Context Management for Long-Context Agents

How agents keep working as their history grows

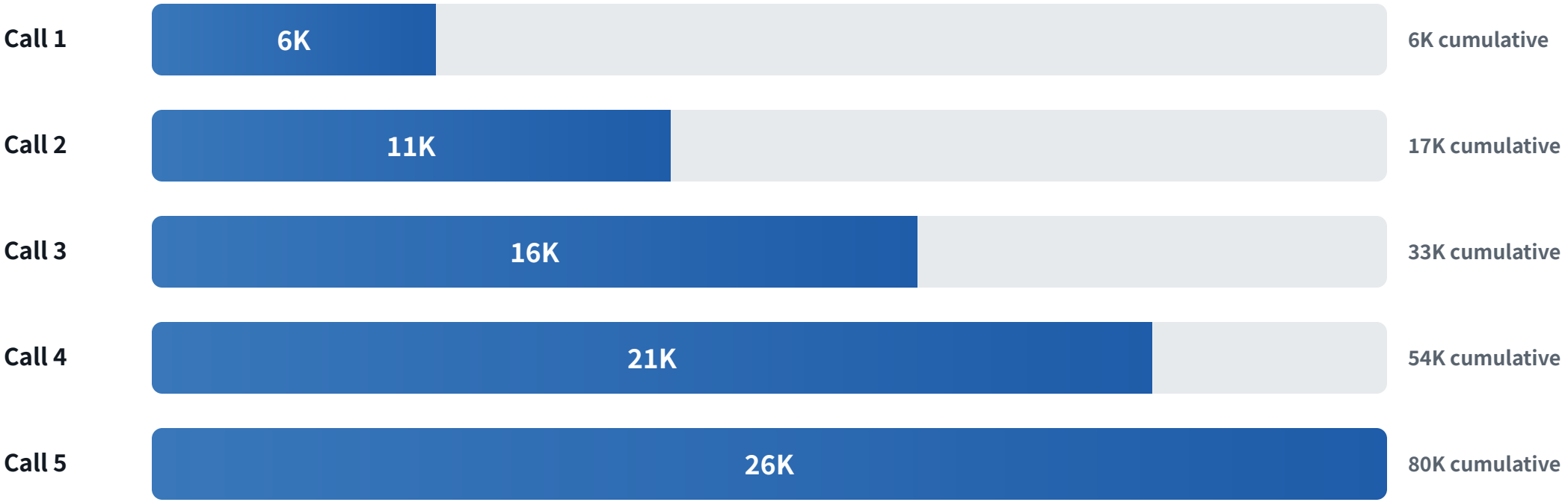
Carnegie Mellon University
Language Technologies Institute

Graham Neubig
Language Technologies Institute

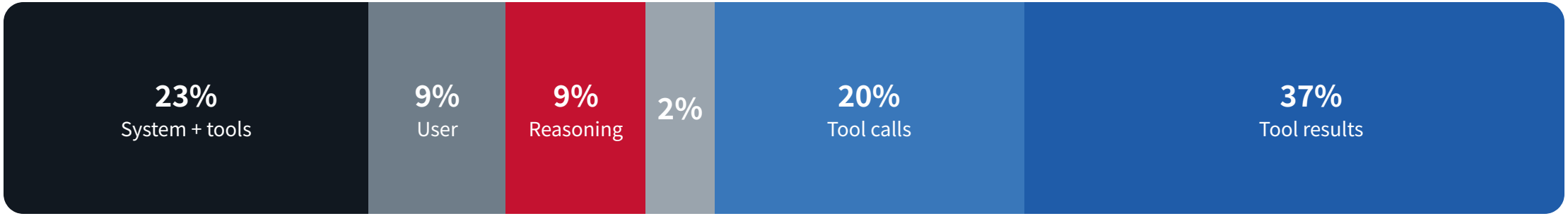
Agent context growth

Quadratic input from linear history growth

$$\text{input}_t = \text{fixed prefix} + \text{history before call } t$$



Agent prompt composition example



77,922 mean tokens per session across 1,500 OpenHands sessions

Two long-context challenges

Capacity

Can the model use the needed evidence?

Architecture, position encoding, and training shape usable context; evaluation measures it.

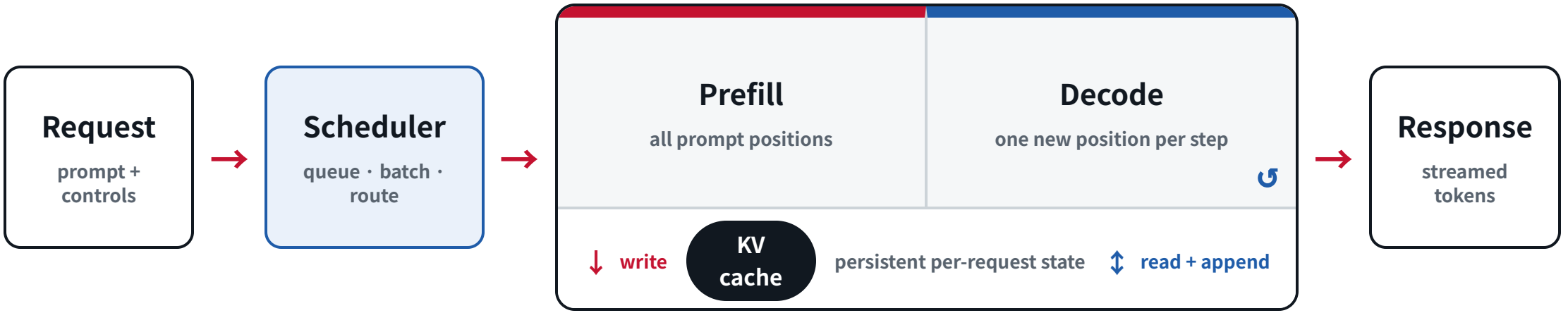
Efficiency

Can the system afford to serve it?

