru, 2026c], 并将分布外`IndentationErrors` 降至 $0/257$在 `MBPP-Sanitized` 和 $0/164$ 在 `HumanEval`。

- 秩-r 因子化绑定嵌入。在 $V = 50$=264 时，未分解查找表占据 16.8M 模型的 77.3%，并在 $d = 1024$ 时占据 51.5M 个参数。对 $E = E\text{tok}P^{\top}\text{proj}$ [进行因子化（Lan 等, 2020] 在 $r = 256$ 上，适用于 `Stage-2` （$L = 20$、$d = 1024$、$d_{ff} = 4096$，312.1M 个参数）），可将词汇表权重减少 74.5%（51.5M $\rightarrow$ 13.1M，并将 95.8% 的权重分配给主干层），且在 `Stage-3` 上（$L = 24$、$d = 2048$、$d_{ff} = 5504$，1, 065.1M 个参数）可节省 75.0%。

- 4:1 分组查询注意力 + 2:1 跨层 KV 共享 + `8K YaRN`。将 16:4 分组查询注意力 [Vaswani 等人, 2017 年; Ainslie 等人, 2023] 与 2:1 跨层 KV 共享 [Sun 等人, 2024] 结合，可将 `350M`的 8,192 token KV 缓存从 640.0 MiB（`MHA`）降至 160.0 MiB（4:1 分组查询注意力），再降至 80.0 MiB（`BF16`，-87.5%）和 40.0 MiB（`INT8`，-93.75%）。固定波长渐进式 `YaRN` [Su 等人, 2024 年; Peng 等人, 2024] 幅度缩放（mscale $= 0.1\ln s + 1$），可使 8K 代码困惑度保持在 1.966（早期架构前五召回率为 66.7%）；而简单的线性位置插值则会使困惑度恶化至 48.37。

3.2 六臂等步架构与 `WSD` 调度器消融实验

表1和图3b展示了在 `RTX 4050` 上进行的受控 320 步（1.31M 词元）风洞测试，涵盖六种架构/调度器配置：(1) 2024] Hu 等人的 `WSD` [，在相同吞吐量（29, 251 令牌/秒）下，-31.07 困惑度优于 `Cosine` （77.68 对 108.75）；在持续令牌扩展（320 $\rightarrow$ 440 步）中，`WSD`的衰减前分支达到 43.33 困惑度，而 `Cosine` 热重启为 70.35 困惑度（并遭受 5.057 纳特的学习率重新预热冲击）；(2) SRAM 融合的 `Dense-GQA-XKV2-WSD` 实现了 30, 068 令牌/秒的训练吞吐量和 120, 051 令牌/秒的 4K预填充吞吐量，快于 HBM物化的 96.4% 稀疏 `ProgSparseAttn`（4K 时为 3, 399 令牌/秒——比 35.3$\times$ 慢）以及纯 PyTorch `Jamba-MambaHybrid`（1, 060 令牌/秒——尽管等步困惑度为 40.40，训练速度仍比 28.4$\times$ 慢）。

架构与学习率臂 （320步，1.31M 清洁词元）	总计 / 激活 参数 (M)	训练速度 (令牌/秒)	验证交叉熵 (纳特)	验证困惑度 (↓)	4K预填充 (令牌/秒)	4K KV内存 (BF16 MiB)
Dense-GQA-Cosine	36.7/36.7	29，120	4.689	108.75	115，939	20.0
Dense-GQA-WSD	36.7/36.7	29,251	4.353	77.68	118，350	20.0
Dense-GQA-XKV2-WSD (已选)	**36.0/36.0**	**30，068**	**4.400**	**81.43**	**120，051**	**10.0**
SparseMoE-8x2-WSD	107.5/36.7	18,891	3.952	52.03	80，755	20.0
ProgSparseAttn-WSD	36.7/36.7	16，594	4.207	67.16	3，399	20.0
Jamba-MambaHybrid-WSD	45.0/45.0	1，060	3.699	40.40	5,225	4.6

表1：在 RTX 4050 （BF16）上进行的受控六臂 Iso-Step 风洞测试。 Dense-GQA-XKV2-WSD在保留 FlashAttention-2 SRAM 融合的同时，在 4:1 分组查询注意力的基础上将 KV 内存减半（BF16 KV 总量减少 87.5%），并实现无尖峰的 WSD持续预训练。

4 数据治理、谱控制与后训练栈

4.1 商业清洁语料库演化（v0.4 → v1.0_clean）与去污染

受 Gunasekar 等人提出的教科书级质量过滤启发， [2023], Shoonya 在保持模型几何结构不变的同时，将预训练数据扩展为四代经审计的数据集：(i) v0.4–v0.5 （4, 300 个 AST 验证的 Python/Rust 函数，0.43M 词元）；(ii) v0.6 （42.66M 词元）；(iii) v0.7–v0.8.5（226.60M 个 256 上下文 + 33.55M、1024 上下文 = 260.15M 三模态词元：63.45% Python、29.03% Rust、7.52% Samyama OpenCypher，覆盖 19 个领域知识图谱）；以及 (iv) v0.9_clean （730.61M 词元） →v1.0_clean（89 个分片共 886.47M uint16词元）。

为确保在 v0.9+ 中不受限制地进行企业部署，自动化的故障关闭式 SPDX 门禁（license_gate.py）清除了 v0.7–v0.8.5 （code_exercises、 Evol-Instruct）中存在的 ∼17.5% CC-BY-NC-SA-4.0 和 OpenRAIL 来源，保留了 0.0% 的 NC-SA 数据（ MIT 、 Apache-2.0 、 BSD 、 PSF-2.0 、 CC0 、 CC-BY-4.0），其中包括经执行验证的 OpenCodeInstruct、 Strandset-Rust-v1，以及在 SamyamaClient.embedded() 中实时执行的 46,005 条 OpenCypher 查询。纯 Rust shoonya-core 通过最佳适应递减打包 100% 抽象语法树完整的文档，速度达 21.26M 令牌/秒，同时执行规范化文档字符串和规范 AST SHA-256 去污染（在所有 593 项**留出基准任务**中，**泄漏**为 0，剔除了 845 份冲突文档）。

