READ MODE · ANSWER FIRST

How Kimi K3 architecture, training and serving work

See how Kimi K3 combines sparse experts, KDA, Attention Residuals, quant-aware SFT and cache-aware serving at 2.8T scale.

CHEAT SHEET · 7 ESSENTIAL IDEAS

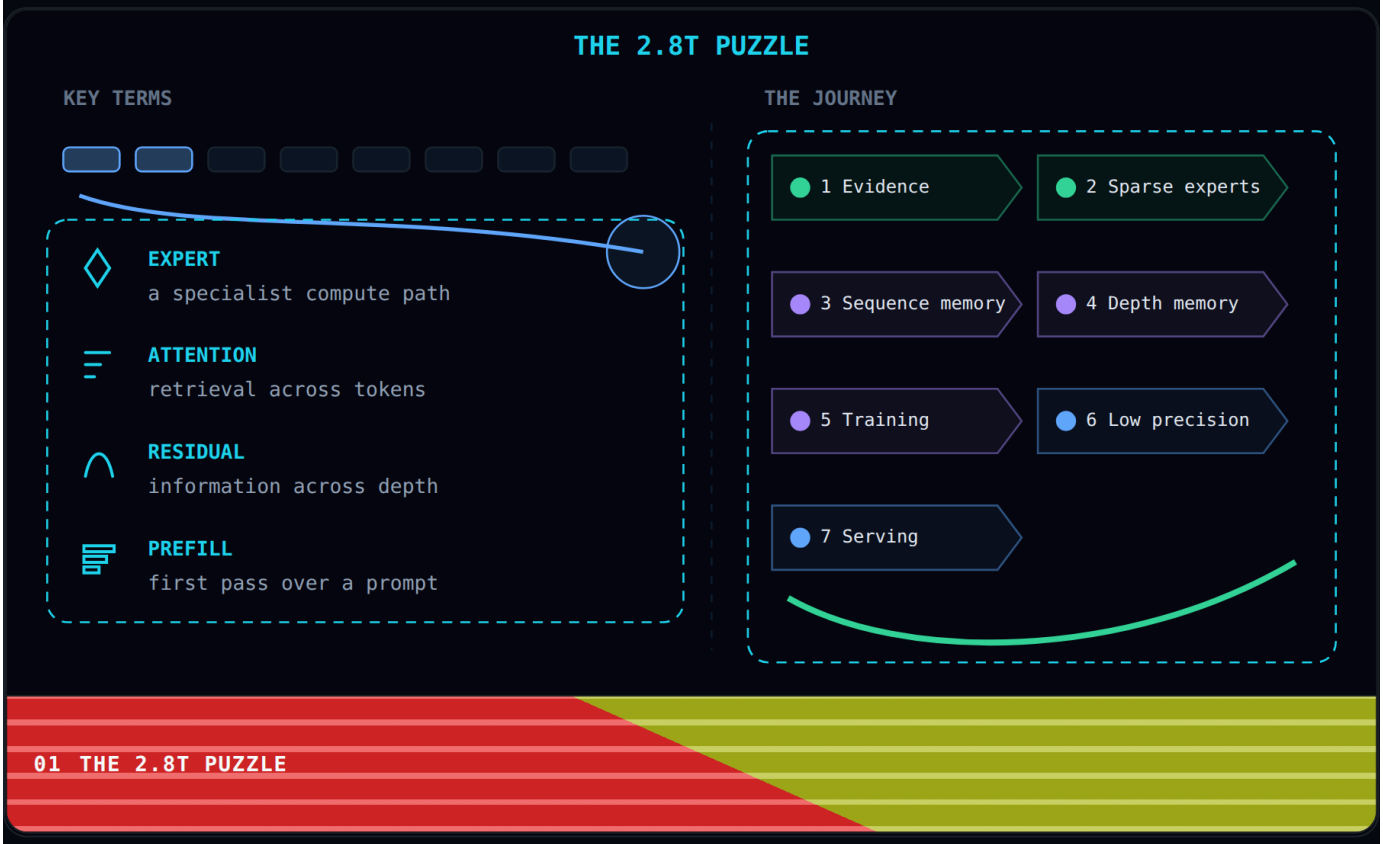
The whole story in 7 lines

Kimi K3 scales by activating less, remembering selectively and treating training precision plus serving topology as one design.

