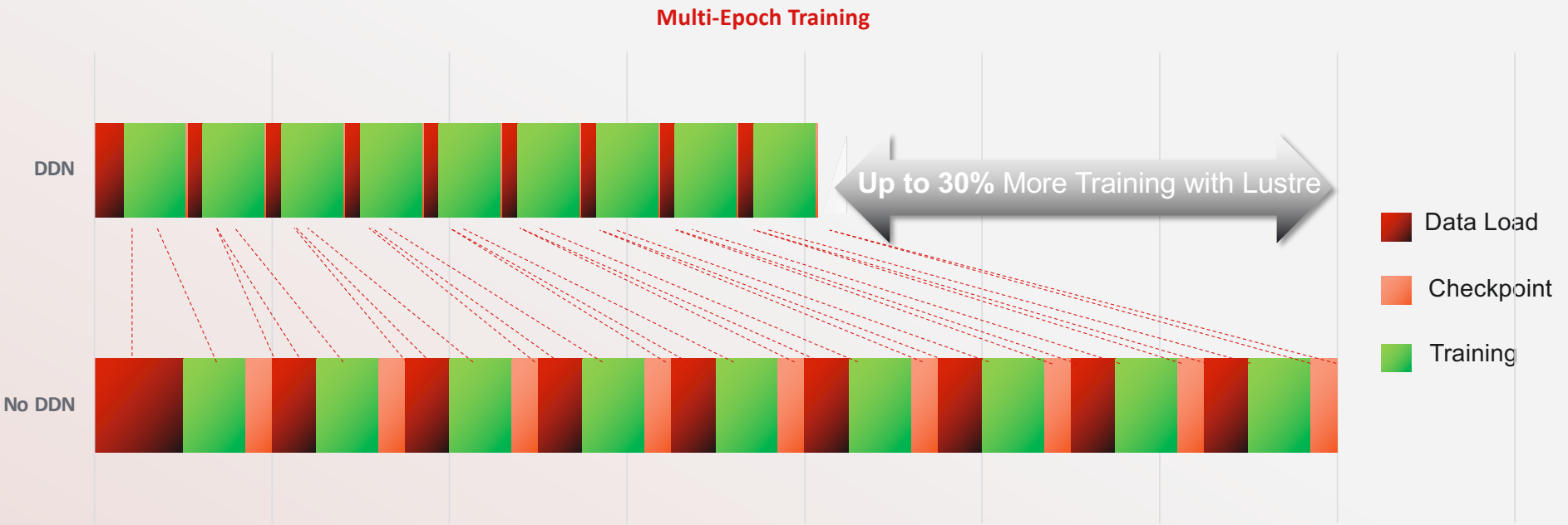
LLM Inference Experiments

Project Update

Morris Skupinsky | 2026-04-28

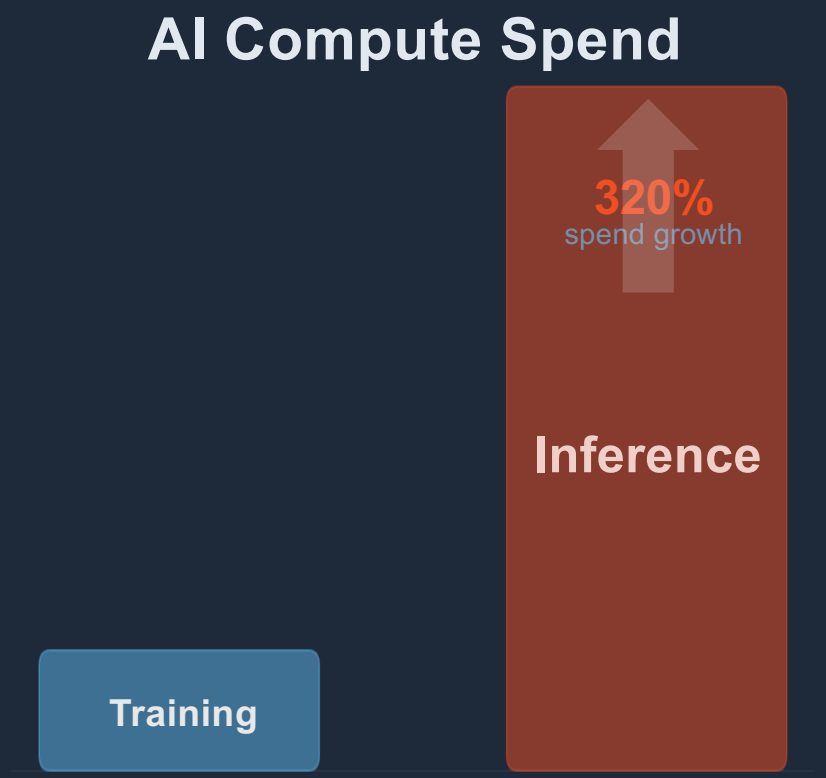
Lustre Can Dramatically Increase AI Productivity

