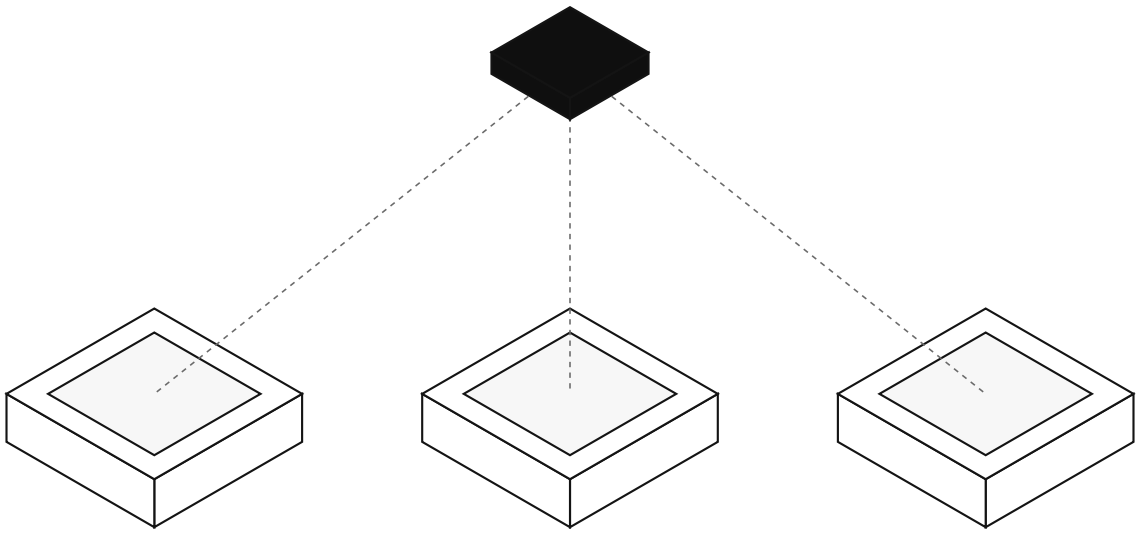
THE UNIVERSAL KV CACHE

UKV

One universal cache that any model can read, and that keeps only what the next turn needs. UKV is a serving architecture for fleets that run several large models. One light shared model reads every conversation once into a **universal cache** — one format for every model in the fleet. A small **translator** per serving model turns that cache into the model's own native cache, so the big models only decode (generate the reply), and a conversation can move to another model for one translation pass instead of a full re-read. Between turns, **safe cache pruning** removes what the next turn will not read, so every conversation carries a much smaller cache while its tasks still complete.



2 November 2026

Executive summary:

A large model keeps a working memory of every conversation it reads — its **native cache**, in a layout only that model understands. UKV gives a multi-model fleet **one universal cache** that any model can read. A shared prefill pool runs **one light model**, Qwen3.8-Flash-Next (Aug 2026), that reads each conversation once into the universal cache. A **translator** (a small translation model, one per serving model) turns the cache into the serving model's own native cache, and the **serving boxes only decode**. Moving a conversation to another model is **one translation pass**, not a re-read. And once per turn a **safe pruner** removes from the cache what the next turn will not read. Measured on our three serving models, GLM-5.3 (Aug 2026), GLM-5.3-Flash (Aug 2026) and DeepSeek-V4.1-Flash (Sep 2026), and served on Qwen3.8-27B (Aug 2026) with Qwen3.5-9B (Feb 2026) as the prefill model. On Qwen3.8-27B the first token comes 44% sooner than on the 27B alone on one GPU and 36% sooner than on standard Mooncake, at Mooncake's time per output token on the same replies.

KV CACHE PER CONVERSATION (ROW OF RECORD)

−45.2%

GLM-5.3-Flash, pruner on; median at task end, 100 tasks; measured.

TOP-1 NEXT-TOKEN AGREEMENT (9 PRODUCTION PAIRS)

89–95%

same next token as the target's own reading; out-of-corpus text; measured.

CAPACITY VS STANDARD MOONCAKE, SAME GPUS

×2.0

Qwen3.8-27B, three GPUs each; 90% of replies within 500 ms, at most 50 ms per token; measured.

IN ONE LINE: one light model prefills for the whole fleet, a translator per serving model writes that model's native cache, and the big models only decode — a model switch costs 15–19% of a full prefill and the target picks the same next token as on its own reading 89–95% of the time; with the pruner on the cache per conversation is 45.2% smaller by task end (42.2–55.1% across all three models) while tasks still complete as often as unpruned; and on our three large models the cost model projects ×1.4–2.0 conversations per dollar-hour against one cluster per model.

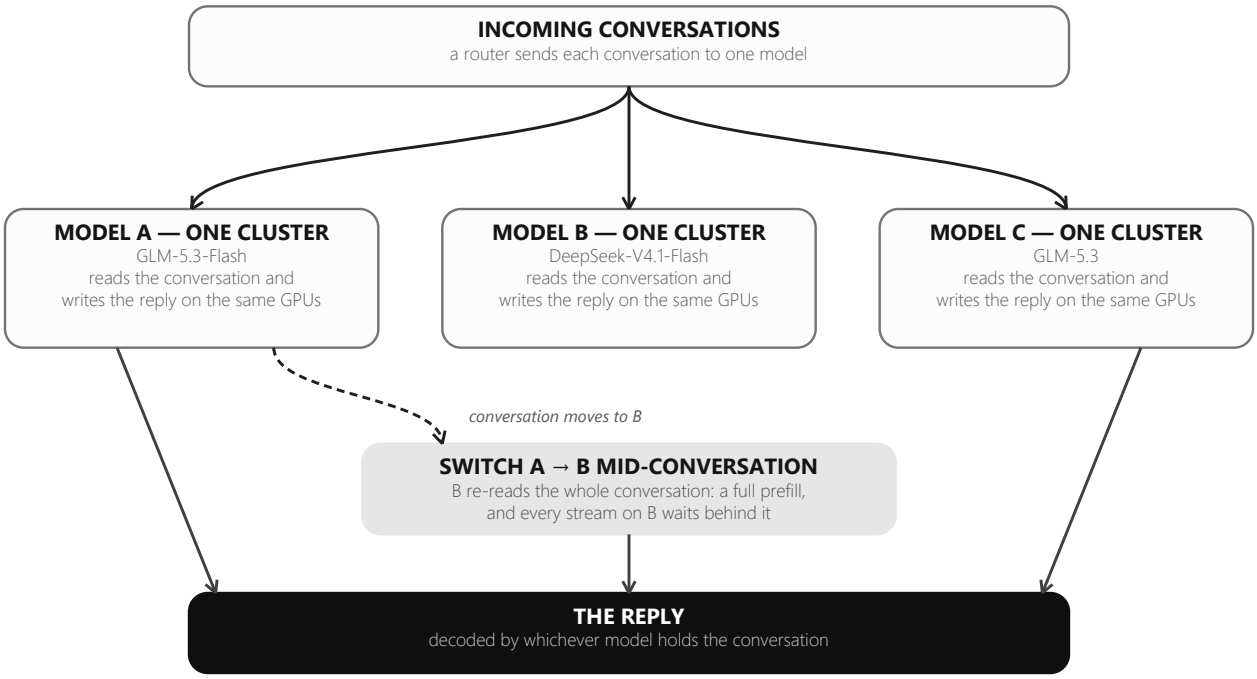
How to read this document

Prefill is a model reading the conversation so far and building its **KV cache**; **decode** is generating the reply one token at a time from that cache. Pages 2–4 state the problem and the idea. **Part I** (pages 5–10) shows the architecture, **Part II** (pages 11–15) the translators, **Part III** (pages 16–20) safe cache pruning and **Part IV** (pages 21–25) the economics; the appendix (pages 26–28) gives the two prefiller options and how they were chosen; the glossary closes the document. **Top-1 agreement** and **task completion** lead every proof; the loss gap and attention share are secondary columns.

TWO KINDS OF CLAIM, KEPT SEPARATE. Numbers marked *measured* were taken on our serving boxes, offline, or on a public agent benchmark. Numbers marked *projected* come from our fleet cost model (assumptions on page 21). The universal prefiller and the pruner on the prefill box are the design: the translators measured here read caches written by our three large models, and the one for Qwen3.8-27B reads a cache written by Qwen3.5-9B; the pruner is measured in place on each serving box.

The Problem: What a Serving Fleet Does Today

Most fleets run **one cluster per model**. Each cluster prefills and decodes on the same GPUs: when a new turn arrives, its prefill is mixed into the decode steps of every other conversation on the box, and every stream waits while it runs. And a model can only use memory it built itself, so if a conversation moves to another model — a cheaper model for an easy turn, a stronger one for a hard one — the new model must **re-read the whole conversation**: 0.97–0.99 s for 32k tokens on GLM-5.3-Flash, paid again on every switch.



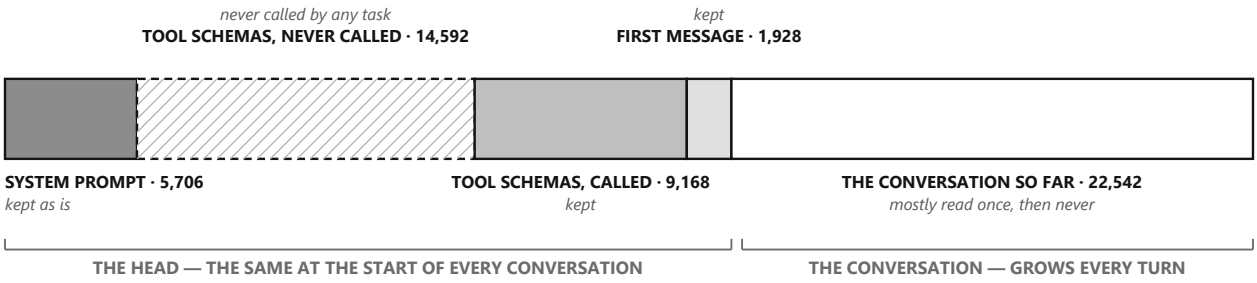
One cluster per model. The grey box is the cost of a switch; the black box is the outcome.

What prefill costs a native box, measured. On a GLM-5.3-Flash box serving 80 conversations, 1,179 of 11,018 sampled steps carried prefill. They ran 32–197 ms against a 20.14 ms decode-only step, so steps carrying prefill took **40.4% of step time**, 33.3% above what decode alone would have cost. Without prefill, every running stream would move at 20.14 ms per step instead of 30.19 ms.

The Problem, Part Two: What the Cache Carries

Every request of a coding-agent conversation begins with the same head: a system prompt of 5,706 tokens and 23,760 tokens of tool schemas — 29,466 tokens before the user has said anything. In our agent sessions about half of that head (14,592 of 29,466 tokens) is tool schemas the session never calls: none of 779 recorded tasks called them. After the head comes the conversation: the user's messages, the model's answers and, above all, tool output (files read, commands run, search results), most of which the model reads once and never returns to. Without pruning, all of it stays in the cache for the rest of the conversation.