Attention, KV memory, caching, and compaction shape cost and latency.

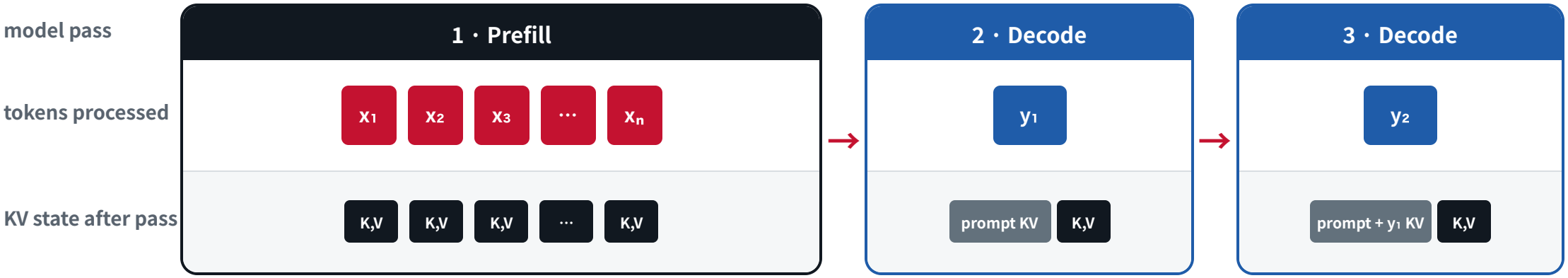
LLM inference

Elements of the serving stack



DistServe · Zhong et al., OSDI 2024 ↗

Inference phases

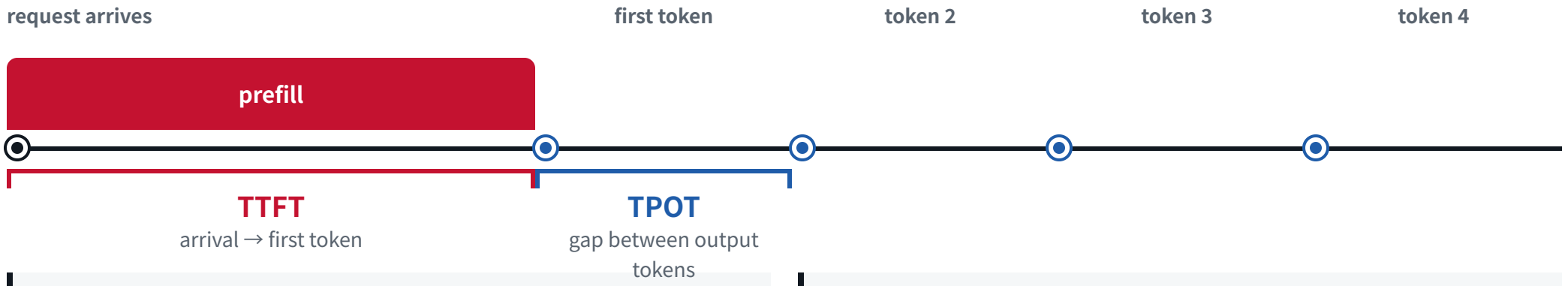


Prefill: compute-intensive; latency grows with prompt length

Decode: sequential and memory-bandwidth-intensive; one step per output token

Prefill/decode execution model · DistServe, OSDI 2024 ↗

Serving metrics



Throughput completed tokens or requests ÷ wall-clock time

Cost input + cached input + output token charges

TTFT and TPOT phase decomposition · DistServe, OSDI 2024 ↗

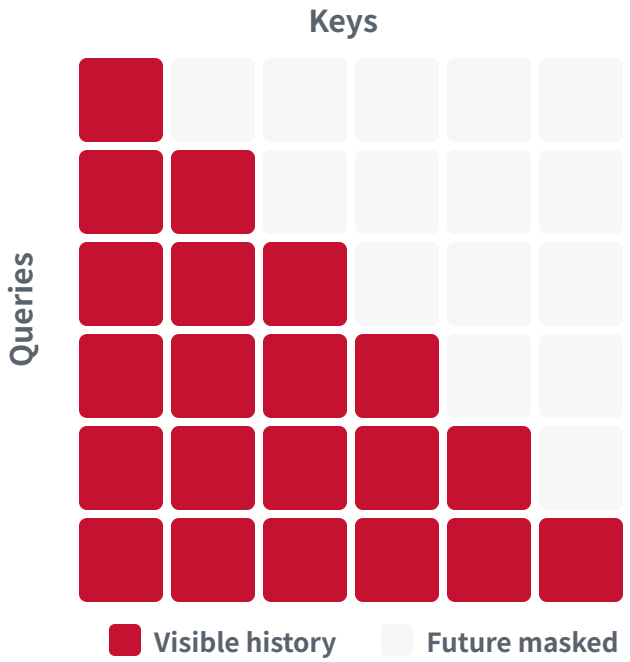
Refresher: attention

Linear projections

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

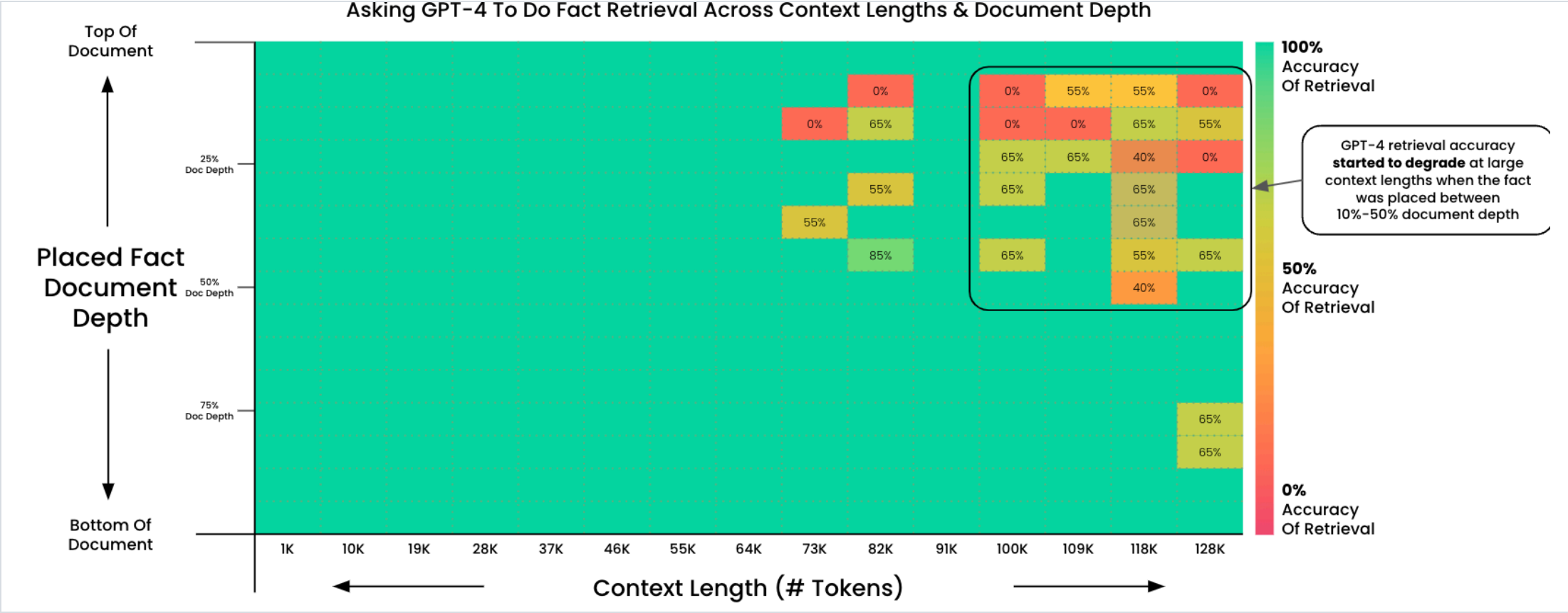