1. K3 facts must stay separate from older Kimi lineage and still unpublished details.
2. Sparse routing selects 16 of 896 experts, while K3 active parameter count remains undisclosed.
3. KDA compresses sequence history while periodic global attention restores selective full recall.
4. Attention Residuals retrieves useful earlier depth states instead of adding every state equally.
5. K3 names new stability methods, but its corpus, token count and complete schedule remain unknown.
6. Quant-aware SFT prepares MXFP4 weights and MXFP8 activations before they become serving constraints.
7. A million-token model needs reusable prefix state and a wide high-bandwidth accelerator domain.

STAGE 1 OF 9

The 2.8T Puzzle



Imagine loading a million-token project into a model with 2.8 trillion parameters. The puzzle is not only size. The system must choose a small amount of useful work at every step.

An expert is a specialist computation path. Attention moves information across tokens, while a residual path moves information across layers. Prefill is the first pass that reads a prompt.

We will trace seven connected questions. Each one follows the same token from public claims through sparse routing, memory, training precision and the machines that finally serve it.

The map now separates model mechanics from evidence quality. Let us begin with the most fundamental idea: what Moonshot has actually confirmed about K3 today.

Evidence Before Architecture

02 EVIDENCE BEFORE ARCHITECTURE

We start by sorting evidence because K3 launched before its full technical report. Moonshot confirms the 2.8T scale, one million token context, native vision and the names of several core mechanisms. That supports a useful skeleton, but not a reproducible blueprint.

Kimi Linear explains KDA in a smaller published model. The Attention Residuals paper explains depth retrieval. K2 documents MuonClip, data curation and agent training, but those remain lineage.

The K3 corpus, token count, active parameter count and complete layer schedule are still blank. Should older K2 numbers fill those empty slots?

PAUSE AND PREDICT

Should older K2 numbers fill K3's unpublished fields?

- **Copy them forward**
- **Keep them separate**

Skip this check

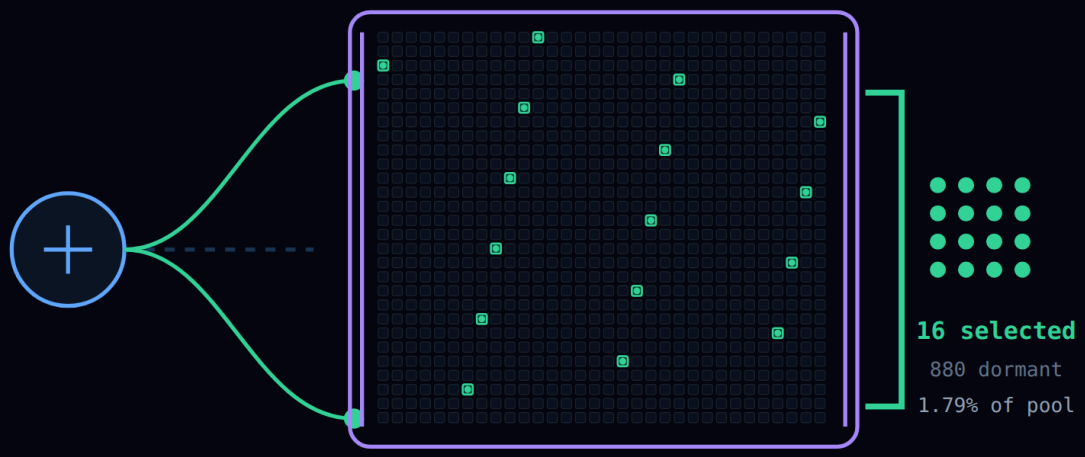
The boundary stays intact. Confirmed K3 facts remain solid, related papers remain supporting lineage and missing fields stay visibly empty until weights or the report supply evidence.

