tensormesh

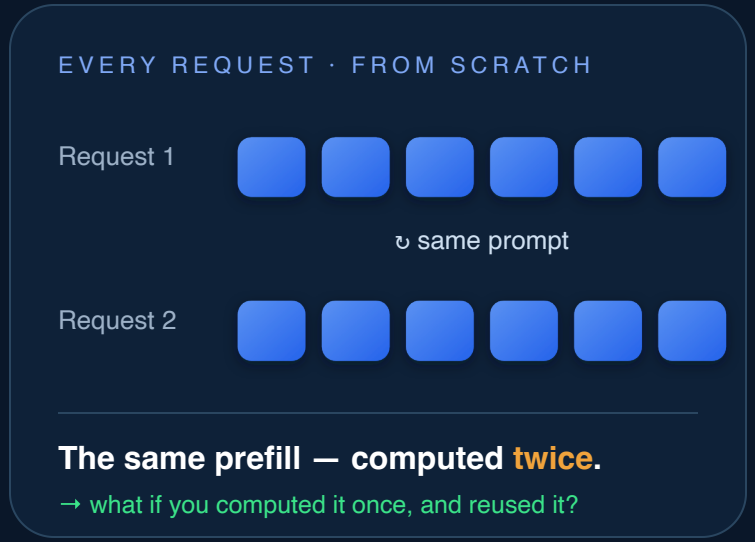
PIE & AI BANGKOK · EMBEDDED AI · 27 JUN 2026

Your GPUs are doing the same work twice

KV-cache reuse for faster, cheaper LLM inference — on hardware you control.

Dai Tran · Solution Architect, Tensormesh

Built on LMCache (open source) · backed by NVIDIA, AMD & CoreWeave



WHO'S BEHIND THIS

tensormesh

The team that turned KV-cache reuse into a field

The researchers

Founded by the people who pioneered KV-cache optimization — from **UChicago, UC Berkeley & CMU**.

LMCache · open source

We build and maintain LMCache, the de-facto open-source KV-caching layer. **~10k ★** · 200+ contributors.

Industry-backed

Investment from **NVIDIA, AMD & CoreWeave** — the companies that set the economics of AI.

Everything labeled **“LMCache”** in this talk is open source — clone it and run it tonight.

tensormesh

On hardware you own, you pay to recompute the same context

If you serve LLMs on-prem or at the edge, you're already paying for this — it just isn't a line on the invoice.

Inference is the recurring cost

Training is a one-time event for a few labs. Inference is the bill that arrives every single day.

Prefill dominates

Every request re-encodes the same system prompts, documents, chat history and tool output — from scratch.

Wasted prefill = lost concurrency

On a fixed node you can't autoscale away. GPU spent re-reading context is the ceiling on users you can serve.

— THE BLIND SPOT

tensormesh

Everyone watches GPU utilization. Almost nobody watches KV-cache hit rate.

Utilization tells you the GPU is **busy**. It doesn't tell you whether it's busy doing **new work** — or redoing work it already did.

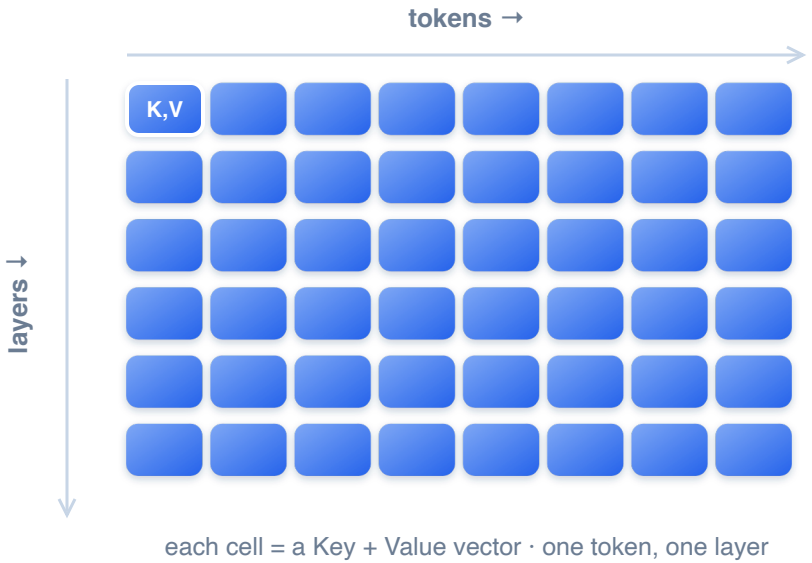
Hit rate is the number that decides your **real cost per token**. If you're not measuring it, you can't see the waste.

