KubeCon CloudNativeCon
North America 2026

From Fine-Tuning to Production: Scaling Agentic AI at Shopify via Inference Perf

Salt Lake City, Utah, United States





Ashok Chandrasekar Navjot Singh

Who We Are & Why We Built This Together



Ashok Chandrasekar

Staff Software Engineer • Google

`kubernetes-sigs/inference-perf` • `@achandrasekar`

- Tech Lead for GKE Inference performance and GKE TPU Inference.
- Core-maintainer of Inference Perf and SIG Benchmarking lead in llm-d.



Navjot Singh

Machine Learning Engineer • Shopify

`Agentic Serving & SFT` • `@Navjot10`

- Machine learning Engineer at Shopify.
- Contributed conversation_replay to Inference Perf to model stateful multi-turn sessions & tool waits.

Better task quality. Lower serving cost.

Our goal: fine-tune open models to beat frontier baselines on our tasks, at a fraction of the cost.

Quality

Held-out task evaluations

Economics

Cost per successful task

Capacity

Useful throughput per GPU

Navjot Singh · Machine Learning Engineer Shopify

Continual learning creates task specialists.

Ground truth and calibrated evaluation guide repeated improvements in quality and cost.

Ground truth

Product specs
+ expert labels

→

Calibrated judge

Align scoring
to human judgment

→

Frontier MVP

Ship a baseline
Collect real failures

REPEAT WITH PRODUCTION FEEDBACK

Self-healing

Diagnose → repair
Re-score accepted outputs

→

Model training

Fine-tune open models
on improved trajectories

→

Serving validation

Replay realistic workloads
Measure latency + capacity

Evaluate candidates on task quality, serving latency and cost per completed job.

Production feedback drives the next data, training and serving iteration.

Connect the model release to a capacity plan.

Pulse research alternates model and tool calls while conversation history grows.

Fine-tune

Task trajectories
Versioned checkpoint

→

Evaluate

Held-out task quality
Frontier comparison

→

Benchmark

inference-perf sweeps
Latency + goodput

→

Deploy

Canary + observe
Production feedback

PULSE CASE STUDY: MULTI-TURN RESEARCH

Sequential turns

conversation_replay
Preserve model-call order

Growing history

Context + token distributions
Exercise the full conversation

Tools + concurrency

tool_call_latency_sec
load.type: concurrent

One benchmark Job and a versioned serving target.

A repeatable run combines configuration, model metadata, request reports and synchronized telemetry.

INFERENCE-PERF JOB • EXCERPT

```
apiVersion: batch/v1
kind: Job
metadata: {name: inference-perf}
spec:
  template:
    spec:
      restartPolicy: Never
      containers:
      - name: benchmark
        command: ["inference-perf"]
        args: ["--config_file",
               "/etc/config/config.yml"]
```

REQUEST PATH

Benchmark Job → Service / gateway

Streaming requests to versioned GPU replicas

Server + GPU telemetry

Queues, KV cache, preemptions and GPU activity

Durable report bundle

Resolved config, revisions and per-stage outputs

Replay Pulse's multi-turn workload shape.

Preserve turn order, growing context and tool waits as concurrent research jobs run.

```
api: {type: completion, streaming: true}
data:
  type: conversation_replay
  conversation_replay:
    seed: 42
    num_conversations: 200
    turns_per_conversation: {type: fixed, mean: 38}
    tool_call_latency_sec: {type: fixed, mean: 8}
    # Set context and token-length distributions
load:
  type: concurrent
  stages:
  - {concurrency_level: 5, num_requests: 2000}
  - {concurrency_level: 10, num_requests: 2000}
```

Growing context

Shared context + history
Input/output length distributions.

Ordered turns

Preserve model-call order.
Carry history into the next turn.

Tool waits + concurrency

Simulate inter-turn delays.
Sweep concurrent conversations.

Example values. Run the full version-pinned configuration with measured workload distributions.

Read quality, latency and capacity evidence together.

Choose a serving configuration using successful task throughput, latency SLOs and cost.

