

On 22 September 2026 Epoch AI, a research group that studies the trajectory of artificial intelligence, published "The plunging price of thought", an estimate that the cost of reaching a given level of performance on its benchmarks has fallen about 47% a quarter since 2023 — roughly thirteen-fold a year. The same week, the price sheets of the large model vendors showed how little a single "price per token" now describes: one OpenAI model carries separate rates for input, cached input, cache writes, output, long prompts, patient batch work and fast service. Behind that list sits the physics of a single request. A model reads a prompt in one parallel sweep, then writes its answer one token at a time, re-reading its weights and a growing per-conversation memory, the key-value cache, at every step. Serving many users at once shares the weights but not the caches, so the number of users a chip can hold, the speed each of them sees and the length of their conversations all draw on the same pool of memory. Fewer bits per number, smarter batching and smaller caches are how the industry has loosened that constraint, and the rates on a price sheet line up closely with it.

1. A price that falls thirteen-fold a year

Epoch's report, by Luke Emberson and David Roodman, measures what it calls the price of thought: the cost of actually answering benchmark questions at a given score, using whichever model can reach that score most cheaply at each date. Its central estimate, from five benchmarks covering mathematics, the hard sciences and games of skill, is a fall of "about 47% per quarter since 2023, or 13× per year". The decline is fastest for performance that has only just been reached — 66% a quarter, or 75-fold a year, on average across its five primary benchmarks — and slows to 32% a quarter, about 4.7-fold a year, two years later. Epoch offers one tentative reason for the pattern: "when a performance level is first achieved, AI companies can briefly charge a premium for it, before competition and technological improvement quickly drive down the price."

Its most vivid example is a single pair of OpenAI models. Epoch estimates that o3, released on 31 January 2025, reached 75% on GPQA Diamond, a multiple-choice examination in doctoral-level physics, chemistry and biology, at an average of 30 cents a question; just under eighteen months later GPT-5.6 Luna matched the score for \$0.0004. Epoch calls that "a 725-fold drop in the price of thought in under 18 months". It is an illustration chosen from the extreme end, and the report's average is the thirteen-fold figure. The report also describes the decline as faster than for "any other transformative technology in history", a comparison that is Epoch's own.

The method matters for reading the number. Because Epoch measures the cost of reaching a score rather than the price of a token, a model that answers in fewer tokens looks cheaper even at the same tariff. For open models that no vendor sells as a service, Epoch used "the cost of their rented hardware" in place of a price. And the report is candid about whom its average describes:

Figure 1. How fast the cost of a fixed level of performance falls, by how new that level is (Epoch AI, average of five benchmarks)



Source: Epoch AI, Luke Emberson and David Roodman, 'The plunging price of thought', published 22 September 2026, section 'Key takeaways' (read 26 September 2026). Per-quarter equivalents given by Epoch: 66%, 47% and 32%. Epoch measures the cost of answering benchmark questions with the cheapest model able to reach each score; it states that its bottom-line numbers 'should not be read as exact'.

▼ Table view

Figure 1. How fast the cost of a fixed level of performance falls, by how new that level is (Epoch AI, average of five benchmarks)

Performance level	Fall in cost per year
Just reached (state of the art at debut)	75× per year
All levels, 2023 to 2026 (headline estimate)	13× per year
Two years after debut	4.7× per year

we implicitly posit an AI user who relentlessly searches for the most cost-effective model for each task, when real users do not switch models so often, and therefore do not reap quite the same savings.

Epoch adds that its "data are incomplete and noisy: the timeframe is barely three years", and that while its bottom-line numbers are "reasonably representative of reality, they should not be read as exact".

2. Two readings that mislead

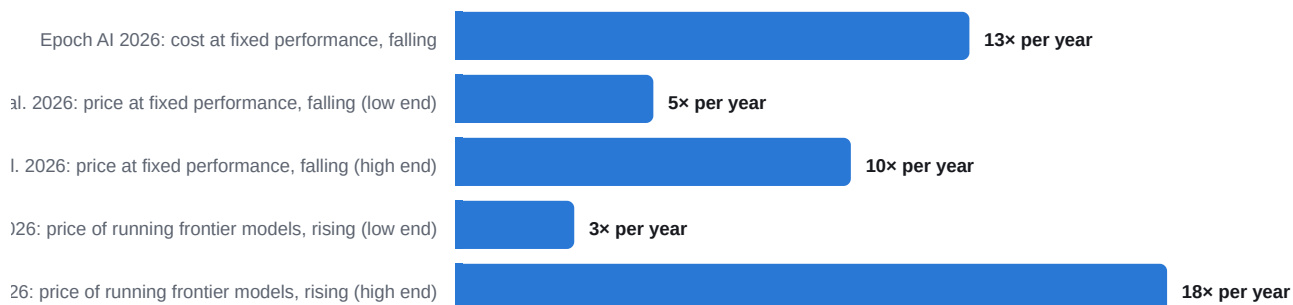
2.1 "The AI people use got thirteen times cheaper"

The headline describes a fixed level of performance bought from the cheapest capable model. It does not describe the strongest models. Hans Gundlach, Jayson Lynch, Matthias Mertens and Neil Thompson of MIT FutureTech, in "The Price of Progress" (arXiv 2511.23455, revised 23 March 2026), find the same kind of decline — the price of a given level of benchmark performance falling "around 5× to 10× per year" for frontier models between April 2024 and November 2025 — and in the same abstract estimate that "the price of running frontier models is rising between 3× to 18× per year due to bigger models and larger reasoning demands". The two statements are compatible: the cost of last year's frontier collapses while the cost of this year's frontier climbs.

Earlier estimates spread widely. An Epoch data insight of 12 March 2025, by Ben Cottier and colleagues, found the fall in per-token price at fixed performance "ranging from 9x to 900x per year" depending on the benchmark and threshold, and noted that "the fastest price drops in that range have occurred in the past

year, so it's less clear that those will persist". An essay by Guido Appenzeller for Andreessen Horowitz in November 2024 put it at ten-fold a year on one knowledge test.