4.2 关闭 d-Loop 频谱治理与无干扰后训练

- 闭环 HTSR 与随机 SVD 遥测。 shoonya-telemetry 同时实现精确循环 Jacobi 奇异值分解和 Halko–Martinsson–Tropp 随机子空间奇异值分解 [Halko 等， 2011] （在 $d = 2048$ 时为 4.75 毫秒，在 $d = 1024$ 时为 3.62 毫秒），在训练期间轮换遍历所有 L 层，以跟踪冯·诺伊曼有效秩 eRank [Roy 和 Vetterli， 2007]、 HTSR 幂律尾部指数 α [Martin 和 Mahoney， 2021], Jensen–Shannon 头部冗余惩罚，以及EWMA+MAD （$z \geq 6.5$）损失尖峰断路。每当 $\alpha < 1.90$时，闭环控制器会将 AdamW 权重衰减提高 1.5× （这在早期运行中自行修复了第 8 轮次从 $\alpha = 1.84 \to 1.88$ 开始的下跌）。图 3a 显示，在 Shoonya-350M （20/20、 $\bar{\alpha} = 1.983$）和 Shoonya-1.3B （24/24、 $\bar{\alpha} = 2.075$）上，重尾层达到 100% 的理想状态；相比之下，使用 4.0T–5.5T 个词元的基线遭受 $\alpha < 1.85$过度压缩（在 Qwen2.5-Coder-0.5B-Instruct、 $\bar{\alpha} = 1.762$上仅有 4/24 个理想层）。

•**零显存 LoRA 参考、密集符号** Step-GRPO**和**OrthoKnOTS。后训练结合了完成掩码监督微调（含 <|think|> 规划块）与执行引导式 Step-GRPO [Shao 等人， 2024]。关闭秩为16/32 的LoRA 适配器

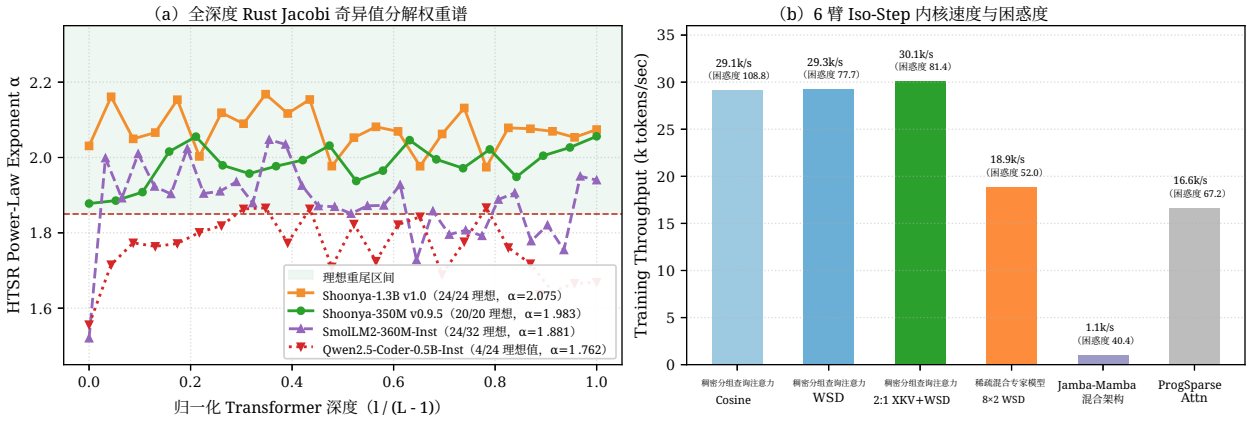

图 3：**(a)** 全深度 Rust 循环雅可比奇异值分解重尾自正则化(HTSR) 指数α 随归一化深度的变化。`Shoonya-1.3B v1.0` （24/24 层，$\bar{\alpha} = 2.075$）和 `Shoonya-350Mv0.9.5` （20/20 层，$\bar{\alpha} = 1.983$）始终严格处于理想的 $\alpha \in [1.85, 3.5]$ 走廊内，而数万亿令牌规模、参数量低于 5 亿的基线则表现出 $\alpha < 1.85$ 相关性陷阱。**(b)** 六臂 Iso-Step 内核的吞吐量与困惑度。

(`set_lora_enabled(False)`) 在不增加显存占用的情况下，精确计算参考对数概率 $\log \pi_{\text{ref}}$ （额外显存为 0 MB）。在 `shoonya-eval` 中使用密集的三阶段符号奖励（抽象语法树/解析有效性 $+0.15$、模块/`rustc`/嵌入式 Cypher 执行 $+0.35$、单元测试/元组匹配 1.00），避免稀疏奖励优势崩溃（$A_i \equiv 0$）。针对领域专门训练的 Python、Rust 和 Cypher LoRA 适配器，通过纯 Rust 实现的共享奇异基进行融合 `KnOTS` [Stoica 等人， 2025] + `TIES`[Yadav 等人， 2023], 并在 `v0.8.5` 中辅以正交零空间清理（`OrthoKnOTS`，移除 28.3% 符号相反的奇异模态，同时保留 99.98% 的能力范数），此外还采用`FLARE` [Jiang 等人， 2023] 熵门控的 `<|search_query|>` 激活 KV 接地（`v0.8.0`），以及零凭证、零出口的 AWS Spot IMDSv2 原子自动恢复机制（< 2.5 秒检查点刷新；同区域 S3 预签名同步）。

5 纵向 陷阱、反向令牌密度陷阱和 AdvKD

5.1 八条从零开始的工程经验教训 (`v0.1` → `v0.9.3`)