- With 3x Faster Data Loads and 15x Faster Checkpoints



Inference Dominates AI Spend

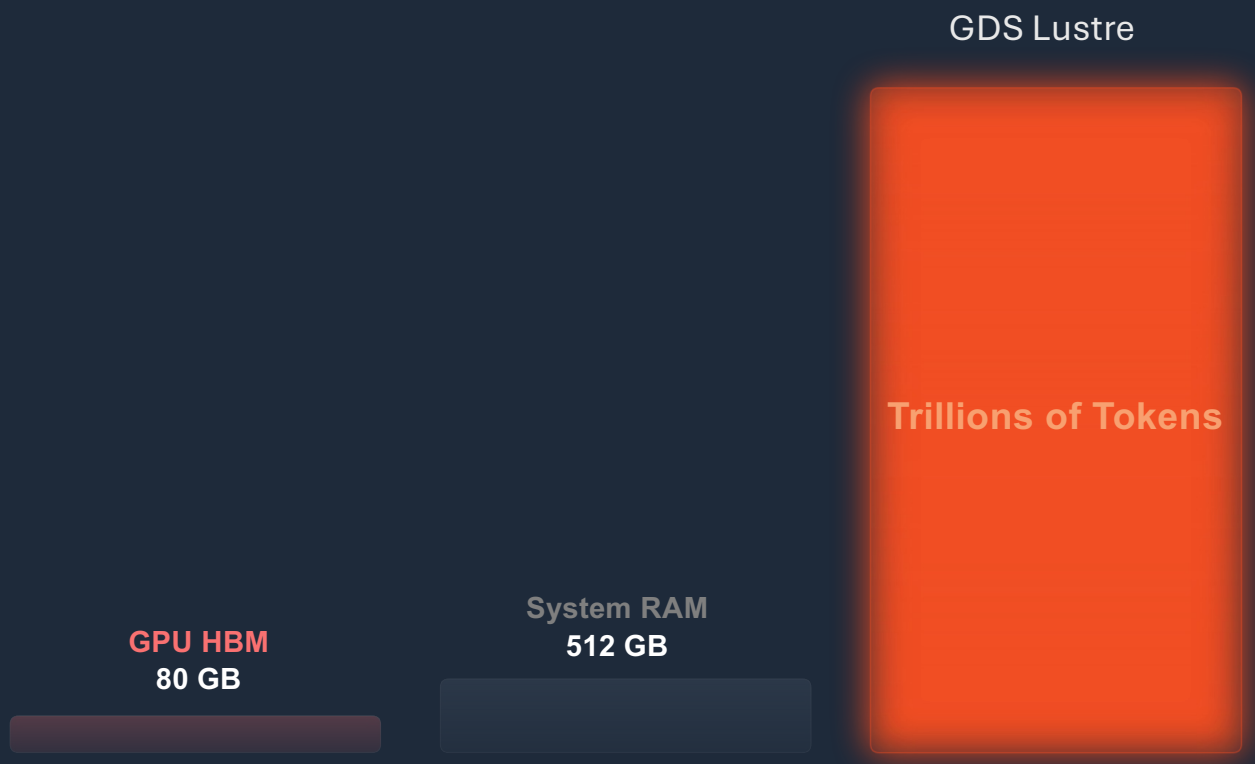
- Inference now accounts for 85% of enterprise AI budgets
- Per-token costs dropped 280x, but total spend grew 320%
- Agentic AI demands 100x more compute per interaction
- Context windows expanding from 4K to 1M+ tokens