Here is the rule for the rest of our tour. We can explain disclosed mechanisms deeply, but we will label every borrowed mechanism detail and every remaining gap. Next comes K3's clearest number.

STAGE 3 OF 9

Sixteen Paths Through 896

★ If you remember one thing · A token selects only 16 of K3's 896 routed experts.



03 SIXTEEN PATHS THROUGH 896

Now that our evidence boundary is clear, let us follow one token. K3 offers 896 routed experts, which behave like a large workshop of specialist computation paths. The field is real capacity, not a decorative crowd.

The complete field represents available routed capacity, not simultaneous work. A router scores the token against the pool and prepares a much smaller path through that field.

Moonshot says K3 effectively activates 16 routed experts for each token. What fraction of the 896 expert cells will light up?

PAUSE AND PREDICT

What share of the routed expert pool is selected?

- About 50%
- About 10%
- About 1.8%

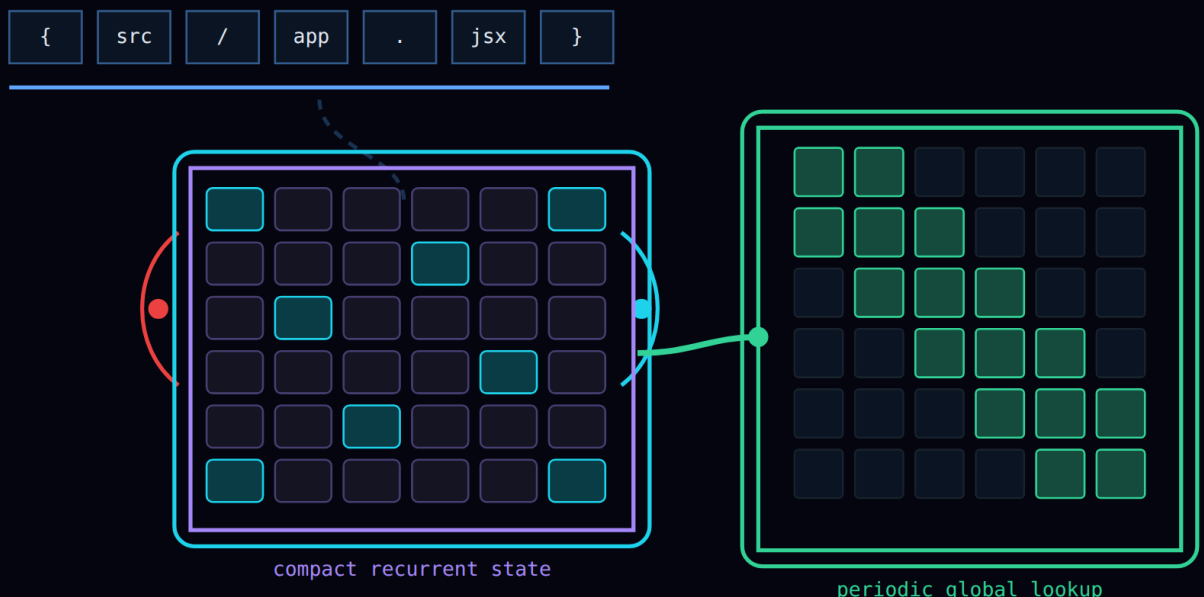
Skip this check

Sixteen bright cells now form one route while 880 remain dark. That is about 1.79 percent of the routed pool, which makes K3's extreme sparsity visible without inventing active parameters.

Sparse routing separates stored capacity from work selected for one token. That small selection is the central tradeoff. Moonshot calls the surrounding framework Stable LatentMoE, but its complete internals await the report. Next we track memory across sequence length.

STAGE 4 OF 9

Memory Across Tokens



04 MEMORY ACROSS TOKENS

The selected expert result returns to the token stream. KDA now handles sequence memory by updating a fixed recurrent state instead of storing one growing key and value pair for every token. Its size does not grow one slot per token.

