Everpure KVA Meets FlashBlade//EXA: Enterprise KV Caching at Neocloud Scale

What happens when you put enterprise-grade KV caching on the fastest data platform on the planet? You get to serve the most demanding inference workloads on earth without the silent compromises that kill user trust, like shorter contexts, smaller models, or aggressive eviction.

That is the short version. Let's get into the details.

A quick refresher on Everpure KVA

Everpure Key-Value Accelerator (KVA) is our persistent, distributed KV cache. It plugs into vLLM and NVIDIA Dynamo, keeps the attention state of your prompts on FlashBlade//S™, and gives it back to the GPU on demand. The point is simple. Prefill is expensive; it gets dramatically more expensive as prompts get longer, and recomputing the same prefixes over and over shows up as latency for chat users and as drag on agents burning through tool calls. KVA speeds both of them up.

Up to this point, the story has been an *enterprise* story. A bank running RAG over a few hundred GBs of policy documents. An insurer serving internal copilots to ten thousand employees. A support org that wants prefix hits on its top 500 tickets. Those workloads live comfortably on FlashBlade//S, and the time-to-first-token (TTFT) improvements are already dramatic.

Enterprise was where KVA earned its reputation. It is not where it stops.

The neocloud problem

If you're running inference for a few thousand concurrent users across a handful of models, FlashBlade//S has you covered. If you're running inference for a few *million* concurrent users across a zoo of models, with context windows blown out to 1M+ tokens, reasoning traces eating tens of thousands of tokens per turn, and agentic workloads that revisit the same system prompts a billion times a day, the math changes.

At that scale, your KV cache is not a side concern. It's a first-class tier of memory that needs to sit somewhere fast, somewhere big, and somewhere every GPU in the fleet can reach.

"Somewhere fast" is the key phrase. You cannot serve neocloud inference off of storage that taps out at a few hundred GB/s. The GPUs will idle, the tail latencies

will blow up, and you'll end up doing the one thing nobody wants to do, which is silently degrading the user experience (shorter contexts, smaller models, aggressive eviction) to make the numbers work.

Why FlashBlade//EXA is built for this

FlashBlade//EXA™ is the answer to that problem. It's the evolution of the FlashBlade® family into a platform purpose-built for neoclouds and AI factories. The headline numbers are unambiguous:

- **10+ TB/s read performance** in a single namespace means a GPU fleet of tens of thousands can pull KV tensors without queuing behind each other.
- Writes that scale up to **50% of read performance,** so ingest keeps pace with inference demand rather than becoming the bottleneck.
- Support for **tens of thousands of GPUs** from a single system.
- **Exabyte-scale** capacity.
- **>20X more files per namespace** than the prior generation, which matters a lot when your KV cache is a haystack of billions of tensor files.

The architectural move that makes this possible is disaggregation. FlashBlade//EXA splits metadata and data into two independently scalable clusters. The metadata core is built on the same proven Purity//FB stack that runs in thousands of FlashBlade deployments. The data nodes are x86 servers running a thin Purity//DN OS, talking NFSv4.1 over RDMA, stitched together with NVMe-oF over RoCEv2 on NVIDIA Spectrum-4 switches.

You do not need to understand the plumbing to understand the payoff. Data and metadata scale on their own curves, nothing sits in the critical path that does not need to be there, and every blade in the metadata core has access to every byte in the system. It's the cleanest way anyone has built this.