将 Shoonya 从 16.8M 参数迭代构建至 1.07B 参数的过程中，在 Milestone `M3` 之前暴露出八种微妙的失效模式（每种都通过回归测试永久锁定）：

L1. 双重因果移位（$t+2$）和填充损失（`v0.1`）： 在数据集加载器和`TransformerLM.forward` 训练位置 t 中都移位标签以预测 $t+2$；单次移位 `ignore_index=-100`立即修复了语法。

L2. 非因果缩放点积注意力和双向滑动窗口（`v0.2`）： 传入 `is_causal=False`时，`attention_mask=None` 泄漏了未来词元（损失 $\to 10^{-4}$，生成率为 0%）；强制使用因果性仅历史掩码修复了自回归。**L3. FIM 哨兵冲突和 AST 中途截断（`v0.3`）：** 将 `<|endoftext|>`（50256）重复用于 FIM 分隔符导致行提前终止；使用不相交的 ID（50257..50260） + 进行 AST 完整打包，消除了表达式中途截断。**L4**

- **CTRL 重复 ition-惩罚符号反转与参数抑制（`v0.4`）：** 除以

负向 logits 乘以惩罚系数 > 1 加剧了重复 [Keskar 等人， 2019], 并惩罚提示词令牌，抑制了函数参数的有效复用（`arr`， `target`）；排除提示词的分段式 CTRL 采样解决了这两个问题。

L5. 未见标识符 $O(n)$ 捷径与小型测试集的假象（`v0.4`–`v0.5`）： 在`binary_search`上， `v0.4` 合成的 O（日志 n）耗时减半（`Cyclomatic=7`， `Halstead=280.0`），但在未见过的重命名提示 `locate_in_sorted` 上，它通过 O（n）线性扫描（`Cyclomatic=3`， `Halstead=155.1`）通过了 100% 的单元断言——这证明功能测试必须与静态抽象语法树复杂度检查配套进行。同样， `v0.5.0` 在 8 项任务的 `Stage-A` 上取得 96.9% 的成绩，耗用 0.43M 个词元；但在 `MBPP-Sanitized`上仅得 0.39%，这促使我们采用两阶段评估门槛。

L6. 重复 `AS name ClauseConflict` (`v0.7.0` → `v0.7.5`)： 投影 `v1.name AS name`、`v2.name AS name` 的双实体 Cypher 模板触发了引擎 `ClauseConflict`，导致在线 Cypher 执行悄无声息地全部归零，直到强制使用不同的语义别名（0.0% → 8.3%，0 次执行错误）。

L7. RLVR 后监督微调的边界软化（`v0.8.0` → `v0.8.5`）： 在 Rust/HumanEval 上进行交叉熵监督微调之后 `Step-GRPO`，边界间隔有所软化；在清理后最后运行 `Step-GRPO OrthoKnOTS`，恢复了单调增益（贪心 MBPP 达 28.02%，4.00% `Cypher-100`）。

L8. 仅编译器 RLVR 的古德哈特化（`v0.8.5` → `v0.9.1`）： 对没有测试的 Rust 代码片段给予满额奖励，导致模型学会生成平凡但可编译的函数体；将无测试时的编译奖励上限设为 0.35，并要求提供 `#[test]`测试框架，使 `Rustc-Exec-64`达到 20.31% TTC（`v0.9.3`）。

5.2 Lesson L9：反向 token 密度蒸馏陷阱与 AdvKD

知识蒸馏 [Hinton 等，2015，Agarwal 等， 2024] 假设较大的教师模型 T（$N_T > N_S$）在各种词元角色上都优于较小的学生模型 S，因为两个模型都是在接近 Chinchilla 词元密度的条件下训练的 $\rho = D/N$ [Hoffmann 等， 2022]。在 Milestone `M3` 的严格 0%-`NC-SA` 计算预算下，我们成熟的 312M 学生模型（`v0.9.3`， $D_S = 796.7$M 词元， $\rho_S = 2.55$ 词元/参数）的词元密度高于新近搭建的 1.07B 教师模型（`1.3B-v0.9.3`， $D_T = 62.9$M 词元， $\rho_T = 0.059$ 词元/参数）。由于其 $d = 2048$ 主干迅速掌握了四种表面形式的 JSON/Cypher 语法（66.7% `ToolPlan/Extraction`，100% `Enrichment`），但在算法代码上仍显不足（MBPP 得分为 10.89%，而学生模型为 31.52%），将未经门控的 top-64 序列 KL 蒸馏用于 `350M-v0.9.4`，引发了严重的能力反转：

命题 *1*（反向令牌密度下的 *Logit* 梯度冲突）。在上下文 c 中，若续写 $y^\star$ 可验证为正确，且由于条件教师熵 $H(p_T(\cdot \mid c)) \gg H(p_{S_0}(\cdot \mid c))$ 更高，高密度学生锚点满足 $p_{S_0}(y^\star \mid c) > p_T(y^\star \mid c)$，则最小化无门控正向 $KL D_{\mathrm{KL}}(p_T^{(K)}(\cdot \mid c) \,\|\, p_S^{(K)}(\cdot \mid c))$ 会在 S_0 附近诱发正梯度 $\partial L^{\mathrm{KL}}_{\partial z_S, y^\star} = p_S^{(K)}(y^\star \mid c) - p_T^{(K)}(y^\star \mid c) > 0$，从而侵蚀学生模型在 $y^\star$ 上的概率质量，转而偏向未经验证的教师尾部令牌。

从实证结果看（图 4b、表 2）， `v0.9.4` 获得了多场景 JSON/Cypher 格式化能力（0% →66.7%），但在 `MBPP` TTC 上下降了 -12.06 个百分点（31.52% → 19.46%），在`HumanEval` TTC 上下降了 -7.93 个百分点（23.78% → 15.85%），并在 `Rustc-Exec-64` TTC 上下降了 -15.62 个百分点（20.31% →4.69%），尽管仍保留了 $20/20$ 个理想奇异值分解层（$\bar{\alpha} = 1.982$）。