The Kimi Linear paper describes KDA as a gated delta rule with fine-grained control. In our illustration, stale marks fade while useful associations are rewritten into the compact state.

Compact memory saves space, but aggressive compression can blur a distant relationship. What restores an occasional direct view across the token history?

PAUSE AND PREDICT

What restores periodic global token access?

- More experts
- Gated MLA
- Attention Residuals

Skip this check

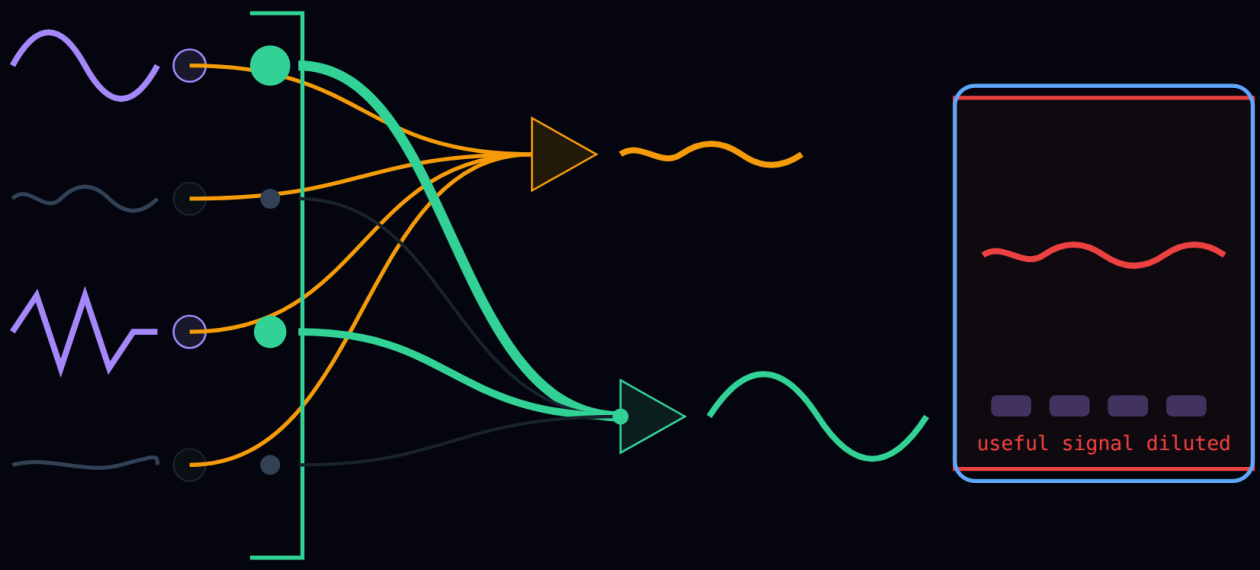
A global attention matrix opens beside the compact KDA state. The launch diagram presents a repeating hybrid motif, while the Kimi Linear paper supplies the detailed three to one lineage.

KDA moves information across sequence length, while Gated MLA restores selective global access. That division helps explain a million-token design. Next we rotate the problem and move information across model depth.

STAGE 5 OF 9

Memory Across Layers

Try it in Watch mode. Keep the useful early feature distinct at the current layer.



05 MEMORY ACROSS LAYERS

We just moved information across tokens. Attention Residuals tackles a different axis by helping the current layer retrieve earlier representations from model depth.

A standard PreNorm residual path keeps adding earlier outputs with fixed unit weight. As depth grows, each useful contribution can become a smaller part of a large accumulated state.

The current token needs a strong early feature and a medium recent feature, while two other states add noise. Should every depth state receive the same weight?

PAUSE AND PREDICT

Should every earlier depth state receive the same weight?

