SOLUTIONS OVERVIEW · TECHNICAL EDITION

KV cache acceleration for Red Hat AI Enterprise with TuringData Cache Fabric

A managed, multi-tier KV cache for the vLLM inference engine: more tokens and a faster first token from the same GPU servers.

Highlights

- 11.7× total token throughput: 5,816 → 67,833 tokens/s
- Time to first token down 94%: 36.7 s → 2.1 s
- Time per output token down 92%: 548 ms → 44 ms
- Software only, with one KV connector setting in vLLM

Technology used

- **Red Hat AI Enterprise (vLLM inference engine)**
- **TuringData Cache Fabric**

Why long-context inference wastes GPU time

The challenges with the existing vLLM technology, long-context inference spends most of its GPU time in prefill, recomputing context the server has already seen. vLLM keeps its KV cache in GPU memory, which is largely taken by the model weights, so cached context is evicted quickly. When that context returns, prefill runs again. Under concurrency, requests queue for prefill and the first token slows.

Optimize GPU investments and reduce inferencing costs

TuringData Cache Fabric gives vLLM a managed KV cache across GPU memory, host DRAM and local NVMe. Repeated context is loaded back from cache and prefill is skipped.

The Red Hat AI Enterprise platform, the models and the applications are not changed.

Key features of the solution

- **Software only.** Runs on your existing GPU servers. No new hardware for the local tiers.
- **One vLLM setting.** Registers through vLLM's KV connector. vLLM itself is not modified.
- **No application change.** Models, prompts and APIs stay as they are.
- **Mixed GPU fleets.** Different GPU models and generations in one cluster.

vLLM only vs. vLLM with TuringData Cache Fabric

	vLLM only	vLLM + TuringData Cache Fabric
Where the KV cache lives	GPU memory	**GPU memory, host DRAM and local NVMe, plus an optional external shared tier**
Repeated context	Recomputed in prefill once it has left GPU memory	**Loaded back from cache, so prefill is skipped**
As concurrency grows	Requests queue for prefill and the first token slows	**First token stays near 2s at 100K context**
Time to first token *	36.7 s (miss hit)	**2.1 s**
Total token throughput *	5,816 tokens/s	**67,833 tokens/s**
What it takes	Already in place	**One KV connector setting in vLLM. No change to models or applications.**

* Joint benchmark with Red Hat: 100K-token context, 3 concurrent requests. One server with 8× NVIDIA H200, GLM-5.2-FP8, vLLM 0.23.0, Red Hat GuideLLM 0.7.3.

How it works

Where it sits in the Red Hat AI stack

TuringData Cache Fabric attaches to vLLM. Everything above it stays as it is.

Layer	Role
Enterprise AI applications	Coding assistants, AI agents, RAG, document analysis, chat
Red Hat AI Enterprise	Enterprise AI platform: deploys, manages and scales inference
vLLM inference engine	Serves the models and exposes the KV connector interface
TuringData Cache Fabric	Container on each GPU server
GPU server	Tier 1 GPU memory · Tier 2 host DRAM · Tier 3 local NVMe

- Cache Fabric runs as a container on each GPU server and registers through vLLM's KV connector interface.
- vLLM asks how much of a request is already cached. Cached KV blocks are loaded before compute starts, and new ones are stored afterwards.

One managed cache across every tier

Cache Fabric tracks which tier holds each KV block, and moves blocks to the GPU when they are needed.

Tier	Role
Tier 1 · GPU memory	The hottest KV blocks, right beside the model
Tier 2 · Host DRAM	More cache capacity on the same server
Tier 3 · Local NVMe	The largest local tier, KV cache is striped across several drives
External Shared tier · optional	TuringData ContextCube or TuringData Platform, when GPU servers share KV cache

A KV block is written to enable tiers when it is generated. A hit returns the block to GPU memory.

