Graid Technology Inc.

Break the Memory Wall and Unlock Faster LLM Inference with SupremeRAID™ AE

Up to 3.26x faster time to first token and 3.20x faster query round completion versus no KV cache offload under the tested memory-pressure workload.

April 2026

Table of Contents

Executive Summary

As long contexts, prompt reuse, and high concurrency compete for limited GPU memory, inference performance can degrade as serving stacks recompute context or evict useful KV cache state. The key-value (KV) cache matters because it allows the serving engine to reuse attention state instead of rebuilding it. When the useful KV working set no longer fits comfortably in GPU memory, the infrastructure needs another tier that is both close enough for inference and resilient enough for production use.

This paper evaluates SupremeRAID™ AE-protected local NVMe RAID5 as a KV cache offload tier for vLLM and LMCache, asking a practical platform question: when GPU memory is constrained, does a protected local NVMe tier improve inference behavior compared to not having a KV cache offload?

Under the tested memory-pressure workload, SupremeRAID™ AE RAID5 reduced query round mean time to first token (TTFT) from 29.451 s to 9.024 s, a 69.4% reduction and a 3.26x speedup versus the no-offload baseline. It also reduced total query round time from 95.077 s to 29.740 s, a 68.7% reduction and a 3.20x speedup.

For comparison, Linux MD RAID5 using default settings measured 36.639 s query round mean TTFT and 117.748 s query round time in the same tested workload.

Together, these results show that KV cache offload depends on the performance of the storage backend, not only on adding local capacity or RAID protection. In this validation, the GPU-accelerated SupremeRAID™ AE path was the protected RAID5 backend that preserved nearly 100% of local NVMe performance to improve the KV cache offload path while adding RAID5 protection.

Key Findings

- SupremeRAID™ AE RAID5 local NVMe KV cache offload substantially lowered both TTFT and query round time in the tested workload.
- RAID protection only delivers value if the storage path preserves NVMe performance. Linux MD RAID5 default settings did not preserve that performance in this tested workload. GPU-accelerated SupremeRAID™ AE was the only protected RAID5 backend that improved KV cache offload performance.
- Local NVMe keeps the KV cache tier close to the inference server without an external storage network, while SupremeRAID™ AE RAID5 protection maintains availability through a single-drive failure, delivering a meaningful resilience improvement over an unprotected local SSD target.
- The results are specific to the tested model, workload, hardware, and software configuration and should not be treated as a universal ranking across all KV cache backends or deployments.

Audience and Scope

This paper is written for AI infrastructure architects, platform engineering teams, storage architects, solution engineers, and technical marketing reviewers evaluating inference infrastructure under GPU memory pressure.

The comparison focuses on a practical deployment decision: whether a protected local NVMe cache tier is useful when the reusable KV working set no longer fits comfortably in GPU memory:

Scenario	Storage Backend	Runtime Notes	Role
Baseline	None	No KV cache offload	Memory-pressure baseline
Baseline	Local Linux MD RAID5	Default setting.	Software RAID5 default comparison
Comparison	Local SupremeRAID™ AE RAID5	Shared GPU resources, NVIDIA Multi-Process Service, RAID state optimal	Main performance comparison

The paper does not attempt to rank every possible KV cache backend. It focuses on a more practical platform question: whether protected local NVMe is a useful offload tier when GPU memory is insufficient. The Linux MD RAID5 result provides a default-setting reference, not a full MD RAID5 tuning study.

Why KV Cache Placement Matters

Transformer inference uses KV cache data to avoid repeating work across long or recurring context. That reuse matters for long-document question answering, repeated prompt prefixes, multi-turn sessions, and other workloads where the same context appears across requests.

GPU memory is the fastest place to keep active KV data, but it is also shared with model weights, runtime state, and concurrent requests. When the working set grows beyond available GPU memory, the serving stack needs another place to hold useful KV data.

KV cache offload provides that second tier. Instead of treating cache eviction and recomputation as unavoidable, the serving stack can move KV data to host memory, local disk, or external storage. In this paper, the offload target is local NVMe protected by SupremeRAID™ AE RAID5.

Why Local NVMe Needs Resilience

Local NVMe is attractive because it keeps cache capacity close to the inference server. It avoids an external storage network for local cache access and integrates cleanly with LMCache through a local filesystem backend.

The challenge is failure behavior. A single unprotected SSD can remove the cache tier from service. The KV cache may be reconstructable, but losing the local tier can still increase recomputation, extend warm-up time, or make response behavior less predictable.