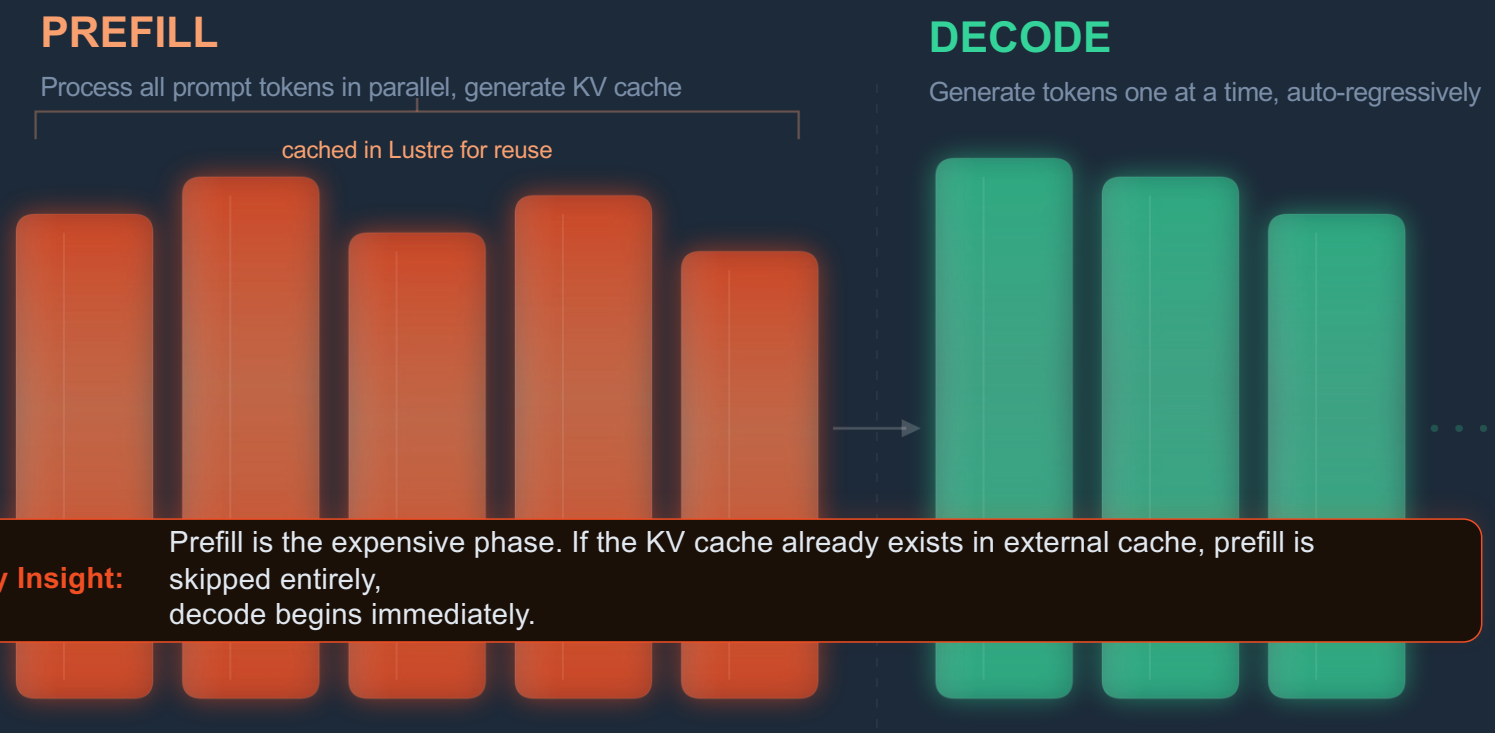
The bottleneck has shifted from GPU compute to data delivery

The Context Memory Problem

AI Memory is the bottleneck to AI Improvement. Lustre extends it.