1 Validate the workload

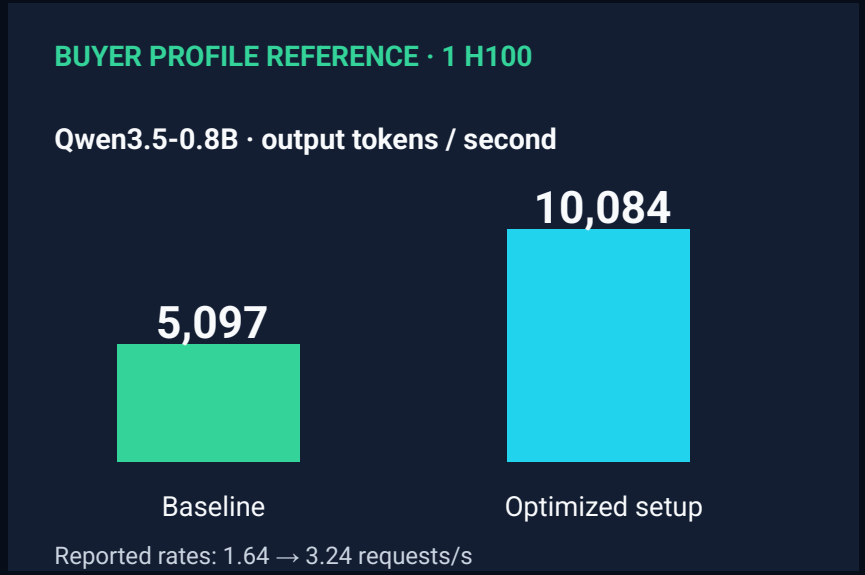
Token lengths, achieved load, completed research jobs + errors.

2 Inspect latency + goodput

TTFT, TPOT / ITL, task latency and SLO-compliant throughput.

3 Explain the result

Align queues, cache, preemptions and GPU telemetry with each stage.



Prompt + runtime changes; synthetic, C=384, 1 run/config. Non-streaming; fp8 KV quality eval pending.

Turn measured service rates into a GPU plan.

Hold the quality bar fixed, run controlled experiments, and validate the selected deployment at fleet scale.

OBSERVATION → NEXT EXPERIMENT

TTFT rises

Test prefill work, queueing and cache reuse.

KV pressure rises

Test active-sequence and context budgets.

GPUs idle with queued work

Inspect tools, client dispatch, CPU and network.

SIZE FROM USEFUL CAPACITY

Replicas = ⌈ demand / safe rate ⌉

Use tasks/hour and measured safe tasks/hour per replica.

Compare a fixed GPU budget

8 × TP1 ↔ 2 × TP4, if memory fits

Plan reserve • test real routing + tool waits • canary • validate latency, backlog and cost/task

Are LLM Performance Benchmarks Reliable?

"What is frequently diagnosed as an infrastructure issue is, in reality, a measurement bias induced by the benchmark itself."



PART 2 THESIS: Trustworthy serving decisions require unbiased clients, standardized workloads & causal replay.

Four Hidden Pitfalls That Corrupt Benchmark Numbers

1. Sampling Variance (21% Shift)

Greedy temp=0.0 vs temp=0.7 shifts output length & tok/s by 21%.

2. Dataset Drift (50% Load Gap)

Unpinned ShareGPT filtering alters total generated tokens by 50%.

3. Unseeded Cache Illusions

Repeated unseeded prefixes trigger silent 95% cache hits across runs.

4. Client/Server Token Drift

Client tokenizers miscount chat templates; track server usage tokens.



PROVENANCE: Inference Perf pins RNG seeds and audits prompt_tokens.cached & token_count_mismatches.

Decoupled Workers Cut Delay: 58s -> 0.63ms

The Python GIL Ceiling

At 5,000 QPS, a single-process client caps at 445 QPS and hallucinates 58s of client delay into TTFT!

Built-In Fidelity Validation

Independent OS workers sustain 4,998 QPS; PerStageLifecycleValidator writes validation.json.



RULE #1: Always inspect schedule_delay & validation.json before blaming GPU serving pods.

One Engine-Agnostic Tool Across APIs & Modalities

APIs & Reasoning Thinking

OpenAI Chat/Completions, Anthropic Messages (thinking budget_tokens), Embeddings & Rerank.

Multimodal Payloads

Configurable Image, Video & Audio blocks testing HTTP serialize + vision encoder prefill.

Auto-Saturation Sweeps

Rate & concurrency binary search auto-discovers your cluster's exact latency/SLO knee.

Realistic Arrival Shapes

Poisson, Gamma (burstiness), Constant, Concurrent & Trace Session Replay schedules.

LoRA Multi-Adapter Traffic

Weighted traffic_split across adapters stressing dynamic GPU LoRA hot-swapping.

Full-Stack Telemetry

Prometheus scraping, OTel Metrics/Trace export, Circuit Breakers & --analyze plots.

SERVER AGNOSTIC: Benchmarks vLLM, SGLang, TGI, TensorRT-LLM, JetStream, Ollama & Frontier APIs.

Turning Tribal Guesses into Workload Blueprints

Statistical Schema • prism.llm-d.ai

```
// workload-catalog/code-generation/config.json
{
  "input_sequence_length": {
    "min": 10000, "max": 990000,
    "mean": 160000, "distribution_type": "log_normal"
  },
  "number_of_turns": {"mean": 540, "max": 3000},
  "time_between_turns": {"mean": 15, "max": 1000},
  "input_sequence_type_token_ratio": 0.03,
  "metadata": {
    "prefill_decode_ratio": 376.47,
    "suitable_optimizations": [
      "Prefix aware routing", "KV cache offloading"
    ],
    "slos": {"ttft_p95_ms": 30000, "itl_p99_ms": 100}
  }
}
```

Two Artifacts Per Blueprint

Machine-readable config.json (distributions + SLOs) paired with ready-to-run inference-perf.yaml.