Figure 2. Published estimates of how fast AI prices change each year



Sources: Epoch AI, 'The plunging price of thought', 22 September 2026, 'Key takeaways'; Gundlach, Lynch, Mertens and Thompson, 'The Price of Progress', arXiv 2511.23455 v2, 23 March 2026, abstract. The first three rows are falls; the last two are rises. The studies use different benchmarks, periods and methods, so the bars are not directly comparable.

▼ Table view

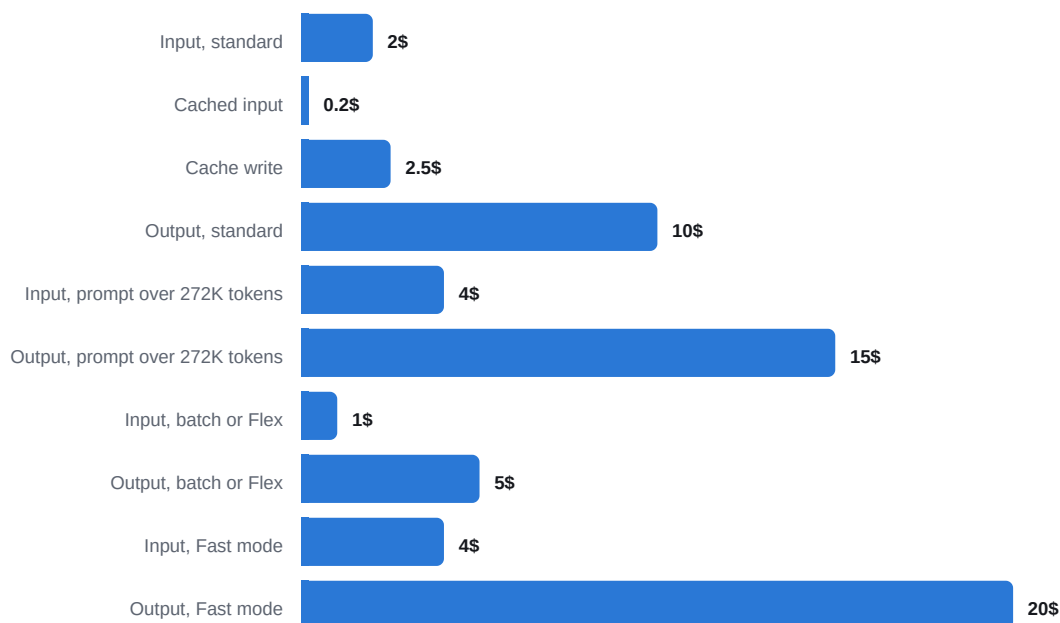
Figure 2. Published estimates of how fast AI prices change each year

Estimate	Change per year
Epoch AI 2026: cost at fixed performance, falling	13× per year
Gundlach et al. 2026: price at fixed performance, falling (low end)	5× per year
Gundlach et al. 2026: price at fixed performance, falling (high end)	10× per year
Gundlach et al. 2026: price of running frontier models, rising (low end)	3× per year
Gundlach et al. 2026: price of running frontier models, rising (high end)	18× per year

A price can also rise with no change to the model. Google launched Gemini 3.8 Flash on 2 September 2026 at an introductory rate of \$0.75 per million input tokens and \$3.75 per million output tokens; its announcement states that from 1 January 2027 "\$1.50/1M input tokens and \$7.50/1M output tokens will apply".

2.2 "A token has a price"

OpenAI's pricing page, read on 26 September 2026, lists its middle model, GPT-6 Sol, at \$2 per million input tokens and \$10 per million output tokens. Input the service has already processed and cached costs \$0.20; writing to that cache costs \$2.50. Once a prompt exceeds 272,000 input tokens every rate rises — input to \$4, output to \$15. Batch and "Flex" processing, which trade speed for price, halve the standard rates, and "Fast mode" doubles them.

Figure 3. One model, ten prices: OpenAI GPT-6 Sol, US dollars per million tokens

Source: OpenAI API pricing page, developers.openai.com/api/docs/pricing, read 26 September 2026 (short context is defined there as ' $\leq 272K$ input tokens'). OpenAI renamed its Priority tier 'Fast mode' on 30 July 2026.

▼ Table view

Figure 3. One model, ten prices: OpenAI GPT-6 Sol, US dollars per million tokens

Rate	Price per million tokens
Input, standard	2\$
Cached input	0.2\$
Cache write	2.5\$
Output, standard	10\$
Input, prompt over 272K tokens	4\$
Output, prompt over 272K tokens	15\$
Input, batch or Flex	1\$
Output, batch or Flex	5\$
Input, Fast mode	4\$
Output, Fast mode	20\$

Yet in arithmetic an input token and an output token cost about the same. A Google paper on serving large models, discussed in section 3, puts the work of a decoder-only model at " $2N$ matmul FLOPs in the forward pass per token seen", for a model of N parameters, whether the token is being read or written. The five-to-one gap between output and input prices comes from somewhere else: from how the work is scheduled and what it has to read.