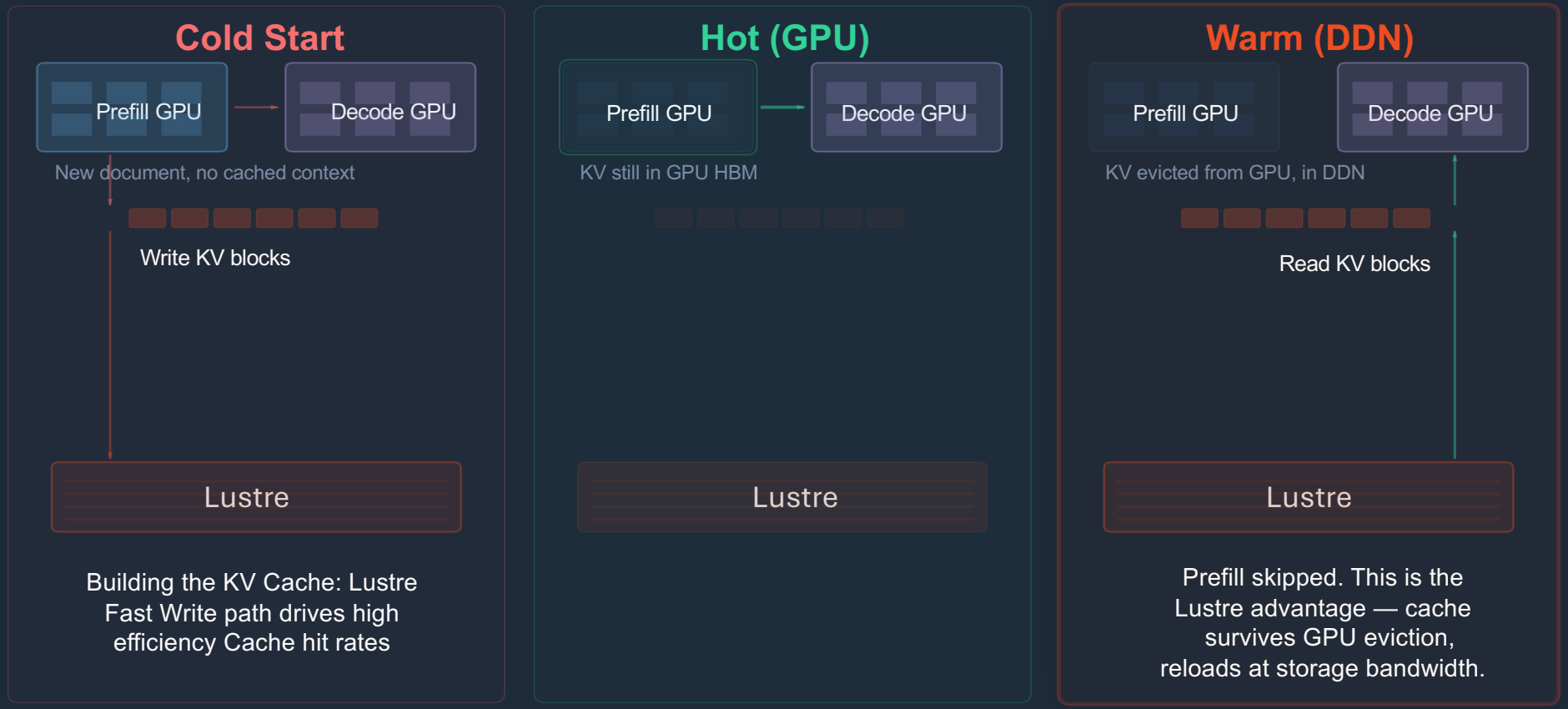


Prefill + Decode: The Inference Pipeline



Key Insight: Prefill is the expensive phase. If the KV cache already exists in external cache, prefill is skipped entirely,
decode begins immediately.

Disaggregated Serving with DDN



EXAScaler vLLM/LMCache Experiments

- Not Typical Nvidia Reference architecture

- Exascaler system deployed on the GPU network, and uses 4x GPU network connections, not the “storage network” specified by Nvidia for storage

EXAScaler vLLM/LMCache Experiments

- NVIDIA DGX H100 | 4x CX-7 100Gb
  - Lustre client - 2.14.0_ddn248
- EXAScaler AI400X2T (single SSU)
  - Lustre server - 2.14.0_ddn248
- Client-side gdsio (POSIX, mode 1) measured to deliver ~63 GB/s Write and ~78 GB/s Read)
- Client-side gdsio (GDS, mode 0) measured to deliver ~30GiB/s Write and Read performance from a single GPU

EXA with vLLM/LMCache using gdsBackend with Llama-3.1-405B-Instruct-AWQ-Int4

=== Model ===
HuggingFace ID: clowman/Llama-3.1-405B-Instruct-AWQ-Int4
Quantization: AWQ Int4 (vLLM auto-converts to awq_marlin kernel)
max_model_len: 131,072 tokens (capped via --max-model-len; YARN factor 64.1 from base 8,192 would otherwise scale to ~525K)