- Yes, keep equal weights
- No, select by content

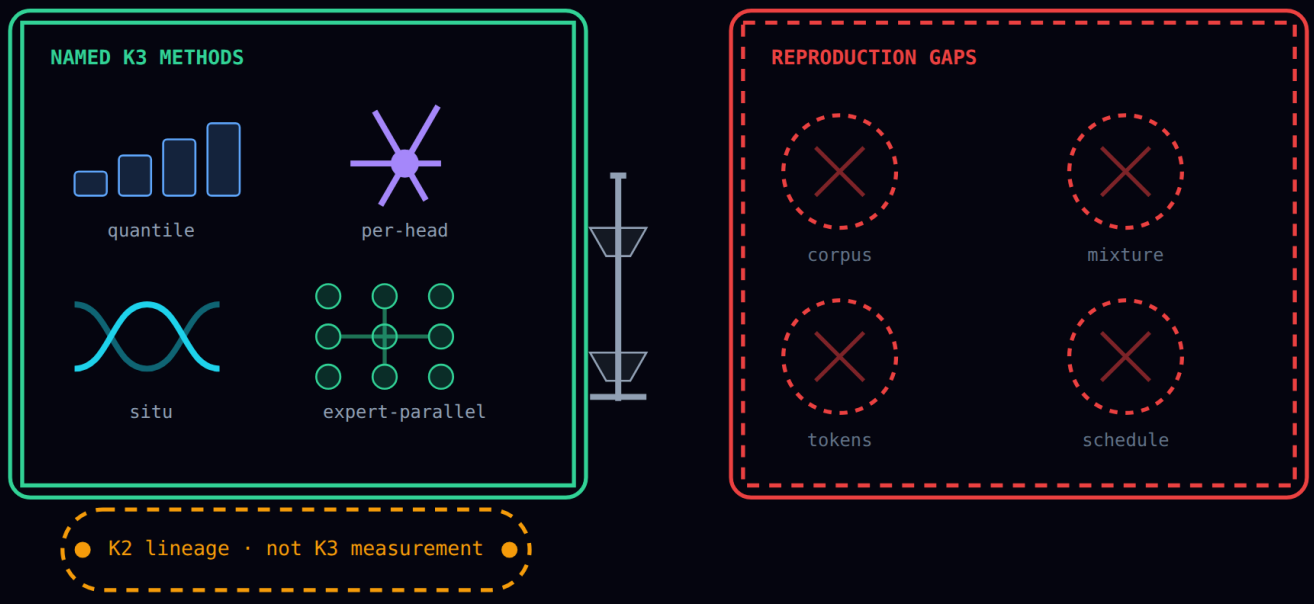
Skip this check

The learned paths now have unequal widths, and the useful waveform remains distinct in the current layer. Block AttnRes groups layers to reduce the memory and communication cost of full depth attention.

Switch the Residual mode control between Uniform sum and AttnRes. Compare the downstream waveform and find the setting where the useful early signal remains distinct.

Attention Residuals turns depth into selective retrieval, while KDA handles sequence memory. Keep those axes separate. Next we examine which K3 training claims are public and which remain missing.

Training: Known and Missing



06 TRAINING: KNOWN AND MISSING

The architecture is only half the story. Moonshot names Quantile Balancing, Per-Head Muon, SiTU and fully balanced expert-parallel training as K3 methods for stable scaling.

K2 showed that MuonClip could control exploding attention logits during a 15.5 trillion token run. That result explains the optimizer lineage, but it does not reveal K3's schedule or token count.

The launch credits refined data and training recipes with the architecture for about 2.5 times better scaling efficiency. Does that claim reveal a reproducible K3 corpus?

PAUSE AND PREDICT

Does the scaling claim reveal a reproducible K3 corpus?