Synthetic + Empirical Trace Schemas

Covers both parameterized distributions and empirical OTel trace-replay characteristics.

Mapped to K8s Optimizations

Links P:D ratio, vocabulary TTR & turn cadence directly to Prefix Routing, KV Offload & P/D Disagg.

EXPLORE INTERACTIVELY: Browse all 7 community-vetted blueprints live at https://prism.llm-d.ai

Multi-Turn Agentic & Long-Context

AGENTIC • MULTI-TURN

code-generation

P:D 376.5 : 1 (Prefill)

```
ISL:   160k (lognorm, 990k)
OSL:   425  (lognorm, 10k)
Turns: 540  |  Wait: 15s
Sys:   3k   |  TTR: 0.03
SLO:   30s TTFT, 100ms ITL
```

Specific: Massive repo context & 0.03 TTR stress KV residency across 540 turns.

⚡ Prefix Routing • KV Offloading

TRACE REPLAY • DAG

agentic-trace-replay

P:D 74.5 : 1 (92% Shared)

```
ISL:   49.1k (up to 1M!)
OSL:   659   (up to 146k)
Turns: 21.9  |  Tools: 20.5
Wait:  13.0s (54% tool wait)
Cache: 44.6k shared / call
```

Specific: Empirical OTel DAGs with 66% tool sessions & 8.8% parallel branches.

⚡ Prefix Routing • KV Offload • P/D

AGENTIC • REACT LOOP

deep-research

P:D 150.0 : 1 (Bimodal)

```
ISL:   45k  (exp, 150k max)
OSL:   300  (bimodal, 4k)
Turns: 15   |  Wait: 25s
Turn+: +3k retrieved docs
SLO:   10s TTFT, 100ms ITL
```

Specific: Short ReAct search calls culminating in a 4k-token final report.

⚡ Prefix Routing • KV Offloading

AGENTIC PATTERN: 74:1 to 376:1 P:D ratios + 13s-25s tool waits make KV prefix routing mandatory.

Interactive Chat, Reasoning & Batch

interactive-chat

INTERACTIVE • 4 TURNS

P:D 13.3 : 1

ISL 4k (lognorm) • OSL 300 • 45s human read wait • TTR 0.35
Specific: Strictest human SLOs — TTFT <= 1.0s P99 & ITL <= 50ms P99!

⚡ Prefix-Aware Routing • P/D Disaggregation

reasoning

REASONING • 1 TURN

P:D 1 : 8 (0.125 : 1)

ISL 1k (lognorm) • OSL 8k mean (exponential up to 32k CoT!)
Specific: Inverted decode-bound regime — long <think> chain, 2s TTFT P99.

⚡ P/D Disaggregation • Speculative Decoding

batch-summarization-rag

OFFLINE • RAG BATCH

P:D 30.0 : 1

ISL 15k (bimodal 1k-80k chunks) • OSL 500 • Loose 15s TTFT P90
Specific: Heavy bimodal RAG document prefill with 630s E2E batch SLO.

⚡ Prefix-Aware Routing • KV Cache Offloading

batch-synthetic-data

OFFLINE • SYNTHETIC

P:D 1 : 8 (0.125 : 1)

ISL 500 + 2k shared sys • OSL 4k (uniform up to 8k) • 2,080s E2E
Specific: Pure offline token generation maximizing tok/s/GPU throughput.

⚡ Wide Expert Parallelism • Fault-Tolerant Queueing

SPECTRUM: From 376:1 prefill-bound coding agents to 1:8 decode-bound reasoning & synthetic data.

Why Stateless Load Generators Fail Agentic AI

Legacy Flat Request Generators

DataGenerator.get_data() Independence:

- Fires pre-baked turns on a Poisson timer even before Turn N-1 finishes decoding!
- Static recorded text ignores model output length shifts & malformed tool JSON.
- Cannot express parallel sub-agent fan-out, join barriers, or context compaction.
- Naive mid-trace slicing orphans 98.6% of downstream sub-agent branches.

Inference Perf Session Engines

LocalUserSession & ReplayGraph DAGs:

- Strict causal & temporal ordering: child turns await parent completion events.
- Live output substitution splices actual streamed responses & tool_calls downstream.
- Bounded session pools (run_session_stage) keep exact N multi-agent workflows active.
- Sticky x-session-token & sub-agent stream IDs preserve real cluster prefix routing.

TWO ABSTRACTIONS: LocalUserSession for linear concurrent sweeps; ReplayGraph for causal multi-agent DAGs.