Causal self-attention

$$A = \mathrm{softmax}\left(\frac{QK^\top + M}{\sqrt{d_k}}\right), \qquad O = AV$$



Notation: $X \in \mathbb{R}^{n \times d}$ has token-state row z_t; lowercase q_i, k_j, v_j denote rows of Q, K, V; d_k is key width; $M_{ij} = 0$ for $j \le i$ and $-\infty$ otherwise.

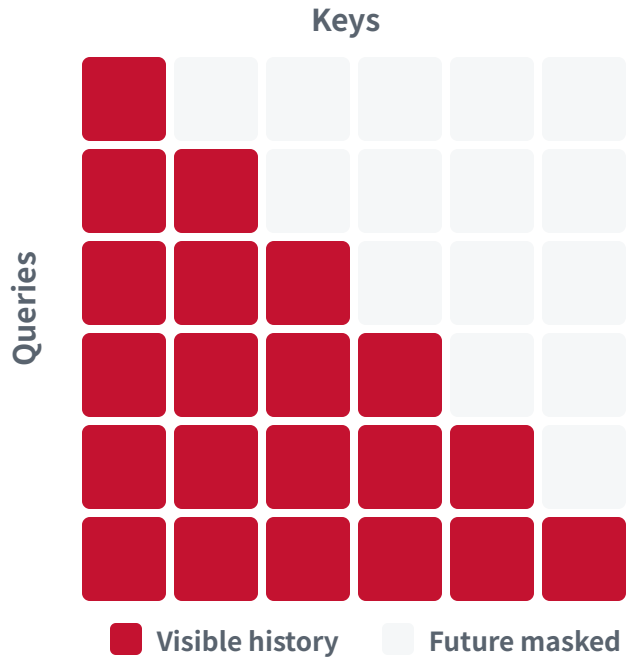
Advertised vs. effective context



GPT-4 needle-in-a-haystack evaluation · Kamradt, 2023 ↗

Architectures for long context

Standard attention is global



Typically, attention makes pairwise comparisons across the entire sequence.

Local modeling

Local computation only considers a constant subset of the other data.

Pairwise-comparison cost

$$\mathcal{O}(n^2) \longrightarrow \mathcal{O}(wn)$$

w is the local window size; for constant w, work is linear in n.