里程碑 `M4` 通过两种相互耦合的机制解决了这个 Trap：

1. $6.53\times$ **`WSD` 持续教师模型扩展（`1.3B-v1.0-Clean`）。**在 $2\times$ L40S GPU 上流式处理未见过的 `v1.0_clean` 分片（385.9M 个词元， $T = 512$时损失为 0.8032； $\alpha = 3.01 + 25.2$M 个词元， $T = 2048$时损失为 0.7086；累计 411.04M 个干净令牌） +

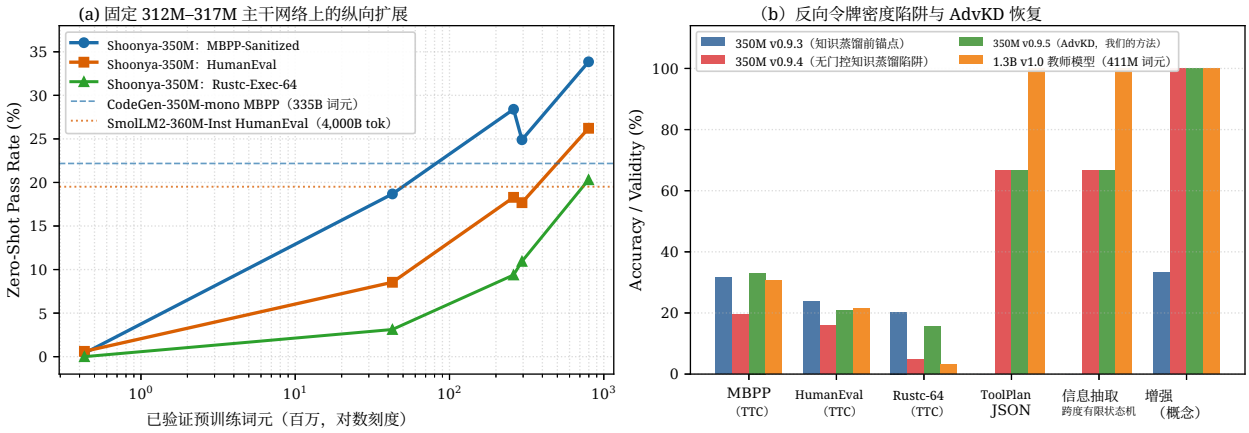

图 4: **(a)** Shoonya-350M（0.43M → 796.7M 经验证令牌）与外部 350M–360M 基线（335B–4,000B 词元）的纵向样本效率。**(b)** 通过 6.53× 教师模型扩展（1.3B-v1.0）和 AdvKD（350M-v0.9.5）解决 M3 反向词元密度 Trap（v0.9.4）。

KnOTS + Step-GRPO 提升教师模型 MBPP 2.82×（10.89% → 30.74% TTC）、HumanEval 1.75×（12.20% → 21.34% TTC），将进程内 Arm 2 Cypher 语法准确率提升至 91.7%，并将 ToolPlan、Extraction和 Enrichment 提升至 100.0%。

2. **零显存验证器优势门控知识蒸馏（AdvKD）**。通过 set_lora_enabled(False) 在额外显存为 0 MB 的条件下评估冻结的 KD 前锚点S_0（v0.9.3），我们仅在教师模型同时满足贪心精确且惊异度低于 S_0 时应用教师模型 Top-64 KL 散度（$\tau = 2.0$）：

$$g_t = \mathbb{I}\Big[\arg\max_{v\in\mathcal{V}} p_T(v \mid y_{<t}) = y_t \ \wedge\ \log p_T(y_t \mid y_{<t}) \geq \log p_{S_0}(y_t \mid y_{<t})\Big], \tag{2}$$

$$\begin{aligned}\mathcal{L}_{\text{AdvKD}}(t) = g_t\Big[(1-\lambda)\mathcal{L}_{\text{CE}}(p_S, y_t) + \lambda\tau^2 D_{\text{KL}}(p_T^{(K),\tau} \,\|\, p_S^{(K),\tau})\Big] \\ + (1-g_t)\Big[\mathcal{L}_{\text{CE}}(p_S, y_t) + \beta D_{\text{KL}}(p_{S_0}^{(K)} \,\|\, p_S^{(K)})\Big].\end{aligned} \tag{3}$$

门控 $g_t = 1$ 在 55.76% 的活跃词元上触发，同时将其余 44.24% 锚定到 S_0，随后经过 36 步的 Step-GRPO 润色，生成 **Shoonya-350M v0.9.5-Clean-AdvKD**。

6 实验评估

6.1 纵向扩展、AdvKD 与同主机竞品结果（593 个问题）

表 2 中的所有模型均在**同一 RTX 4050 主机**（BF16、t =0 Std 和 $N = 4$ 无预言信息 Search-TTC）上，针对全部 593 个经 SHA-256 去污染的问题（Stage-A 8、MBPP-Sanitized 257 [Austin 等人、2021]、OpenAI HumanEval 164 [Chen 等人、2021]、Rustc-Exec-64 64以及内嵌的 Cypher-Bench-100 100）进行了连续评估：

- **纵向阶段转换（v0.5 → v0.9.3）**。在保持 312M–317M 主干网络不变的情况下，将干净令牌数量从 0.43M (v0.5.0) 扩展到 260.15M (v0.7.0–v0.8.5)，再扩展到 796.71M (v0.9.1–v0.9.3)，使 MBPP-Sanitized 86.8× (0.39% → 33.85% TTC) 提升，HumanEval 43.0×

