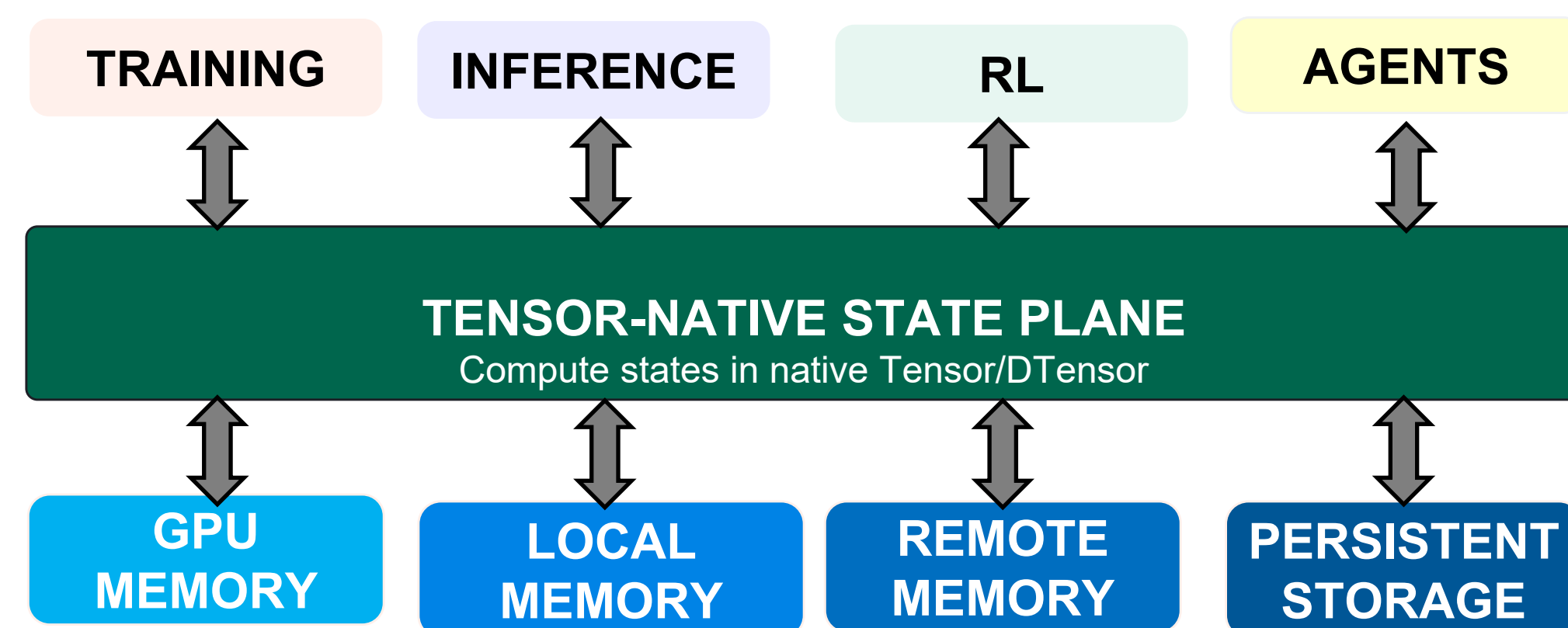


AI Compute State and Physical Infrastructure

AI compute states, such as model weights and KV cache, are native tensors, but their identities are currently tightly coupled with physical infrastructure. As models grow larger and infrastructure becomes distributed, elastic, heterogeneous, compute states need to move across GPU and topology boundaries.