=== Topology ===
config/dynamo-aggregated-exa-llama405b-awq-tp8-1PD.yaml
- 1 aggregated worker, TP=8 across GPUs 0-7
- gpu_memory_utilization: 0.55 (leaves ~28 GB/GPU for cuFile 20 GB buffer)
- connector: lmcache, gds_path on EXAScaler

=== Container ===
nvcr.io/nvidia/ai-dynamo/vllm-runtime:0.9.1-2516_bugfix
(CUDA 12, LMCache 0.4.1 with PR 2516 cache-hit fix, Lustre GDS thread-pool patch, numpy 1.26.4 pin)

=== vLLM launch (identical across both runs) ===
Invoked via:

```
source setup_environment.sh
./run_vllm_topology.sh start \
--topology config/dynamo-aggregated-exa-llama405b-awq-tp8-1PD.yaml \
--delete-cache
```

Script generates this vllm serve invocation inside the container:

```
vllm serve clowman/Llama-3.1-405B-Instruct-AWQ-Int4 \
  --port 8000 \
  --tensor-parallel-size 8 \
  --gpu-memory-utilization 0.55 \
  --max-num-seqs 2 \
  --no-enable-prefix-caching \
  --max-model-len 131072 \
  --load-format runai_streamer \
  --kv-transfer-config
'{"kv_connector":"LMCacheConnectorV1","kv_role":"kv_both","kv_connector_extra
_config":{"discard_partial_chunks":false}}' \
  --kv-events-config '{"enable_kv_cache_events": false}' \
  --hf-overrides '{"rope_parameters":
{"rope_type":"yarn","factor":64.1,"original_max_position_embeddings":8192}}'
```

=== LMCache config: config/lmcache-config-EXA-gdspath.yaml ===

```
chunk_size: 256
save_unfull_chunk: false  avoids LMCache 0.3.x cache-hit assert (PR 2516)
local_cpu: false
gds_path: "/x2e09/sub/KVCACHE3"
cufile_buffer_size: 20480  20 GiB per GPU
extra_config:
use_cufile: true
use_direct_io: true
gds_io_threads: 32
```

Results: `long_doc_qa.py`
EXA with vLLM/LMCache using gdsBackend with Llama-3.1-405B-Instruct-AWQ-Int4

Metric	Run 1 (inflight=1)	Run 2 (inflight=2)
Warmup mean TTFT (cold cache)	43.474 s	59.822 s
Query mean TTFT (warm cache)	0.958 s	1.426 s
Cache speedup (warmup ÷ query TTFT)	**45.4×**	**41.9×**
Warmup round time	914.758 s	894.325 s
Query round time	257.731 s	169.872 s
Cache speedup (round time)	3.55×	5.27×
Avg store throughput	15.81 GB/s	14.25 GB/s
Avg retrieve throughput	9.14 GB/s	9.00 GB/s
Cross-run: query prompts/sec	0.310	0.471 (1.52× over Run 1)

=== Benchmark command ===
Differs from inflight=2 run only in the final flag (1 vs 2):

```
source venv/bin/activate && \
cd LMCache/benchmarks/long_doc_qa && \
python long_doc_qa.py \
  --model clowman/Llama-3.1-405B-Instruct-AWQ-Int4 \
  --port 8000 \
  --num-documents 20 \
  --document-length 100000 \
  --output-len 100 \
  --repeat-count 4 \
  --repeat-mode tile \
  --max-inflight-requests 1 # <-- varies between experiments
```

Results: `aiperf`
EXA with vLLM/LMCache using gdsBackend with Llama-3.1-405B-Instruct-AWQ-Int4