tensormesh

What is the KV cache?

As the model reads your prompt — the **prefill** step — every layer computes a **Key** and **Value** for every token. That's the model's working memory of the context, and generation attends back to it.

It's big: roughly **tokens × layers × heads × 2** — tens of MB to tens of GB for long context. And today it's discarded the moment the request ends.



tensormesh

Treat the KV cache as data — not disposable scratch

Compute it once. Then reuse it — across requests, across instances, across storage tiers.

Whoever reuses this layer best wins on cost and concurrency.

WHERE STANDARD CACHING STOPS

tensormesh

Prefix caching helps — until your context isn't a prefix

Your engine already caches the shared **prefix** — great for a fixed system prompt. But it only fires when the reusable text sits at the very front, identically, every time:

- **RAG breaks it** — retrieved chunks change order every query, so there's no shared prefix to hit.
- **Agents break it** — context compaction and repetitive tool output aren't prefix-stable.

The ceiling

On real shared-context workloads, prefix caching alone tops out around ~40–50% reuse.

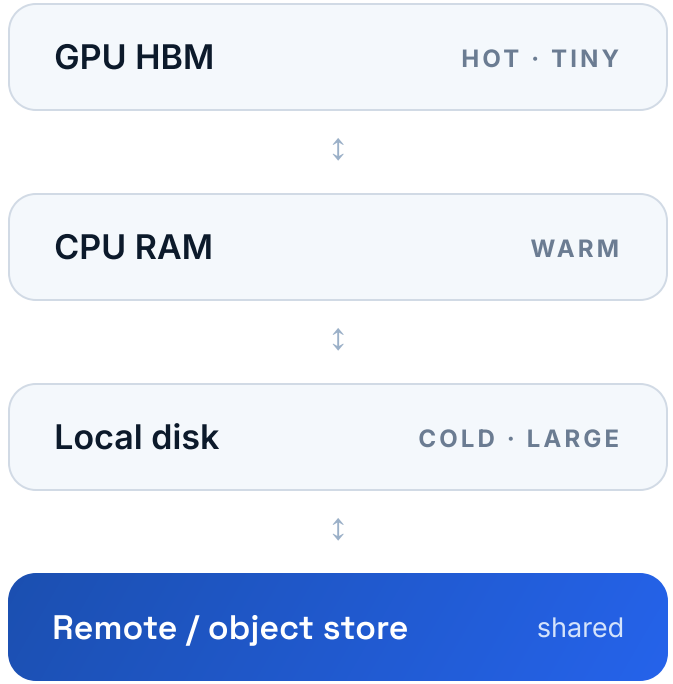
The reuse you actually want is in the middle of the prompt — where prefix caching can't reach.

tensormesh

LMCache — give the cache somewhere to live

Step one is simply: **stop deleting it.** LMCache gives the KV cache a memory hierarchy, and lets it survive across requests, sessions — even process restarts.