Periodic global access in hybrid models

`N × local computation → 1 × global computation → repeat`

Long-context hybrid architectures

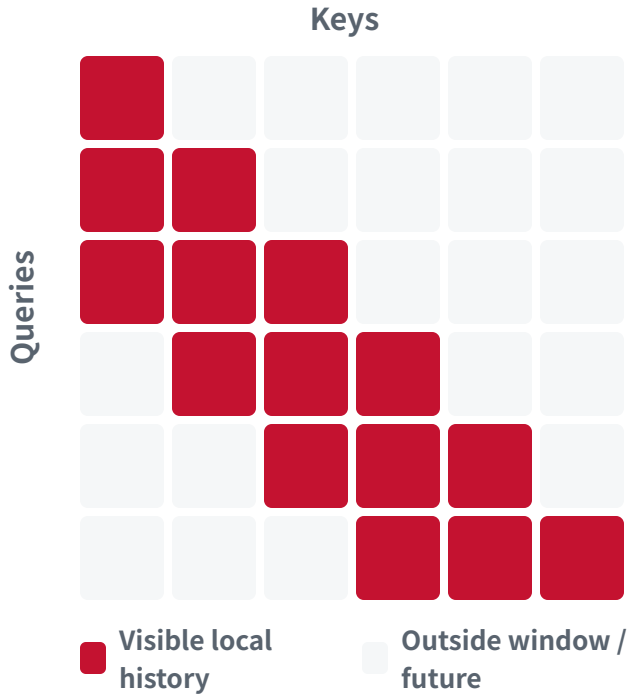
Model	Local / stateful calculation	Global calculation	Local : global
Qwen3.8-Flash-Next	Gated DeltaNet	QSA sparse retrieval	36 : 12 ≈ 3:1
GLM-5.3-Flash	Kimi Delta Attention	DSA + IndexPool	34 : 11 ≈ 3:1
Kimi K3	Kimi Delta Attention	dense Gated MLA	69 : 24 ≈ 3:1
Nemotron 3 Ultra	Mamba-2	dense GQA	48 : 12 = 4:1
Inkling-Small	512-token window	dense GQA	35 : 7 = 5:1
DeepSeek-V4-Pro-0813	128-token window branch	compressed sparse / dense	interleaved branches

Qwen ↗ GLM ↗ Kimi ↗ Nemotron ↗ Inkling ↗ DeepSeek ↗

Four recurring mechanisms

- Sliding-window attention
- Linear attention or recurrent models
- Sparse attention
- KV compression

Sliding-window attention



New variable: w is the number of causal positions retained by each query.

Causal window mask

$$M_{ij}^{(w)} = \begin{cases} 0, & 0 \le i - j < w \\ -\infty, & \text{otherwise} \end{cases}$$

Using the same attention equation, work falls from $O(n^2)$ to $O(nw)$.

A local layer cannot directly retrieve distant tokens.

Longformer · Beltagy et al., 2020 ↗

Softmax attention

1 Softmax | 2 Linear | 3 DeltaNet | 4 Gated DeltaNet | 5 KDA

Past representation

Growing KV cache

Retain every prior k_i, v_i; compare q_t with every k_i.

explicit token-level access · O(t) decode comparisons

Normalized causal score

$$a_{ti} = \frac{\exp(q_t^\top k_i / \sqrt{d_k})}{\sum_{j=1}^{t} \exp(q_t^\top k_j / \sqrt{d_k})}$$

Read from explicit values

$$o_t = \sum_{i=1}^{t} a_{ti} v_i$$

New variable: a_{t_i} is the normalized causal attention weight from query t to key i.

Attention Is All You Need · Vaswani et al., NeurIPS 2017 ↗

Linear attention

Past representation

Fixed matrix state

$S_t^\top$ stores an online mapping from keys to values.

additive writes · interference is not explicitly erased

1 · Start from standard attention

$$o_t = \sum_{i=1}^{t} \mathrm{softmax}_i\left(q_t^\top k_i / \sqrt{d_k}\right) v_i$$

2 · Replace normalization with a bilinear score

$$\mathrm{softmax}_i\left(q_t^\top k_i / \sqrt{d_k}\right) \quad \longrightarrow \quad q_t^\top k_i$$

3 · Reassociate, then update recurrently

$$o_t = \sum_{i=1}^{t} (q_t^\top k_i) v_i = \left(\sum_{i=1}^{t} k_i v_i^\top\right)^\top q_t = S_t^\top q_t,$$
$$S_t = S_{t-1} + k_t v_t^\top.$$

Key step: replacing query-normalized softmax weights with bilinear scores lets us reorder the sum. $S_t \in \mathbb{R}^{d_k \times d_v}$ stores that fixed-size recurrent memory.

Transformers are RNNs · Katharopoulos et al., ICML 2020 ↗

DeltaNet

Memory operation

Targeted correction

Read the current association, then write only its prediction error.

correct along k_t · unrelated memory persists

Prediction and error

$$\hat{v}_t = S_{t-1}^\top k_t, \qquad e_t = v_t - \hat{v}_t$$

Delta-rule write

$$S_t = S_{t-1} + \beta_t k_t e_t^\top, \qquad o_t = S_t^\top q_t$$

New variables: $\hat{v}_t$ is the current memory prediction; e_t is its error; $\beta_t \in [0,1]$ is the correction strength.

Linear Transformers Are Secretly Fast Weight Programmers · Schlag et al., ICML 2021 ↗

Gated DeltaNet

1 Softmax | 2 Linear | 3 DeltaNet | 4 Gated DeltaNet | 5 KDA

New operation

Scalar forgetting

Decay the entire memory before applying the same targeted correction.

α_t: global retention · β_t: targeted edit

Whole-state decay

$$\tilde{S}_{t-1} = \alpha_t S_{t-1}$$

Correction after decay

$$S_t = \tilde{S}_{t-1} + \beta_t k_t \left(v_t - \tilde{S}_{t-1}^{\top} k_t \right)^{\top}$$

New variables: $\alpha_t \in [0,1]$ is a scalar retention gate; $\tilde{S}_{t-1}$ is the decayed memory. Setting $\alpha_t = 1$ recovers DeltaNet.