RAID5 protection makes the local cache tier more operationally useful. The goal is not to turn KV cache into permanent application data. The goal is to keep the local offload target available through a single-drive degraded condition so inference can continue to benefit from KV cache offload.

At the same time, protection cannot come at the cost of making the cache tier too slow for the inference path. KV cache offload needs a storage backend that can protect local NVMe while preserving enough performance for reuse to be faster than recomputation or inefficient request handling.

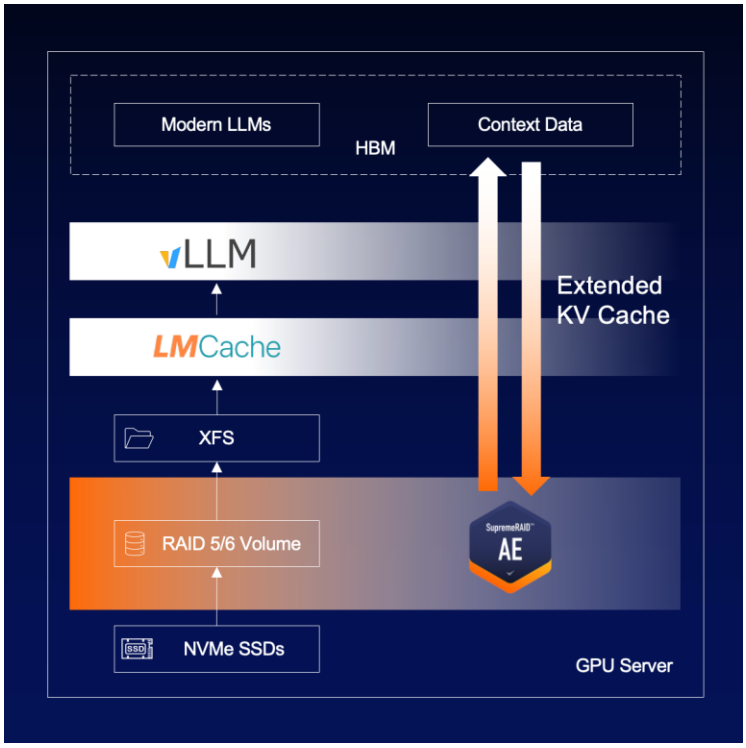
SupremeRAID™ AE as the Local KV Cache Tier

SupremeRAID™ AE provides a protected local NVMe RAID target for AI infrastructure. In this architecture, SupremeRAID™ AE exposes the NVMe devices as a RAID5 filesystem target, and LMCache uses that filesystem path as its local disk backend.

Layer	Components	Role
Inference	vLLM, Qwen3-235B-A22B-Instruct-2507, four NVIDIA H200 GPUs	Serves the model and executes requests
KV cache	LMCache connector and local filesystem backend	Moves reusable KV data outside constrained GPU memory
Storage	SupremeRAID™ AE RAID5 over local NVMe	Provides the protected local cache target
Runtime	NVIDIA Multi-Process Service, vLLM container, Graid service	Allows vLLM and Graid to share selected GPU resources in this validation

The design keeps the cache tier local to the server while adding RAID5 protection. That combination is the core of the architecture: fast local access for the offload path, paired with a storage layer that can remain available when the RAID set is degraded.

Reference Architecture



Conceptual reference architecture for SupremeRAID™ AE KV cache offload

This reference architecture focuses on the local KV cache offload path built from vLLM, LMCache, XFS, and SupremeRAID™ AE. vLLM provides the serving layer and exposes an OpenAI-compatible API. LMCache extends KV cache capacity beyond accelerator memory and uses a local filesystem target to persist and reuse KV data. XFS provides that local filesystem layer, while SupremeRAID™ AE presents the protected local NVMe RAID5 volume underneath it.

The model layer in this design is intentionally generic. The diagram is meant to show the integration pattern rather than tie the architecture to the specific validation model used later in this paper. The architectural idea is straightforward: vLLM serves requests, LMCache manages KV cache extension and reuse, XFS provides the local filesystem interface, and SupremeRAID™ AE supplies the protected NVMe tier underneath that stack.

Validation Methodology

The validation tested whether SupremeRAID™ AE RAID5 local KV cache offload improved inference behavior compared with no KV cache offload and Linux MD RAID5 using default settings.