模型（$t=0$ 标准 / $N=4$ TTC）（同一主机 RTX 4050 BF16）	参数 (M)	预训练词元	Stage-A (8)	MBPP-San (257)	HumanEval (164)	Rustc-64 (64)	Cypher-100 (100)	理想奇异值分解 ($\bar{\alpha}$)
外部子500*M* 基线（贪心 $t=0$）								
CodeGen-350M-mono [Nijkamp等人， 2023]	356.7	335B	62.5%	22.18%	12.80%	0。00%	0。00%	17/20 (1.92)
SmolLM2-360M-Inst [Allal等人， 2025]	361.8	4，000B	87。5%	38。13%	19.51%	15.62%	1.00%	24/32 (1.88)
Qwen2.5-Coder-0.5B [Hui 等人， 2024]	494.0	5,500B	100.0%	47。08%	27.44%	46。88%	8.00%	1/24 (1。76)
Qwen2.5-Coder-0.5B-Inst [Hui 等， 2024]	494.0	5,500B	87。5%	42.80%	50。61%	32。81%	0。00%	4/24 (1。76)
此前的 *Shoonya* 谱系（*v0.5.0 → v0.8.5-OrthoKnOTS*；317.3*M* 参数）								
350M v0.5.0（风洞种子）	317.3	0。0004B	96。9%	0.39%	0。61%	0。00%	0。00%	20/20 (2.12)
350M v0.6.0(42.7M 步)	317.3	0.043B	93.8%	18.68%	8.54%	3.12%	0。00%	20/20 (2.03)
350M v0.7.5-RLVR(KnOTS+GRPO)	317.3	0.260B	87。5%	27.24/26.85%	15.24/16.46%	9.38%	0.00%	20/20 (2.01)
350M v0.8.0-Grounded(FLARE)	317.3	0.260B	87。5%	27.63/28.40%	15.24/15.85%	9.38%	2.00/3.00%	20/20 (2.01)
350M v0.8.5-OrthoKnOTS	317.3	0.260B	87。5%	28.02/28.02%	15.24/17.07%	3.12/6.25%	4.00/4.00%	20/20 (2.01)
商业清洁 *Shoonya* 系列 (0.0% *NC-SA, WSD + 2:1 XKV;* 标准 / 验证器 *TTC)*								
350M v0.9.0-Clean-Inst	312.1	0.295B	100/100%	24.90/24.90%	17.68/17.68%	6.25/10.94%	1.00/1.00%	20/20 (2.00)
350M v0.9.1-Clean-Inst	312.1	0.797B	100/100%	32.68/33.85%	21.95/26.22%	8.59/17.19%	1.50/2.00%	20/20 (1。98)
350M v0.9.2-KnOTS-Think	312.1	0.797B	100/100%	33.46/33.07%	19.82/26.22%	10.16/12.50%	1.00/4.00%	20/20 (1.98)
350M v0.9.3-EntNLQ(S_0)	312.1	0.797B	100/100%	30.74/31.52%	18.60/23.78%	16.41/20.31%	1.00/1.00%	20/20 (1。98)
350M v0.9.4-UnGatedKD(Trap)	312.1	0.797B	93.8/87.5%	18.68/19.46%	13.41/15.85%	3.91/4.69%	0.00/0.00%	20/20 (1.98)
350M v0.9.5-AdvKD(最新)	**311.7**	**0。797B**	**100/100%**	**30.93/33.07%**	**17.68/20.73%**	**9.38/15.62%**	**1.00/3.00%**	**20/20 （1。98）**
1.3B v0.9.3-Clean(M3 种子)	1，065。1	0.063B	100/100%	9.14/10.89%	8.23/12.20%	0.00/0.00%	0.00/0.00%	24/24 (2.11)
1.3B v1.0-Clean(最新)	**1，065。1**	**0。411B**	**87.5/87.5%**	**28.21/30.74%**	**17.68/21.34%**	**3.12/3.12%**	**0.50/1.00%**	**24/24 （2。07）**

表 2：在全部 593 个经 SHA-256 去污染的问题上，以及完整深度 Rust 循环雅可比奇异值分解中，完整纵向进展（v0.5.0 → v0.9.5-AdvKD 和 1.3B-v1.0）与同主机竞品评估。

（0。61% → 26.22% TTC）， Rustc-Exec-64 从 0。00% → 20.31% TTC，到 4。00%（96。0%–99。0% 的有效语法），同时将幻觉生成的 Python 标识符减少 −91.4%，并将 ECE 减少 −90.6%（0.323 → 0.030）。

- 420×–5，019× **样本效率**。在 0。797B 0%-NC-SA 词元规模下， Shoonya-350M v0.9.1-Clean在所有基准测试中均优于 CodeGen-350M-mono （335B 词元）（+11.67 MBPP TTC 分，+13.42 HumanEval TTC 分，+17.19 Rust 分），并在 Stage-A （100% 对 87。5%）、 HumanEval （21。95% 标准 / 26。22% TTC 对 19。51%）、Rustc-Exec-64 （17。19% 对 15。62%）以及 Cypher-100 （2。00%–4。00% 对 1。00%）上超过 SmolLM2-360M-Instruct （4，000B 词元）。此外请注意， Qwen2.5-Coder-0.5B Base → Instruct 在 Python HumanEval上提升了 27。44% → 50.61%，但由于以 Python 为主导的聊天调优，在 Rust 上回退了 46。88% → 32.81%，在 OpenCypher 上回退了 8。00% → 0.00%。

- **AdvKD 恢复率（v0.9.5）**。 Shoonya-350M v0.9.5-AdvKD在 +12.25 **标准 /** +13.61 **TTC 点**上恢复了 MBPP (19.46% → 33.07% TTC，优于 KD 前 v0.9.3 的 31.52%)，在 +4.88 **TTC 点**上恢复了 HumanEval (20.73%)，在 3.33× **TTC** 上恢复了 Rustc-Exec-64 (4.69% → 15.62%)，并在 Cypher-100 上恢复了 3.00% TTC，同时保留了 20/20 理想 SVD 层（$\bar{\alpha}=1.983$）。

