

DeepSeek DeepInfer: Kernels Shipped, Serving Layer Withheld

TechRounder PDF Edition

Live article: <https://www.techrounder.com/ai/deepinfer-withheld-kvcache-load-balancers-128-gpu-floor/>

By Vipin PG | Published October 21, 2026 | Updated October 21, 2026 | Format: Deep Dive | 10 min read

In brief

DeepSeek's DeepInfer release ships kernels and a detailed deployment Q&A, but leaves out the KV cache store and the routing its 128-GPU floor depends on.

DeepSeek released DeepInfer on October 20, 2026, with optimized inference kernels, a model runtime, and support for its DeepSeek-V4.1-Flash model. The release also makes an important limitation explicit. The KV cache management system and request load balancers used in DeepSeek's production infrastructure are not fully open source.

That distinction matters for anyone considering DeepInfer for a production deployment. The code provides an offline inference path and several optimizations used by DeepSeek. Its deployment Q&A describes much of the infrastructure around that code, including a practical deployment size of 128 GPUs and a dedicated distributed storage layer for cached attention state. Reproducing the complete service involves components that are documented but not supplied.

What DeepSeek published

The repository is licensed under Apache 2.0. DeepSeek describes it as the framework behind its large-scale inference service, while specifying that this public release supports offline inference. It currently lists one supported model, 'DeepSeek-V4.1-Flash', with 'DeepSeek-V4.1-Flash-1017' as the default checkpoint.

DeepInfer includes optimized kernels, model and sampling runtimes, expert parallelism, speculative decoding, sliding-window attention optimizations, asynchronous scheduling, and structured output generation. It supports NVIDIA SM100 and Huawei Ascend 950 hardware, although the published performance measurements use NVIDIA GPUs.

DeepSeek also released deepseek-recipe, a separate library for converting API requests and model token sequences. The company says its processing logic matches the official DeepSeek API. That library is MIT-licensed.

What is missing from the open-source release?

DeepSeek's README identifies three requirements for a complete inference service that are not included in the current release:

- A KV cache system for retaining and retrieving reusable attention state.
- Vision Encoder, Prefill, and Decode disaggregation, although DeepSeek says most of the required engine functionality is already implemented.
- Load balancing across data-parallel prefill and decode replicas.

The deployment documentation gives the reason: "The KV cache management and load balancing components of the inference system are tightly coupled to our internal infrastructure, so that code is not yet fully open-sourced."

Some KV-related kernel operators are present in the repository, including 'load_paged_kv_cache.py', 'store_kv_cache.py', and 'state_cache.py'. These handle kernel-level operations. They should not be confused with the distributed cache service, storage coordination, and request routing described in the deployment documentation. The provided examples focus on offline inference.

DeepInfer's production deployment architecture

DeepSeek's production design separates the vision encoder, prefill, and decode stages. A dedicated storage cluster holds reusable KV cache data. Expert parallelism distributes the mixture-of-experts workload across GPUs inside a node or larger scale-up domain.

Component: Vision encoder | Recommended configuration: One GPU per instance | Deployment guidance: Lower-cost GPUs are recommended.

Component: Prefill | Recommended configuration: EP2-8 / DP2-8 | Deployment guidance: Use the smallest parallel configuration that fits GPU memory.

Component: Decode | Recommended configuration: EP64-128 / DP64-128 | Deployment guidance: Use the largest supported expert-parallel configuration.

Component: KV cache storage | Recommended configuration: Dedicated 3FS cluster | Deployment guidance: Approximately 14.4 TB of logical storage per GPU.

EP means expert parallelism, and DP means data parallelism. DeepSeek's architecture sets the number of attention DP replicas equal to the number of EP ranks. The deployment Q&A says tensor parallelism and context parallelism are not currently used. The README nevertheless lists "Attention DP / balanced CP" among supported features. The public documents do not explain that difference.