THE CACHE AT A TYPICAL CUT TURN — 53,936 TOKENS (GLM-5.3-FLASH, MEASURED)



WHAT THE NEXT TURN READS (HELD-OUT CONVERSATIONS, MEASURED)



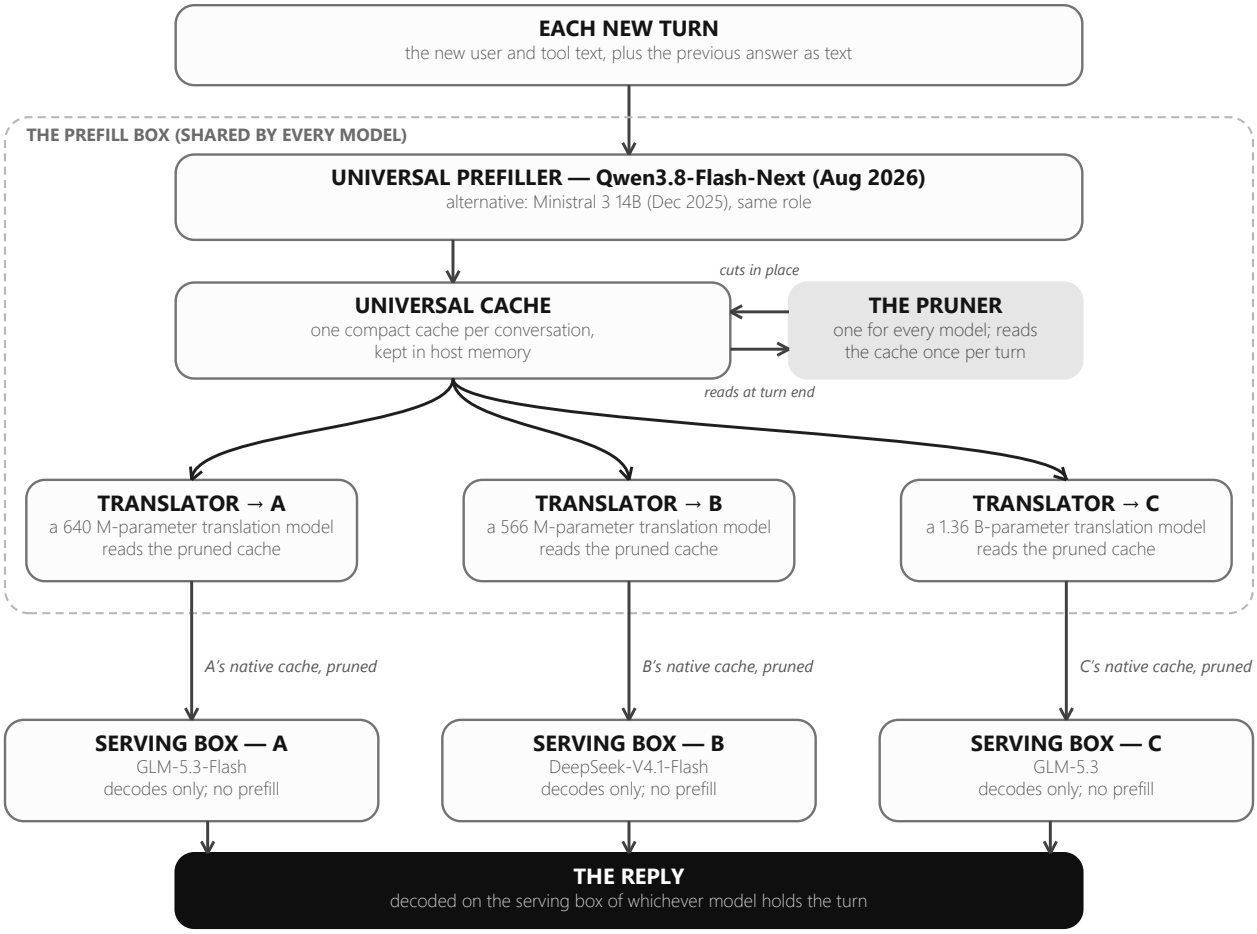
Top: one resident cache on the GLM-5.3-Flash serving box at a typical cut turn (median), drawn to scale, without pruning. Bottom: on held-out conversations, the pieces the pruner cuts make up a large share of the conversation but held a small share of the next turn's attention to it.

WHAT IS IN THE CACHE	TOKENS	SENT
System prompt	5,706	Every request
Tool schemas the session never calls	14,592	Every request
Tool schemas the session calls	9,168	Every request
The conversation so far	grows every turn	Once, then again on every later turn

Tokens *are counted with the GLM-5.3-Flash tokenizer; the other two models differ by a few hundred.* **Attention** *is how much the model draws on a piece of earlier text when it writes its next answer.*

The Idea: One Cache, a Translator per Model, Kept Small

Keep every conversation in **one universal cache** — one format that any model can read — and give each serving model a small **translator** that turns that cache into the model's own native cache. The serving model is not changed: it reads memory in its own format, as if it had read the text itself, and adding a model to the fleet means training one translator for it. Then **keep the cache small**: once per turn, on the prefill box, the pruner removes what the next turn will not read, before any translator reads the cache — so one cut serves every model.



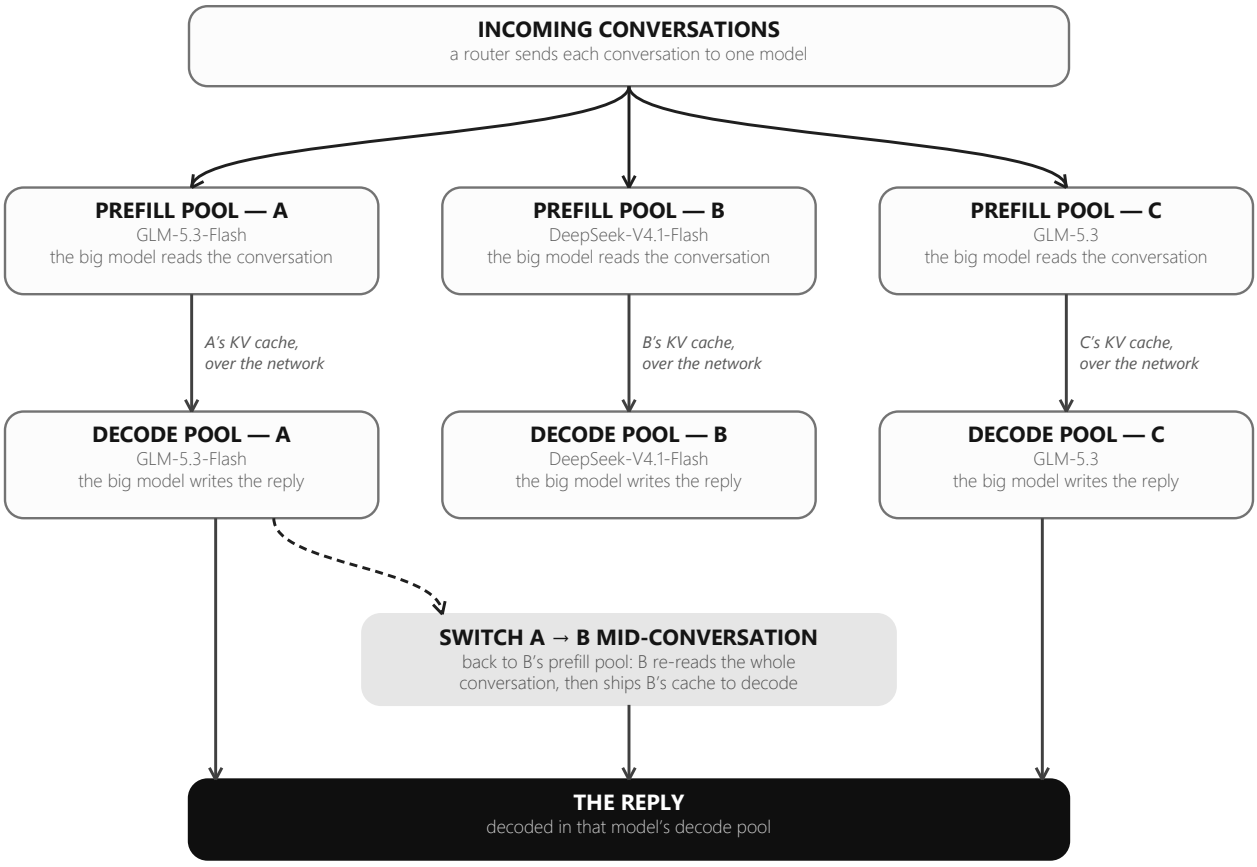
The system. Everything inside the dashed frame runs on one shared prefill box; grey is the pruner, black the outcome. Part I walks through the boxes, Part II the translators, Part III the pruner.

The design. The translators measured in this document read caches written by our three large models, and the one for Qwen3.8-27B reads a cache written by Qwen3.5-9B; the pruner is measured today in place on each serving box (page 17).

PART I · HOW IT WORKS

Standard Mooncake: Prefill and Decode Apart

Mooncake — the disaggregated design Moonshot AI published for serving Kimi — separates the two jobs. Each model gets a **prefill pool** that reads conversations and a **decode pool** that writes replies; the KV cache is shipped from one to the other over the network. Decode boxes no longer stall on prefill. But every model still needs its own prefill pool running the big model, and the cache each pool writes is readable only by that one model. A switch mid-conversation still sends the conversation back to the new model's prefill pool for a full re-read.

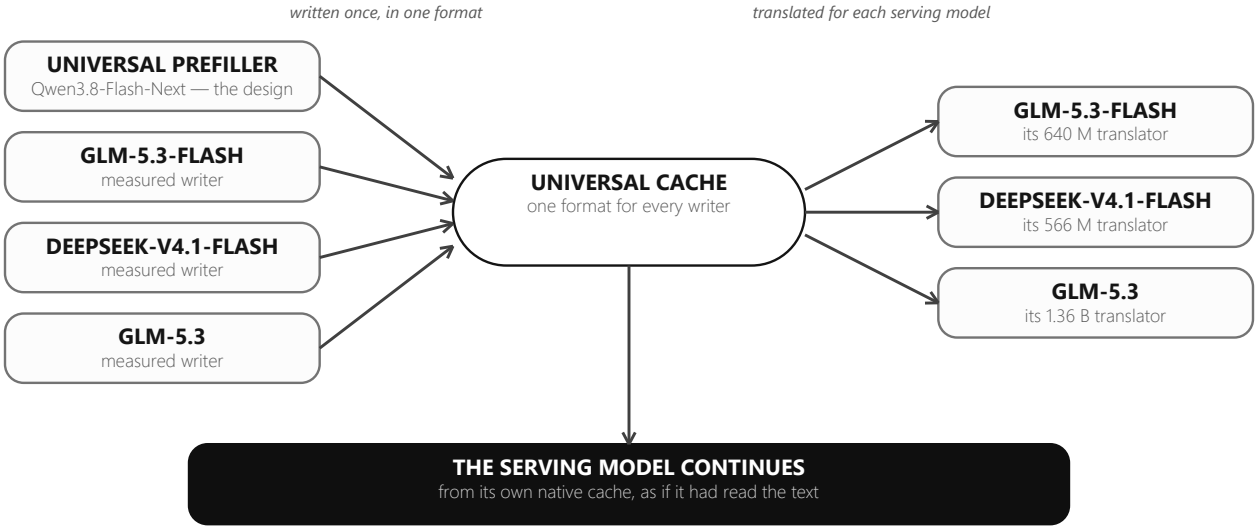


Standard Mooncake, one prefill pool and one decode pool per model.

Mooncake fixes the stall; it does not make the cache portable. The prefill work is the same big-model work as before — 0.47–1.47 GPU-seconds per turn on our three models (measured) — only moved to other boxes.

UKV: One Universal Prefill Pool

UKV keeps Mooncake's split and changes who prefills. One **universal prefiller**, the open Qwen3.8-Flash-Next model, reads every conversation for every model in the fleet into one **universal cache**: a compact, shared cache format. On the same **prefill box**, a **translator** per serving model — 640 M, 566 M and 1.36 B parameters — turns the universal cache into that model's native cache rows and hands them to a **serving box that only decodes**. Any writer, any reader: the universal cache is one format, so one translator per serving model reads it whichever model wrote it.



What the cache does: any writer, any reader. Where each part runs is the drawing on page 4.