- ✓ Offload KV beyond GPU → **CPU RAM → local disk → remote / object store**
- ✓ Decoupled connector — works with **vLLM** (most mature) and **SGLang**; TensorRT-LLM coming soon
- ✓ Transparent reuse: the engine just finds the cache already there



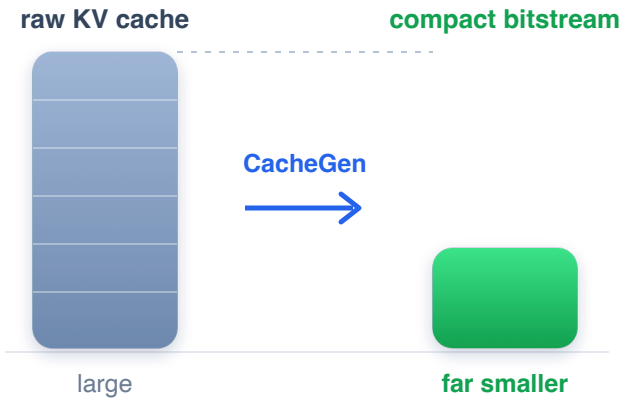
tensormesh

CacheGen — make the cache cheap to move and store

KV tensors are big — and if you're going to tier them or share them between nodes, size decides whether offload actually pays off.

CacheGen encodes the cache into **compact bitstreams** from its distributional structure, with **negligible decode overhead**. Smaller cache → cheaper to store, faster to move.

On edge: no NVLink, limited disk — compression is what makes offload practical.



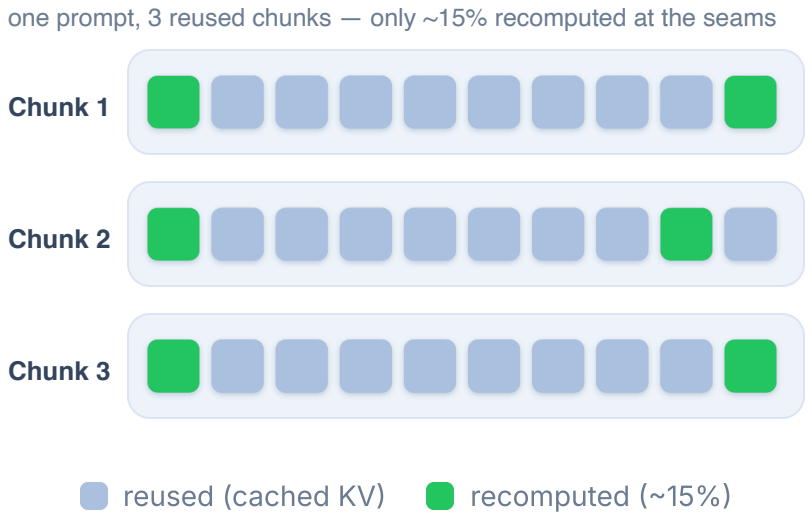
tensormesh

CacheBlend — reuse any chunk, not just the prefix

Reuse cached KV for **any** repeated chunk, anywhere in the prompt. The catch: blindly concatenating cached chunks loses the **cross-attention** between them → right facts, wrong relationships.

The insight: only a **small fraction (~15%)** of tokens carry that cross-chunk attention. Recompute just those, on every layer → **full-prefill quality.**

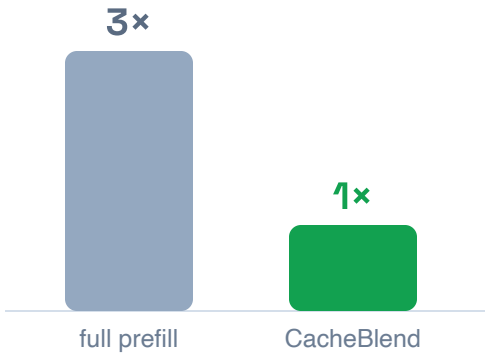
Pushes effective reuse toward **~90–95%** on shared-context workloads · peer-reviewed at **ACM EuroSys 2025** · open source.



tensormesh

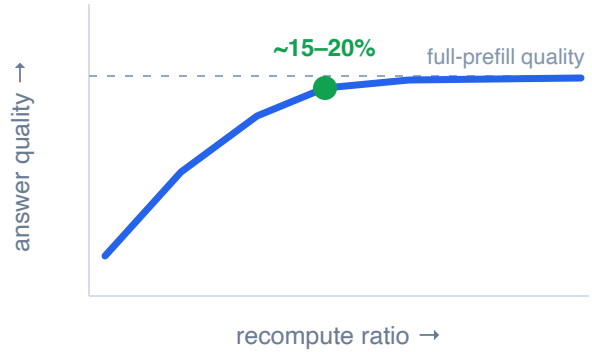
Published, measured — and honest about the conditions

≈ 3× faster TTFT at equal quality



Setup: 2WikiMQA · Llama-70B · 2×A40 (CacheBlend, EuroSys 2025)

Quality holds at a small recompute ratio



Quality matches full prefill at a low recompute fraction.