Test Environment

Category	Value
CPU	2 x Intel Xeon Platinum 8562Y+
System memory	1 TB
GPUs	4 x NVIDIA H200
Inference engine	vLLM
Container image	vllm/vllm-openai:v0.18.1
Model	Qwen3-235B-A22B-Instruct-2507
Serving GPU pinning	CUDA_VISIBLE_DEVICES=0,1,2,3
KV cache software	LMCache
LMCache backend	Local filesystem
Host KV cache path	/mnt/graid/lmcache
Container KV cache path	/lmcache
Model host path	/mnt/graid/models
Container model path	/workspace
Graid GPU note	In the example setup, Graid runs on GPU 0
Operating System	Ubuntu 24.04.4 LTS with kernel 6.8.0-110-generic
Shared GPU mechanism	NVIDIA MPS

Storage Configuration

Item	Value
Protected storage layer	SupremeRAID™ AE for the main comparison; Linux MD RAID5 for the default-setting comparison
RAID level	RAID5
Host cache path	/mnt/graid/lmcache
Container cache path	/lmcache
LMCache local disk setting	file:///lmcache
O_DIRECT	Enabled for the local RAID5 filesystem comparisons
NVMe drive	8 x 3.84 TB Phison Pascari X200P U.2 NVMe SSDs, PCIe Gen5 x4 drives operating at PCIe Gen4 x4 due to backplane limitation
Filesystem and mount options	XFS

The local filesystem backend used the following LMCache configuration:

```
chunk_size: 256
local_disk: "file:///lmcache"
max_local_disk_size: 64
extra_config: {'use_odirect': True}
```

Benchmark Workload

The benchmark used LMCache long doc qa against the vLLM OpenAI-compatible API endpoint. In plain terms, it simulates long-context question answering against 100 documents of length 10000 with up to 100 inflight requests.

Parameter	Value
Benchmark	LMCache long_doc_qa
Documents	100
Document length	10000
Output length	100
Repeat count	2
Repeat mode	tile
Maximum inflight requests	100
API endpoint	http://127.0.0.1:8000/v1
Primary metric 1	Query round mean TTFT
Primary metric 2	Query round time

The serving command enabled LMCache through the vLLM KV transfer configuration:

```
vllm serve /workspace/Qwen3-235B-A22B-Instruct-2507/ \
  --served-model-name Qwen3-235B-A22B-Instruct-2507 \
  --tensor-parallel-size 4 \
  --gpu-memory-utilization 0.9086 \
```

```
  --kv-transfer-config '{"kv_connector":"LMCacheConnectorV1", "kv_role":"kv_both"}'
```

The benchmark was run from the host:

```
python3 /mnt/graid/LMCache/benchmarks/long_doc_qa/long_doc_qa.py \
  --model "Qwen3-235B-A22B-Instruct-2507" \
  --base-url http://127.0.0.1:8000/v1 \
  --num-documents 100 \
  --document-length 10000 \
  --output-len 100 \
  --repeat-count 2 \
  --repeat-mode tile \
  --max-inflight-requests 100
```

Metric Definitions

Metric	Definition
Query round mean TTFT	Mean time to first token across responses in the benchmark query round reported by long_doc_qa. TTFT captures how quickly the service begins generating output after requests enter the query phase.
Query round time	Elapsed wall-clock time for the benchmark query round to complete across the configured workload. It is a round-completion metric, not a tokens-per-second throughput metric.

In this test, the query round used 100 documents, document length 10000, output length 100, repeat count 2, repeat mode tile, and up to 100 inflight requests.

Test Controls and Limitations

The primary scenarios used the same model, inference engine, container image, tensor-parallel size, GPU memory utilization setting, serving GPU count, GPU pinning, and LMCache long_doc_qa workload parameters. The main variable was KV cache placement: no offload versus LMCache local filesystem offload to local RAID5 storage.

Control Area	Treatment
Model and serving stack	Qwen3-235B-A22B-Instruct-2507 served by vLLM in vllm/vllm-openai:v0.18.1
Serving configuration	Tensor parallel size 4 and GPU memory utilization 0.9086
Serving GPU placement	Four NVIDIA H200 GPUs pinned with CUDA_VISIBLE_DEVICES=0,1,2,3
Benchmark workload	Same long_doc_qa document count, document length, output length, repeat count, repeat mode, and maximum inflight requests
KV cache variable	No KV cache offload for baseline; LMCache local filesystem offload for Linux MD RAID5 and SupremeRAID™ AE RAID5 comparisons

Scope and Applicability

These results reflect the tested workload and configuration. They characterize one specific scenario: GPU-memory-constrained inference with a large MoE model. They should not be read as a universal performance ranking across all LLMs, backends, concurrency profiles, RAID implementations, or storage layouts. For production planning, validate against the intended model, prompt lengths, reuse profile, and hardware.