BEFORE AND AFTER: MOVING ONE CONVERSATION TO MODEL B

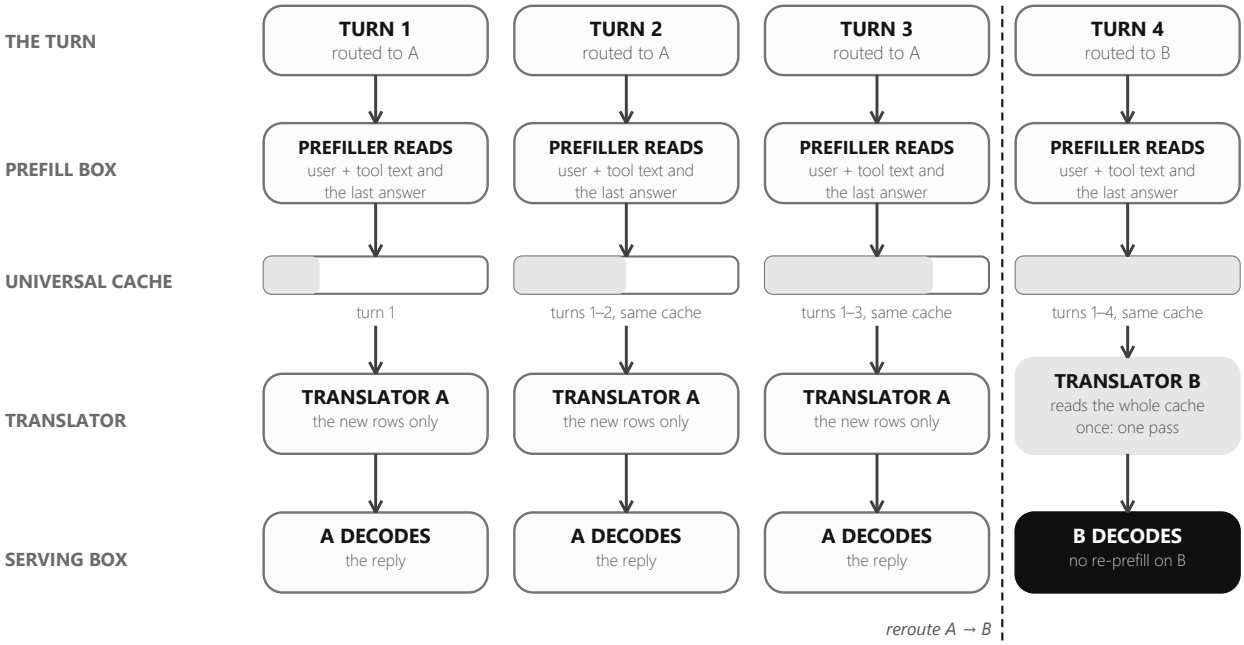
	RE-READ FROM TEXT	UKV
What model B reads	The whole conversation, as text	**The conversation's universal cache, translated into B's own cache**
Time for 32k tokens	0.97–0.99 s (a full re-read, measured)	**0.15–0.18 s (one translation pass, measured)**
Per model added to the fleet	Nothing to add; every switch pays full price	**One translator**

Both times measured on GLM-5.3-Flash for 32k tokens on one GPU. **Why Qwen3.8-Flash-Next (August 2026).** *The universal cache is what the translators read, so the prefiller must write a per-token cache at least as rich as our proven writer, GLM-5.3-Flash (11 attention layers with per-token rows). Qwen3.8-Flash-Next keeps per-token KV in 12 of its 48 layers and passes; among the DeepSeek, Qwen and Gemma models that pass, it is the cheapest to prefill, 31.3 GFLOP per token at 64k context (appendix, pages 27–28); the alternative is on page 26. Its weights are open (Qwen Community 1.0).*

A Model Switch, Step by Step

ILLUSTRATION · NOT A BENCHMARK RESULT

A conversation runs three turns on model A, and the router moves turn 4 to model B. The universal cache does not change hands or formats: it is the same cache, grown by one turn each time. What changes is which translator reads it. Translator B reads the cache for the whole conversation once and writes B's native cache; B then decodes as if it had read the conversation itself. **Nothing is re-prefilled on B.**



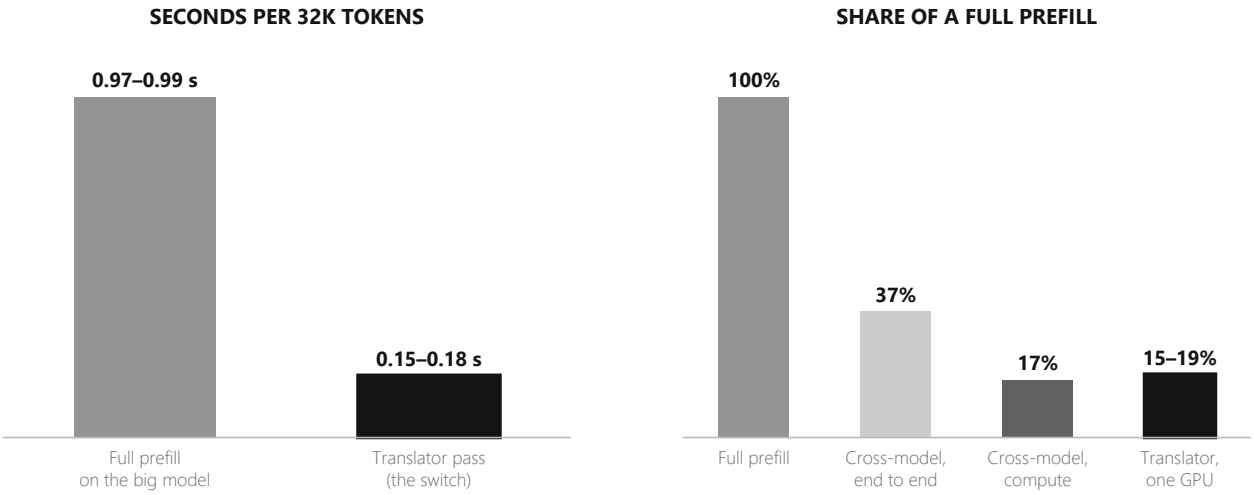
Illustrative conversation. The dashed line marks the reroute.

#	WHAT HAPPENS	WHERE	COST
1	Turns 1–3: the prefiller reads each turn's new text; translator A writes the new rows into A's cache; A decodes.	Prefill box → A's box	0.10–0.17 GPU-s per turn (projected)
2	The router moves the conversation to B.	Your router	—
3	Translator B reads the universal cache for the whole conversation once and writes B's native cache.	Prefill box → B's box	0.15–0.18 s per 32k tokens (measured)
4	B decodes the reply. Its box never prefills the conversation.	B's box	—
5	Turn 5 onward runs as turns 1–3, with translator B.	Prefill box → B's box	as step 1

Illustrative values: the turn numbers are an example. The costs are the cost model's per-turn figure (step 1) and the measured translator time (step 3).

What a Switch Costs: Nearly Free

"Nearly free" here means a number. On 32k tokens of conversation, a translator writes a big model's native cache in **0.15–0.18 s**. A full prefill of the same tokens on the big model takes **0.97–0.99 s** (GLM-5.3-Flash, 32,640 tokens, cold). The switch therefore costs **15–19% of a full prefill**, and it runs on the prefill box rather than stalling the serving box. Across models — a cache written by GLM-5.3-Flash, read by GLM-5.3 — the measured translation is 0.17× the compute of GLM-5.3's own prefill, and 0.37× end to end including the network.



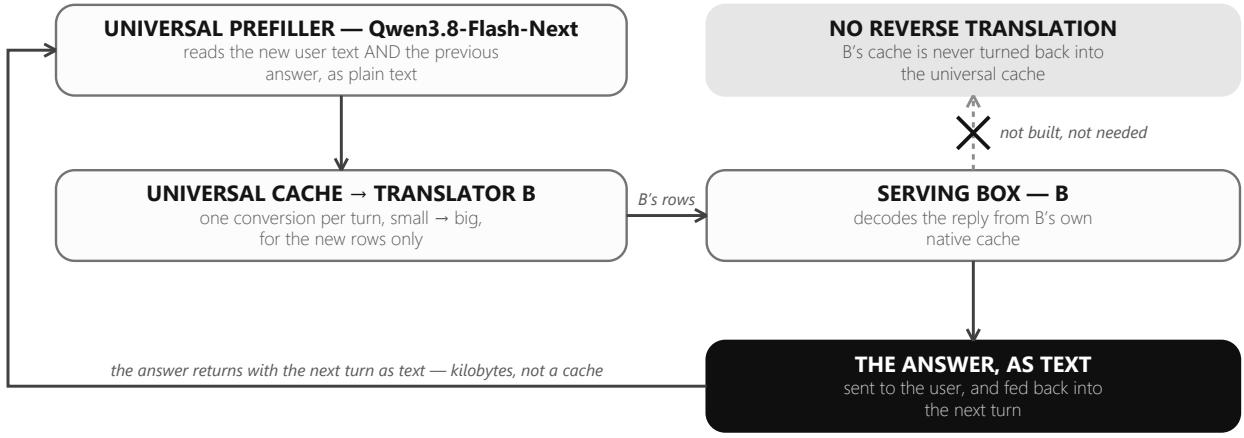
All values measured. Range bars are drawn at the top of their range.

PER 32K TOKENS	WALL TIME	NOTE
Full prefill on the new model	0.97–0.99 s	1.44–1.45 s for 49,280 tokens
Translator pass, one GPU	**0.15–0.18 s**	**15–19% of a full prefill**
Translator pass, GLM-5.3-Flash cache read by GLM-5.3	0.66 s	on one H100; 0.17× compute, 0.37× end to end

Basis: *translator times for the GLM-5.3-Flash and DeepSeek-V4.1-Flash translators, each measured on one GPU; prefill time for GLM-5.3-Flash on its own 8×H200 box, cold; the cross-model pass on one H100. The translator also reads a short pass of the target model's own front end, measured at 0.27 s per 32k tokens on GLM-5.3; in UKV both run on the prefill box.*

No Reverse Path: the Prefiller Reads the Answer

A two-way design would translate the big model's cache back into the universal cache after every answer. UKV does not. The answer leaves the serving box as **text**; on the next turn the universal prefiller simply reads that text along with the new user message. Reading a thousand or two new tokens on the prefiller is cheaper than any reverse translation, so there is **one conversion per turn, small → big only**. Native serving already works this way: on our boxes the prompt grows 809 tokens per turn while 2,041 are prefilled, because the re-rendered answer is read again.



One turn of UKV. The crossed arrow is the reverse step UKV leaves out.

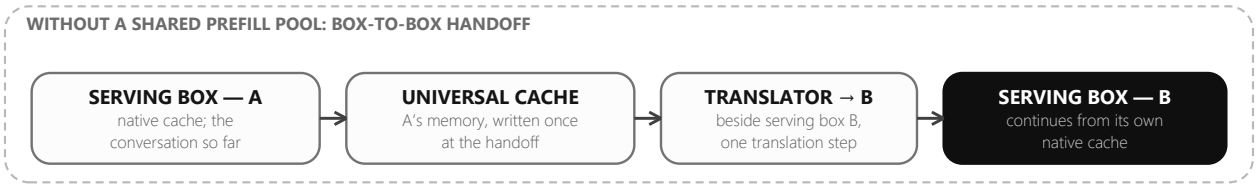
WHAT ONE TURN COSTS THE PREFILL BOX (PROJECTED)

STEP	GPU-SECONDS
The prefiller reads the new text (1,196–1,871 tokens at 62k context)	0.067–0.146
Target model's short front-end pass over the new rows	0.009–0.014
Translator over the new rows	0.008–0.024
Writing the new rows into the universal cache	0.001
Total per turn (loading the cache from host memory overlaps with compute)	**0.102–0.172**

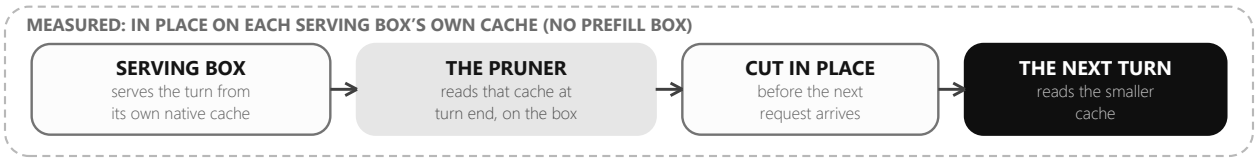
For comparison, the big models spend 0.47–1.47 GPU-seconds per turn on their own prefill today (measured). Per-turn figures are from the cost model (page 21).

How It Fits into Your Fleet

UKV adds one machine: the **prefill box** holds the prefiller, the translators, the pruner and the cache pool, and hands translated cache rows to your serving boxes, whose models, weights and engine stay as they are (the drawing on page 4). A translator is trained once per prefiller → serving-model pair as part of deployment — a supervised training run of hours on a single node — and ships with the fleet. The same two mechanisms also work without a shared prefill pool, box to box — and that is how they are measured.



The translator without a shared prefill pool: it sits beside the serving box that takes the conversation over.