Real, but **workload-dependent** — the win scales with how much context you reuse, your bandwidth, and the model. Measure your workload first.

WHERE IT PAYS MOST

tensormesh

The workloads where reuse is basically free money

Reuse only helps when there's something to reuse. These three are everywhere in production — and they're exactly where prefix caching falls short.

RAG & long documents

The same documents and system prompts, retrieved again and again — reorderable, so non-prefix reuse is the unlock.

Agents & coding loops

Long, repetitive context and tool output across many steps. Agentic AI lives or dies on context reuse.

Long multi-turn chat

Every turn re-reads the whole conversation — reuse keeps latency flat as it grows.

FROM CAPABILITY TO PRODUCTION

tensormesh

The science is open. Running it well is the work.

Going from “the repo supports it” to “it's serving my users reliably — in my environment” is real engineering.

Configure & scale

Tiered offload sizing · blending config · multi-node cache sharing.

Operate & observe

Observability incl. hit rate · on-prem deploy, upgrades & lifecycle · enterprise support.

Tuned to you

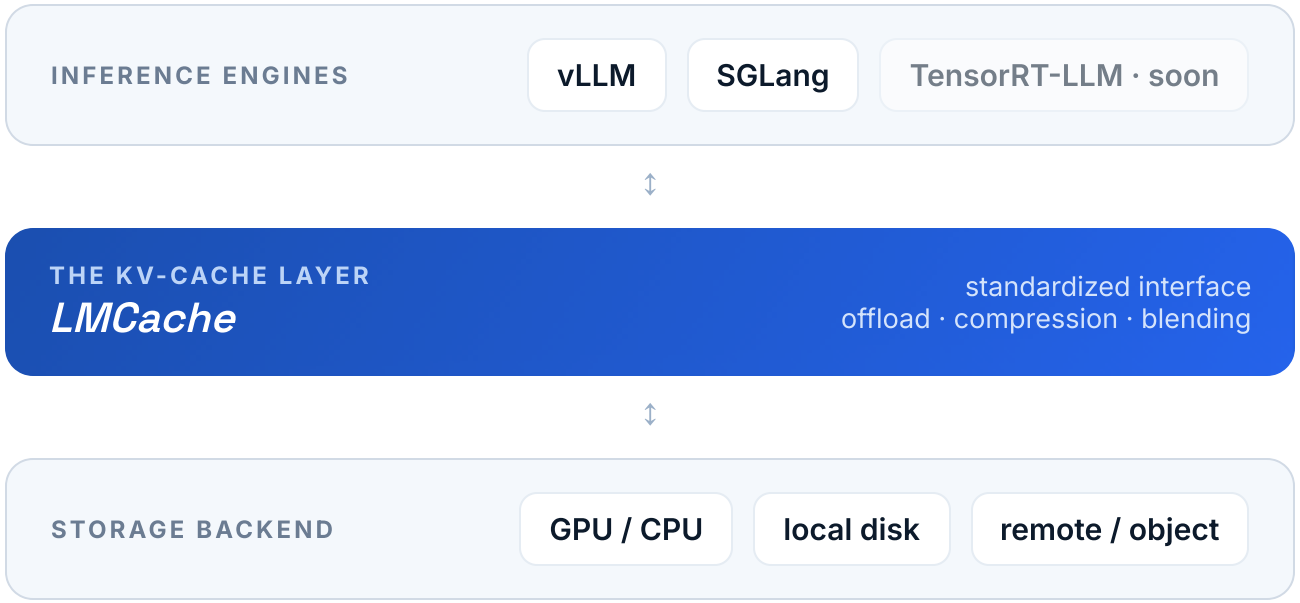
Your models, your hardware, your traffic — with SLAs.