6.2 进程内 samyama-graph-enterprise 四表面结果

表 3 报告了在 samyama-graph-enterprise v1.15.0 临床企业知识图谱上的实时零套接字执行（12 个分层自然语言查询到 Cypher 任务： + 6智能体式 ToolPlan、 ExtractionAgent和 Enrichment 个任务 + 实时 DDL 演进）。Shoonya-1.3B v1.0 INT8 （1，162.9 MiB 常驻集大小，20.3 μ秒多版本并发控制检查）达到 75.0% 的原始 Arm 1 语法准确率、91.7%（11/12）的精简 Arm 2语法准确率，以及在 ToolPlan JSON、源跨度有依据的 Extraction 和Enrichment/弃答方面的 100.0%。Shoonya-350M v0.9.5-AdvKD INT8 （389.8 MiB 常驻集大小，59.8 令牌/秒）保持了 66.7% 的原始 Arm 1 语法准确率、75.0% 的 Arm 2 语法准确率、66.7% 的 ToolPlan/Extraction （借助M5.A的 JSON 转义的 Cypher 有限状态机，将实时 ToolPlan 多版本并发控制执行率从 0.0% 提升至 → 33.3%）、100%/100% 的 Enrichment，以及 50.0%（2/4）的一级一跳求解率，泄漏率为 0.0%。

检查点 (samyama v1.15.0)	精确率 (PagedKV)	RSS (MiB)	热启动首令牌时间 (加速比)	速度 (令牌/秒)	MVCC (μs)	Arm1/Arm2 语法	一级 已解决	Arm4 Leak	Plan / Extr	Enrich (概念/抽象)
350M v0.9.2(M1)	INT8	389.2	920.8 毫秒 (5.3×)	28.7	55.3	0.0/16.7%	0.0%	**0.0%**	0/0%	0/0%
350M v0.9.3(M2)	INT8	389.5	400.4 毫秒 (4.9×)	67.9	64.3	33.3/66.7%	**50.0%**	**0.0%**	0/0%	33/100%
350M v0.9.3(M2)	BF16	685.3	612.4 毫秒 (5.9×)	42.8	62.6	16.7/50.0%	25.0%	**0.0%***	—	—
350M v0.9.4(Trap)	INT8	389.6	396.0 毫秒 (5.2×)	63.6	66.2	58.3/58.3%	**50.0%**	**0.0%**	67/67%	100/100%
350M v0.9.5-AdvKD	**INT8**	**389.8**	**427.7 毫秒 (5.0×)**	**59.8**	**64.6**	**66.7/75.0%**	**50.0%**	**0.0%**	**67/67%**	**100/100%**
350M v0.9.5-AdvKD	BF16	685.3	651.4 毫秒 (5.9×)	39.6	62.6	66.7/75.0%	50.0%	**0.0%**	—	—
1.3B v0.9.3(M3)	INT8	1,163.0	1,052.6 毫秒 (6.0×)	25.0	19.6	33.3/50.0%	0.0%	**0.0%**	67/67%	100/100%
1.3B v1.0-Clean	**INT8**	**1,162.9**	**1,193.1 毫秒 (5.8×)**	**22.0**	**20.3**	**75.0/91.7%**	**25.0%**	**0.0%**	**100/100%**	**100/100%**
1.3B v1.0-Clean	BF16	2,200.5	2,109.3 毫秒 (6.0×)	12.6	21.3	66.7/91.7%	25.0%	**0.0%**	—	—

表 3：在 CPU 上对 samyama-graph-enterprise v1.15.0 进行实时零套接字评估（未修改任何数据库文件；目录有限状态机构建耗时 29.8–37.5 μ秒）。在 v0.9.3 BF16 上停用目录有限状态机（Arm 3）会产生 8.33% 的假阳性泄漏； Arm 4 （目录有限状态机 + 多版本并发控制门控）在所有运行中将泄漏率控制在 0.0%。

7 相关工作

嵌入式数据库与结构化大语言模型服务。DuckDB 表明，进程内分析型执行可消除线协议开销 [Raasveldt 和 Mühleisen， 2019],而 Samyama 在 Rust 中统一了属性图、向量和优化工作负载 [Mandarapu andKunkunuru，2026b]。在.....方面 PagedAttention [Kwon 等人， 2023], SGLang [Zheng 等人， 2024], 以及 Outlines[Willard 和 Louf， 2023] 通过静态令牌字符串缓存和预编译正则表达式优化进程外服务， shoonya-infer则嵌入数据库管理系统进程中，将 KV 有效性绑定到事务性 DDL 周期，并构建 36 μs 基于实时列式 GraphStore 结构的 logit 掩码。

紧凑型代码小语言模型、子空间合并和蒸馏。包括 Phi-1[Gunasekar 等、 2023]、 CodeGen [Nijkamp等人、 2023]、 MiniCPM [Hu 等、 2024]、 SmolLM2 [Allal等人、 2025], 以及 Qwen2.5-Coder [Hui 等在内的 10 亿参数以下模型，使用 70 亿至 5.5 万亿词元进行训练。容量差距蒸馏已有相关研究 $N_T \gg N_S$ [Mirzadeh 等人，2020 年，Agarwal 等， 2024]；我们在严格的商业合规数据预算下，分离出正交反向令牌密度区间（$\rho_T \ll \rho_S$），并通过零显存 LoRA 门控 AdvKD加以解决，同时补充 KnOTS/TIES 合并 [Stoica 等人，2025年，Yadav 等人， 2023] 以及 GRPO [Shao 等， 2024]。