Four Agentic Generators: Choosing the Right Engine

conversation_replay

`load: concurrent (LocalUserSession)`

Inputs:
0 trace files; seeded token & wait dists.

Topology:
Linear multi-turn chain (25-57+ turns).

KV & History:
2-tier prompt + live output + 225k reset.

Best For: Replica capacity sweeps.

otel_trace_replay

`trace_session_replay (ReplayGraph DAG)`

Inputs:
OTel GenAI spans (JSON / Dir / HF).

Topology:
Inferred causal + temporal DAG.

KV & History:
[shared|output|unique] live Event splice.

Best For: Faithful prod workflow replay.

weka_trace_replay

`trace_session_replay (ReplayGraph DAG)`

Inputs:
Privacy-safe block hash_ids + subagents.

Topology:
Multi-turn + nested sub-agent streams.

KV & History:
SHA-256 corpus blocks + LCP rewind align.

Best For: Exact KV block-boundary tests.

synthetic_agentic

`trace_session_replay (ReplayGraph DAG)`

Inputs:
0 trace files; DAG topology parameters.

Topology:
Recursive dispatch sub-agent fan-out.

KV & History:
Tool catalogs + compaction sawtooth.

Best For: Pre-prod fan-out & compaction.

COMPLETE COVERAGE: Parameterize distributions, replay OTel or Weka hashes, or synthesize future topologies.

How conversation_replay Pins Multi-Turn State

2-Tier System Blueprint

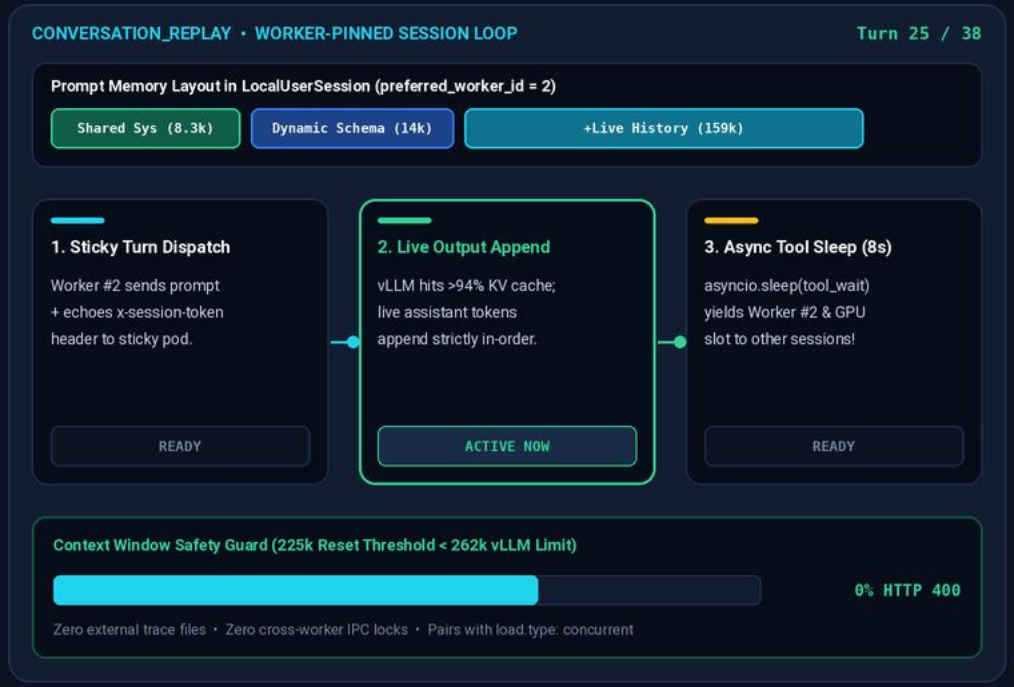
Combines fleet-wide shared_system_prompt + per-session dynamic_system_prompt.

Worker-Pinned LocalUserSession

preferred_worker_id pins all turns to 1 worker—zero cross-process IPC locks!

Non-Blocking Tool Latency

asyncio.sleep(tool_call_latency_sec) frees worker & GPU slots; echoes x-session-token.



IDEAL FOR: Sweeping N concurrent sessions (5 -> 10 -> 20 -> 40) to find replica KV saturation knees.