Vision: Tensor-Native State Plane

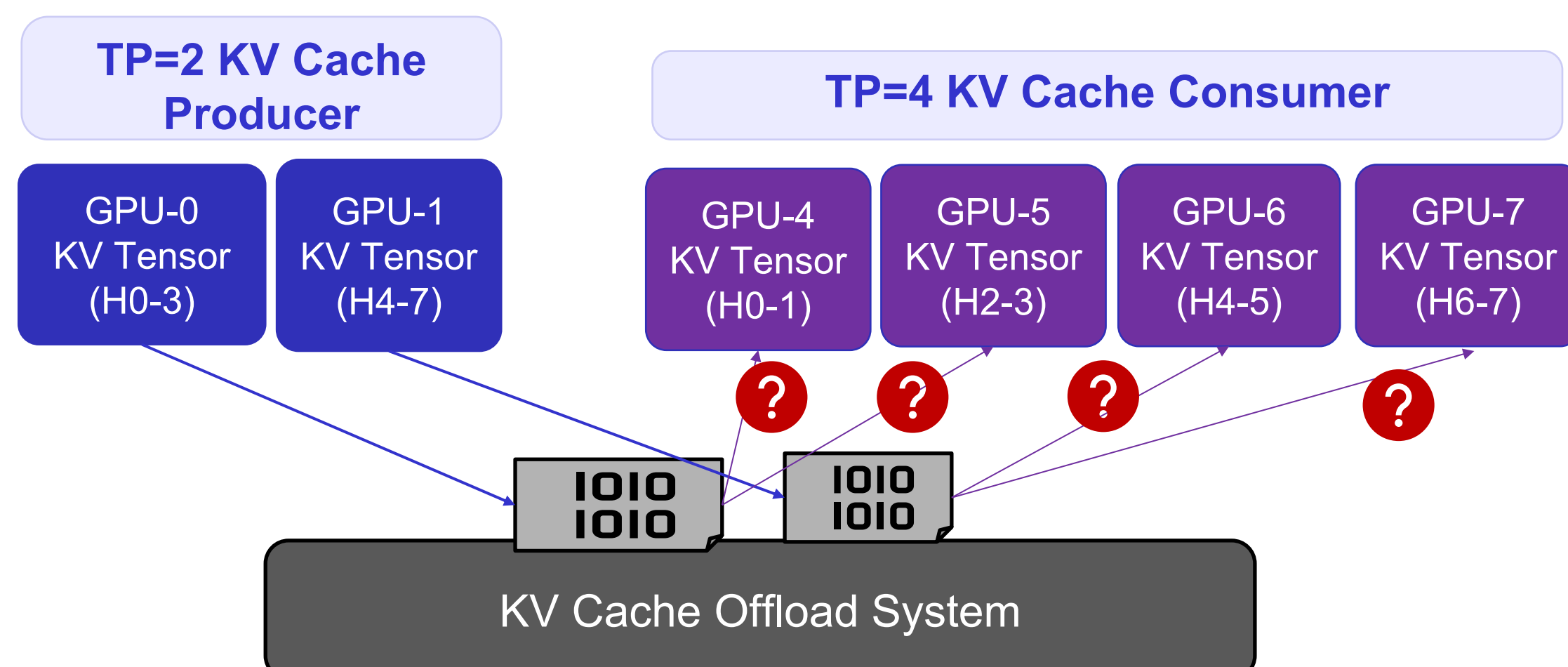
A tensor-native state plane maintains a logical view of AI compute state across GPU memory, pooled host memory, remote memory, and persistent storage.



Compute systems operate on Tensors and DTensors, while the state plane manages data residency, sharding, data transfer and tiering.

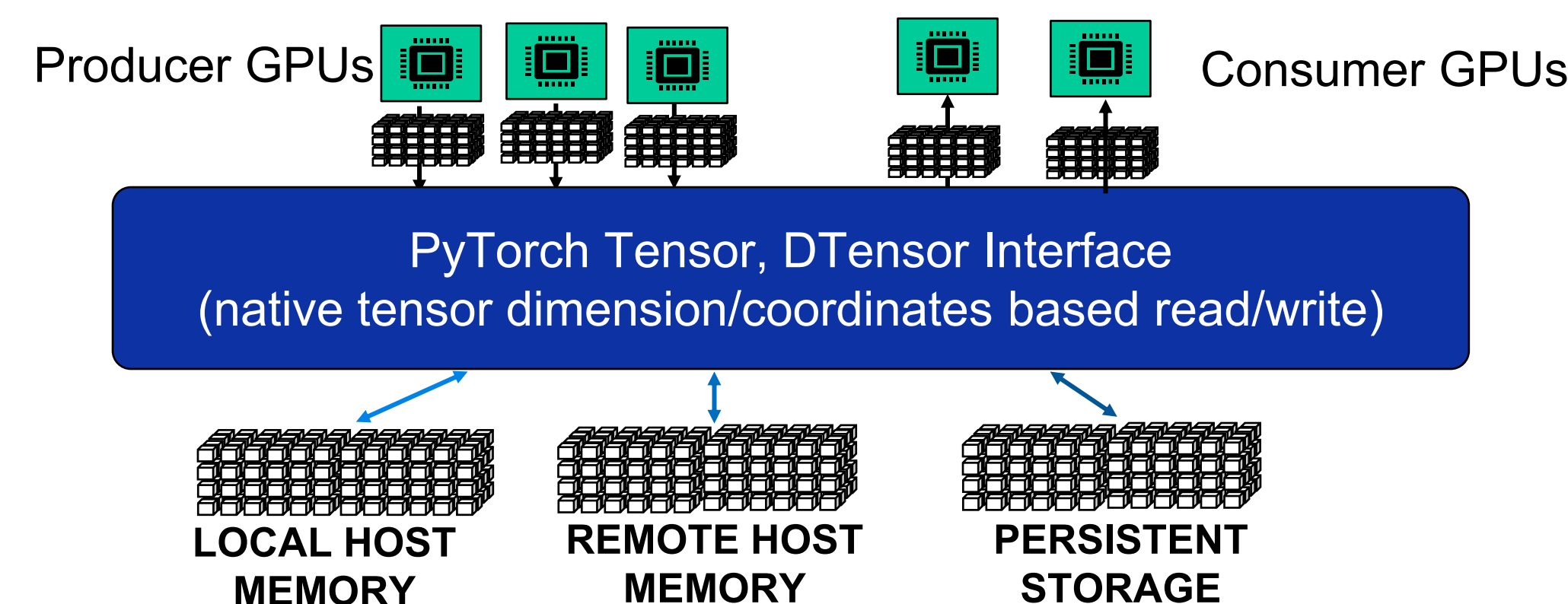
KV Cache Is a Concrete Compute State Example