Gated Delta Networks · Yang et al., 2024 ↗

Kimi Delta Attention

1 Softmax | 2 Linear | 3 DeltaNet | 4 Gated DeltaNet | 5 KDA

New operation

Channelwise forgetting

Give each key-space feature its own memory lifetime.

fine-grained retention · same targeted correction

Per-channel decay

$$D_t = \mathrm{Diag}(\boldsymbol{\alpha}_t), \qquad \widetilde{S}_{t-1} = D_t S_{t-1}$$

KDA recurrence

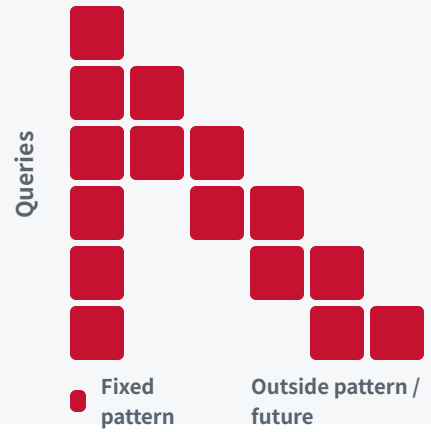
$$S_t = (I - \beta_t k_t k_t^\top) D_t S_{t-1} + \beta_t k_t v_t^\top$$

New variables: $\alpha_t \in [0,1]^{d_k}$ is the vector retention gate; D_t is its diagonal matrix; I is the $d_k \times d_k$ identity. Equal retention in every channel recovers Gated DeltaNet.

Kimi Linear / KDA · Kimi Team, 2025 ↗

Sparse attention

Fixed pattern



Input-independent

Visible keys

$$\mathcal{S}_i^{\text{fixed}} = \{j \le i : i - j < w\} \cup (G \cap [1, i])$$

Predictable cost; the pattern cannot adapt to the query.

Content-dependent

Query-specific

Visible keys

$$\mathcal{S}_i^{\text{dyn}} = \text{TopK}_{j \le i}\ g(q_i, k_j)$$

Adaptive retrieval; selector errors exclude relevant keys.

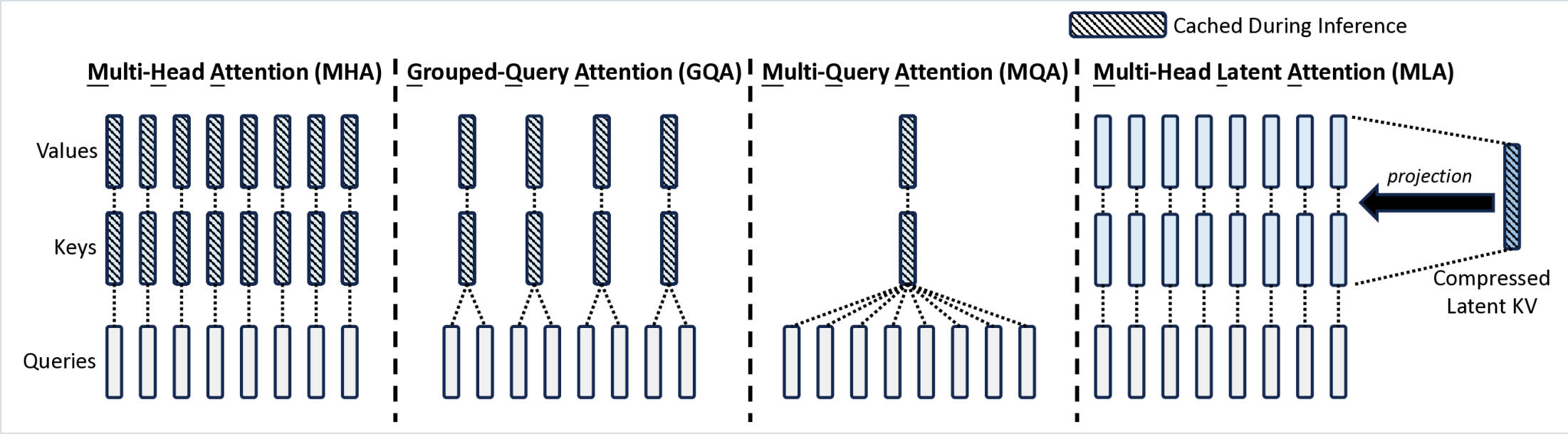
Both renormalize over the retained set

$$A_{ij} = \text{softmax}_{j \in \mathcal{S}_i}\left(\frac{q_i^\top k_j}{\sqrt{d_k}}\right), \qquad o_i = \sum_{j \in \mathcal{S}_i} A_{ij} v_j$$

New variables: G is a fixed set of global positions; g is a learned selector; $|\mathcal{S}_i^{\text{dyn}}| = K$.

Fixed local + global pattern · Longformer, 2020 ↗ Content-dependent top-K · DeepSeek Sparse Attention, 2025 ↗

KV compression



MLA caches the latent and reconstructs head-specific keys and values

$$c_t^{KV} = W_D^{KV} z_t, \qquad k_t^{(h)} = W_{UK}^{(h)} c_t^{KV}, \qquad v_t^{(h)} = W_{UV}^{(h)} c_t^{KV}$$

New variables: c_t^{KV} is the cached latent; h indexes heads; W^D compresses and W^{UK}, W^{UV} recover per-head K,V.