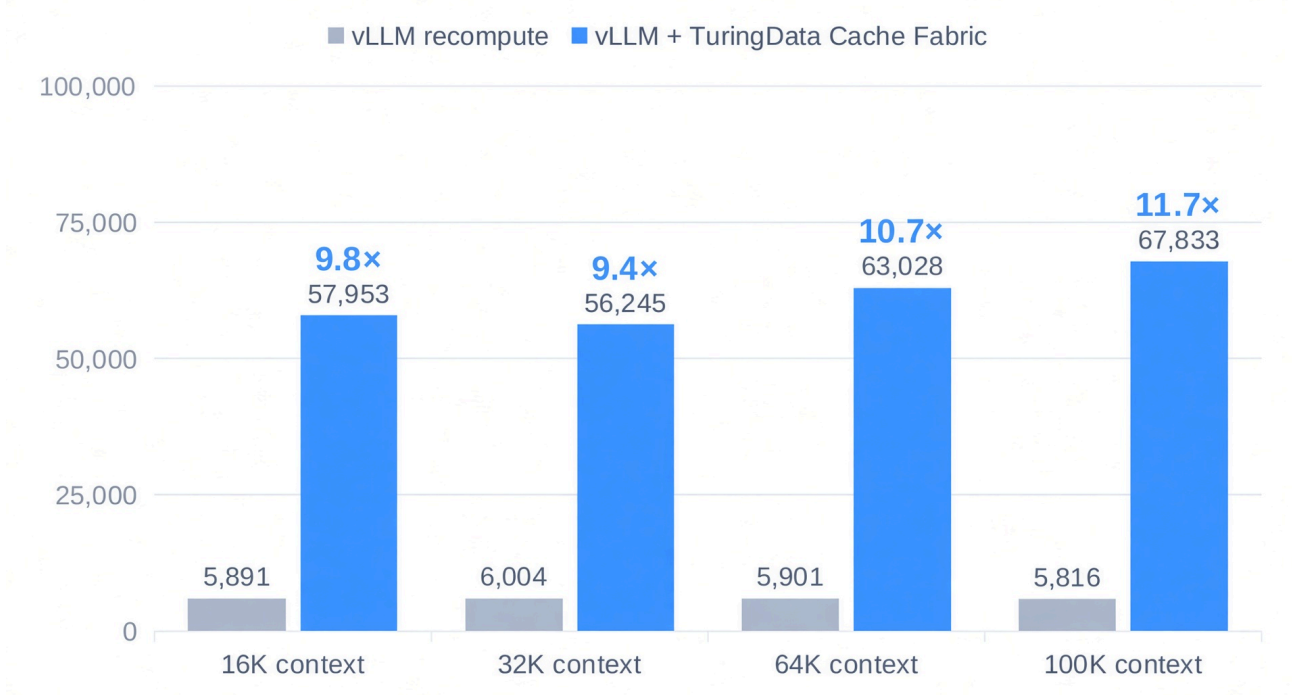
Benchmark results: joint benchmark with Red Hat

Same server, same model and same settings. The only difference between the two runs is the KV cache connector.

Test setup: one server with 8× NVIDIA H200; GLM-5.2-FP8; vLLM 0.24.0; Red Hat GuideLLM 0.7.3; 16K, 32K, 64K and 100K-token input context with 100 output tokens per request; 18 operating points from 1 to 20 concurrent requests; cache tiers used: GPU memory, host DRAM and local NVMe (no shared tier).

Total token throughput

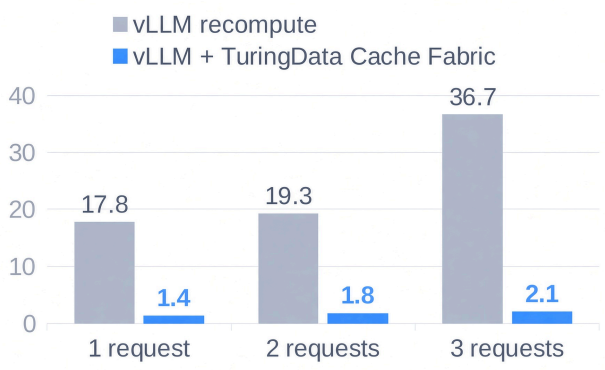
Input context	Concurrent requests	vLLM recompute	vLLM + Cache Fabric	Gain
16K tokens	20	5,891 tokens/s	57,953 tokens/s	**9.8×**
32K tokens	20	6,004 tokens/s	56,245 tokens/s	**9.4×**
64K tokens	7	5,901 tokens/s	63,028 tokens/s	**10.7×**
100K tokens	3	5,816 tokens/s	67,833 tokens/s	**11.7×**