KV cache offload is a proof point for the need of a tensor-native state plane. KV caches are large, distributed, 5-dimensional tensors [*layer, K/V, tokens, heads, head_dim*]. Block-based KV cache offload treating KV tensor as binary data coupled with producer physical layout, would cause a false cache miss if consumers have different TP from producers.



Tensor KV Cache Concept

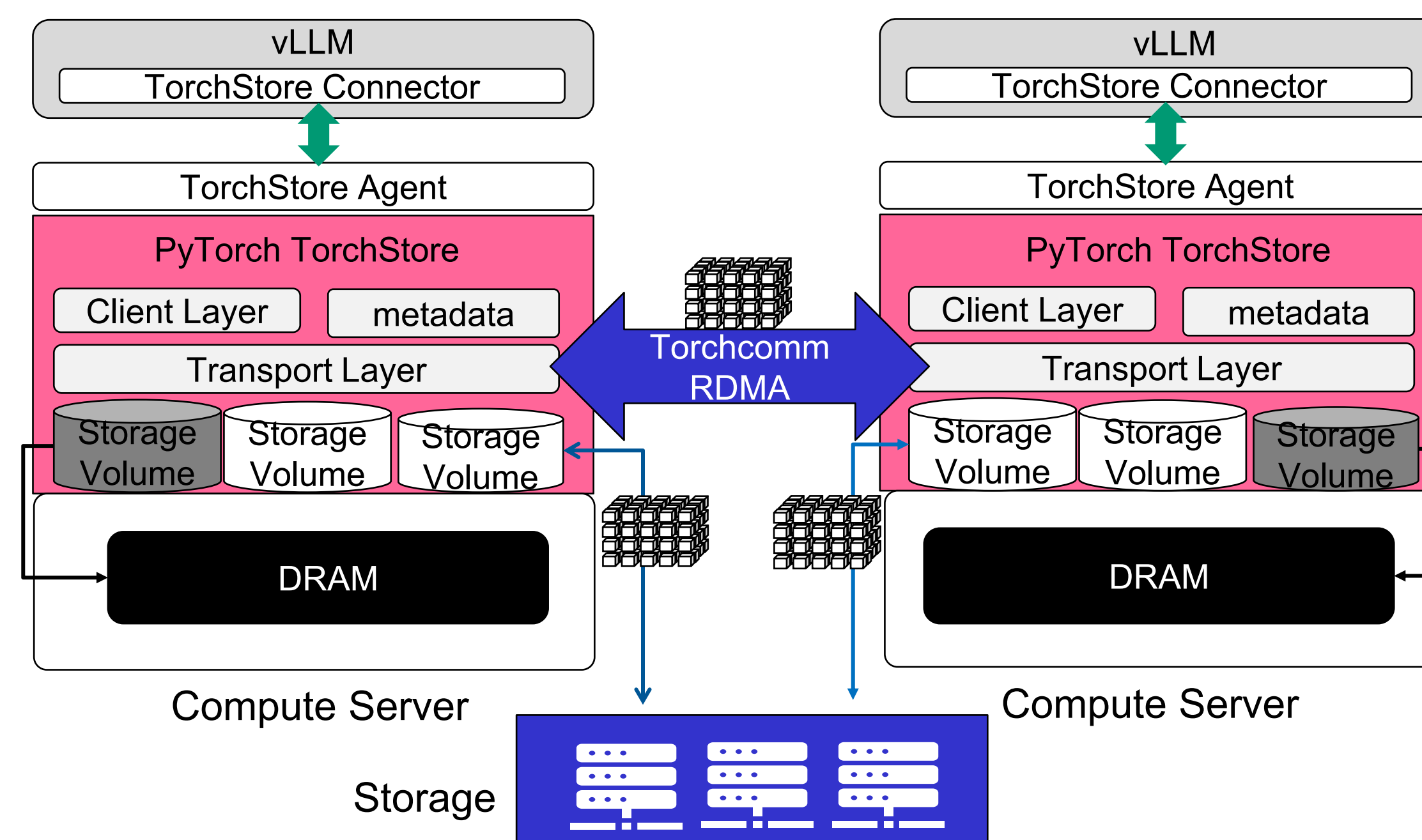
- KV cache tensors are stored as PyTorch DTensor in tensor-native store across host memory and persistent storage
- KV cache tensors are consumed in DTensor slices, independent of physical deployment