3. The idea: one request, two phases, one pool of memory

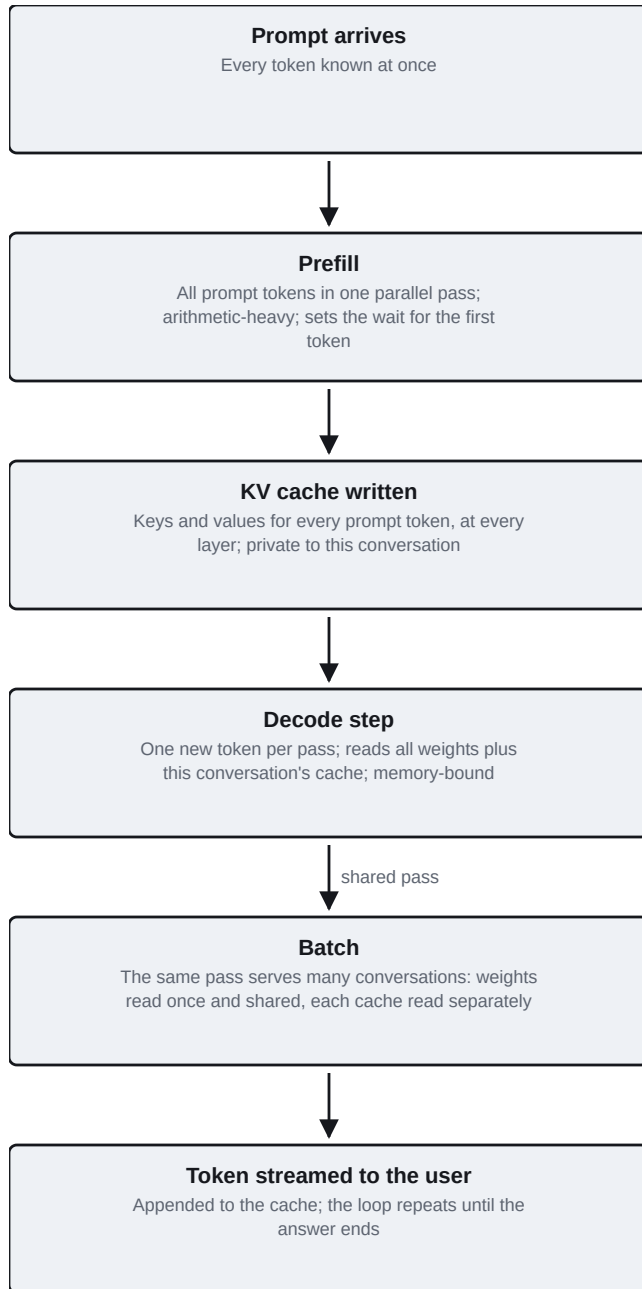
3.1 Reading and writing

A request to a language model is served in two phases that behave almost nothing alike. In the first, the model reads the prompt. Every token of it is available at once, so the whole prompt can be pushed through the network in a single pass, or a few large ones, and the chip's arithmetic units are kept busy. In the second, the model writes its answer one token at a time, and each new token needs a complete pass through the model before the next can begin.

The names now standard for the two phases appear in "Efficiently Scaling Transformer Inference" by Reiner Pope and colleagues at Google (arXiv 2211.05102, 9 November 2022), which splits the latency of a request into "the time to process the input tokens present at the start of the inference", which the authors call "prefill", and "the time to autoregressively generate output tokens", which they call "decode". Prefill largely sets how long a user waits for the first word — the "time to first token" of the trade. Decode sets how quickly the rest follows.

The paper's headline results show how differently the two behave on the same machine. Serving Google's 540-billion-parameter PaLM on TPU v4 chips, it reported "a low-batch-size latency of 29ms per token during generation (using int8 weight quantization) and a 76% MFU during large-batch-size processing of input tokens". MFU, model FLOPs utilisation, is the share of the chips' peak arithmetic doing useful work. Reading prompts in bulk kept three-quarters of it busy; generation was reported as a delay per token, because in generation the arithmetic is rarely what binds.

The reason is the roofline logic of accelerator design. To generate a token for a single user, a chip must stream every weight the model uses from its high-bandwidth memory into its arithmetic units, and it performs only about two operations for each weight it reads. That is far below the ratio of operations to bytes at which any current accelerator's arithmetic becomes the constraint, so a single user's decode is paced by memory bandwidth, not by arithmetic.

Figure 4. One request's journey through a serving system

Schematic, after the two-phase account in Pope et al., 'Efficiently Scaling Transformer Inference', arXiv 2211.05102, 9 November 2022, section 2. Systems that separate the phases run prefill and decode on different machines and move the cache between them.

▼ Table view

Figure 4. One request's journey through a serving system — stages

#	Stage	Note
1	Prompt arrives	Every token known at once
2	Prefill	All prompt tokens in one parallel pass; arithmetic-heavy; sets the wait for the first token
3	KV cache written	Keys and values for every prompt token, at every layer; private to this conversation
4	Decode step	One new token per pass; reads all weights plus this conversation's cache; memory-bound

#	Stage	Note
5	Batch	The same pass serves many conversations: weights read once and shared, each cache read separately
6	Token streamed to the user	Appended to the cache; the loop repeats until the answer ends

Figure 4. One request's journey through a serving system — connections

From	To	Label
Prompt arrives	Prefill	
Prefill	KV cache written	
KV cache written	Decode step	
Decode step	Batch	shared pass
Batch	Token streamed to the user	

3.2 The cache that makes decoding possible, and expensive

Decoding has a second thing to read. To choose each new token, a Transformer's attention layers compare it with every token that came before. Rather than recompute what it needs about those earlier tokens at every step, a serving system keeps, for each earlier token and at every layer, two vectors called a key and a value. The store is the key-value, or KV, cache. Pope and colleagues describe it as "the attention key and value tensors of each layer, which we refer to as the KV cache", which "must also be stored in memory for the duration of decoding".

Its cost was diagnosed three years earlier. Noam Shazeer of Google wrote in "Fast Transformer Decoding: One Write-Head is All You Need" (arXiv 1911.02150, 6 November 2019) that

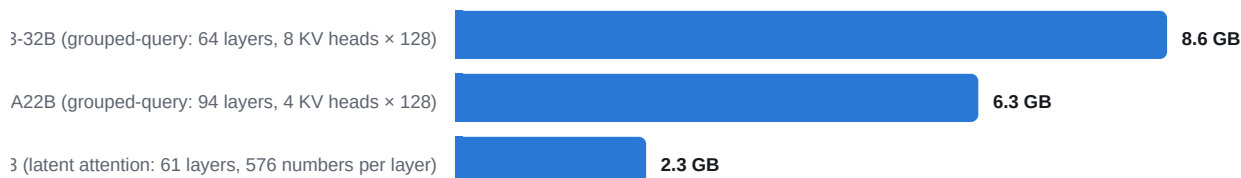
the speed of incremental Transformer inference on modern computing hardware is limited by the memory bandwidth necessary to reload the large "keys" and "values" tensors which encode the state of the attention layers.

His remedy, multi-query attention, lets all of a layer's attention heads share one set of keys and values, which he reported could "indeed be much faster to decode, and incur only minor quality degradation from the baseline". Later designs sit between the extremes. Grouped-query attention (Joshua Ainslie and colleagues, Google, arXiv 2305.13245, May 2023) shares keys and values within groups of heads, because multi-query attention "can lead to quality degradation". DeepSeek-V2's multi-head latent attention (arXiv 2405.04434, May 2024) stores a compressed version; DeepSeek reported that, compared with its earlier 67-billion-parameter model, V2 "reduces the KV cache by 93.3%".