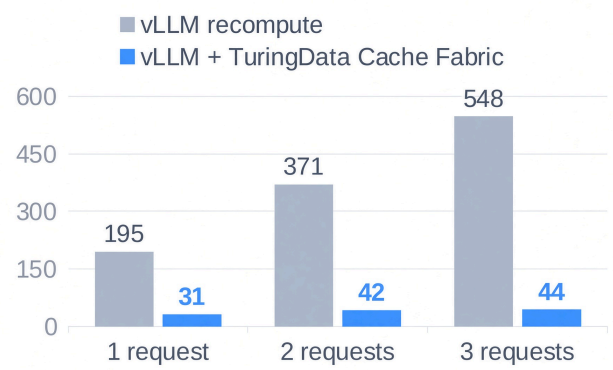
Total token throughput, tokens per second. Each pair is one measured point at the highest concurrency tested for that context length.

Latency under load, 100K-token context

Time to first token, seconds



Time per output token, milliseconds



Medians reported by GuideLLM, at 1, 2 and 3 concurrent requests. With Cache Fabric the first token arrives in 1.4 to 2.1 s; vLLM alone takes 17.8 to 36.7 s. Time per output token is 31 to 44 ms with Cache Fabric, against 195 to 548 ms.

Where it pays off most

Any workload where long context repeats from one request to the next.

Use case	What repeats
Coding assistants	The same repository and files go in with every request.
Long-context analysis	Many questions on the same long contract, report or log.
AI agents	Instructions, tools and task history repeat at every step.
RAG	Shared instructions and popular documents keep coming back.
Multi-round conversation	The full chat history is sent again on every turn.

The longer the context and the more it repeats, the bigger the gain.

Time to value and implementation approach

TuringData Cache Fabric is software that runs on the GPU servers you already own, so there is no hardware lead time. The first measured result arrives in days, and production follows in about four weeks.

Phase	Timing	What happens	Outcome
1. Discover and scope	Week 1	Environment survey, model check and agreed target metrics.	Agreed test plan
2. Proof of concept	Weeks 1–2	vLLM baseline, then Cache Fabric at the same test points; answers checked. 1 day on one server, 2 to 4 days on site.	Side-by-side test report
3. Pilot	Weeks 2–3	Enable on production inference servers for the first workload. Size the DRAM and NVMe tiers.	First workload live
4. Production rollout	Week 4	Roll out across GPU servers. Add the optional shared tier where needed.	Solution in production

Indicative timeline for a software-only rollout. Actual timing depends on the model and on how ready the environment is.

Interested in finding out more?

Start with a proof of concept: a vLLM baseline against TuringData Cache Fabric at the same test points.

- **Customers Provide:** a technical owner, one dedicated GPU server and access to it.
- **TuringData Provides:** engineers to deploy and run the tests, and the test report.

Contact TuringData at marketing@turingdata.io or visit www.turingdata.io.

About TuringData

TuringData Pte. Ltd. is a Singapore-based AI storage company. It helps enterprises and AI cloud providers get more from every GPU.

STORAGE FOR AI TRAINING
TuringData Platform
TuringData Flash

KV CACHE FOR AI INFERENCE
TuringData Cache Fabric
TuringData ContextCube

WEBSITE
www.turingdata.io

EMAIL
marketing@turingdata.io

ADDRESS
18 Cross Street, #02-101
Singapore 048423