Benchmark Results

In the tested workload, SupremeRAID™ AE RAID5 reduced both query round mean TTFT and total query round time relative to the no-offload baseline. The same workload also measured Linux MD RAID5 using default settings.

Scenario	Storage Backend	Runtime Notes	Query Round Mean TTFT (s)	Query Round Time (s)
Baseline	None	No KV cache offload	29.451	95.077
Baseline	Local Linux MD RAID5	Default setting, RAID state optimal	36.639	117.748
Comparison	Local SupremeRAID™ AE RAID5	Shared GPU resources, MPS, RAID state optimal	9.024	29.740

Interpreting the Results

Relative to the no-offload baseline, SupremeRAID™ AE RAID5 in the optimal state reduced query round mean TTFT by 69.4% and query round time by 68.7%, equivalent to a 3.26x TTFT speedup and a 3.20x query round time speedup.

Relative to Linux MD RAID5 using default settings, SupremeRAID™ AE RAID5 in the optimal state reduced query round mean TTFT by 75.4% and query round time by 74.7%, equivalent to a 4.06x TTFT speedup and a 3.96x query round time speedup.

This comparison is important because it shows that protected RAID5 capacity alone does not guarantee an effective KV cache tier. The storage backend must preserve enough NVMe performance for the offload path to help the inference workload.

The reduction and speedup values use the following calculations:

Metric	Calculation
TTFT reduction	(baseline TTFT - comparison TTFT) / baseline TTFT
TTFT speedup	baseline TTFT / comparison TTFT
Query round time reduction	(baseline round time - comparison round time) / baseline round time
Query round time speedup	baseline round time / comparison round time

In the tested memory-pressure workload, the optimal-state result shows the practical value of a local KV cache tier. With no KV cache offload, query round mean TTFT was 29.451 s and query round time was 95.077 s. With Linux MD RAID5 using default settings, TTFT was 36.639 s and query round time was 117.748 s. With SupremeRAID™ AE RAID5 local offload in the optimal state, TTFT dropped to 9.024 s and query round time dropped to 29.740 s.

The Linux MD RAID5 result should be read as a default-setting comparison point for this tested configuration, not as a statement about every possible MD RAID5 tuning profile. Its role in the validation is to make the storage-backend requirement explicit: for KV cache offload, data protection must be paired with a storage path fast enough to preserve the practical benefit of local NVMe.

Deployment Considerations

KV cache offload is most useful when GPU memory is insufficient for the active and reusable KV working set. Teams should identify the conditions creating that pressure: model size, prompt length, concurrency, cache reuse pattern, and GPU memory allocation.

The storage tier should be sized around the target model and request pattern. Long-context and high-concurrency workloads can create different read, write, and reuse behavior, so cache chunk size, storage media, RAID layout, filesystem, and container runtime all matter.

For production planning, teams should repeat the validation using the intended model, prompt lengths, reuse profile, concurrency, GPU memory setting, storage layout, and software versions.

Conclusion

GPU memory is a finite resource, and as model size, context length, and concurrency grow, the gap between what fits in accelerator memory and what inference requires will only widen. KV cache offload is one practical answer to that gap, but only if the offload tier is fast enough to matter and reliable enough to trust in production.

This validation demonstrated that SupremeRAID™ AE RAID5 meets both bars. Under a memory-pressure workload with a 235B-parameter MoE model, it reduced mean TTFT by 69.4% (3.26x) and total query round time by 68.7% (3.20x) compared to no offload, while keeping the cache tier local to the inference server and protected against single-drive failure. In the same workload, Linux MD RAID5 using default settings measured higher TTFT and query round time than both the no-offload baseline and SupremeRAID™ AE RAID5, demonstrating why storage-backend performance is central to the KV cache offload use case.

If your inference stack is hitting GPU memory limits, the question is not whether to offload KV cache; it is whether your offload tier is fast enough and protected enough to depend on in production. Linux MD RAID5 default settings provide RAID protection but fail the performance requirement in this tested workload. SupremeRAID™ AE RAID5 answers both.

Learn more at Graidtech.com/ai.