How large the cache grows can be worked out from a model's published configuration: layers $\times$ key-value heads $\times$ numbers per head $\times$ 2 (a key and a value) $\times$ bytes per number. Alibaba's Qwen3-32B, released in April 2025, has 64 layers and 8 key-value heads of 128 numbers each. At two bytes a number that is 262,144 bytes — a quarter of a megabyte — for every token of every conversation. A conversation at the model's

native limit of 32,768 tokens needs about 8.6 gigabytes of cache; at the 131,072 tokens the model card says it supports with an extension method, about 34 gigabytes. The weights themselves, 32.8 billion parameters at two bytes each, occupy about 65.5 gigabytes.

Figure 5. Key-value cache for one 32,768-token conversation, from each model's published configuration (16-bit values)



Arithmetic on the config.json files published on Hugging Face (read 26 September 2026): layers × KV heads × head dimension × 2 × 2 bytes per token, times 32,768 tokens. DeepSeek-V3 caches a compressed latent of 512 + 64 numbers per layer; had it kept full keys and values for all 128 heads, the same conversation would need about 164 GB. The simplification ignores cache precision (8-bit caches halve every figure), allocator overhead and how a cache is replicated when a model is split across chips.

▼ Table view

Figure 5. Key-value cache for one 32,768-token conversation, from each model's published configuration (16-bit values)

Model (cache design)	Cache for one conversation
Qwen3-32B (grouped-query: 64 layers, 8 KV heads × 128)	8.6 GB
Qwen3-235B-A22B (grouped-query: 94 layers, 4 KV heads × 128)	6.3 GB
DeepSeek-V3 (latent attention: 61 layers, 576 numbers per layer)	2.3 GB

3.3 Batching: shared weights, private caches

Because every decode step reads the whole model anyway, a server can produce the next token for many conversations in the same pass. The weights are read once and shared across the batch; each conversation adds only its own cache. Pope and colleagues put the asymmetry in a clause:

(unlike the weights) the KV cache is unique for each sequence in the batch.

Batching is older than language models — a 2011 Google paper on running speech-recognition networks on ordinary processors, by Vincent Vanhoucke, Andrew Senior and Mark Mao, listed "batching of the computation" among the techniques behind its speed-up — but two systems made it the core of language-model serving. Orca, from Seoul National University and FriendlyAI (Gyeong-In Yu and colleagues, OSDI, July 2022), observed that under the batching then in use "requests that have finished earlier than other requests in a batch cannot return to the client, while newly arrived requests have to wait until the current batch completely finishes". Its answer was "iteration-level scheduling": the batch is re-formed at every step, so a finished answer leaves and a new request joins at once. The authors reported a "36.9× throughput improvement at the same level of latency" over NVIDIA's FasterTransformer on a 175-billion-parameter model — their own measurement, against a baseline they chose.

vLLM, from a team led from the University of California, Berkeley (Woosuk Kwon and colleagues, arXiv 2309.06180, SOSP 2023), attacked the memory side. Profiling the systems then available, its authors found that "only 20.4% - 38.2% of the KV cache memory is used to store the actual token states", the rest lost to reservations and fragmentation. Their PagedAttention stores each cache in small fixed-size blocks, "inspired by the classical virtual memory and paging techniques in operating systems", and they reported throughput "2-4×" that of the systems they compared against "with the same level of latency".

The trade-off batching creates is spelled out by SemiAnalysis, an industry research firm, in its description of InferenceMAX, a serving benchmark it launched on 9 October 2025. It defines throughput as "the rate at which each GPU can process tokens (tok/s/gpu)" and interactivity as "the rate at which tokens are generated for each individual user (tokens/sec/user)", and writes:

Large batches enable better GPU utilization and higher token throughput, but they split available resources across more requests, slowing down token processing per user.

3.4 Quantisation: fewer bytes per number

The fourth lever is to store numbers in fewer bits. The idea predates language models: Vanhoucke and colleagues' 2011 paper reported "a 4× speedup over an aggressively optimized floating-point baseline at no cost in accuracy" for one speech-recognition system, using eight-bit fixed-point arithmetic among other techniques, and Benoit Jacob and colleagues at Google described in December 2017 "a quantization scheme that allows inference to be carried out using integer-only arithmetic" (arXiv 1712.05877), paired with a training procedure designed to preserve accuracy. For large language models the step came in August 2022, when Tim Dettmers and colleagues reported that with their LLM.int8() method "a 175B parameter 16/32-bit checkpoint can be loaded, converted to Int8, and used immediately without performance degradation" (arXiv 2208.07339). The method keeps a small set of outlier features at sixteen bits while "more than 99.9% of values are multiplied in 8-bit". Two months later GPTQ (Elias Frantar and colleagues, arXiv 2210.17323) reported cutting weights to "3 or 4 bits per weight, with negligible accuracy degradation" on models of 175 billion parameters. Whether quantisation costs quality depends on the method, the bit width, the model and the test.