Compiling OTel Spans into Causal ReplayGraphs

1. Span & Message Extraction

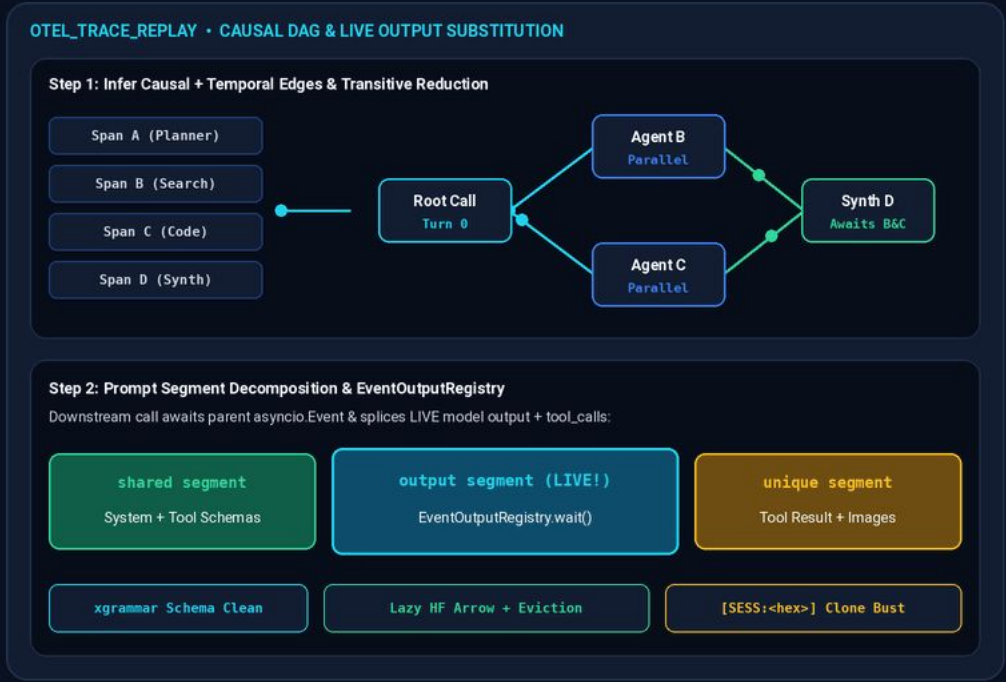
Parses gen_ai.input.messages from JSON files, directory shards, or Hugging Face.

2. Causal + Temporal DAG

Infers history-prefix & non-overlap edges, then runs transitive reduction.

3. Live Output Substitution

Splits prompts into [shared | output | unique]; EventOutputRegistry splices live text!



FIDELITY: Parallel sub-agents fire concurrently while join nodes await live upstream outputs via asyncio.Event.

Production Hardening: Tools, Multimodal & Lazy Scale

otel-prod-replay.yaml

```
load:
  type: trace_session_replay
  trace_session_replay:
    num_concurrent_sessions: 32

data:
  type: otel_trace_replay
  otel_trace_replay:
    hf_dataset_name: org/prod-agent-traces
    tool_execution_mode: force_recorded
    bad_tool_call_handling: use_recorded
    duplicate_sessions_target: 500
    request_max_retries: 2
```

Tool Schema Clean

force_recorded vs as_recorded; cleans xgrammar schemas & recovers bad JSON.

Multimodal Tool Turns

Preserves inline base64 screenshots & images inside user and tool turns.

Lazy Arrow Eviction

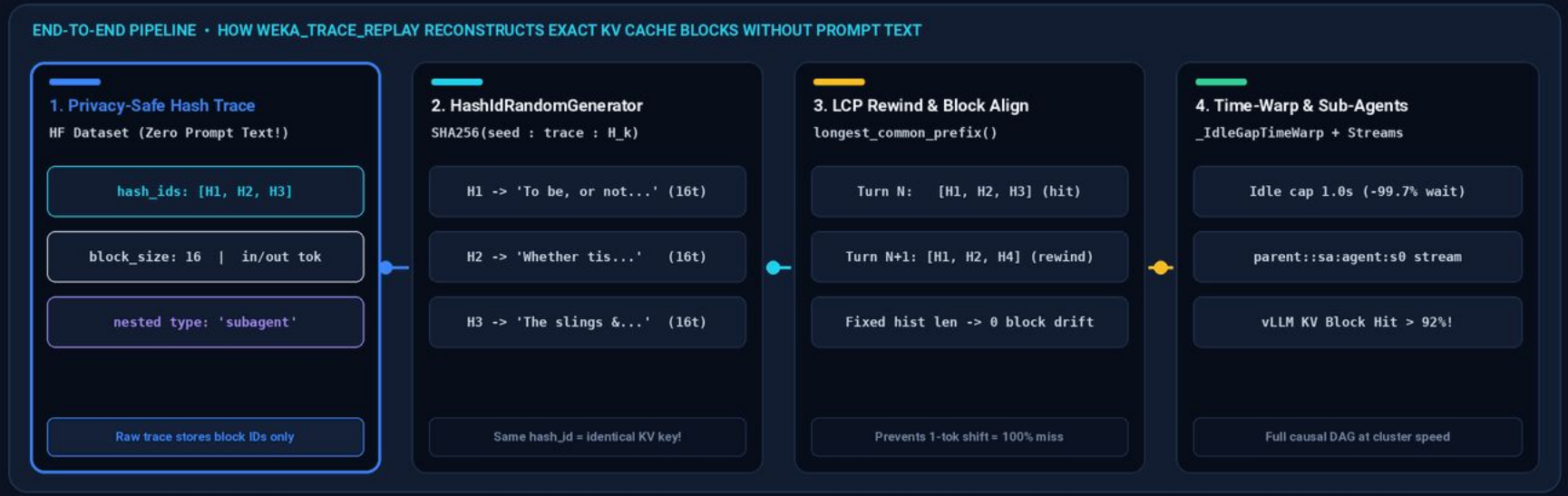
Compiles 1 session on-demand from HF Arrow & evicts finished DAGs.