That productization — by the team that wrote the science — **is Tensormesh.**

ARCHITECTURE

tensormesh

A layer that drops into the stack you already run



- ✓ **Decoupled** — independent of any single engine or storage vendor
- ✓ **Future-proof** — new KV techniques ship without touching your engine
- ✓ **Drop-in** — slots into the stack you already run; no re-platform

TWO WAYS TO RUN IT

tensormesh

Managed, or entirely in your environment

Tensormesh Inference

Hosted · cloud-native SaaS

✓ GA today

- OpenAI / Anthropic-compatible API — point your client at it
- Deploy open-weight models on public GPUs in minutes
- Zero infrastructure overhead; unified cost dashboard

Tensormesh Operator

In your cluster · on-prem / VPC / BYOC

Beta expected early July 2026

- The same engine, inside your environment — you keep the GPUs and the data
- The sovereignty answer for regulated / on-prem buyers
- **Design-partner POC available now**

Same engine — the only difference is **where it runs and who operates it.**

SELF-HOSTED & SOVEREIGN

tensormesh

Run it entirely inside your own walls

For most teams here, the reason to self-host isn't one thing — it's some mix of these four. Tensormesh Operator is built for all of them.

Data residency & security

Inference data never leaves your VPC or data center — no third party in the path. Aligns with **PDPA, PIPL, APPI, DPDP**.

Cost at scale

When GPU is a top line item, reuse turns into real budget — on capacity you've already bought.

Latency & control

Serve in-region, on your network, on your terms — no shared-tenant surprises.

Your own models

Fine-tuned or proprietary weights stay in-house — bring your own vLLM, keep your IP.

IS THIS WORTH IT FOR YOU?

tensormesh

A 30-second self-check

The more of these are true, the more KV-cache reuse is probably your single biggest cost lever.

- You run models on your own GPUs (not just calling APIs)
- Your workload has shared context — RAG, agents, long multi-turn
- GPU is a top cost line — or you're capacity / latency-bound
- You have a team that can run a quick POC

3 of 4? Let's measure your real hit rate together — that's a conversation worth having.

IT'S REAL

tensormesh

Already in production — and trusted upstream

≈4×

GMI Cloud

LLM performance boost, SSD-augmented KV cache

Cohere RAG

via CoreWeave

cheaper inference, enterprise data stays private

AWS HyperPod

SageMaker

tiered KV cache as an L2 backend under high concurrency

Open-source traction: **~10k ★ · 200+ contributors** · backed by NVIDIA, AMD & CoreWeave