Why DeepSeek recommends 128 GPUs

The Q&A says a configuration with one prefill instance and one decode instance is deployable. For an efficient production unit, however, DeepSeek recommends roughly one prefill GPU for every decode GPU. Because decode uses EP64 or larger, the documented minimum efficient configuration is 64 prefill GPUs plus 64 decode GPUs.

This is a deployment recommendation, not a claim that DeepInfer cannot run on smaller hardware. The repository supports offline inference, and the published performance benchmark uses four NVIDIA SM100 GPUs for prefill and 64 for decode. That benchmark configuration should not be mistaken for the recommended 128-GPU production unit.

Wide expert parallelism reduces memory traffic during decoding and can improve tokens per second when expert load balancing works well. It also creates a larger failure domain. In a large-EP instance, failure of one GPU can make the entire instance unavailable until the platform reorganizes the remaining healthy GPUs.

DeepSeek says separating prefill and decode across data centers is technically possible because the model uses a compressed KV cache. Its guidance is to budget approximately 400 Gbps of interconnect bandwidth for every eight GPUs. The company says its own inference nodes operate in different but adjacent data centers, where interconnect reliability remains a concern.

The KV cache system and its storage requirements

DeepSeek describes a dedicated 3FS distributed file system cluster with approximately 100 PiB of SSD storage for KV cache data. Its stated retention periods are three days for agent traffic and seven days for chatbot traffic. DeepSeek reports that shortening chatbot retention to about three days reduces the cache hit rate by roughly 0.5 to 1 percentage point, although that result is based on its own traffic.

The architecture stores different types of KV state in different places:

- In-flight block and state KV cache remain in GPU memory.
- Sliding-window attention KV cache resides in local host memory. Cross-machine retrieval uses an in-memory directory and a high-bandwidth network.
- Paged block KV cache resides in the dedicated 3FS cluster and is managed by the separate 'kvcache-store' service.

DeepSeek has released 3FS itself under the MIT license. The missing part is the application-level KV cache management system that connects the inference service to this storage architecture.

Its recommended storage-server configuration is 20 SSDs of 30.73 TB each, with at least 40 GB/s read throughput and 5 GB/s write throughput per server. DeepSeek advises sizing the cluster to retain 24 hours of incoming data while preserving the longer effective cache retention policy.

Storage uses 10+2 erasure coding and can tolerate two simultaneous node failures within an erasure-coding group. For reads, the documentation describes sending requests to 11 storage nodes and completing recovery after any 10 suitable responses arrive.

Cache expiration is handled by 3FS garbage collection rather than an expiration process inside 'kvcache-store'. A worker on each storage node reclaims least-recently-used cache blocks when necessary to keep disk use below a configured threshold.

Keeping cached prefixes in shared storage allows a prefill instance to retrieve previous attention state without requiring requests to return to the GPU that handled their earlier tokens. That gives the request balancer more freedom to distribute work, and it lets compute and storage capacity be adjusted separately.

The missing request load balancers

The Q&A identifies separate load balancers for prefill, decode, vision encoding, and expert parallelism. The distinction is important: DeepInfer includes expert-parallel load balancing (EPLB), but the request-level routing systems for prefill and decode replicas are among the missing components.

Prefill requests are divided into four context-length groups: below 16K tokens, 16K to 64K, 64K to 256K, and 256K tokens or more. The number of engines assigned to each group is adjusted according to recent compute demand. Requests do not require prefix-affinity routing because the shared KV cache allows any prefill engine to reuse eligible prefixes.

Prefill scheduling also adapts chunk size to available compute capacity. DeepSeek describes a fitted quadratic cost model, $\text{cost}(q, c)$, used to choose the largest token chunk that fits a step's budget, up to a limit of 32,768 tokens. Decode scheduling considers both the number of active requests and their total KV cache length.

The released EPLB implementation has documented starting settings. For prefill, DeepSeek recommends four redundant experts per GPU, a 100-step measurement window, and reassignment every 1,000 steps. For decode, it recommends one redundant expert per GPU before expert-count rounding, a 1,000-step measurement window, and reassignment every 10,000 steps.