GQA · Ainslie et al., 2023 ↗ Attention comparison and MLA · DeepSeek-V2, 2024 ↗

Training long-context models

Short-to-long training

Short pretraining

abundant data · lower cost

4K–8K

→

Continued training

long documents · packed sequences

32K–128K

→

Long adaptation

task traces · synthetic curricula

128K+

Absolute positional encodings

Input to the first layer

$$z_t = E[x_t] + p_t$$

Fixed sinusoidal encoding

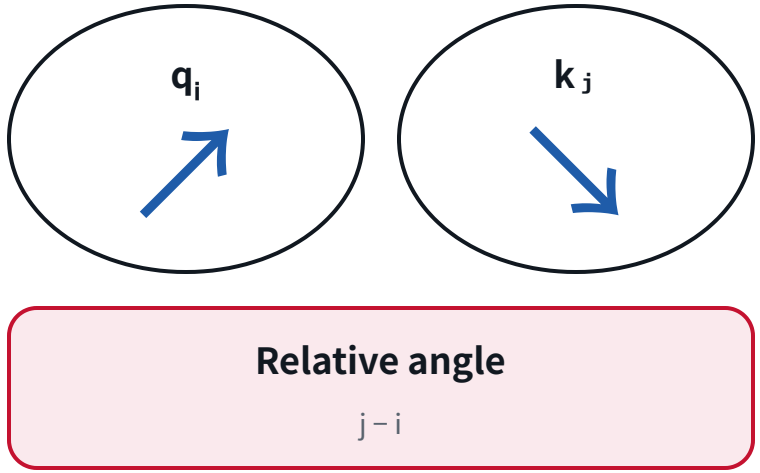
$$p_{t,2\ell} = \sin(t/\omega_\ell), \quad p_{t,2\ell+1} = \cos(t/\omega_\ell), \qquad \omega_\ell = 10000^{2\ell/d}$$

New variables: x_t is a token ID; E is the embedding table; p_t is the positional vector; ℓ indexes sinusoid pairs.

- Position enters **before** the projections that produce Q, K, and V.
- Learned absolute embeddings have a fixed table; sinusoids define positions analytically.

Transformer positional encoding · Vaswani et al., 2017 ↗

Rotary position embeddings (RoPE)



New variable: R(t) is block-diagonal in 2D rotations with angles $t\theta_l$.

Rotate rows of Q and K

$$\tilde{q}_i = R(i)q_i, \qquad \tilde{k}_j = R(j)k_j$$

The score depends on relative displacement

$$\tilde{q}_i^\top \tilde{k}_j = q_i^\top R(j-i)k_j$$

- V is unchanged; attention uses the same equation with rotated Q and K.

RoFormer · Su et al., Neurocomputing 2024 ↗

No positional encoding (NoPE)

token state only		model layers
$E[x_t]$	→	causal order

Input to the first layer

$$z_t = E[x_t]$$

Attention equation is unchanged

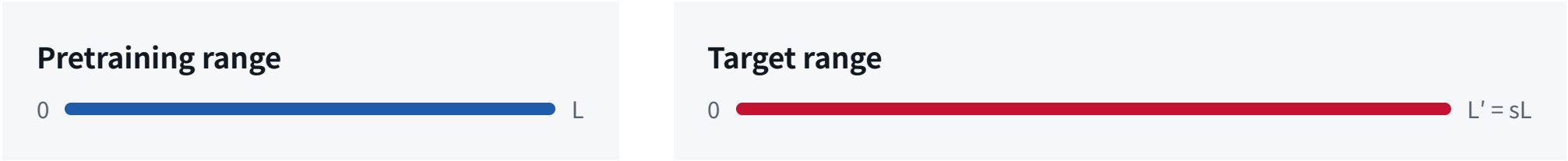
$$A = \mathrm{softmax}\left(\frac{QK^\top + M}{\sqrt{d_k}}\right), \qquad O = AV$$

- Order is inferred from causal computation or recurrent dynamics—not from p_t or R(t).
- This removes position extrapolation, but not the need to train long-range behavior.

NoPE length generalization · Kazemnejad et al., NeurIPS 2023 ↗

Kimi K3 · NoPE with recurrent KDA ↗

Position interpolation



evaluate target position t using the familiar phase t / s

Uniform position interpolation

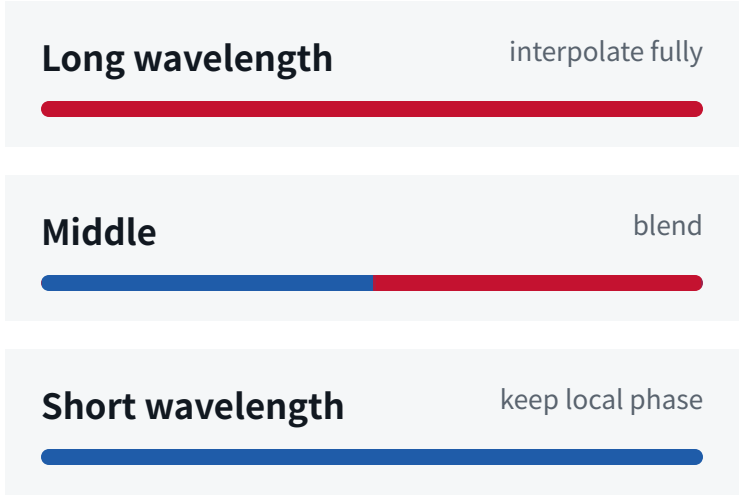
$$R(t) \longrightarrow R(t/s) \qquad \text{equivalently} \qquad \theta_\ell \longrightarrow \theta_\ell / s$$

New variables: L and L′ are original and target limits; s = L′/L.

- Every RoPE dimension is stretched by the same scale s.
- Long positions stay inside the phase range observed during pretraining.

Position Interpolation · Chen et al., 2023 ↗

YaRN



Frequency-selective interpolation

$$\theta'_\ell = \left(1 - \gamma_\ell\right)\frac{\theta_\ell}{s} + \gamma_\ell\theta_\ell, \qquad 0 \le \gamma_\ell \le 1$$

Attention-temperature adjustment

$$A = \mathrm{softmax}\left(\frac{\tilde{Q}\tilde{K}^\top + M}{\tau\sqrt{d_k}}\right), \qquad \sqrt{1/\tau} = 0.1\ln s + 1$$

New variables: γ_l ramps from 0 for long wavelengths to 1 for short wavelengths; τ is attention temperature.