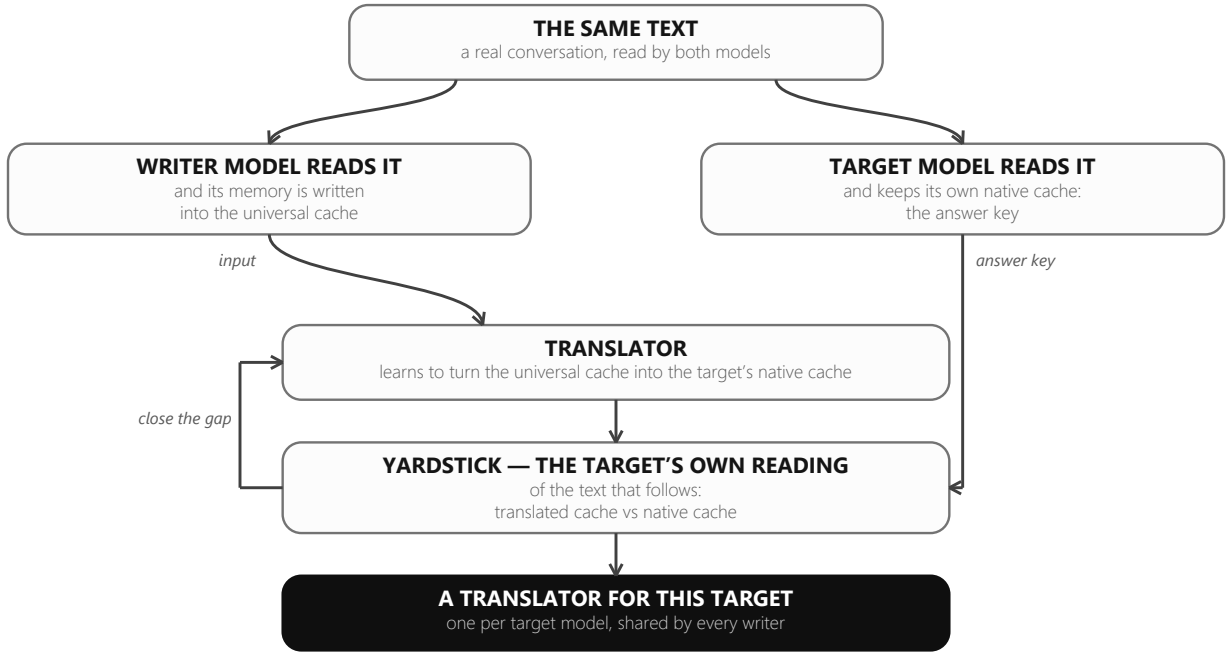
The pruner without a prefill box: in place on each serving box's own cache, as measured in Part III. Grey: the pruner.

WHAT	WHAT CHANGES	WHERE IT RUNS
A prefill box is added	Its GPUs hold the prefiller, the translators, the pruner and the cache pool (about 65 GB of 80 per GPU, projected)	Prefill box
A model switch	One translator pass (0.15–0.18 s per 32k tokens, measured) instead of a full re-read	Prefill box, or beside the serving box
A translator beside a serving box	1.1–2.5 GiB of weights; resident on a serving box it cost nothing measurable (196.5 vs 196.6 tokens/s, measured)	Beside the serving box
The pruner	Between turns, off the request path; nothing is added to a request. Measured today in place on each serving box's own cache	Prefill box (design); serving box (today)

PART II · THE TRANSLATORS

What a Translator Is, and How It Learns

A **translator** is a small **supervised** translation model, one per serving model and shared by every writer. The three built for our large serving models have 566 M, 640 M and 1.36 B parameters, at most 0.2% of the model each one serves; the one for Qwen3.8-27B has 1.64 B, about 6%. It learns from worked examples with a known right answer: the **same text read by two models**. The writer's universal cache of that text is the input; the target's own native cache of the same text is the answer key. The yardstick is the target's own reading of the text that follows: training closes the gap between reading the translated cache and reading its own. The target model itself is never changed.

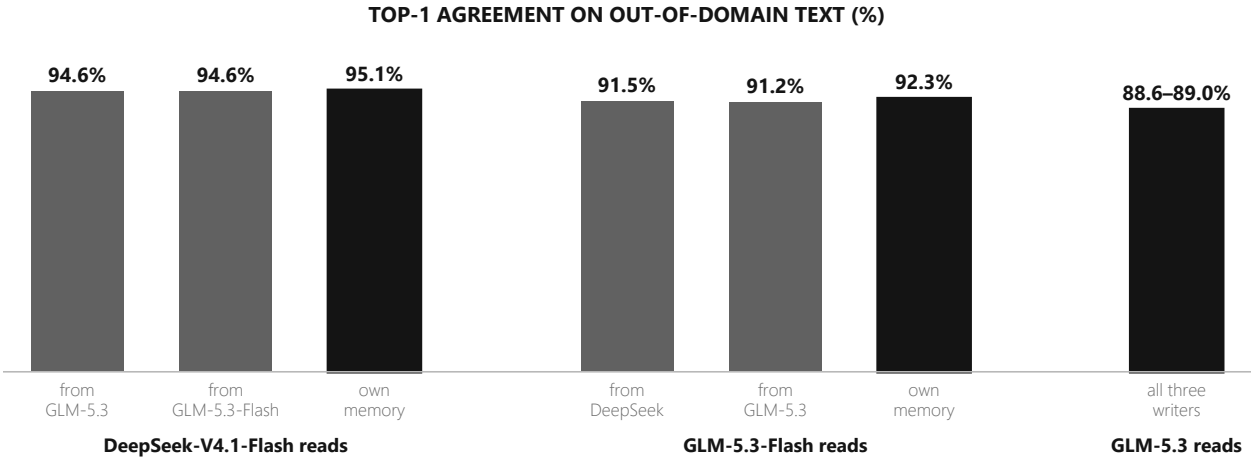


PART	WHAT IT IS	NOTE
Input	The writer's universal cache of a text	Any writer model in the fleet; one translator serves them all
Answer key	The target's own native cache of the same text	Produced by the unchanged target model
Yardstick	The target's own reading of the text that follows	Scored as top-1 agreement; the loss gap is the secondary measure

A translated cache is good when the target makes the same choices from it: **top-1 agreement** *is the share of tokens on which the target picks the same most likely next token as on its own reading. The* **loss gap** *(translated minus native next-token loss, in nats per token) is kept as the technical measure.*

The Target Picks the Same Next Token

The measure a buyer feels first: reading the translated memory, does the target model choose the same next token it would choose reading its own? Across all nine writer → target pairs of the three production models, it does on **89–95%** of tokens of out-of-domain text, scored token by token against the target's own reading of the same text.



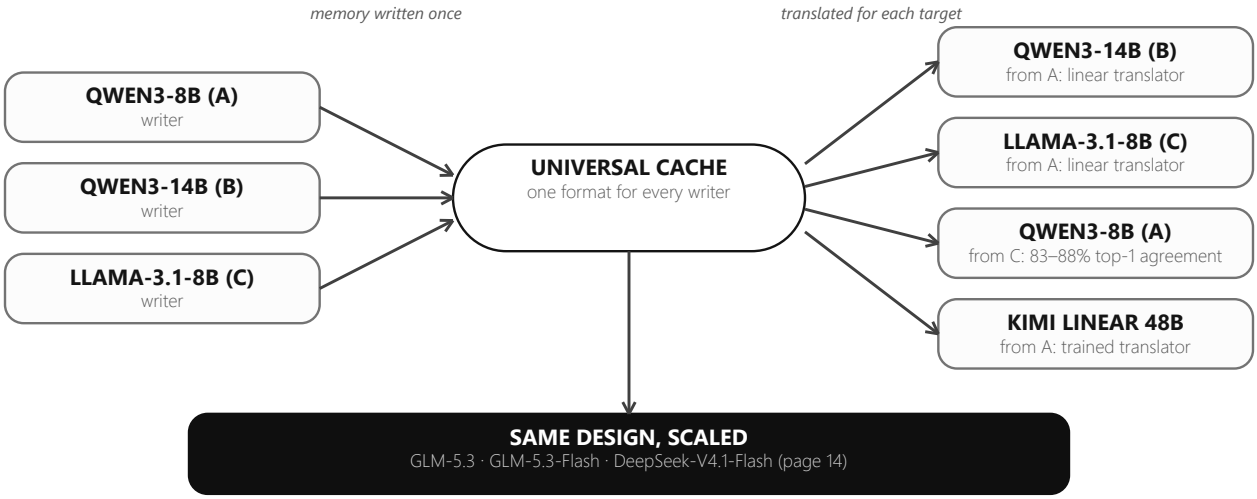
WRITER → TARGET	TOP-1 AGREEMENT (OUT-OF-DOMAIN TEXT)	LOSS GAP PER TOKEN (SECONDARY)
DeepSeek → DeepSeek-V4.1-Flash	**95.1%**	+0.089
GLM-5.3 → DeepSeek-V4.1-Flash	94.6%	+0.100
GLM-5.3-Flash → DeepSeek-V4.1-Flash	94.6%	+0.098
DeepSeek → GLM-5.3-Flash	91.5%	+0.175
GLM-5.3 → GLM-5.3-Flash	91.2%	+0.187
GLM-5.3-Flash → GLM-5.3-Flash	**92.3%**	+0.150
All three writers → GLM-5.3	**88.6–89.0%**	+0.11–0.13

Basis: 20 held-out out-of-domain documents per pair, scored token by token; GLM-5.3's three writers are reported as one range.

On out-of-domain text the target picks the same next token 89–95% of the time, and reading another model's memory costs at most 1.1 points of agreement against reading its own: the writer barely matters, which is what makes the cache universal.

Cross-Family Hops, Proved First on Small Models

Before scaling, the design was proved on the small open models current when these experiments ran: Qwen3-8B (Apr 2025) (A), Qwen3-14B (Apr 2025) (B), Llama-3.1-8B (Jul 2024) (C) and the hybrid Kimi Linear 48B (Oct 2025). Inside the Qwen family the simplest translator — one linear map per target — carries the hop, within 0.021 nats of a native-to-native transfer. Across families the trained translator carries it: reading Llama's translated memory, Qwen3-8B picks its own next token on **83–88%** of tokens, as often as Qwen3-14B agrees with Qwen3-8B on the same text (79–85%).



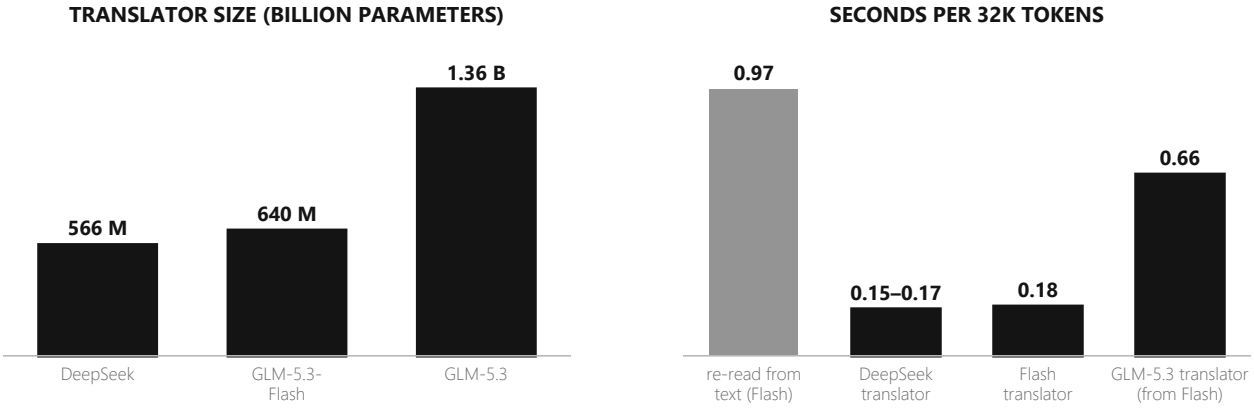
HOP	TRANSLATOR	TOP-1 AGREEMENT	LOSS GAP PER TOKEN (SECONDARY)
A → B (same family)	linear	—	+0.091
A → B → A (same family, chain)	linear	—	+0.026
A → C (cross family)	linear	—	+0.226
C → B (cross family)	linear	—	+0.285
A → C → B (cross family, chain)	linear	—	+0.131
C → A (cross family)	**trained**	**83–88%**	−0.104 to +0.076
A → Kimi Linear 48B (cross family, hybrid)	**trained**	**—**	−0.015

Basis: held-out conversations for each hop; the C → A row spans four held-out sets up to 131k tokens; a chain is a conversation written by the models in turn and read by the last.