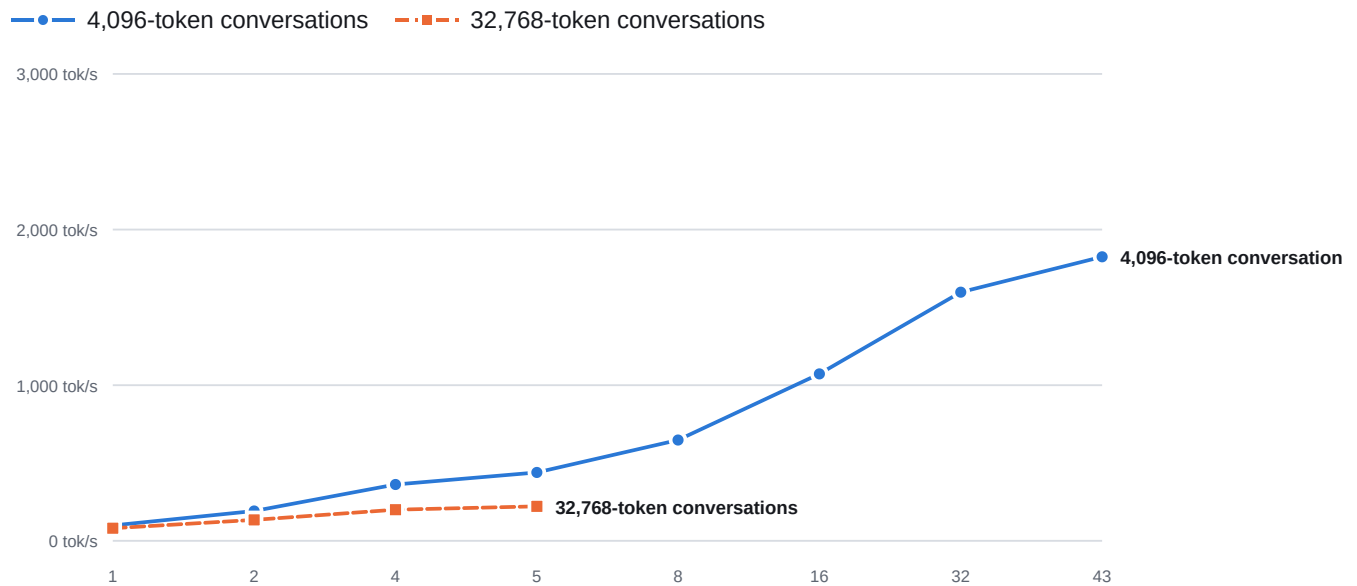
For serving, quantisation does two things at once. Halving the bytes per weight halves the traffic each decode step must read, and it frees memory for more caches. Z.ai's GLM-5.3 illustrates the first: its default download moved from sixteen-bit to eight-bit weights, the same 753 billion parameters in about half the bytes.

3.5 The pull between latency, throughput and context length

Put the pieces on one chip. NVIDIA's H100 has 80 gigabytes of memory and 3.35 terabytes a second of memory bandwidth, according to NVIDIA's product page. With Qwen3-32B's weights at eight bits, about 47 gigabytes remain for caches: room for about 43 conversations of 4,096 tokens, or 5 of 32,768. With sixteen-bit weights, one conversation of 32,768 tokens fits.

On the idealised assumption that each decode step is paced only by reading the weights once plus every conversation's cache — which ignores the arithmetic, the attention computation, the software's own reserve and every overhead of a real server — one user alone would receive about 99 tokens a second. Forty-three users sharing each pass would receive about 42 each, but about 1,800 between them. Five users with 32,768-token conversations would receive about 44 each and about 220 between them.

Figure 6. Idealised decode on one H100, Qwen3-32B with 8-bit weights: total tokens per second as users are added

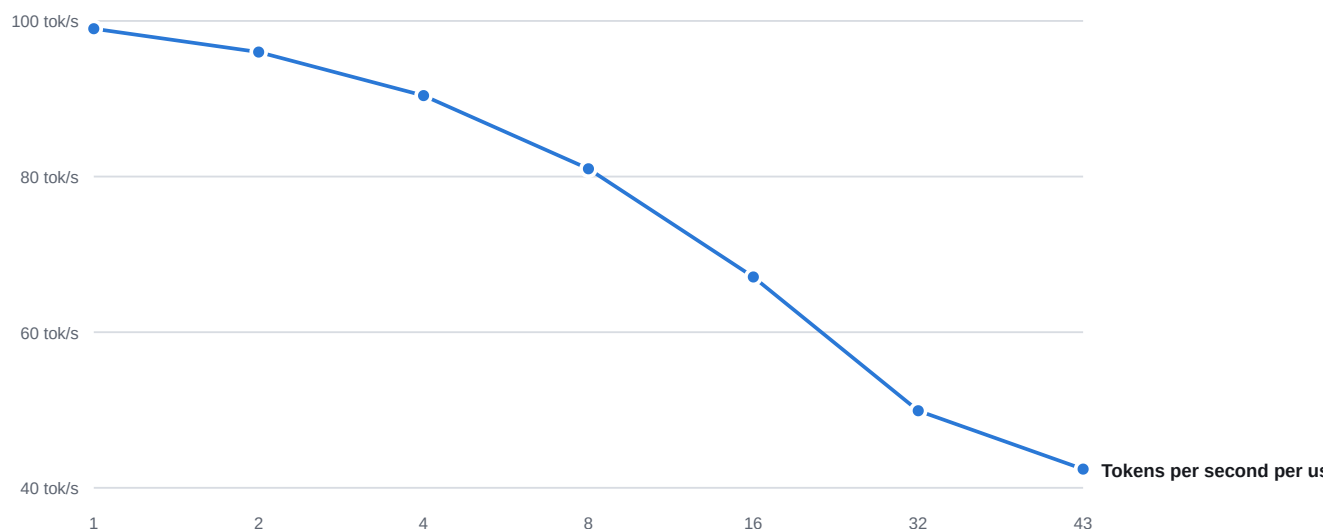


The show's own arithmetic (memory traffic only): $\text{step time} = (\text{weights } 32.8 \text{ GB} + \text{users} \times \text{cache per conversation}) \div 3.35 \text{ TB/s}$. Cache per token 262,144 bytes (Qwen3-32B config.json); H100 80 GB and 3.35 TB/s (NVIDIA product page, read 26 September 2026). Only five 32,768-token conversations fit beside the weights, so that series stops. Real servers achieve less; the shape, not the level, is the point.

▼ Table view

Figure 6. Idealised decode on one H100, Qwen3-32B with 8-bit weights: total tokens per second as users are added

Conversations in the batch	4,096-token conversations	32,768-token conversations
1	99 tok/s	81 tok/s
2	191.9 tok/s	134.2 tok/s
4	361.6 tok/s	199.6 tok/s
5	439.3 tok/s	221.2 tok/s
8	648.1 tok/s	—
16	1,073.2 tok/s	—
32	1,597.1 tok/s	—
43	1,825 tok/s	—

Figure 7. The same idealised chip: speed each user sees as users are added (4,096-token conversations)

Same assumptions as Figure 6. Total output rises roughly eighteen-fold from one user to 43 while each user's speed falls by more than half. SemiAnalysis's InferenceMAX benchmark measures the same trade-off on real systems as tokens per second per GPU against tokens per second per user.