Clone KV Busting

duplicate_sessions_target injects [SESS:<hex>] to stop fake KV hits!

SCALE READY: Replays 10,000+ multi-agent sessions with flat worker RAM and SessionLifecycleMetric reports.

How Inference Perf Reconstructs KV Blocks for Semi Analysis' Weka Trace



1. Zero-Text Hash IDs

Stores block_size=16/64 & hash_ids—privacy-safe.

2. Corpus Synthesis

SHA-256 slices Shakespeare -> identical KV keys.

3. Fixed History Align

Prevents 1-tok shift from busting 100% of KV!

4. Sub-Agent Streams

Caps 99.7% idle wait & routes ::sa:<agent>:s0.

KEY INSIGHT: Even a 1-token output length delta shifts every subsequent block boundary—LCP + fixed history solves it.

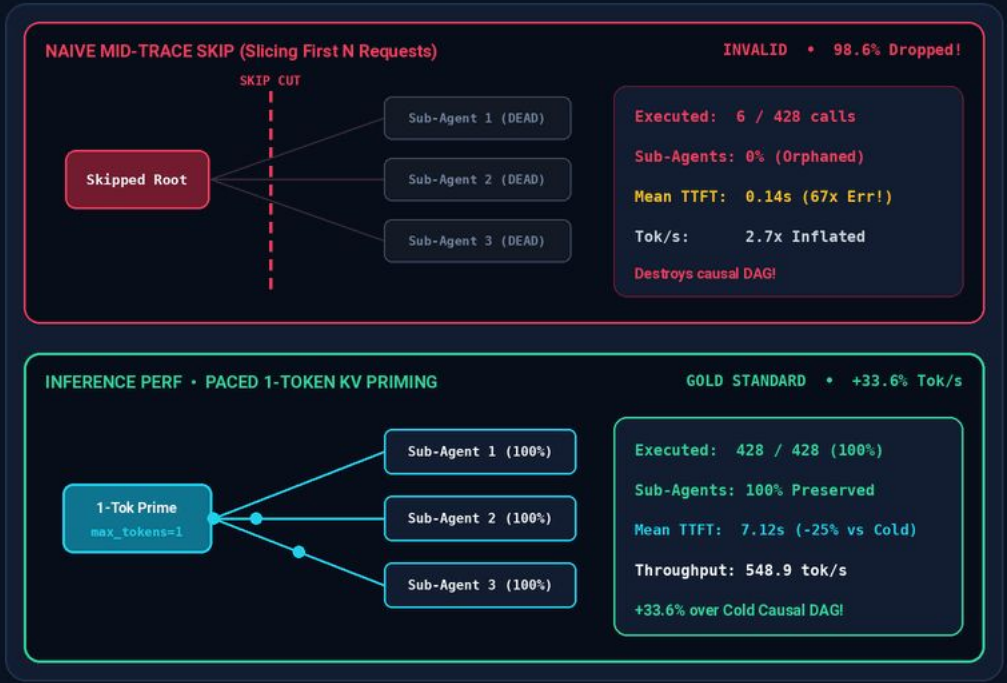
Why Mid-Trace Skipping Fails vs. 1-Token KV Priming

The Mid-Trace Skip Trap

Skipping the first N requests to 'warm up' slices root spans—orphaning 98.6% of sub-agents (6/428 left) & under-reporting TTFT by 67x!

Paced 1-Token KV Priming

Replays prefix history with max_tokens=1 to warm vLLM & EPP routers first—preserving 100% of sub-agents & boosting tok/s +33.6%!



BENCHMARK RULE: Never slice a causal agent trace mid-stream; warm prefix caches with 1-Token Priming.