Inside the Qwen family a linear translator carries the hop; across families the trained translator gives 83–88% agreement up to 131k tokens — as often as a sibling model agrees with the target.

Scaled to the Three Production Models

The same design runs on GLM-5.3, GLM-5.3-Flash and DeepSeek-V4.1-Flash — models of 321B to 763B parameters. Each has one translator, which reads the memory of any of the three. The translators are small enough to sit beside the prefiller on the prefill box, and a translation is a fraction of what re-reading the text costs.



SERVING MODEL	TRANSLATOR SIZE	TRANSLATION TIME PER 32K TOKENS	TOP-1 AGREEMENT FROM THE OTHER TWO WRITERS
GLM-5.3	**1.36 B**	0.66 s	88.6–89.0%
GLM-5.3-Flash	**640 M**	0.18 s	91.2–91.5%
DeepSeek-V4.1-Flash	**566 M**	0.15–0.17 s	94.6%

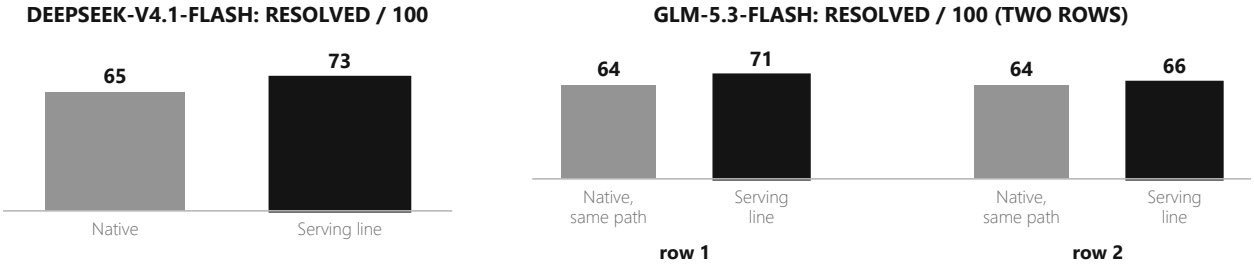
Basis: translation time on one GPU (GLM-5.3 reading GLM-5.3-Flash's memory); agreement on 20 held-out out-of-domain documents (GLM-5.3: one range for all three writers).

Translators of 566 M to 1.36 B parameters serve models of 321B to 763B. A translation takes 0.15–0.17 s per 32k tokens on DeepSeek against 0.97 s to re-read from text, and reading another model's memory costs at most 1.1 points of agreement against reading the model's own.

The Serving Line on Real Work

The translators ship with the UKV serving line: the serving box with its native cache pruned in place. On the 100-task SWE-bench Verified set — real issues from open-source repositories, worked by a Claude Code agent and graded by each project's own hidden tests — the serving line resolves in band with native on every row of record, on both families. On the serving line, no turn at these loads needed a translated cache: 0 of 387.4 M served prompt tokens on the DeepSeek-V4.1-Flash row came from a translator. The last row is Qwen3.8-27B served from translated caches, with Qwen3.5-9B as the prefill model, on 114 τ^2-bench retail tasks.

MEASURED: THE SERVING LINE'S TASK COMPLETION, AND QWEN3.8-27B SERVED FROM TRANSLATED CACHES. CAPABILITY: THE SERVING LINE SERVING TRANSLATED CACHES.



ROW	TASKS COMPLETED (OURS VS NATIVE)	IN BAND (P)
DeepSeek-V4.1-Flash	**73 vs 65**	in band (p = 0.057)
GLM-5.3-Flash, row 1	**71 vs 64**	in band (p = 0.092)
GLM-5.3-Flash, row 2	**66 vs 64**	in band (p = 0.774)
Qwen3.8-27B, prefill model Qwen3.5-9B (Feb 2026)	**91 vs 97**	in band (p = 0.286)

Basis: the 100 SWE-bench Verified tasks, official grading, exact paired (McNemar) test; one DeepSeek task received another repository's text (72 / 100 without it); the Qwen3.8-27B row: 114 τ^2-bench retail tasks, native is the 27B alone.

WHAT THE LOSS GAP MEANS IN PRACTICE (SECONDARY)

LOSS GAP	OF EACH TOKEN'S PROBABILITY	THE AGREEMENT IT CORRESPONDS TO
0.09–0.10 nats	keeps 90–91%	DeepSeek, all three writers, out-of-domain text: same next token on 94.6–95.1% of tokens
0.15–0.19 nats	keeps 83–86%	GLM-5.3-Flash, all three writers, out-of-domain text: same next token on 91.2–92.3% of tokens
−0.10 to +0.08 nats	—	Llama-3.1-8B → Qwen3-8B (page 13): 83–88% agreement; Qwen3-14B against Qwen3-8B: 79–85%

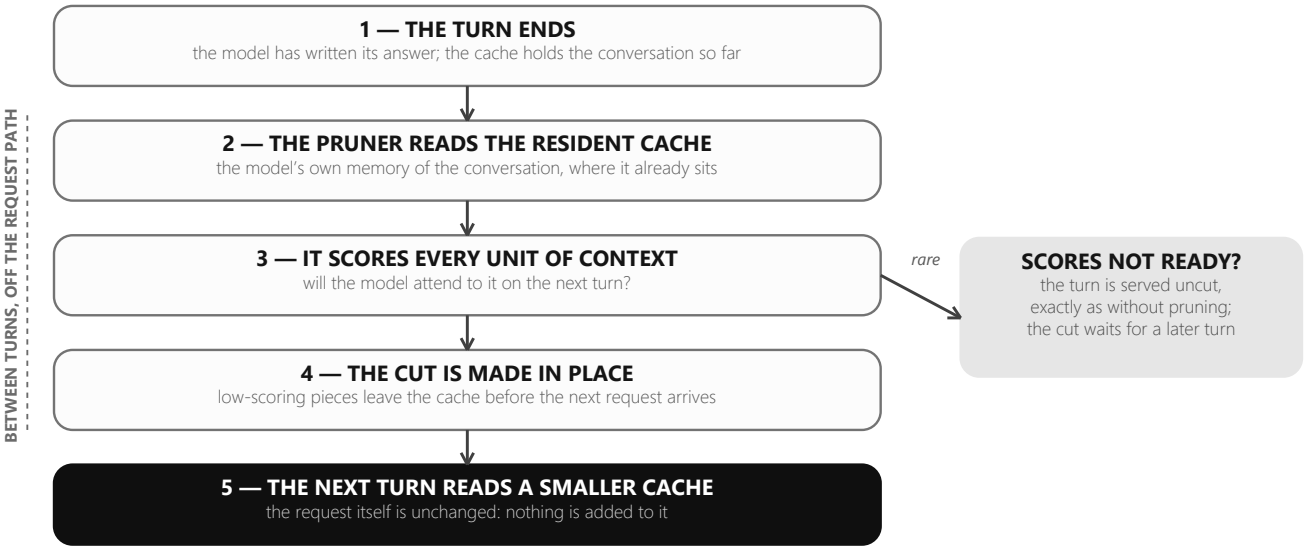
For scale, published 4-bit weight compression of two open models costs +0.064 / +0.13 nats.

PART III · SAFE CACHE PRUNING

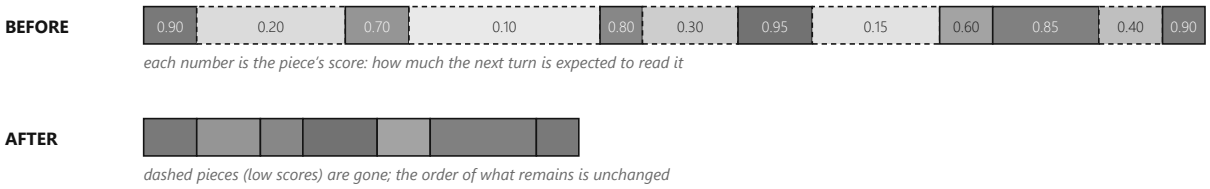
The Pruner, Step by Step

ILLUSTRATION · NOT A BENCHMARK RESULT

One turn of a conversation, from the moment the model finishes its answer to the moment the next request is served. The pruner learned from the model's own attention which earlier text a next turn actually reads; one pruner serves all three models. Its cut thresholds are set per operating point.



One turn, top to bottom. Black is the outcome; grey is the fallback.

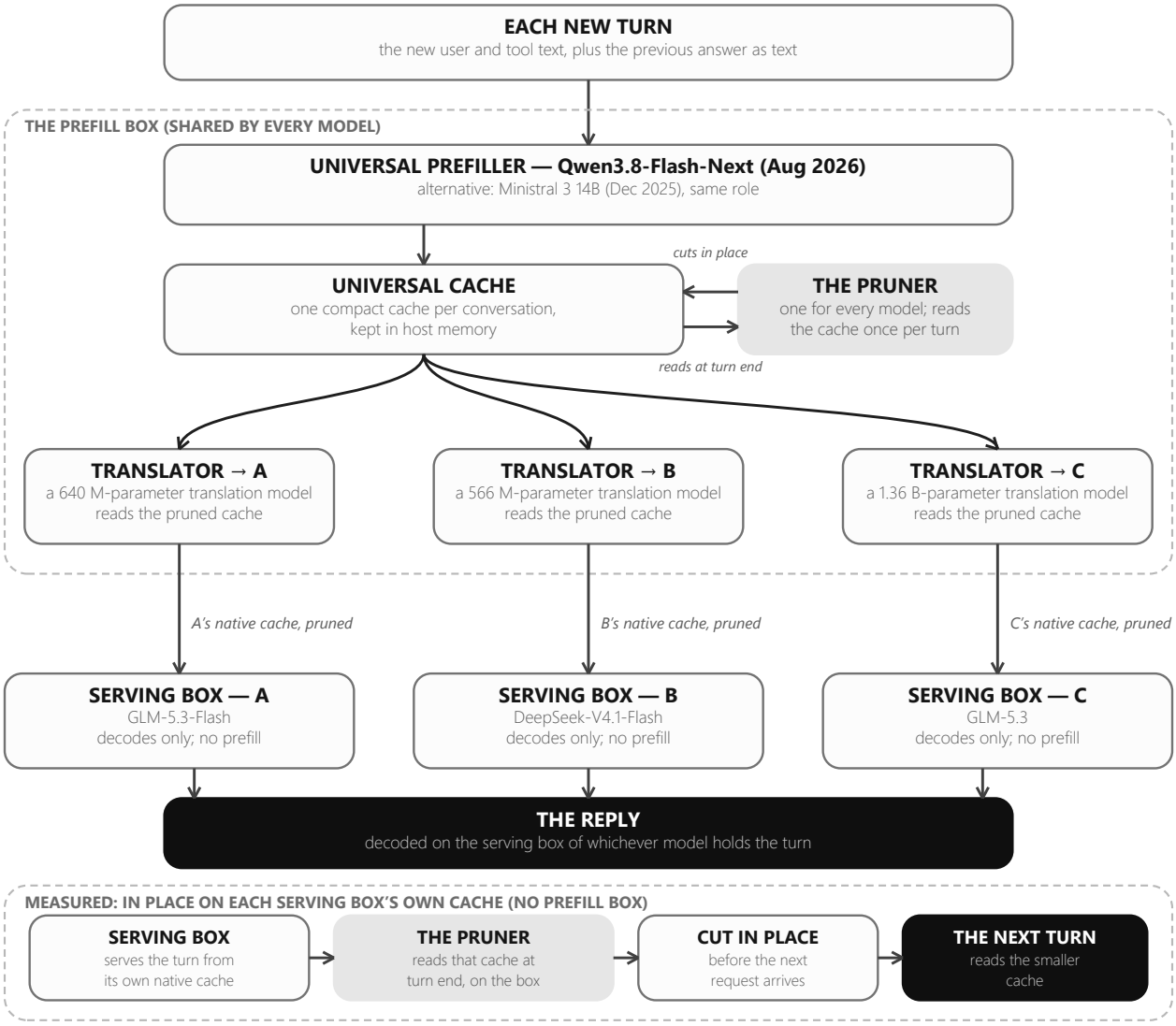


Illustrative values: one conversation's context as pieces, shaded by score (darker = more likely to be read next turn).