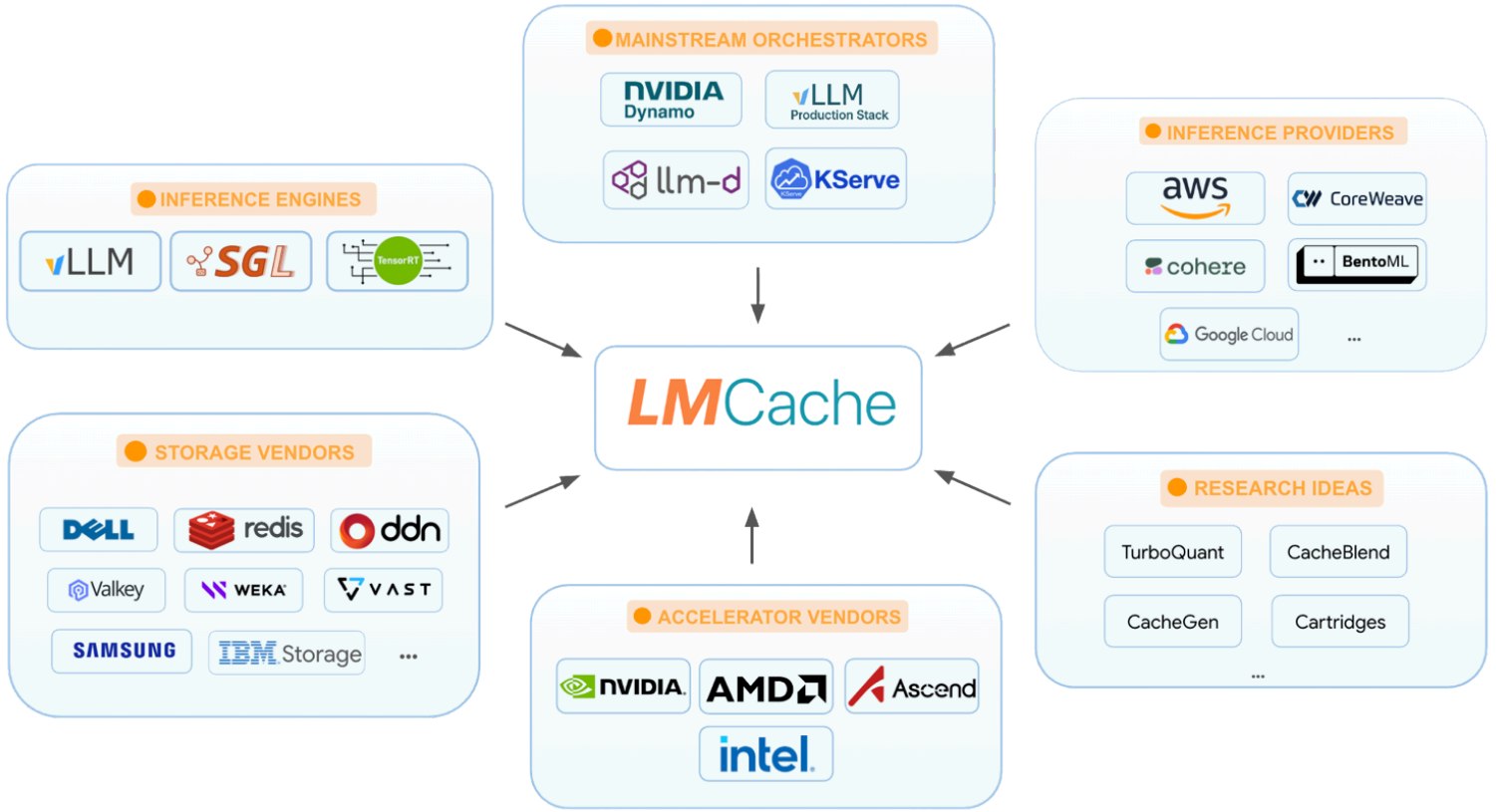
Sources: GMI Cloud, Cohere / CoreWeave and AWS published benchmarks. Results vary by workload.

ECOSYSTEM

tensormesh

An integral layer in the LLM inference ecosystem

Community-driven integration across serving engines, inference frameworks, hardware vendors, storage systems and infrastructure providers.



COMMUNITY

Built by a fast-growing community

Developers, researchers, industry adopters and partners building the next generation of efficient LLM inference.

WORK WITH US

tensormesh

Start with a design-partner POC

Low commitment, real numbers — measured on your workload and your hardware — on-prem or in the cloud.

- **Bring a real workload** — a representative slice of your traffic
- **We measure** KV-cache hit rate, cost-per-token and TTFT — before and after
- **~30-day POC** in your environment — decide on your numbers, not our slides

For partners

Regional co-sell for Southeast Asia

Enablement, margin / referral model, joint deal registration — plus upstream & research collaboration on LMCache.

BEFORE YOU GO

tensormesh

Tell us about your setup

forms.gle/WeHuv5nRtGs5anHf6

A 60-second survey — mostly about **self-hosting**. It helps us, and we'll send you what's most relevant back:

- Do you self-host on your own GPUs — and how big is that footprint?
- Do you track your KV-cache hit rate today?
- What would push you to self-host — cost, latency, residency, your own model?

Thank you — questions?

tensormesh.ai
dai@tensormesh.ai