DeepSeek reports a prefill 'moe_lb' ratio, measured as slowest rank divided by the mean, of 1.074 for EP4/DP4 and 1.045 for EP2/DP2. The decode documentation reports 1.090. Lower values closer to 1 indicate more even work distribution under this measure.

The company estimates that its broader load-balancing arrangement reaches approximately 80% of a perfectly balanced offline benchmark. The public Q&A does not provide enough detail about the benchmark workload and cluster configuration to reproduce that comparison directly.

What reproducing the KV cache path would involve

DeepSeek's description of the cache lookup process shows why the unpublished layer matters. An endpoint first checks cached-prefix metadata through a local 'kvcache-store' process operating in a CHECK role. Prefix matching uses a hash chain, and the cache-hit length must be known before the request can be routed.

On a hit, cached data is fetched asynchronously from 3FS into shared host memory. The engine then transfers the required KV blocks into GPU memory, overlapping transfers with computation where possible. DeepSeek says its implementation uses GPUDirect RDMA for fine-grained transfers because ordinary CUDA copy-engine behavior is less efficient for that workload.

Sliding-window attention has separate storage rules. Its cached state is kept in host memory, with snapshots written at 128-token boundaries. The documented footprint is 20 layers multiplied by 134 token positions and 528 bytes, or 1,415,040 bytes per sequence. Cross-machine hits can be retrieved using a directory service and a 400 Gbps network.

Autoscaling depends on the same coordination layer. Prefill capacity follows the volume of uncached input tokens, while decode capacity follows active request count. DeepSeek says scale-in is blocked when quota utilization exceeds 0.8, and engine restarts take minutes. Extra capacity and request-level rerouting are therefore part of its fault-tolerance strategy.

These details provide useful design guidance. They are not a drop-in implementation of the distributed cache service or the request router.

What the released inference engine can do

DeepInfer includes DSpark speculative decoding with confidence-scheduled, variable-length verification. Speculative decoding allows a model to verify drafted tokens in batches instead of generating every token through a separate full decode step.

DeepSeek reports that its variable-length verification cuts the average number of drafted tokens verified per step by approximately 30%, while keeping accepted length close to the original configuration. At around 60 concurrent sequences per rank on its EP64/DP64 setup, the company reports a throughput improvement of 4% to 7%. Its published results vary by concurrency, including a small regression at one tested setting.

Sliding-window attention bounded replay reduces repeated work during prefill by recomputing only a limited recent context in decoder layers. DeepSeek describes replaying the last 128 prompt tokens rather than 2,560 positions across 20 layers, with a substantial reduction in total prefill computation.

The engine also includes asynchronous scheduling, which overlaps CPU scheduling work with GPU execution. Structured outputs support JSON Schema, regular expressions, and EBNF. Grammar compilation happens separately before the constrained request enters the inference engine.

Two advertised capabilities have hardware restrictions: batch-invariant inference and deterministic sampling are marked NVIDIA-only in the release documentation.

DeepSeek also acknowledges that the public FlashMLA implementation lacks an internal hand-tuned SASS optimization. It estimates that the released decoding kernel is 5% to 10% slower in the affected comparison, while the overall end-to-end difference is at most 5%. Those are vendor-reported figures.

NVIDIA and Huawei support has different evidence

DeepInfer provides dependency configurations for NVIDIA SM100 and Huawei Ascend 950. The NVIDIA stack uses CUDA, while Ascend uses Huawei's CANN software stack. This is a meaningful portability goal for an inference engine built around large parallel accelerator groups.

The performance evidence is uneven. DeepSeek's published benchmarks use NVIDIA SM100 GPUs, and some features are explicitly NVIDIA-only. The repository therefore demonstrates two supported software targets without establishing equivalent throughput, latency, or operating costs across them.

The New Stack has also noted the absence of corresponding public Huawei benchmark results. Independent Ascend 950 measurements will be needed to assess how the port performs outside DeepSeek's reported configurations.