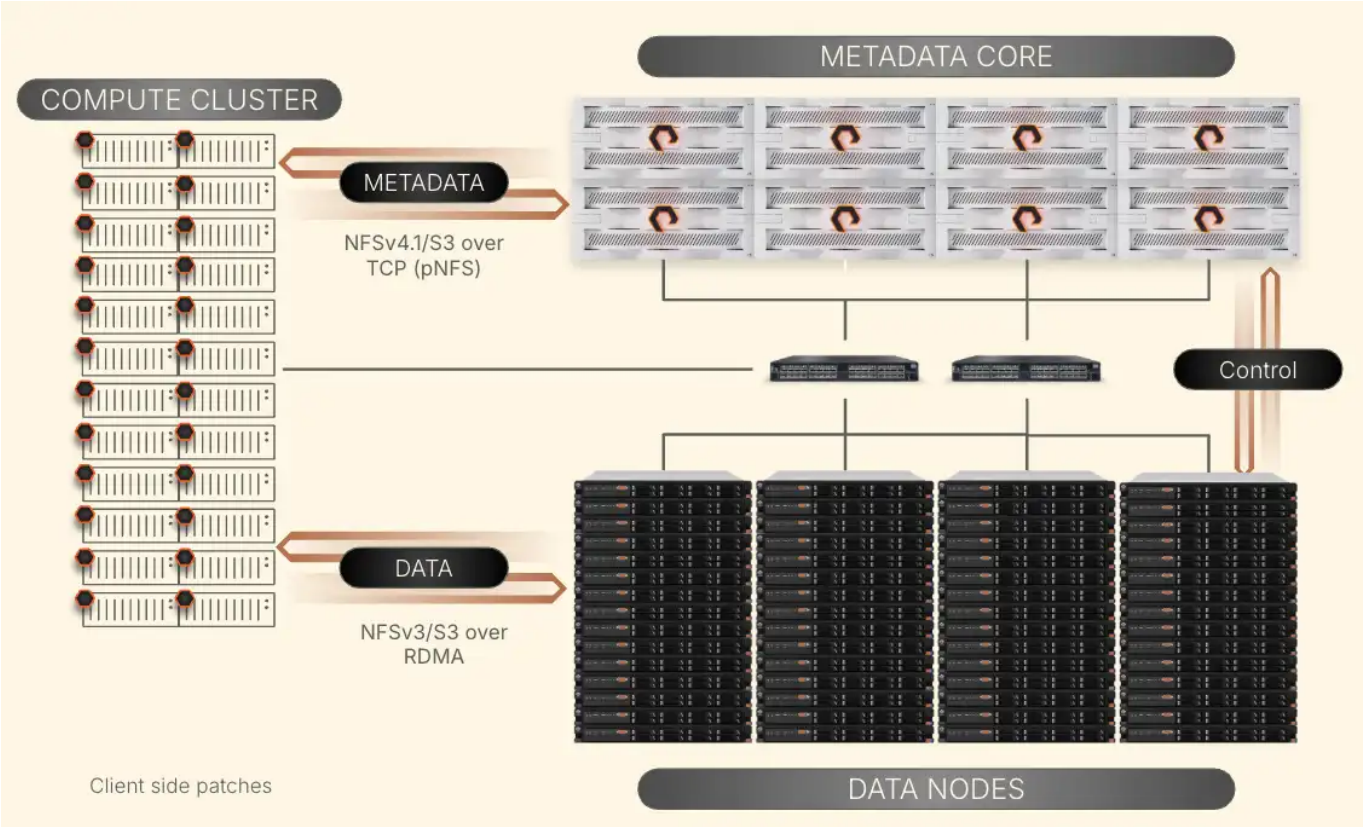

Figure 1: FlashBlade//EXA architecture.

What changes for Everpure KVA

Everpure KVA on FlashBlade//EXA is not a port. The cache logic is the same. The API integrations are the same. The failure modes, the semantics, and the operational model are all the same. You should expect the same TTFT and throughput characteristics that KVA delivers on FlashBlade//S, just on a much, much bigger canvas.

What changes is the addressable workload.

A neocloud running inference for a public API product can now keep a globally shared prefix cache that is measured in petabytes, fronting a GPU fleet that is measured in tens of thousands of accelerators, and serve it all out of a single namespace at 10+ TB/s. The cache becomes, effectively, free memory that every GPU in the cluster can hit. The HBM bottleneck disappears. The “do I cache this user’s context or evict it” tradeoff disappears. You stop nerfing context lengths to fit into memory budgets because memory is no longer the budget.

This is the part that matters for the inference era. The industry has spent the last two years obsessing over training infrastructure. The next two are about *inference economics*, which are dominated by how well you can reuse work. KV caching is the primary lever. The question is whether the storage layer underneath it can keep up with a GPU fleet that doubles every 18 months.

FlashBlade//EXA can keep up. Can your current infrastructure say the same?

Where this goes

The integration is complete. KVA is validated on FlashBlade//EXA, and the same reference designs that apply to enterprise KVA deployments extend upward into FlashBlade//EXA territory without architectural surprises. If you’re a hyperscaler or neocloud waiting for a production-grade answer to KV cache offload, this is it.

The inference era is not going to be won by whoever has the most GPUs. It’s going to be won by whoever wastes the fewest tokens. Everpure KVA plus FlashBlade//EXA is how you stop wasting them.