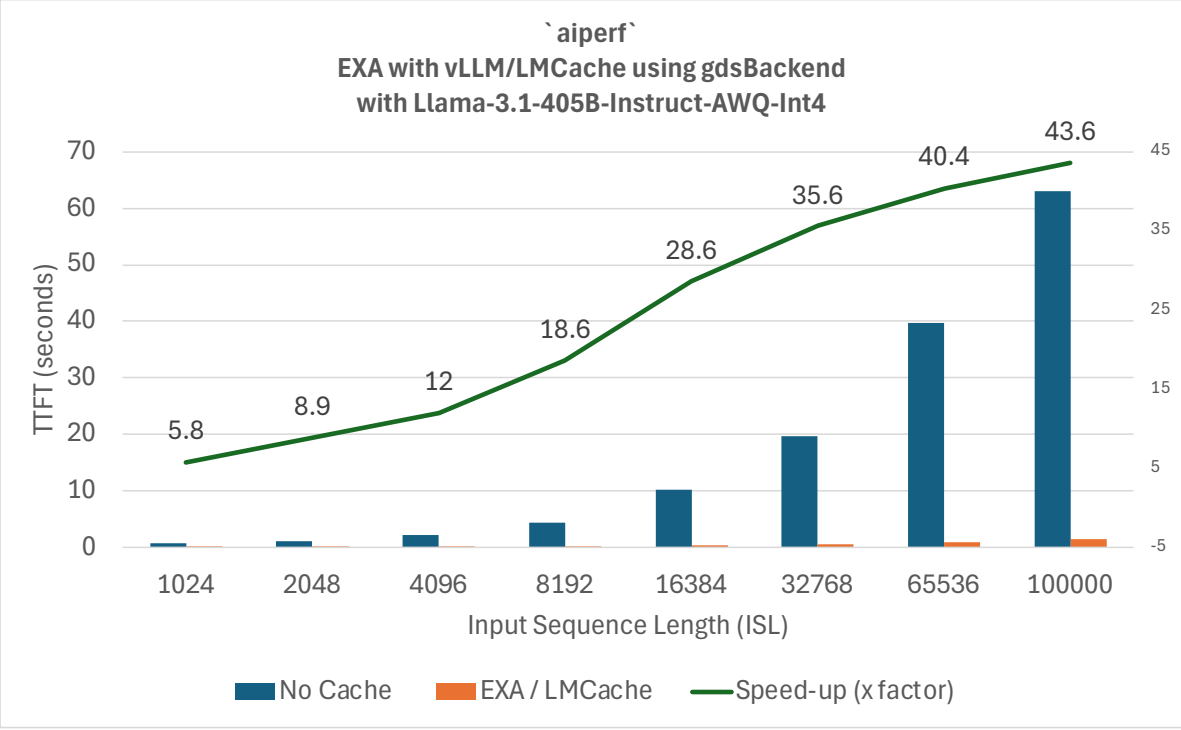
TTFT: nocache vs. cache

ISL	nocache CONC=1	cache CONC=1	Speed -up	nocache CONC=2	cache CONC=2	Speed-up
1K	385.6	87.7	4.4×	744.9	129.5	5.8×
2K	736.8	101.5	7.3×	1,191.8	134.6	8.9×
4K	1,448.2	118.3	12.2×	2,191.4	182.7	12.0×
8K	2,913.0	165.4	17.6×	4,381.6	235.8	18.6×
16K	5,900.9	233.7	25.2×	10,206.9	356.8	28.6×
32K	12,205.9	376.9	32.4×	19,654.2	552.7	35.6×
64K	26,274.6	658.8	39.9×	39,747.3	984.6	40.4×
100K	44,384.3	1,014.5	**43.7×**	63,061.3	1,446.7	**43.6×**

Benchmark command

```
aiperf profile \
  --model 'clowman/Llama-3.1-405B-Instruct-AWQ-Int4' \
  --tokenizer 'clowman/Llama-3.1-405B-Instruct-AWQ-Int4' \
  --endpoint-type 'chat' \
  --url 'http://localhost:8000' \
  --streaming \
  --concurrency 1 \
  --request-count 100 \
  --num-prefix-prompts 1 \
  --synthetic-input-tokens-mean 1 \
  --synthetic-input-tokens-stddev 0 \
  --prefix-prompt-length 102400 \
  --output-tokens-mean 100 \
  --output-tokens-stddev 0 \
  --random-seed 176 \
  --dataset-sampling-strategy 'shuffle' \
  --warmup-request-count 4 \
  --extra-inputs 'ignore_eos:true' \
  --extra-inputs '{"nvext":{"ignore_eos":true}}' \
  --artifact-dir 'benchmark_results/aiperf_cache_2026-04-25/ISL100K_CON1_OSL100/'
```