8 局限性与前沿替换就绪度

表 4 展示了 Shoonya 在四种 samyama-graph-enterprise 大语言模型接口上的生产就绪度。尽管本体 Enrichment （100%/100%）和目录+跨度掩码 ExtractionAgent（100% 有依据）在本地已可独立就绪，且 Tier-1 NLQPipeline + AgentPlanner 可安全地充当零泄漏 Tier-0 路由器，但仍有三项局限构成当前里程碑 M5 工作的重点：（1）由于 4/11 投影别名不匹配、 4/11 多跳箭头方向偏差以及 3/11 未建立索引的字符串字面量，Tier-2/3 多跳 OpenCypher（OPTIONAL MATCH、 WITH..HAVING）在 M4 的贪心精确匹配率仍为 0.0%；（2）与 350M INT8 （400 毫秒）相比， 1.3B INT8 CPU 预热首令牌时间（1.19 秒）更适合异步丰富，而非交互式输入；（3）目前正在将实时进程内基准测试从 18 项临床企业任务（+593 公开任务）扩展至多租户企业图。

samyama-graph-enterprise 表面	最佳本地准确率（M4/ M5.A）	Leak	部署模式
4. 本体 Enrichment& 弃答	100.0% 概念 / 100.0% 弃答	0.0%	**独立本地**
3. ExtractionAgent (目录+跨度有限状态机)	100.0% JSON / 100.0% 有依据	0.0%	**独立本地**
2. AgentPlanner （ToolPlanJSON）	100.0% JSON（1.3B）；33.3% 实时执行（M5.A）	0.0%	**混合第0层**
1a. NLQPipeline 一级（单跳）	50.0% 已解决（350M），100.0% 精确率	0.0%	**混合第0层**
1b. NLQPipeline Tier-2/3（多跳）	91.7% 语法，0.0% 贪心精确匹配（M4）	0.0%	**升级至前沿模型**

表 4：四个 samyama-graph-enterprise 表面上诚实的前沿模型替代就绪度。

9 结论

在企业级图数据库的地址空间中嵌入零依赖纯 Rust 语言模型运行时，可将数据库内部机制——DDL 世代、列式属性目录和微秒级只读 MVCC 快照——转变为一等推理原语。结合秩分解 KV 共享骨干网络、闭环 Rust 雅可比奇异值分解治理、经执行验证的Step-GRPO，以及验证器优势门控知识蒸馏（AdvKD），Shoonya 可在390 MiB CPU 内存中实现独立的本体增强和有依据的抽取，并提供零泄漏 Tier-0 查询路由。开放权重（Apache-2.0）、分片清单和谱证书可在 https://huggingface.co/VaidhyaMegha/Shoonya获取。

参考文献

Rishabh Agarwal、Nino Vieillard、Yongchao Zhou、Piotr Stanczyk、Sabela Ramos Garea、Matthieu Geist 和 Olivier Bachem。同策略蒸馏语言模型：从自生成的错误中学习。载于国际学习表征会议 *(ICLR)*，2024。

Joshua Ainslie、James Lee-Thorp、Michiel de Jong、Yury Zemlyanskiy、Federico Lebrón 和 Sumit Sanghai。GQA：从多头检查点训练广义多查询 Transformer 模型。载于 *2023*年自然语言处理经验方法会议 *(EMNLP)* 论文集，第 4895–4901 页，2023 年。

Loubna Ben Allal、Anton Lozhkov、Elie Bakouch、Gabriel Martín Blázquez、Guilherme Penedo、Lewis Tunstall、Andrés Marafioti、Hynek Kydlíček、Agustín Piqueres Lajarín、Vaibhav Vahtola等。SmolLM2：小模型如何做大——以数据为中心的小型语言模型训练。*arXiv* 预印本 *arXiv:2502.02737*，2025年。

Jacob Austin、Augustus Odena、Maxwell Nye、Maarten Bosma、Henryk Michalewski、David Dohan、Ellen Jiang、Carrie Cai、Michael Terry、Quoc Le 和 Charles Sutton。使用大型语言模型进行程序合成。*arXiv* 预印本 *arXiv:2108.07732*，2021年。

Mohammad Bavarian、Heewoo Jun、Nikolas Tezak、John Schulman、Christine McLeavey、Jerry Tworek 和 Mark Chen。训练语言模型高效填充中间内容。*arXiv* 预印本 *arXiv:2207.14255*，2022年。

Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。评估在代码上训练的大型语言模型。*arXiv* 预印本 *arXiv:2107.03374*，2021年。