DeepInfer's vLLM origins and earlier release concerns

The README credits vLLM as DeepInfer's primary starting point, with additional ideas from SGLang and TensorRT-LLM. That lineage is relevant because the company previously explained why publishing its full inference engine was difficult.

In a 2025 explanation, DeepSeek cited divergence from its older vLLM fork, dependence on internal infrastructure, and limited maintenance capacity. The 2026 release addresses part of that problem by publishing the engine code it can share and documenting several infrastructure components that remain internal.

The new repository does not establish that those original maintenance and infrastructure problems have been fully resolved. For developers planning to build around DeepInfer, the missing orchestration layer remains a practical consideration.

The \$38 per GPU-hour revenue claim

The deployment Q&A says one DeepSeek cluster handles 155 trillion tokens per day and earns US\$38 in revenue per GPU-hour at what it calls "valley pricing". The statement attracted attention because it places a revenue figure beside an unusually high token-volume claim.

Revenue should not be read as profit. The documentation does not disclose the cluster's GPU count, capital costs, power costs, storage expenditure, utilization, depreciation, or operating margin. It also does not define the off-peak pricing period used in the calculation. Without those details, the \$38 figure cannot establish profitability or the economics of a comparable deployment.

The token-volume and revenue figures are DeepSeek's own claims. They should be treated as reported operating figures rather than independently audited performance or financial results.

What developers discussed after the release

A Hacker News discussion on October 20 focused heavily on the revenue claim and the hardware costs implied by DeepSeek's deployment figures. Some participants estimated possible returns from \$38 per GPU-hour. Others pointed out that revenue says little about profit without an account of infrastructure and operating costs.

The missing KV cache service also drew attention. Commenters identified shared cache storage and intelligent request routing as substantial parts of the production system that a third party would still have to build. That concern follows directly from DeepSeek's own release notes: the source code provides a working inference foundation, while the distributed serving layer remains incomplete.

Community reactions are useful for identifying the questions developers want answered, but they do not substitute for independent benchmarks or a published cost model.

What DeepInfer means for outside deployments

DeepInfer gives developers access to inference kernels, scheduling techniques, and a runtime for DeepSeek-V4.1-Flash, along with unusually detailed documentation of the production architecture around them. Teams studying speculative decoding, expert parallelism, or GPU scheduling can evaluate those released components without reconstructing DeepSeek's entire service.

A production service matching the company's documented large-scale configuration is a different project. DeepSeek's recommended minimum efficient unit uses 128 GPUs, and its cache architecture assumes dedicated 3FS storage, a high-bandwidth interconnect, prefix matching, cache transfer coordination, and request-level load balancing. Several of those pieces are unavailable as part of the DeepInfer release.

The most useful next evidence will be code releases for 'kvcache-store' and the data-parallel request balancers, independent throughput tests on NVIDIA and Huawei hardware, and third-party deployments that report complete serving costs. Until then, DeepInfer is best evaluated as a substantial open-source inference-engine release with a documented but partly proprietary production serving layer.

References

1. github.com - deepseek-ai / DeepInfer - <https://github.com/deepseek-ai/DeepInfer>
2. github.com - deepseek-ai / DeepInfer - https://github.com/deepseek-ai/DeepInfer/blob/main/doc/inference_deployment_qa.md
3. github.com - deepseek-ai / deepseek-recipe - <https://github.com/deepseek-ai/deepseek-recipe>
4. github.com - deepseek-ai / 3FS - <https://github.com/deepseek-ai/3FS>
5. thenewstack.io - deepseek-deepinfer-huawei-nvidia - <https://thenewstack.io/deepseek-deepinfer-huawei-nvidia/>
6. github.com - vllm-project / vllm - <https://github.com/vllm-project/vllm>
7. github.com - deepseek-ai / open-infra-index - https://github.com/deepseek-ai/open-infra-index/blob/main/OpenSourcing_DeepSeek_Inference_Engine/README.md
8. news.ycombinator.com - item - <https://news.ycombinator.com/item?id=50163277>