#	WHAT HAPPENS	ON THE REQUEST PATH?
1–2	Reads the cache where it already sits; nothing is copied off the model's machine	No
3	Scores are ready before the next request on 99.0% of turns (measured, GLM-5.3-Flash row of record)	No
4	The cut is applied before the next request; when scores are late, the turn is served uncut and the cut waits	No
5	The request, the model and its settings are unchanged; only the cache is smaller	Only the lookup it always does

Where and When It Happens

The system drawing of page 4, read for the pruner. The pruner sits on the **prefill box** and acts on the **universal cache**, once per conversation turn, after the prefiller has added the new turn and before any translator reads the cache. So one cut serves every model: whichever serving box takes the next turn receives a cache that is already pruned, and the serving boxes still only decode.

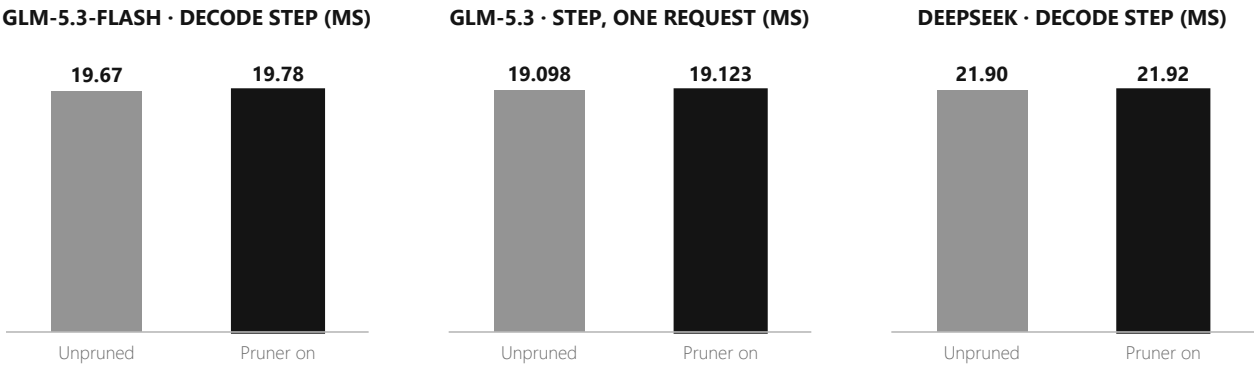


Top: the design — the pruner on the prefill box (projected). Bottom: the measured placement, in place on each serving box. Grey: the pruner.

	WHAT	CLAIM
When	Once per turn, after the answer and before the next request	Measured
Where (design)	Prefill box, on the universal cache, before translation	Projected
Where (today)	Each serving box, in place on its own cache	Measured

Why It Is Safe

Safe means two things a buyer feels: tasks still complete in band, and the model runs at near parity — the pruner does not slow the step that generates each token. The pruner learned from the model's own attention, so it cuts what the model itself was not going to read. It works between turns, off the request path; every cut is served exactly as planned, and turns the pruner did not cut are re-read from scratch no more often than without it.



MODEL	DECODE STEP VS NATIVE	CUT TURNS SERVED EXACTLY	RE-PREFILL
GLM-5.3-Flash	19.78 vs 19.67 ms	1,834 of 1,834	1 of 4,885 turns
GLM-5.3	19.12 vs 19.10 ms	182 of 182	0 of 1,349 turns
DeepSeek-V4.1-Flash	21.92 vs 21.90 ms	373 of 373	0 turns

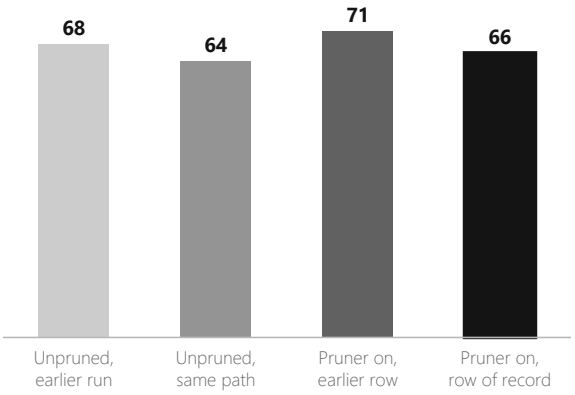
Decode step vs native: *engine time per generation step at matched load, pruner on vs the same box unpruned (DeepSeek-V4.1-Flash: on its row of record, at the same number of conversations in flight; GLM-5.3: one request).* **Re-prefill:** *uncut turns that had to be read again.* **Served exactly:** *the cache the model read equals the planned cut.*

Near parity on all three models: the step with the pruner on is within a fraction of a percent of unpruned, and every cut is served exactly.

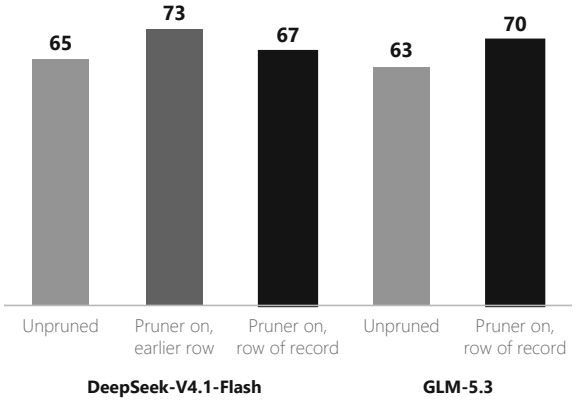
Pruned, Tasks Still Complete

Real coding-agent tasks (100 in each run: real issues from open-source Python projects; a task counts when the agent's patch passes the project's hidden tests), served on our serving boxes with the pruner on, against the same model unpruned. These are serving-box results: the cut is made in place on each model's own cache, which the model read directly (translators were not in the path).

GLM-5.3-FLASH · TASKS COMPLETED / 100



DEEPSEEK AND GLM-5.3 · TASKS COMPLETED / 100

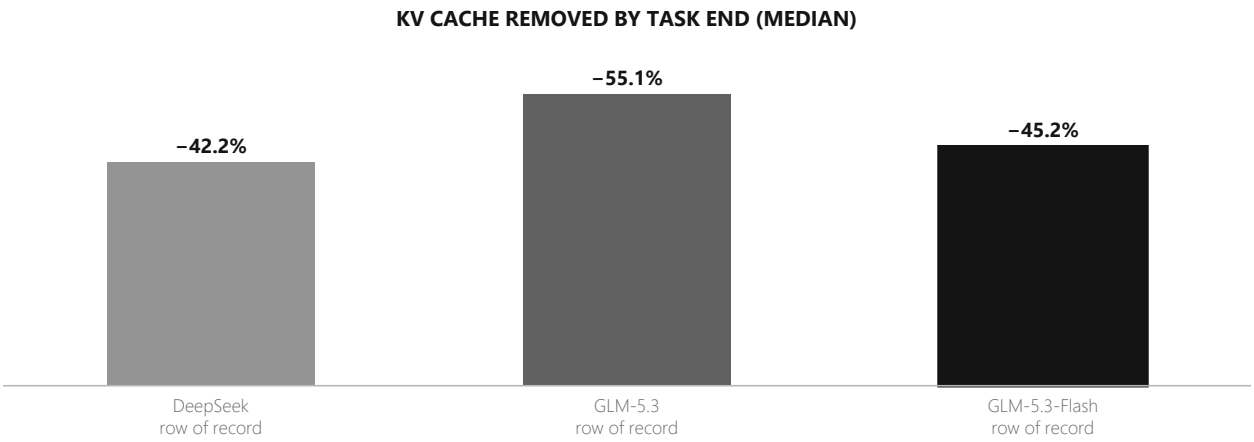


In band: *tasks completed by the pruned run but not the unpruned run, vs the reverse, by the exact two-sided paired (McNemar) test: GLM-5.3-Flash row of record 7 vs 5 (p 0.77); earlier row 10 vs 3 (p 0.09); DeepSeek-V4.1-Flash row of record 8 vs 6 (p 0.79); earlier row 11 vs 3 (p 0.06); GLM-5.3 row of record 13 vs 6 (p 0.17). The DeepSeek-V4.1-Flash earlier row ran the earlier build of the same pruner. Translators wrote 0 of the cache rows served.*

On the rows of record, GLM-5.3-Flash completed 66 of 100 tasks against 64 unpruned on the same path, DeepSeek-V4.1-Flash 67 against 65 and GLM-5.3 70 against 63; on the earlier rows, GLM-5.3-Flash completed 71 against 64 and DeepSeek-V4.1-Flash completed 73 against 65. In band on every row.

How Much Smaller the Cache Is

The first measure of the pruner: the share of a conversation's cache removed by its end, against the same conversation unpruned, turn for turn.



MODEL	BASIS	KV CACHE PER CONVERSATION AT TASK END, MEDIAN (P90)	TASKS COMPLETED (OURS VS SAME PATH)
GLM-5.3-Flash	**Row of record**	**−45.2% (−53.3%)**	**66 / 100 vs 64 / 100**
GLM-5.3-Flash	Row of record (earlier)	−29.5% (−41.8%)	71 / 100 vs 64 / 100
DeepSeek-V4.1-Flash	Row of record	−42.2% (−49.4%)	67 / 100 vs 65 / 100
DeepSeek-V4.1-Flash	Row of record (earlier)	—	73 / 100 vs 65 / 100
GLM-5.3	Row of record	−55.1% (−62.7%)	70 / 100 vs 63 / 100

On the GLM-5.3-Flash row of record, 16.1 points of the reduction came from unused tool schemas and 28.8 from the conversation; on the DeepSeek-V4.1-Flash row of record, 18.5 and 23.4; on the GLM-5.3 row of record, 11.4 and 43.5.

The GLM-5.3-Flash cache is 45.2% smaller by task end on the row of record (median; p90 53.3%), with 66 of 100 tasks completed against 64 unpruned; the DeepSeek-V4.1-Flash and GLM-5.3 rows of record show 42.2% and 55.1%.

PART IV · THE ECONOMICS

How We Measured, and What We Projected

Measured numbers come from four places: our serving boxes (8×H200, one large open model each: GLM-5.3-Flash, DeepSeek-V4.1-Flash and GLM-5.3), offline evaluation of the translators, a real agent benchmark graded by its official grader, and Qwen3.8-27B served on 114 τ^2-bench retail tasks. A result is **in band** when the number of tasks resolved is statistically indistinguishable from native serving on the same tasks (McNemar's test on the paired per-task outcomes). Projected numbers come from the fleet cost model below.