- **Yes, fully**
- **No, details are missing**

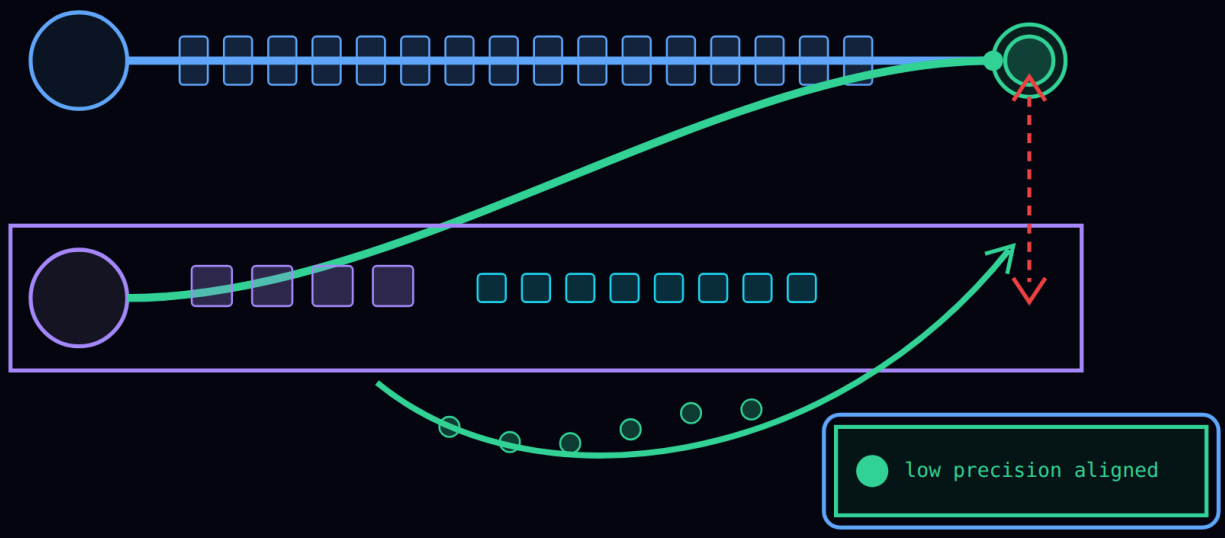
Skip this check

The training board now has solid method cards beside empty recipe slots. We can discuss what each named method tries to control, but we cannot reconstruct the full K3 run from this disclosure.

The honest training lesson has two parts. K3 introduces concrete stability machinery, while its complete data and optimization recipe still awaits the report. Next we inspect one unusually specific post-training disclosure.

STAGE 7 OF 9

Low Precision Starts in SFT



07 LOW PRECISION STARTS IN SFT

The missing corpus limits our training story, but Moonshot gives one precise post-training fact. Quantization-aware training begins at supervised fine-tuning rather than after the model is already finished.

The deployment path uses MXFP4 weights and MXFP8 activations. Fewer bits reduce storage and data movement, but rounding can shift a computation away from its reference target.

A post-hoc conversion first learns only the wider path, then compresses it later. What changes when the compact path participates during supervised fine-tuning?

PAUSE AND PREDICT

What changes when low precision participates during SFT?