▼ Table view

Figure 7. The same idealised chip: speed each user sees as users are added (4,096-token conversations)

Conversations in the batch	Tokens per second per user
1	99 tok/s
2	96 tok/s
4	90.4 tok/s
8	81 tok/s
16	67.1 tok/s
32	49.9 tok/s
43	42.4 tok/s

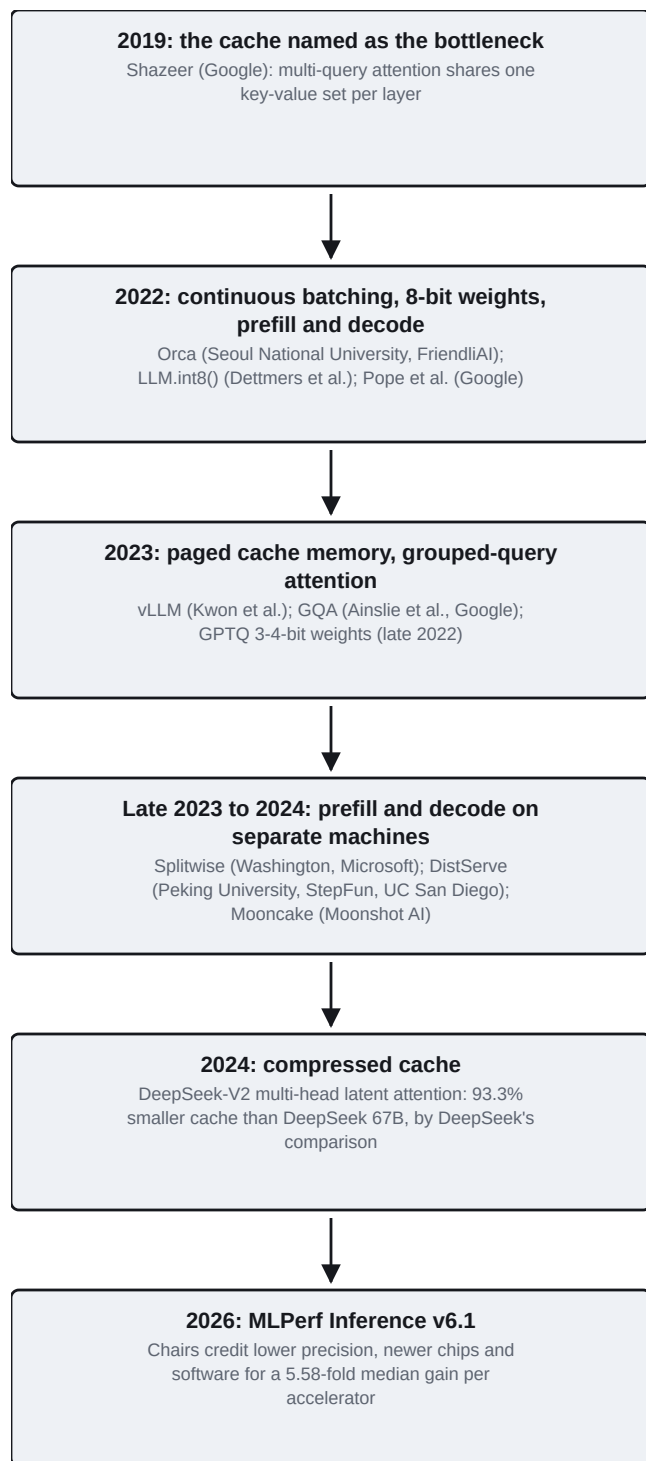
That is the pull the day's question asks about. Batching raises a chip's total output and lowers each user's speed. Longer conversations take more cache, so fewer fit and the total falls. Serving one user fast means small batches, which leaves the chip's arithmetic idle and makes each token dearer in chip-time. Pope and colleagues stated the last part in 2022:

Lower latency can often be achieved with smaller batch sizes, but smaller batch sizes also result in worse MFU, resulting in a higher total cost (in terms of chip-seconds or dollars) per token.

They also showed where long contexts lead: for a model of more than 500 billion parameters with conventional multi-head attention, "for batch size 512 and context length 2048, the KV cache totals 3TB, which is 3 times the size of the model's parameters".

4. How the machinery changed

Epoch's report measures the decline without apportioning it; Epoch's 2025 data insight named some well-known reasons — "models becoming smaller and hardware becoming more cost-effective" — and added that other important factors "might be difficult to determine from public information". The machinery of serving a single request also changed in ways that can be dated.

Figure 8. The serving levers, in the order they were published

Dates are first public versions (arXiv or conference). Each system's performance claims are its authors' own measurements against baselines they chose.

▼ **Table view**

Figure 8. The serving levers, in the order they were published — stages

#	Stage	Note
1	2019: the cache named as the bottleneck	Shazeer (Google): multi-query attention shares one key-value set per layer

#	Stage	Note
2	2022: continuous batching, 8-bit weights, prefill and decode	Orca (Seoul National University, FriendlyAI); LLM.int8() (Dettmers et al.); Pope et al. (Google)
3	2023: paged cache memory, grouped-query attention	vLLM (Kwon et al.); GQA (Ainslie et al., Google); GPTQ 3-4-bit weights (late 2022)
4	Late 2023 to 2024: prefill and decode on separate machines	Splitwise (Washington, Microsoft); DistServe (Peking University, StepFun, UC San Diego); Mooncake (Moonshot AI)
5	2024: compressed cache	DeepSeek-V2 multi-head latent attention: 93.3% smaller cache than DeepSeek 67B, by DeepSeek's comparison
6	2026: MLPerf Inference v6.1	Chairs credit lower precision, newer chips and software for a 5.58-fold median gain per accelerator

Figure 8. The serving levers, in the order they were published — connections

From	To	Label
2019: the cache named as the bottleneck	2022: continuous batching, 8-bit weights, prefill and decode	
2022: continuous batching, 8-bit weights, prefill and decode	2023: paged cache memory, grouped-query attention	
2023: paged cache memory, grouped-query attention	Late 2023 to 2024: prefill and decode on separate machines	
Late 2023 to 2024: prefill and decode on separate machines	2024: compressed cache	
2024: compressed cache	2026: MLPerf Inference v6.1	