WHAT WAS MEASURED, AND HOW

WHAT	BASIS
Prefill share of step time	8×H200 serving box, GLM-5.3-Flash, 80 conversations, 11,018 sampled engine steps
Translators	One GPU, 32k tokens, against a cold full prefill of 32,640 tokens; offline agreement on every writer → target pair (two out-of-corpus sets of 20 documents)
Serving on real work	100 SWE-bench Verified tasks on one 8×H200 box, 80 and 45 conversations at once (GLM-5.3-Flash, DeepSeek-V4.1-Flash), against native on the same tasks
Cache pruning	In place on each serving box: the GLM-5.3-Flash, DeepSeek-V4.1-Flash and GLM-5.3 rows of record (100 tasks each); step time at matched load
Serving Qwen3.8-27B	114 τ^2-bench retail tasks on an 8×H100 box: the 27B alone on one GPU, standard Mooncake and UKV on three GPUs each, at one new conversation every 3.9 s and at rising load with the same replies on every system; on the same tasks UKV's own replies ran longer than Mooncake's

WHAT THE COST MODEL ASSUMES (EVERY PROJECTED NUMBER)

ASSUMPTION	VALUE
Serving box	8×H200 at $50.84/h
Prefill box	8×H100 at $3.60 per GPU-hour ($28.80/h), 70% utilised
Prefiller cost per turn	new tokens × 31.3 GFLOP per token ÷ 400–560 TFLOP/s per H100, plus the front-end pass, the translator and the cache write; the dense-model rate, applied to the mixture-of-experts pick (assumed)
Cache pool	conversations × 64k tokens of our universal cache (0.516 GB), plus 0.115 GB of the prefiller's fixed recurrent state: 0.631 GB per conversation unpruned
Turn shape	1,196–1,871 new tokens per turn, 62k context, one turn every 26–34 s per conversation
Pruning	The reductions measured on the serving boxes' own caches carry over to the universal cache, conversation for conversation (assumed)
Quality	Tasks resolved, turns per task and reply length unchanged (assumed)

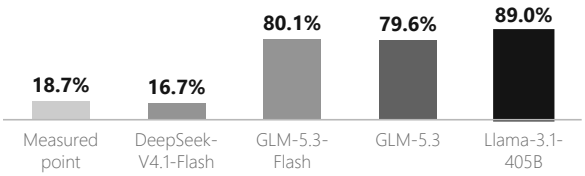
Not priced: *the per-turn token mapping between tokenizers and the network beyond the stated rates.*

What a Smaller Cache Buys

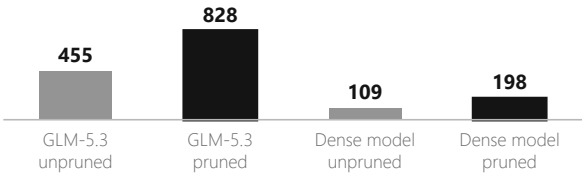
PROJECTED FROM PUBLISHED SPECIFICATIONS · ONE MEASURED POINT, LABELLED

Decode is **memory-bound**: every decode step reads the weights the batch touches plus the cache of every conversation in the batch, and the cache part grows with every conversation and every token. A cache 45.2% smaller therefore cuts the cache's part of every step's memory traffic by 45.2%, and where cache memory is what limits a box it holds about **1.8× the conversations**. What that is worth depends on the cache's share of the step: on a current HGX B300 node (8 × 288 GB, 8 TB/s per GPU) at production batch and 64–128k contexts it is 79.6%–89.0% for GLM-5.3 and a dense frontier model, Llama-3.1-405B (Jul 2024), so time per token falls 35.8%–40.0% at the same batch, or the node holds 1.82× the conversations at the same speed.

CACHE SHARE OF EACH DECODE STEP (64K)



CONVERSATIONS PER NODE (64K)



MODEL (HGX B300, 64K)	CACHE SHARE OF STEP	TIME PER TOKEN	CAPACITY	$ / CONVERSATION-H
GLM-5.3	**79.6%**	**−35.8%**	**1.82×**	**$0.167 → $0.092**
Llama-3.1-405B (dense)	**89.0%**	**−40.0%**	**1.82×**	**$0.697 → $0.383**
GLM-5.3-Flash	80.1%	−28.6%	1.00×	$0.074 → $0.074
DeepSeek-V4.1-Flash	16.7%	−7.2%	1.00×	$0.074 → $0.074

Projected *rows: decode step time = bytes read ÷ memory bandwidth; cache pool = 90% of memory minus weights; at most 1,024 conversations decoding at once per node; the cache cut is the row of record's 45.2%; priced at $9.50 per GPU-hour on demand ($76.00 per node-hour), list prices of 20 October 2026. On Rubin-class parts time per token falls 11.8 → 7.6 ms.* **The measured point** *is last-generation hardware at low batch: weights 328 GB against 75 GB of cache per step and the cache at most 49.7% full, so on that generation the smaller cache shows as room — 80 → 116 conversations in the same memory — at near parity (measured).*

A 45.2% smaller cache cuts the cache's traffic in every decode step by 45.2% and lifts the cache-bound ceiling 1.8× (projected). For GLM-5.3 on current Blackwell nodes that is −35.8% time per token, or 455 → 828 conversations per node; on last-generation hardware at low batch it shows as room in the same memory.

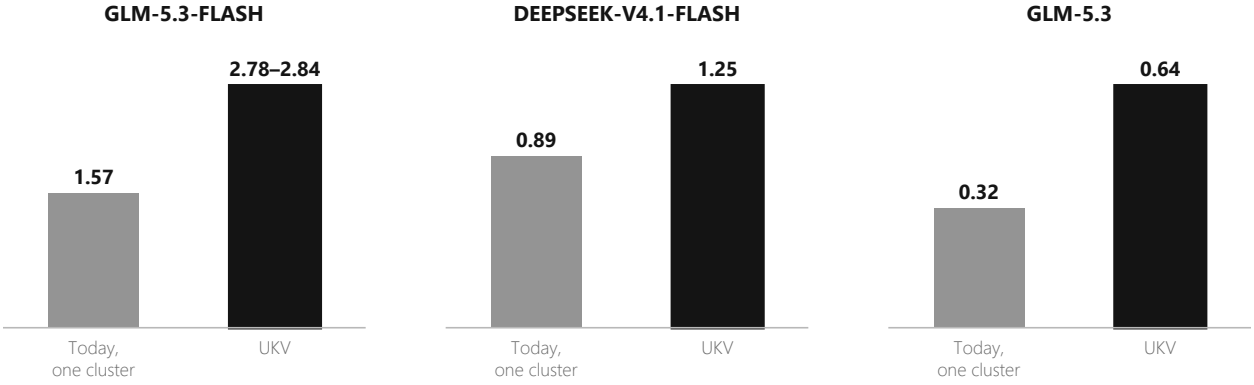
Fleet Economics

PROJECTED · FROM THE COST MODEL, NOT A BENCHMARK RESULT

A serving box that only decodes holds more conversations at the same per-stream speed, because no prefill interrupts its steps. The cost model puts that at **1.4–2.0× today's conversations per box**. One prefill box running Qwen3.8-Flash-Next feeds 900–1,840 conversations — 5.8–7.6, 21–28 or 34–43 serving boxes depending on the model — and is only 1–9% of fleet cost, so conversations per dollar-hour rise by about the same factor: **×1.8 (GLM-5.3-Flash), ×1.4 (DeepSeek-V4.1-Flash), ×2.0 (GLM-5.3)**. These figures are before pruning; what the smaller cache buys on one box is on page 22.

What the prefiller choice changes. With the dense 9 B candidate (appendix) one prefill box would feed 1,100–2,260 conversations, because each turn would cost it 0.08–0.14 instead of 0.10–0.17 GPU-seconds — but its cache has been read only by Qwen3.8-27B, a model of its own family. Conversations per $/h barely move (0.64–2.88 with the dense model, 0.64–2.84 with the pick) because the prefill box is a small share of fleet cost; the prefiller choice moves fan-out, not the price. The prefiller's throughput is a computed rate, not a measurement.

CONVERSATIONS PER $/H, WHOLE FLEET (PROJECTED)



SERVING MODEL	CONVERSATIONS PER BOX	SERVING BOXES PER PREFILL BOX	CONVERSATIONS PER $/H	CHANGE
GLM-5.3-Flash	80 → 155	5.8–7.6	1.57 → 2.78–2.84	**×1.8**
DeepSeek-V4.1-Flash	45 → 65	21–28	0.89 → 1.25	**×1.4**
GLM-5.3	16 → 33	34–43	0.32 → 0.64	**×2.0**

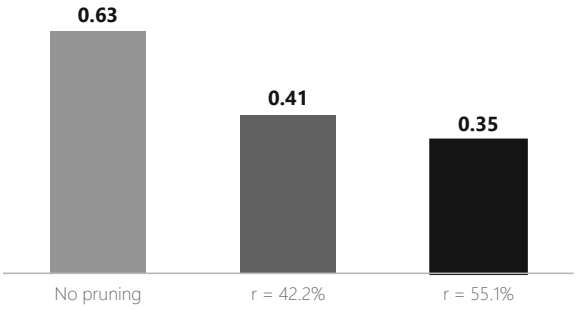
Today *is measured capacity priced at $50.84/h per box;* **UKV** *is projected.* **Break-evens:** *the design wins once a decode-only box holds 2–8% more conversations than today's box; all three clear that by 40–100%. The gain fades toward ×1.1 as conversations grow past about 86k (GLM-5.3-Flash) and 107k (GLM-5.3) resident tokens, against 48.7k and 58k today; DeepSeek-V4.1-Flash holds to about 320k.*

The Pruned Cache Pool

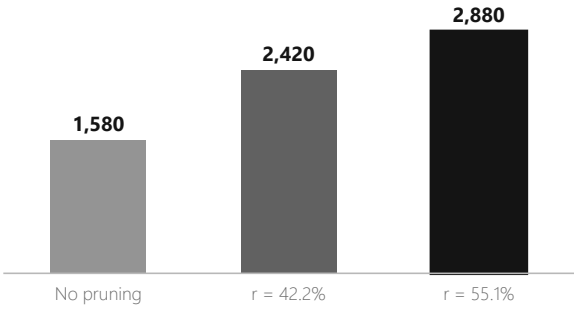
ALL NUMBERS ON THIS PAGE ARE PROJECTED

The prefill box keeps the universal cache of every open conversation in host memory. The cost model sizes that pool at 0.631 GB per conversation (64k tokens of our universal cache plus the prefiller's fixed state, page 21). The pruner shrinks the growing part by the reduction ***r*** measured with the pruner on: 42.2% (DeepSeek-V4.1-Flash, row of record) to 55.1% (GLM-5.3, row of record), with the GLM-5.3-Flash row of record at 45.2%. The fixed state does not shrink.

UNIVERSAL CACHE PER CONVERSATION (GB)



CONVERSATIONS PER TB OF HOST MEMORY



QUANTITY	FORMULA	NO PRUNING	WITH PRUNING
Universal cache per conversation	growing part × (1 − *r*) + fixed state = 0.5159 × (1 − *r*) + 0.1154 GB	0.631 GB	**0.347–0.414 GB (34–45% smaller)**
Conversations held per TB of host memory	a terabyte ÷ cache per conversation	1,580	**2,420–2,880**
Translation at a model switch	twice the translator time per 32k tokens × (1 − *r*), at 64k context; time taken as proportional to the cache	0.30–0.36 s	**0.13–0.21 s**
Conversations one prefill box feeds	unchanged if the pruner's work hides between turns; worst case, its measured per-turn time (0.030 s) added to the box's 0.102–0.172 GPU-s per turn	900–1,840	**760–1,430 (worst case) to 900–1,840**

Assumptions: *the reductions measured on serving models' own caches carry over to the universal cache conversation for conversation; the pool and fan-out figures are the cost model's projections at the prefiller throughput on page 21; the pruner's cost per turn is as measured on a serving box.*

Projected: the universal cache per conversation falls from 0.631 GB to 0.347–0.414 GB, so one TB of host memory holds 2,420–2,880 conversations instead of 1,580, and a model switch translates a smaller cache (0.13–0.21 s instead of 0.30–0.36 s).

One Cluster, Mooncake, UKV

The same fleet three ways. UKV is the last column. Rows marked measured were taken on our boxes or offline; rows marked projected come from the cost model.

	ONE CLUSTER PER MODEL	STANDARD MOONCAKE	UKV
Where prefill runs	On the serving model's own GPUs, mixed into decode steps	A prefill pool per model, running that big model	**One shared pool running Qwen3.8-Flash-Next for every model**
Where decode runs	The same GPUs as prefill	A decode pool per model	**Serving boxes that only decode**
Prefill compute per turn	0.47–1.47 GPU-s on the big model (measured)	The same 0.47–1.47 GPU-s, on the big model's prefill pool	**0.10–0.17 GPU-s on the light model's box (projected)**
Cost of switching models mid-conversation	Full re-prefill on the new model: 0.97–0.99 s per 32k tokens (measured)	Full re-prefill on the new model's prefill pool	**One translator pass: 0.15–0.18 s per 32k tokens, 15–19% of a prefill (measured)**
Cache per conversation	Everything read so far, kept to the end	The same, in the decode pool	**45.2% smaller by task end with the pruner on (measured in place, row of record); 42.2–55.1% on all three models**
Time to first token, Qwen3.8-27B[a]	186 ms (measured)	162 ms (measured)	**104 ms (measured)**
Capacity at the latency target, Qwen3.8-27B[a]	0.26 new conversations per second on one GPU (measured)	0.51 on three GPUs (measured)	**1.02 on three GPUs (measured)**
Conversations per $/h at today's per-stream speed	×1.0 (today)	—[b]	**×1.4–2.0 by model (projected)**[c]
Fidelity vs native	Native	Native	**Same next token as native on 89–95% of out-of-corpus tokens (measured offline); pruned tasks complete in band (measured)**

[a] Serving Qwen3.8-27B: one cluster is the 27B alone on one GPU; standard Mooncake and UKV run on three GPUs each, and UKV's prefill model is Qwen3.5-9B. Time to first token is the median at one new conversation every 3.9 s. The latency target is 90% of replies starting within 500 ms and taking at most 50 ms per token after that, with every system writing the same replies. [b] The cost model prices today's fleet and UKV only; a per-model Mooncake fleet was not modelled. Its decode pools gain like UKV's serving boxes, but it pays for a big-model prefill pool per model. [c] Projected: ×1.8 (GLM-5.3-Flash), ×1.4 (DeepSeek-V4.1-Flash), ×2.0 (GLM-5.3), assuming resolve rate, turns per task and reply length unchanged, the box prices, turn shape and prefiller speed listed on page 21.