PyTorch TorchStore-based Architecture

PyTorch TorchStore Provides a Solid Foundation for Tensor KV Cache

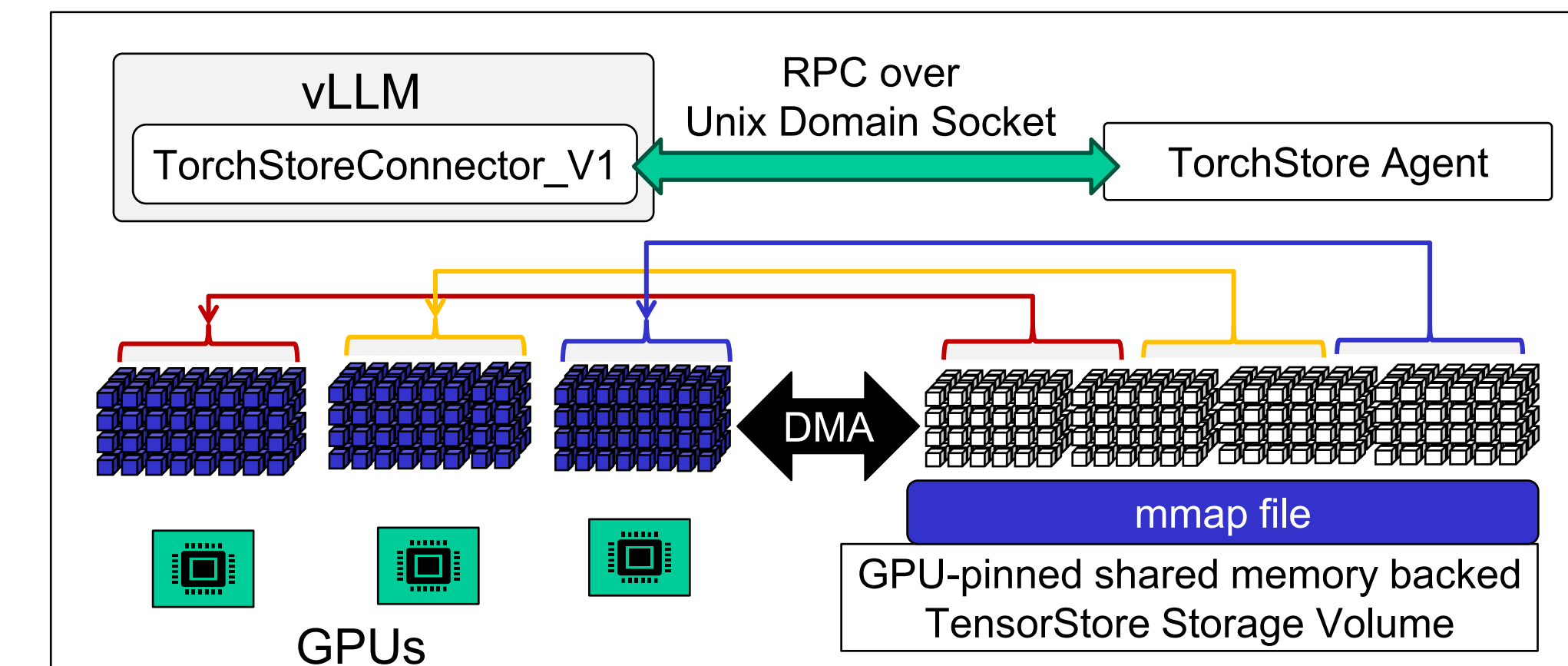
- PyTorch native Tensor and DTensor interface, tensor resharding
- Local host memory and pooled remote host memory backed storage volume
- RDMA sharing of DTensor between compute servers