The idea of separating the two phases follows directly from their different appetites. Splitwise, from the University of Washington and Microsoft (Pratyush Patel and colleagues, arXiv 2311.18677, 30 November 2023), described "a compute-intensive prompt computation phase" and a memory-intensive "token generation phase" and argued that "token generation does not need the compute capability of the latest GPUs and can be run with lower power and cost". DistServe (Yinmin Zhong and colleagues, Peking University, StepFun and UC San Diego, arXiv 2401.09670, January 2024) argued that serving both phases on the same machines "not only leads to strong prefill-decoding interferences but also couples the resource allocation and parallelism plans for both phases". Mooncake (arXiv 2407.00079, June 2024) is Moonshot AI's own description of "the serving platform for Kimi", which "features a KVCache-centric disaggregated architecture that separates the prefill and decoding clusters". Not everyone split them: Sarathi-Serve (Amey Agrawal and colleagues, Microsoft Research India and Georgia Tech, arXiv 2403.02310, March 2024) kept both phases on the same chips and cut long prompts into chunks so that new arrivals do not stall answers already being written — a scheduler, in its own title, for "Taming Throughput-Latency Tradeoff".

The latest independent snapshot came on 16 September 2026, when MLCommons, an industry consortium, published MLPerf Inference v6.1: results from 30 submitters on 120 systems. On one large-language-model test that has run for six rounds, the working-group chairs, Miro Hodak and Frank Han, reported "median per-accelerator performance for Server scenario submissions, which has improved 5.58x over 6 runs", and gave three reasons: some submissions "use FP4 precision, whereas earlier rounds used FP8" — within the

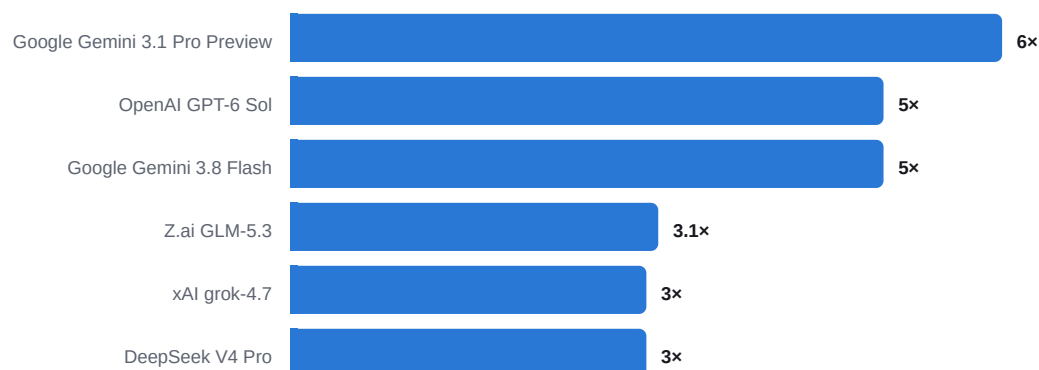
benchmark's accuracy requirements, so that "the performance gains did not come at the cost of accuracy loss" as the benchmark defines it — newer accelerators, and software improvements "even on the same hardware". Two of the three are the levers described above. The results are the submitters' own, published under rules the submitters review; one submitter, MangoBoost, reported "the first prefill/decode-disaggregated results on AMD Instinct GPUs".

5. What a price sheet encodes

None of the price sheets examined for this piece — OpenAI, Google, xAI, DeepSeek and Z.ai, all read on 26 September 2026 — states why output costs more than input. The mapping below is therefore an inference, and a price is not a cost; but the pattern is consistent across vendors.

Vendor · model	Input	Cached input	Output	Output ÷ input	Long prompts	Batch	Faster service
OpenAI · GPT-6 Sol	\$2.00	\$0.20 (write \$2.50)	\$10.00	5.0	above 272K tokens: \$4 in, \$15 out	half	Fast mode, double
Google · Gemini 3.1 Pro Preview	\$2.00	\$0.20 + \$4.50 per million tokens per hour stored	\$12.00	6.0	above 200K tokens: \$4 in, \$18 out	half	Priority, 1.8×
Google · Gemini 3.8 Flash (to 31 Dec 2026)	\$0.75	\$0.075 + \$0.50 per million tokens per hour	\$3.75	5.0	none listed	half	Priority, 1.8×
xAI · grok-4.7	\$2.00	\$0.50	\$6.00	3.0	from 200K tokens, every token of the request at \$4 in, \$12 out	offered	Priority, 2×
DeepSeek · V4 Pro (peak hours)	\$1.32	\$0.044	\$3.96	3.0	none; 1M-token window	—	off-peak hours at half price
Z.ai · GLM-5.3	\$1.40	\$0.26	\$4.40	3.1	none listed	—	—

US dollars per million tokens, standard tier, from each vendor's pricing page read 26 September 2026 (Google's page last updated 24 September 2026; xAI's dated 21 September 2026).

Figure 9. Output price as a multiple of input price, September 2026

Standard short-context rates from each vendor's pricing page, read 26 September 2026. In the arithmetic of Pope et al. (2022) a token costs about the same number of operations whether it is read or written; the difference in price tracks how the work is scheduled, not how much arithmetic it takes.

▼ Table view

Figure 9. Output price as a multiple of input price, September 2026

Model	Output ÷ input
Google Gemini 3.1 Pro Preview	6×
OpenAI GPT-6 Sol	5×
Google Gemini 3.8 Flash	5×
Z.ai GLM-5.3	3.1×
xAI grok-4.7	3×
DeepSeek V4 Pro	3×

Output at three to six times the price of input corresponds to decode: one token per pass, each pass reading the weights and the cache. Cached input at a tenth of the input price or less corresponds to prefill that need not be repeated; OpenAI's prompt-caching guide describes the benefit as:

Avoid recalculating a prompt prefix that the model has already processed.

A kept cache has to be stored, and Google charges for storing it by the token-hour: \$4.50 per million tokens per hour for Gemini 3.1 Pro. OpenAI lists cache writes at 1.25 times the input price. Batch rates at half price correspond to customers who let the provider fill its batches at its convenience; OpenAI describes its Flex tier as offering "lower costs ... in exchange for slower response times and occasional resource unavailability". Premium tiers correspond to the other end of the trade: OpenAI's Fast mode, renamed from Priority processing on 30 July 2026, "delivers up to 2.5× faster speeds" at twice the standard rate, and xAI's Priority Processing "gives text requests higher scheduling priority for lower latency" and is "billed at a 2x premium over standard rates".