Procedural Sub-Agent Fan-Out & Compaction Sawtooths

Zero-Trace Topology Sweeps

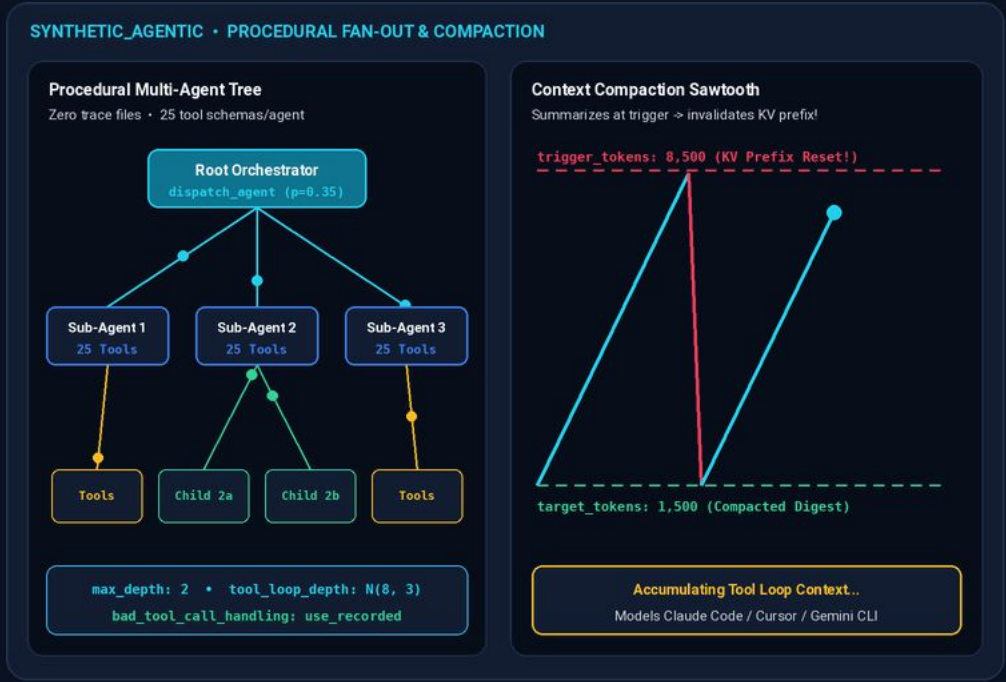
Controls tool_loop_depth, parallel_tool_calls & 25+ tool schemas per agent.

Recursive Sub-Agent Fan-Out

Forces dispatch_agent calls (fanout_probability) spawning 2-4 child agents to max_depth.

Context Compaction Sawtooth

trigger_tokens -> target_tokens models coding agents summarizing & resetting KV!



WHAT-IF ENGINE: Stress-test sub-agent storms, tool-schema caching & compaction before prod traces exist.

Beyond Raw Tok/s: TTFO, NTPOT, Session & Goodput

TTFO vs. TTFT (Reasoning)

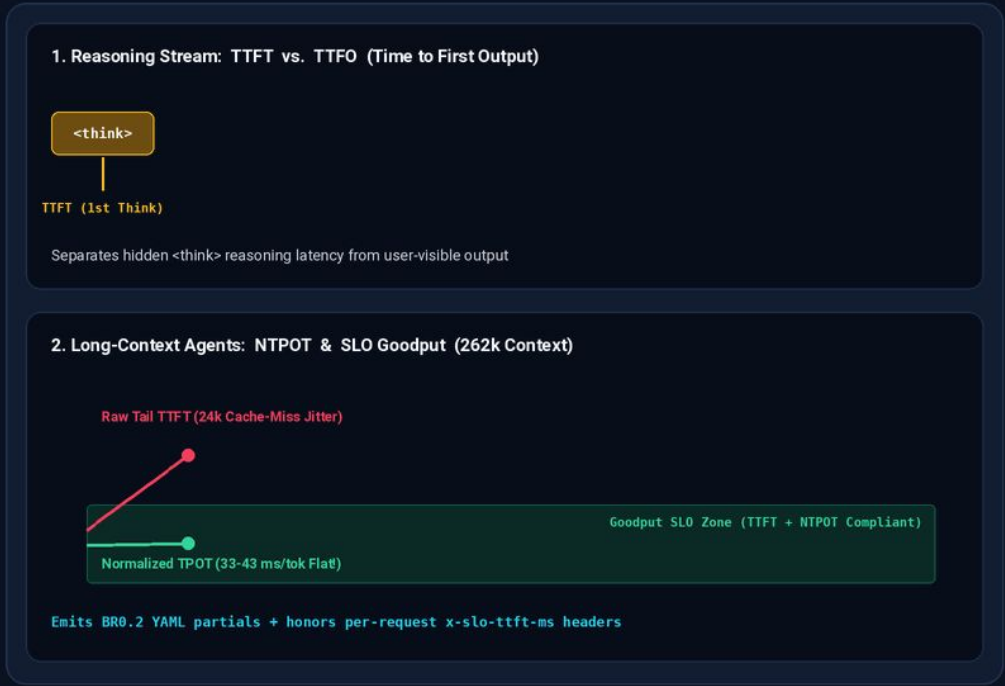
Separates <think> tokens from Time to First visible Output token (TTFO).