- Long-wavelength dimensions are interpolated; short-wavelength dimensions preserve local distinctions.
- The temperature correction counteracts attention-entropy drift at larger extension scales.
- The published formula for τ was fit on LLaMA models; it is a recipe, not a universal constant.

YaRN · Peng et al., ICLR 2024 ↗

Model-specific extension paths

Model	Context-length training	Position method	How it reaches maximum context
Qwen3.8-Flash-Next	262K native; full curriculum undisclosed	**partial RoPE**	YaRN extension to 1M
DeepSeek V4	4K → 16K → 64K → 1M	**partial RoPE**	progressive training + YaRN
GLM-5.3-Flash	1M native; exact curriculum undisclosed	**NoPE in main attention**	no RoPE rescaling
Kimi K3	8K → 64K → 256K → 1M	**NoPE**	progressive training to 1M
Nemotron 3 Ultra	1M long-context CPT; SFT to 512K	**implicit order in Mamba; no RoPE**	trained/evaluated at long lengths
Inkling	1M context; curriculum undisclosed	**relative position embedding**	learned relative representation

Qwen ↗ DeepSeek ↗ GLM ↗ Kimi ↗ Nemotron ↗ Inkling ↗

Long-context data examples

Long documents

coherent natural structure

Books, codebases, repositories, and multi-document corpora.

Packed examples

high token utilization

Efficient, but boundaries and cross-example leakage matter.

Agent trajectories

actions + observations

Teach long-horizon recovery, state tracking, and tool use.

Synthetic tasks

controlled dependencies

Place evidence and distractors at designed positions.

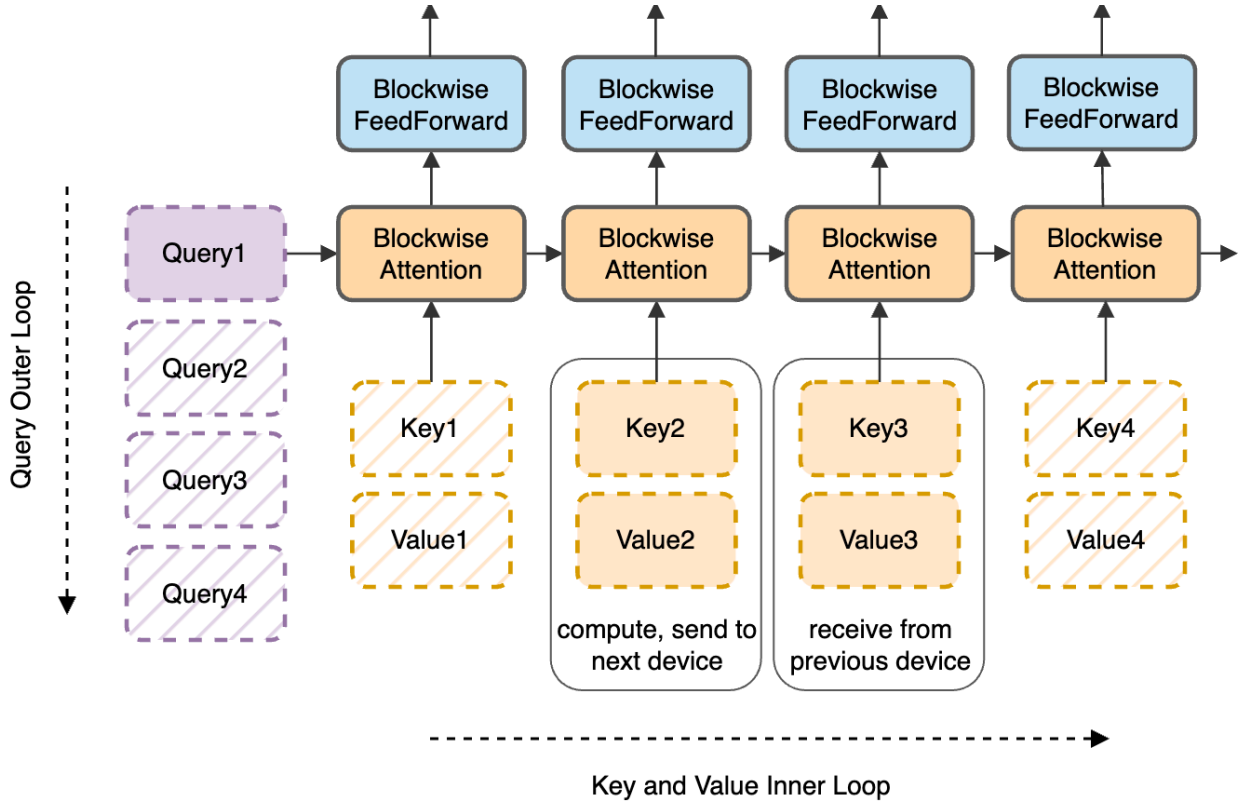
Long-context data mixtures

Model	Natural long data	Constructed supervision	Length curriculum
Kimi K3	cleaned, upsampled documents + video	permuted multimodal subtasks with evidence across the full context	**8K → 64K → 256K → 1M**
GLM-5	documents; repository files, issues, PRs, and diffs	synthetic dependencies + agent trajectories	**32K → 128K → 200K**
Nemotron 3 Ultra	long-document QA	multi-document reasoning, sequential scans, synthetic tables	**33B-token 1M CPT; SFT to 512K**
DeepSeek V4	scientific papers + technical reports	agentic post-training after long-context pretraining	**4K → 16K → 64K → 1M**

The shared recipe is **coherent long sources + tasks whose answer depends on distant evidence + progressive length growth**.