- **The model can adapt**
- **Nothing changes**

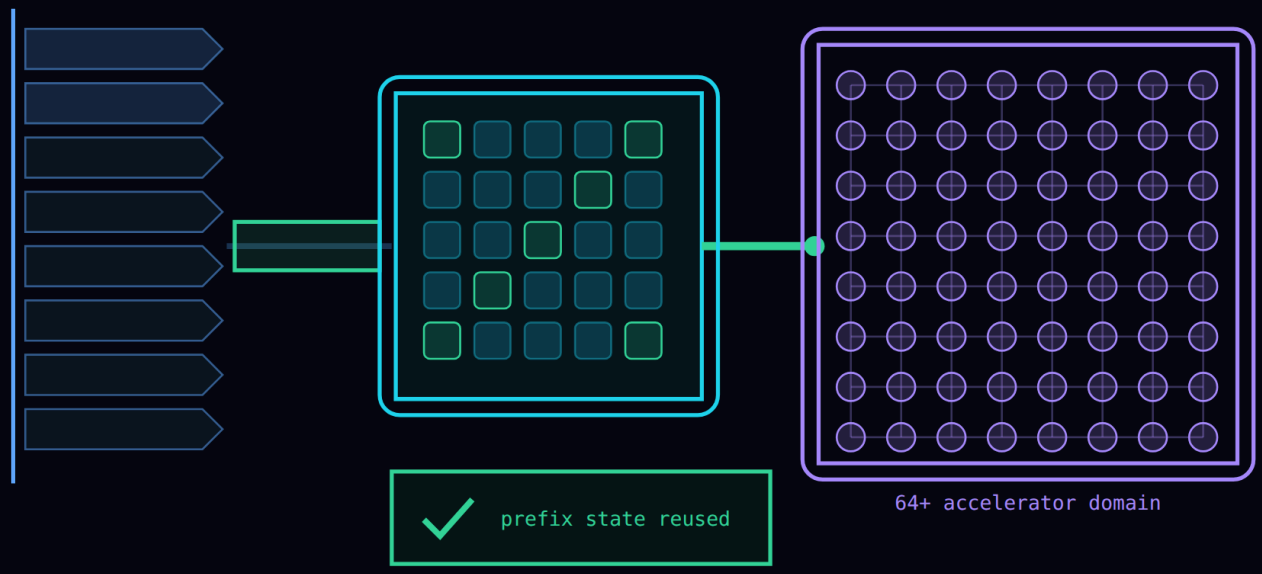
Skip this check

The compact computation now lands on the reference target in our illustration. This demonstrates the purpose of quant-aware SFT without claiming Moonshot's undisclosed calibration algorithm or numerical error.

Switch the Preparation control between After training and During SFT. Compare the compact endpoint and find when the low precision path stays aligned with the target.

K3 treats MXFP4 weights and MXFP8 activations as training constraints from SFT onward. That connects optimization directly to deployment. Next we place the million-token model onto its serving system.

Serving a Million Tokens



08 SERVING A MILLION TOKENS

Quant-aware SFT prepares the numbers, but the serving system still faces a million-token prompt. Prefill reads that prompt before generation and can dominate repeated long-context requests.

Conventional prefix caches are built around key and value blocks. Moonshot says KDA creates new caching challenges and describes a corresponding vLLM implementation for reusable prefill state.

K3 also spreads sparse expert work across a large communication domain. How wide is the supernode configuration Moonshot recommends for efficient inference?

PAUSE AND PREDICT

How many accelerators does Moonshot recommend at minimum?

- 8
- 32
- 64 or more

Skip this check

The reusable prefix state feeds a 64-node accelerator mesh instead of rebuilding every prompt block. The picture links context length, recurrent memory and high-bandwidth expert communication as one serving problem.

Switch the Prefix state control between Cold and Reusable. Compare the prompt blocks that travel into the mesh and find when repeated prefill work disappears.

Serving K3 is not just loading large weights. It requires cache logic shaped for KDA and enough high-bandwidth accelerators for sparse expert traffic. Now let us step back and see the whole picture together.

STAGE 9 OF 9

One Coordinated System

09 ONE COORDINATED SYSTEM

We started with the evidence boundary. Confirmed launch facts stayed separate from Kimi Linear, Attention Residuals and K2 lineage, while undisclosed K3 fields remained empty.

Then we learned how K3 stores enormous routed capacity. Each token selects 16 of 896 experts, although the exact active parameter count is still unpublished.

Next we followed sequence memory. KDA updates compact recurrent state while Gated MLA periodically restores selective global access across a long token history.

We rotated from sequence to depth. Attention Residuals learns unequal retrieval weights so useful earlier representations do not disappear inside one uniform accumulated sum.

The training boundary revealed both progress and uncertainty. Moonshot names several stability methods, but the full K3 corpus, token count and optimization schedule remain unavailable.

Quant-aware supervised fine-tuning connected learning to deployment. MXFP4 weights and MXFP8 activations enter during SFT, allowing the compact path to influence adaptation.

Finally, serving joined context memory with hardware topology. Reusable KDA prefill state reduces repeated prompt work, while Moonshot recommends a domain of 64 or more accelerators.

The complete lesson now converges. K3 scales by activating a tiny expert slice, retrieving information selectively and preparing both low precision arithmetic and cache-aware serving as one coordinated system.