Tensor KV Cache Extends PyTorch TorchStore

- vLLM connector to integrate with vLLM
- TorchStore agent to interact with vLLM connector
- Persistent storage volume in TorchStore for persistent tier of KV Cache
- Orchestration of tensor data between hosts and storage tiers

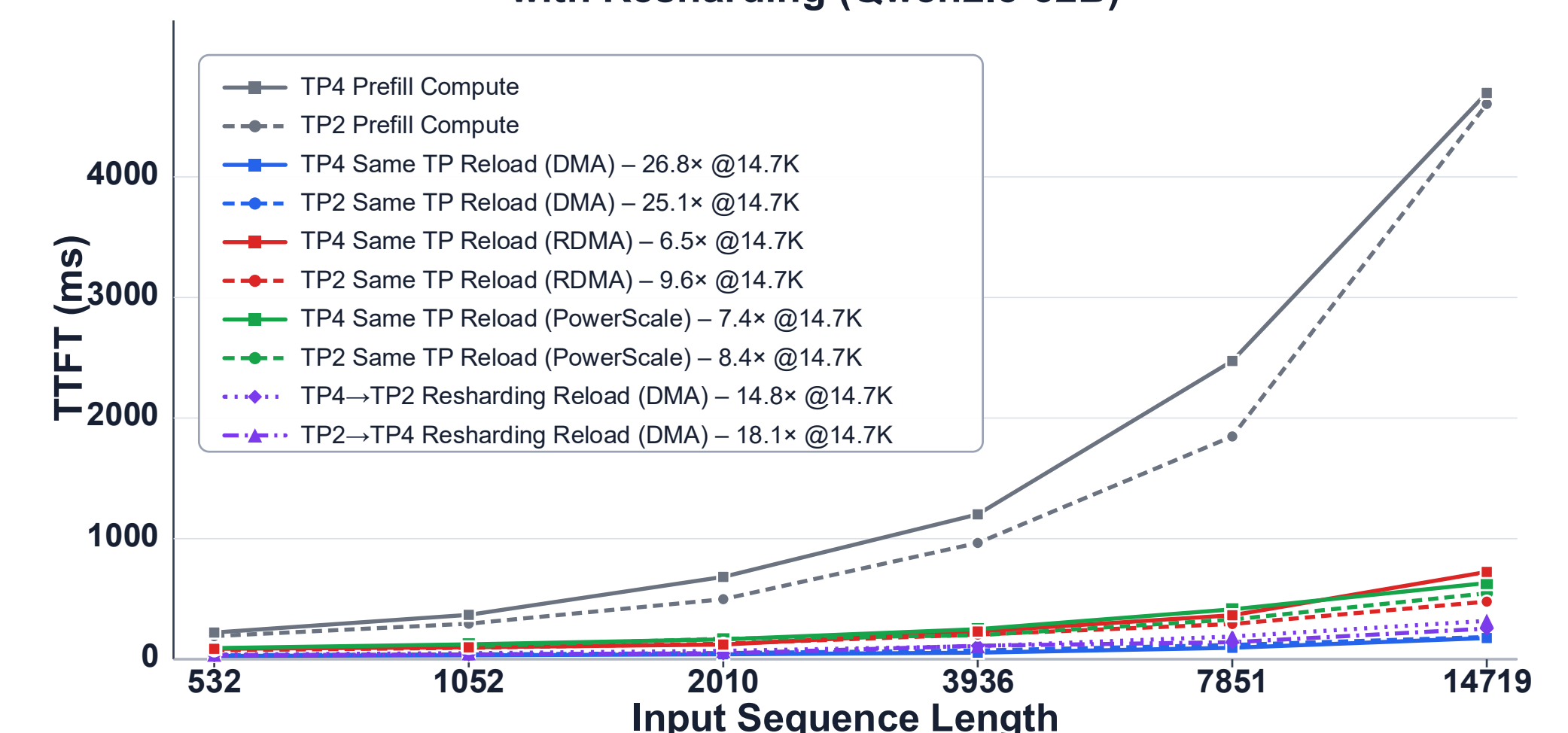
Layout Independent KV Cache Load



- KV Cache tensors are stored as native PyTorch DTensors, not as binary blocks
- KV Cache consumers load their desired slice of KV Cache tensor based on its own TP configuration, independent of producer's configuration

Preliminary Performance Results

TTFT: Prefill vs DMA/RDMA/Storage Reloading with Resharding (Qwen2.5-32B)



Future Works

- Support latest attention mechanisms such hybrid attention
- Metadata system hardening
- KV Cache tiering optimization
- Support SGLang and other LLM engines
- Extend to model weights and other AI compute states