Results: `aiperf`
EXA with vLLM/LMCache using gdsBackend with Llama-3.1-405B-Instruct-AWQ-Int4



Key-Value Caching and the "Memory Wall"

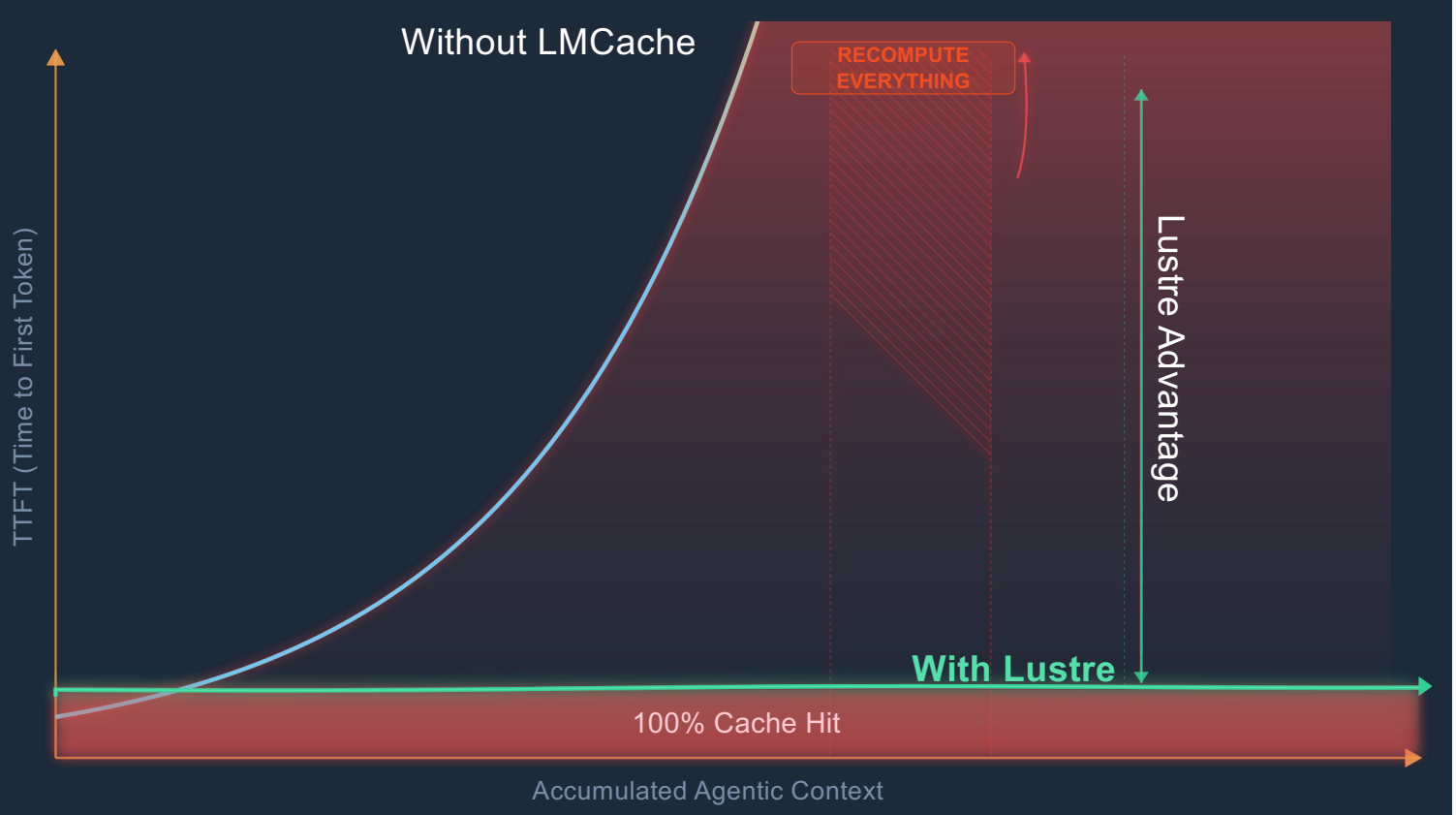
Model context is growing beyond the capacity of HBM causing huge inference limitation

→Lustre can cache more tokens due to superior write performance

→Cache Capacity increase → *infinity*

→ Approach 100% KV cache hits

→ Make KV cache available to all GPUs




ddn