NTPOT & SessionLifecycleMetric

NTPOT filters 24k cache-miss spikes; tracks full DAG duration & fan-out degree.

SLO Goodput & BR0.2 Reports

Measures SLO-compliant tok/s, injects x-slo-ttft-ms headers & exports BR0.2 YAML.



DIAGNOSTIC: On 262k agents, raw P99 TTFT triggers false alarms at 25% load; NTPOT reveals the true knee.

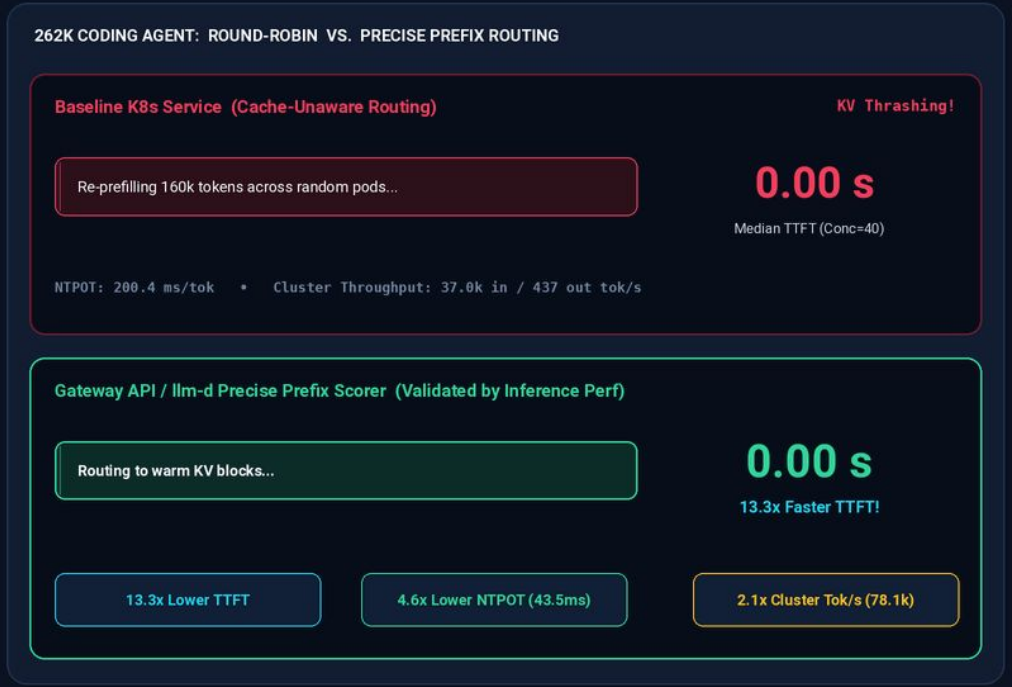
13.3x Faster Median TTFT with Prefix-Aware Routing in llm-d

262k Coding Agent Cluster

Qwen3-Coder-480B (FP8) • 64 Chips
workload-catalog/code-generation • C=40

Validated via Inference Perf

- 13.3x Lower Median TTFT
  17.39s -> 1.31s
- 4.6x Lower Median NTPOT
  200.4ms -> 43.5ms / tok
- 2.1x Cluster Throughput
  37.0k -> 78.1k input tok/s



STACK SYNERGY: Inference Perf + Gateway API Inference Extension (llm-d precise-prefix-cache-scorer).

Start Benchmarking with Inference Perf Today

Quickstart CLI

```
$ pip install inference-perf

# Run any standardized Prism Workload Catalog blueprint:
$ inference-perf --config_file \
    workload-catalog/code-generation/inference-perf.yaml

# Generate Goodput & latency comparison charts:
$ inference-perf --analyze reports/
```

Official Links & Kubernetes Community

- GitHub Repo & Docs: github.com/kubernetes-sigs/inference-perf
- Prism Workload Explorer: prism.llm-d.ai
- Kubernetes Slack: #inference-perf & #wg-serving (slack.k8s.io)
- Community Calls: Bi-weekly Kubernetes WG-Serving Meetings

Star & Contribute

kubernetes-sigs/
inference-perf

OPEN SOURCE: Apache-2.0 under Kubernetes SIGs — PRs, trace adapters & catalog blueprints welcome!

Thank You!

Questions & Discussion

Come chat with us about 250k+ agentic serving, benchmark observer bias, trace replay, or routing for agentic workloads.



Ashok Chandrasekar • **Google** • **@achandrasekar**



Navjot Singh • **Shopify** • **@Navjot10**

THANK YOU SALT LAKE CITY! • **github.com/kubernetes-sigs/inference-perf** • **prism.llm-d.ai**