Tri Dao。FlashAttention-2：通过更好的并行性和工作划分实现更快的注意力。在国际学习表征会议（*ICLR*），2024。

Nadime Francis、Alastair Green、Paolo Guagliardo、Leonid Libkin、Tobias Lindaaker、Victor Marsault、Stefan Plantikow、Mats Rydberg、Petra Selmer 和 Andrés Taylor。Cypher：一种不断演进的属性图查询语言。在*2018*年国际数据管理会议（*SIGMOD*）论文集，第1433–1445页，2018年。Suriya Gunasekar、Yi Zhang、Jyoti Aneja、Caio César Teodoro Mendes、Allie Del Giorno、Sivakanth Gopi、Mojan Javaheripi、Piero Kauffmann、Gustavo de Rosa、Olli Saarikivi 等。《教科书就是你所需要的一切》。*arXiv*预印本 *arXiv:2306.11644*，2023 年。Nathan Halko、Per-Gunnar Martinsson 和 Joel A. Tropp。以随机性发现结构：用于构造近似矩阵分解的概率算法。*SIAM*评论，53（2）：217–288，2011年。Geoffrey Hinton、Oriol Vinyals 和 Jeff Dean。蒸馏神经网络中的知识。*arXiv*预印本 *arXiv:1503.02531*，2015年。Jordan Hoffmann、Sebastian Borgeaud、Arthur Mensch、Elena Buchatskaya、Trevor Cai、Eliza Rutherford、Diego de Las Casas、Lisa Anne Hendricks、Johannes Welbl、Aidan Clark 等。训练计算量最优的大语言模型。在神经信息处理系统进展（*NeurIPS*），第35卷，第30016–30030页，2022年。Shengding Hu、Yuge Tu、Xu Han、Chaoqun He、Ganqu Cui、Xiang Long、Zhi Zheng、Yewei Fang、Yuxiang Huang、Weilin Zhao 等。MiniCPM：借助可扩展训练策略，揭示小型语言模型的潜力。*arXiv* 预印本 *arXiv:2404.06395*，2024。Binyuan Hui、Jian Yang、Zeyu Cui、Jiaxi Yang、Dayiheng Liu、Lei Zhang、Tianyu Liu、Jiajun Zhang、Bowen Yu、Keming Lu 等。Qwen2.5-Coder技术报告。*arXiv*预印本 *arXiv:2409.12186*，2024。Zhengbao Jiang、Frank F. Xu、Luyu Gao、Zhiqing Sun、Qian Liu、Jane Dwivedi-Yu、Yiming Yang、Jamie Callan 和 Graham Neubig。主动检索增强生成。在*2023*年自然语言处理经验方法会议（*EMNLP*）论文集，第7969–7992页，2023 年。Nitish Shirish Keskar、Bryan McCann、Lav R. Varshney、Caiming Xiong 和 Richard Socher。CTRL：一种用于可控生成的条件变换器语言模型。*arXiv*预印本 *arXiv:1909.05858*，2019年。Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph E. Gonzalez、Hao Zhang 和 Ion Stoica。利用PagedAttention高效管理大语言模型服务的内存。在第*29*届操作系统原理研讨会（*SOSP*）论文集，第611–626页，2023 年。Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。ALBERT：用于语言表示自监督学习的轻量级BERT。在国际学习表征会议（*ICLR*），2020年。Madhulatha Mandarapu 和 Sandeep Kunkunuru。知识图谱 grounding 仅在训练外知识场景下对大语言模型有帮助：一项关于临床问答的对照研究。*arXiv*预印本 *arXiv:2606.22419*，2026a。

Madhulatha Mandarapu 和 Sandeep Kunkunuru。Samyama：一种统一的图向量数据库，具备数据库内优化、智能体增强和硬件加速功能。*arXiv* 预印本 *arXiv:2603.08036*，2026b。Madhulatha Mandarapu 和 Sandeep Kunkunuru。可移除与不可约：多语言分词税的令牌成本账本。*arXiv* 预印本 *arXiv:2609.00378*，2026c。Charles H. Martin 和 Michael W. Mahoney。深度神经网络中的隐式自正则化：来自随机矩阵理论的证据及其对学习的影响。《机器学习研究期刊》，22(165):1–73，2021年。Seyed Iman Mirzadeh、Mehrdad Farajtabar、Ang Li、Nir Levine、Akihiro Matsukawa 和 Hassan Ghasemzadeh。通过教师助手改进知识蒸馏。载于*AAAI*人工智能会议论文集，第34卷，第5191–5198页，2020年。Erik Nijkamp、Bo Pang、Hiroaki Hayashi、Lifu Tu、Huan Wang、Yingbo Zhou、Silvio Savarese 和 Caiming Xiong。CodeGen：一种用于代码、支持多轮程序合成的开放式大语言模型。载于国际学习表征会议（*ICLR*），2023 年。Bowen Peng、Jeffrey Quesnelle、Honglu Fan 和 Enrico Shippole。YaRN：高效扩展大型语言模型的上下文窗口。载于国际学习表征会议（*ICLR*），2024。Mark Raasveldt 和 Hannes Mühleisen。DuckDB：一种可嵌入的分析型数据库。载于*2019*年数据管理国际会议（*SIGMOD*）论文集，第1981–1984页，2019年。Olivier Roy 和 Martin Vetterli。有效秩：一种衡量有效维度的方法。载于第*15*届欧洲信号处理会议（*EUSIPCO*），第606–610页，2007年。Zhihong Shao、Peiyi Wang、Qihao Zhu、Runxin Xu、Junxiao Song、Xiao Bi、Haowei Zhang、Mingchuan Zhang、Y. K. Li、Y. Wu 和 Daya Guo。DeepSeekMath：突破开放语言模型数学推理的极限。*arXiv* 预印本 *arXiv:2402.03300*，2024。George Stoica、Pratik Ramesh、Boglarka Ecsedi、Leshem Choshen 和 Judy Hoffman。使用奇异值分解合并模型以绑定 KnOTS。载于国际学习表征会议（*ICLR*），2025年。Jianlin Su、Murtadha Ahmed、Yu Lu、Shengfeng Pan、Wen Bo 和 Yunfeng Liu。RoFormer：采用旋转位置嵌入的增强型 Transformer。《神经计算》，568:127063，2024。Yutao Sun、Li Dong、Yi Zhu、Shaohan Huang、Wenhui Wang、Shuming Ma、Quanlu Zhang、Jianyong Wang 和 Furu Wei。只需缓存一次：用于语言模型的解码器-解码器架构。载于神经信息处理系统进展（*NeurIPS*），第37卷，2024。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。注意力就是你所需要的一切。载于神经信息处理系统进展（*NeurIPS*），第30卷，2017年。Brandon T. Willard 和 Rémi Louf。面向大型语言模型的高效引导生成。*arXiv* 预印本 *arXiv:2307.09702*，2023 年。Prateek Yadav、Derek Tam、Leshem Choshen、Colin A. Raffel 和 Mohit Bansal。TIES合并：解决模型合并时的干扰。载于神经信息处理系统进展（*NeurIPS*），第36卷，2023 年。

Lianmin Zheng、Liangsheng Yin、Zhiqiang Xie、Chuyue Sun、Jeff Huang、Cody Hao Yu、Shiyi Cao、Christos Kozyrakis、Ion Stoica、Joseph E. Gonzalez、Clark Barrett 和 Ying Sheng。SGLang：高效执行结构化语言模型程序。载于神经信息处理系统进展（*NeurIPS*），第37卷，2024。