The sharpest difference between vendors is over long prompts, where the cache is largest. OpenAI raises its rates above 272,000 input tokens and Google above 200,000. xAI's rule is harsher: "Models with long context pricing bill the long context rates for all tokens in a request once its prompt reaches the model's long context threshold", 200,000 tokens for grok-4.7. DeepSeek sells a one-million-token window with no long-context tier and charges half its peak rates outside its peak hours — "01:00 - 04:00 and 06:00 - 10:00 UTC, Monday through Friday". DeepSeek's pricing page does not explain the choice. Its published model designs since 2024 compress the cache, and the configuration of DeepSeek-V3 caches 576 numbers per layer per token where full keys and values for its 128 heads would need 40,960; whether that design is why DeepSeek can price long prompts flat is not stated anywhere in its documentation.

6. What to watch

- **21 November 2026.** OpenAI's pricing page says "GPT-5.6 Sol's promotional pricing is available at least through November 21, 2026" (its standard rates are listed at \$4 input and \$20 output per million tokens). Whether that row changes after the date is checkable on the same page.
- **1 January 2027.** Gemini 3.8 Flash's introductory price ends. Input rises from \$0.75 to \$1.50 per million tokens, output from \$3.75 to \$7.50, and cache storage from \$0.50 to \$1.00 per million tokens per hour — the same model at twice the price.
- **The first MLPerf Endpoints results.** MLCommons says that "moving forward, MLPerf Endpoints will replace Inference in our family of benchmarks for the datacenter", without giving a date. Whether the new benchmark reports speed per user beside output per accelerator will decide whether the latency-throughput trade-off becomes visible in its headline numbers.

7. The idea to keep

The key-value cache is the idea that organises the rest. The weights of a model are shared by every conversation in a batch; the cache belongs to one conversation and grows with every token of it. Batching spreads the cost of reading the weights; long contexts and fast service use up the room that batching needs; quantisation and compressed caches make more room. On that reading, most of the rates on a modern price sheet line up with one side of the split or the other.

Three questions follow for any quoted price of an answer. Is the token being read, re-read from a cache, or written? How long is the conversation behind it? And how quickly must it arrive? The fullest single account remains Reiner Pope and colleagues, "Efficiently Scaling Transformer Inference", arXiv 2211.05102 (2022).

Sources

Source	Date	What it supports
Epoch AI (Luke Emberson, David Roodman), <i>The plunging price of thought</i>	22 Sep 2026	47% per quarter, 13× a year; 725-fold example; caveats

Source	Date	What it supports
Epoch AI (Ben Cottier and colleagues), <i>LLM inference prices have fallen rapidly but unequally across tasks</i>	12 Mar 2025	9× to 900× a year; reasons for price drops
Gundlach, Lynch, Mertens, Thompson, <i>The Price of Progress</i> , arXiv 2511.23455 v2	23 Mar 2026	5× to 10× a year falling; frontier 3× to 18× rising
OpenAI, API pricing page; prompt caching, Fast mode and Flex guides	read 26 Sep 2026	GPT-6 Sol rates; Fast mode rename; caching and Flex descriptions
Google, Gemini Developer API pricing; Gemini 3.8 Flash announcement	24 Sep 2026; 2 Sep 2026	Gemini rates, storage price, introductory price
xAI, API pricing page	21 Sep 2026	grok-4.7 rates, long-context rule, Priority Processing
DeepSeek, Models & Pricing	read 26 Sep 2026	V4 Pro rates, 1M context, off-peak hours
Z.ai, pricing page	read 26 Sep 2026	GLM-5.3 rates
Pope et al. (Google), <i>Efficiently Scaling Transformer Inference</i> , arXiv 2211.05102	9 Nov 2022	prefill and decode; KV cache; latency-cost trade-off
Shazeer (Google), <i>Fast Transformer Decoding</i> , arXiv 1911.02150	6 Nov 2019	decode bound by reloading keys and values; multi-query attention
Ainslie et al. (Google), <i>GQA</i> , arXiv 2305.13245	May 2023	grouped-query attention
DeepSeek-AI, <i>DeepSeek-V2</i> , arXiv 2405.04434	May 2024	93.3% smaller KV cache than DeepSeek 67B
Yu et al., <i>Orca</i> , OSDI 2022	Jul 2022	iteration-level scheduling; 36.9× claim
Kwon et al., <i>PagedAttention / vLLM</i> , arXiv 2309.06180 (SOSP 2023)	12 Sep 2023	20.4–38.2% of KV memory used; 2–4× claim
Vanhoucke, Senior, Mao (Google), <i>Improving the speed of neural networks on CPUs</i>	2011	batching and 8-bit arithmetic on CPUs
Jacob et al. (Google), arXiv 1712.05877	15 Dec 2017	integer-only inference
Dettmers et al., <i>LLM.int8()</i> , arXiv 2208.07339	15 Aug 2022	8-bit inference at 175B parameters
Frantar et al., <i>GPTQ</i> , arXiv 2210.17323	31 Oct 2022	3–4-bit weights
Patel et al., <i>Splitwise</i> , arXiv 2311.18677	30 Nov 2023	two phases with different demands
Zhong et al., <i>DistServe</i> , arXiv 2401.09670	18 Jan 2024	prefill-decode interference
Agrawal et al., <i>Sarathi-Serve</i> , arXiv 2403.02310	4 Mar 2024	chunked prefill on shared machines
Qin et al. (Moonshot AI, Tsinghua), <i>Mooncake</i> , arXiv 2407.00079	24 Jun 2024	disaggregated serving for Kimi
SemiAnalysis, InferenceMAX launch post	9 Oct 2025	throughput versus interactivity
MLCommons, MLPerf Inference v6.1 results and chairs' analysis	16–17 Sep 2026	5.58× median gain; three reasons; MLPerf Endpoints
Alibaba Qwen team, Qwen3-32B model card and config; DeepSeek-V3 and Qwen3-235B-A22B configs	read 26 Sep 2026	cache arithmetic
NVIDIA, H100 product page	read 26 Sep 2026	80 GB, 3.35 TB/s