UKV moves prefill off the big GPUs onto one light shared model, so the 33.3% that prefill adds to a native box's step time goes away; a model switch costs 15–19% of a full prefill; the target then picks the same next token as on its own reading 89–95% of the time; and the cache each conversation carries is 45.2% smaller with tasks still in band. The cost model projects ×1.4–2.0 conversations per $/h.

APPENDIX

Two Prefiller Options

Why two. The architecture is shown with two prefillers from different labs so that it never depends on one. Qwen3.8-Flash-Next (Aug 2026), from the Alibaba Qwen team, is the pick by the numbers (Qwen Community 1.0 licence). Ministral 3 14B (Dec 2025), from Mistral AI, is the alternative (Apache-2.0). Either one writes the universal cache; the translators, the pruner, the serving boxes and your router are the same.

Why these sizes. The pick runs 6 B of its parameters per token. The alternative is the largest model in Mistral's small dense Ministral line, 14 B dense, with a 262,144-token context — the size in that line built to read a long agent conversation. Prefill per token costs 0.29–0.84× (pick) and 0.62–1.81× (alternative) of a serving model's own prefill (37.7, 37.1 and 108.5 GFLOP for GLM-5.3-Flash, DeepSeek-V4.1-Flash and GLM-5.3); the gain is that prefill leaves the big GPUs, where it adds 33.3% to every step. Both write a per-token cache in enough layers for the translators to read: 12 of the pick's layers, and every one of the alternative's 40 because it is dense. Our proven writer, GLM-5.3-Flash, uses 11, and other models read its cache with 89–95% top-1 agreement.

Why these makers. Among the current open-weight models we considered, these are the two of the right size that write a rich enough per-token cache (page 28). The prefill box is 1–9% of fleet cost with the pick and 2–16% with the alternative, so conversations per $/h are ×1.8 / ×1.4 / ×2.0 and ×1.7 / ×1.4 / ×2.0: **the choice between them barely moves the price**.

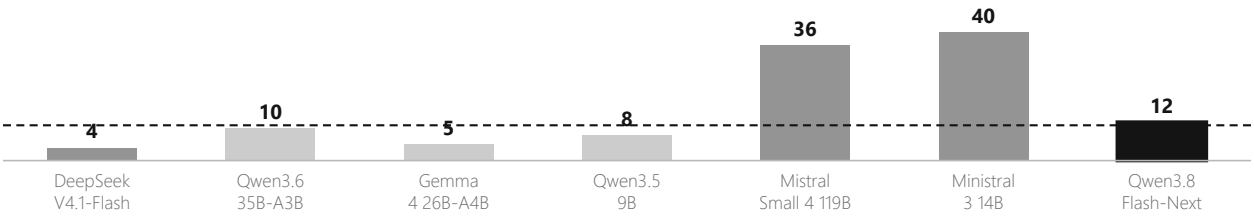
MODEL (RELEASED)	PARAMS TOTAL / ACTIVE	PREFILL GFLOP PER TOKEN	CACHE PER CONVERSATION (OUR CODEC, 64K)	CONVERSATIONS PER PREFILL BOX	LICENCE
Qwen3.8-Flash-Next (Aug 2026)	**125 B / 6 B (+51 B n-gram tables)**	**31.3**	**0.63 GB**	**900–1,840**	**Qwen Community 1.0**
Ministral 3 14B (Dec 2025)	14 B dense	67.3	3.44 GB	450–950	Apache-2.0

Projected with the cost model on page 21; the cache column is our universal cache for each writer, before pruning (page 24).

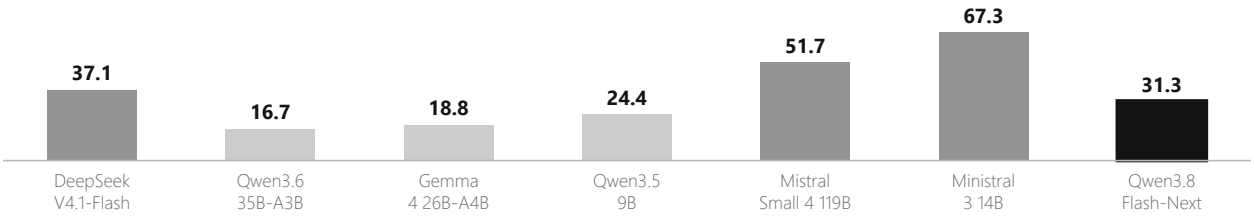
Choosing the Universal Prefiller

The prefiller writes the universal cache that every translator reads, so the first test is **richness**: the model must keep per-token KV over the whole conversation in at least as many attention layers as our proven writer, GLM-5.3-Flash (11), or have a measured writer result of its own. Qwen3.5-9B (8 layers) is the prefill model measured with Qwen3.8-27B, a model of its own family that shares its tokenizer; for a fleet of other labs' models it is held to the layer count. Among the models that pass, the cheapest to prefill wins, then reading quality on long agent work, then licence and context. Candidates are the newest open-weight DeepSeek, Qwen and Gemma models with 3–20 B active parameters, plus the Mistral alternative (page 26) and the larger Mistral Small 4 119B (Mar 2026), checked on 20 October 2026; release dates are from each model card or announcement.

LAYERS THAT WRITE A PER-TOKEN CACHE (OUR PROVEN WRITER: 11, DASHED)



PREFILL GFLOP PER TOKEN AT 64K



Light grey: too few per-token layers; mid grey: enough; black: the pick. A dense model writes a per-token cache in every layer.

The pick: Qwen3.8-Flash-Next — 12 of 48 layers write a per-token cache (our proven writer uses 11) and it costs 31.3 GFLOP per token to prefill, the lowest of the models with enough per-token layers.

The Candidates

MODEL (RELEASED)	PARAMS TOTAL / ACTIVE	PREFILL GFLOP PER TOKEN	CACHE PER CONVERSATION (OUR CODEC, 64K)	CONVERSATIONS PER PREFILL BOX	LICENCE
Qwen3.8-Flash-Next (Aug 2026)	**125 B / 6 B (+51 B n-gram tables)**	**31.3**	**0.63 GB**	**900–1,840**	**Qwen Community 1.0**
Ministral 3 14B (Dec 2025)	14 B dense	67.3	3.44 GB	450–950	Apache-2.0
DeepSeek-V4.1-Flash (Sep 2026)	552 B backbone / 8 B prefill, 16 B decode	37.1	0.88 GB	770–1,600	MIT
Qwen3.6-35B-A3B (Apr 2026)	35 B / 3 B	16.7	0.49 GB	1,480–3,000	Apache-2.0
Gemma 4 26B-A4B (Apr 2026)	26 B / 3.8 B	18.8	0.42 GB	1,360–2,760	Apache-2.0
Qwen3.5-9B (Feb 2026)	9 B dense	24.4	0.74 GB	1,100–2,260	Apache-2.0
Mistral Small 4 119B (Mar 2026)	119 B / 6.5 B	51.7	0.48 GB	580–1,200	Apache-2.0

Projected with the cost model on page 21; cache per conversation is our universal cache at 64k tokens.

Why the others are out. DeepSeek-V4.1-Flash (Sep 2026) is a measured writer, but the cache it was measured with needs a full pass at 37.1 GFLOP per token. Qwen3.6-35B-A3B (Apr 2026, 10) and Gemma 4 26B-A4B (Apr 2026, 5) write a per-token cache in fewer layers than our proven writer's 11; so does Qwen3.5-9B (Feb 2026, 8), whose measured result is with Qwen3.8-27B alone. Mistral Small 4 119B carries 119 B of weights for the same role; the alternative is the right-sized Mistral model.

How the columns are computed. *GFLOP per token = twice the active parameters + attention over the context in the full-attention layers at 64k tokens. Per-token layers are counted from each model's published configuration. The pick is from the DeepSeek, Qwen and Gemma lines; the alternative is the Ministral model of the right size.*

The other models named in this document. *Serving models, measured: GLM-5.3 (Aug 2026), GLM-5.3-Flash (Aug 2026), DeepSeek-V4.1-Flash (Sep 2026), Qwen3.8-27B (Aug 2026). Small-model experiments (page 13), the open models current when they ran: Qwen3-8B (Apr 2025), Qwen3-14B (Apr 2025), Llama-3.1-8B (Jul 2024), and Kimi Linear 48B (Oct 2025) as a reader. Release months for every model named were checked against its model card or the lab's announcement on 20 October 2026.*

Glossary

WORD	DEFINITION
Universal cache	One compact cache format for a conversation that every model in the fleet can read, through its translator.
Universal prefiller	The light shared model (Qwen3.8-Flash-Next) that reads every conversation for every serving model and writes the universal cache.
Translator	A small supervised translation model, one per serving model, that turns the universal cache into that model's own native cache — elsewhere in our technical notes called a decoder.
KV cache	The per-token working memory a model builds while reading a conversation and attends to while writing.
Native cache	A model's own KV cache, in a layout only that model can read.
Writer / target	The writer is the model whose memory is in the universal cache; the target continues from the translated cache and is never changed.
Prefill	Reading the conversation so far and building the KV cache.
Decode	Generating the reply, one token at a time, from the KV cache. In UKV only the serving boxes decode.
Serving box	One multi-GPU machine running one large model. In UKV it only decodes.
Prefill box	The machine that holds the universal prefiller, the translators, the pruner and the cache pool, and hands translated rows to the serving boxes.
Mooncake	A serving design that splits each model into a prefill pool and a decode pool and ships the KV cache between them.
Model switch (reroute)	Moving a conversation to a different model between turns; with UKV, a different translator reading the same cache.
Top-1 agreement	The share of tokens on which the target, reading a translated cache, picks the same most likely next token as reading its own.
Loss gap	Next-token loss reading the translated cache minus reading the native cache, in nats per token; the secondary measure.
Head	What every conversation starts with: the system prompt and the tool schemas.
Pruner	One small learned model, shared by every serving model, that scores every piece of the context — system material and tool schemas included — by whether the next turn will attend to it, and removes the rest in place, between turns, off the request path.
Cut in place	Removing pieces from the cache where it sits, without copying it elsewhere or re-reading the conversation.
Near parity	The generation step with the pruner on runs within a small margin of the same box unpruned.
Serving line	The serving box with its native cache pruned in place — the line that will serve translated caches.
Row of record	A full run of real benchmark tasks, graded, on the serving setup we stand behind.
Model family, cross-family hop	A line of models from one maker (Qwen, Llama, GLM, DeepSeek, Kimi); a hop translates between two families.

Cosine	How closely the target's attention output on the translated cache matches its own; 1.000 is identical.
Held-out, out-of-corpus	Text never used to train the translator; out-of-corpus text also comes from outside the training sources.
SWE-bench Verified	A public benchmark of real issues from open-source Python repositories, graded by each project's hidden tests.
In band	Tasks resolved statistically indistinguishable from native serving, or from the same model unpruned, on the same tasks (exact paired McNemar test).
Measured / projected	Measured: on our boxes, offline or on the benchmark. Projected: from the fleet cost model and its listed assumptions.
GPU-second	One GPU busy for one second.