Kimi K3 ↗ GLM-5 ↗ Nemotron 3 Ultra ↗ DeepSeek V4 ↗

Context parallelism

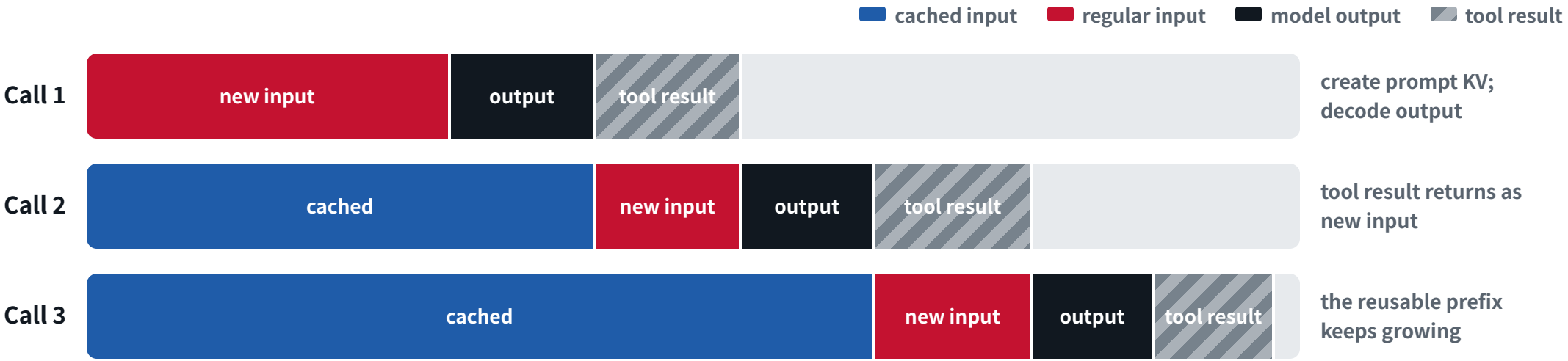


Memory / device $O(nd_k/P)$ | **Communication** overlapped with block attention | **Total work** $O(n^2d_k)$ · exact, not sparse

Ring Attention · Liu et al., 2023 ↗ Megatron context parallelism ↗

Prompt and KV caching

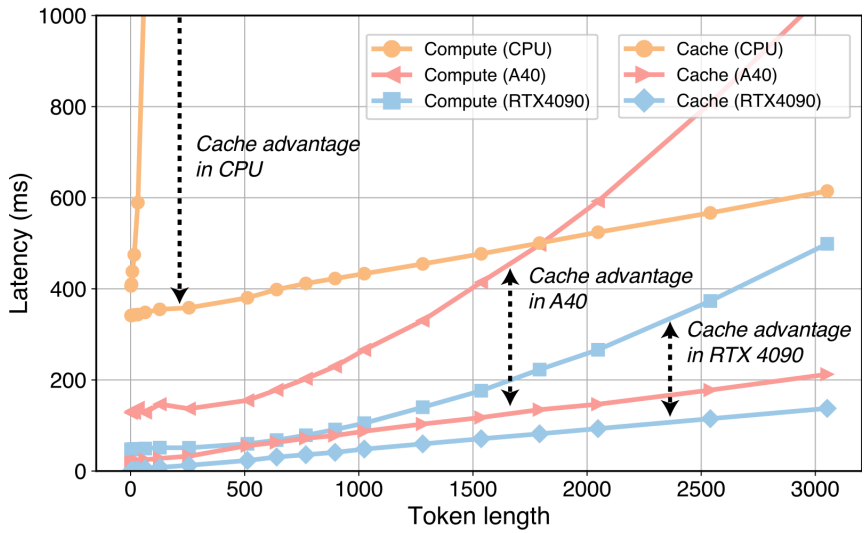
KV caching during decoding



Within a call: output tokens extend the KV cache during decode. **Between calls:** the tool result returns as new input; after prefill, it joins the reusable prefix.

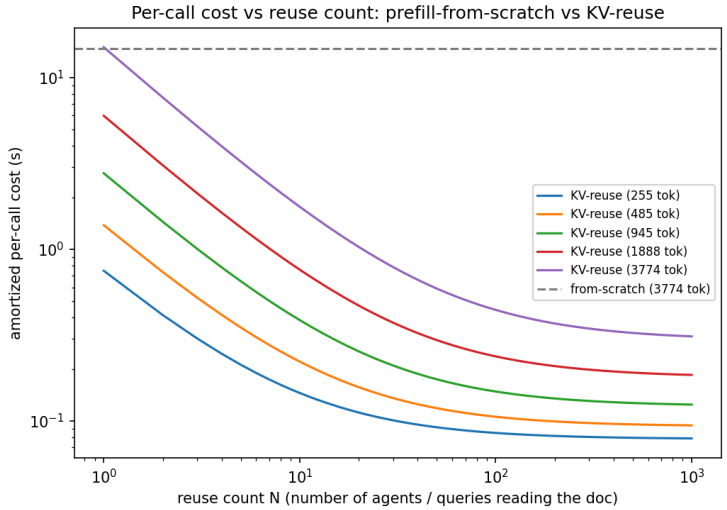
KV and prompt-state reuse · Prompt Cache, MLSys 2024 ↗

Empirical benefits of cache reuse



Prompt Cache

Compute grows faster than loading cached attention state; the crossover depends on hardware.



Can I Buy Your KV Cache?

One-time prefill is amortized across readers; per-call cost approaches the reuse-step floor.

Measured physical cost and provider API price are different quantities.

Prompt Cache · Gim et al., 2024 ↗ Can I Buy Your KV Cache? · 2026 ↗

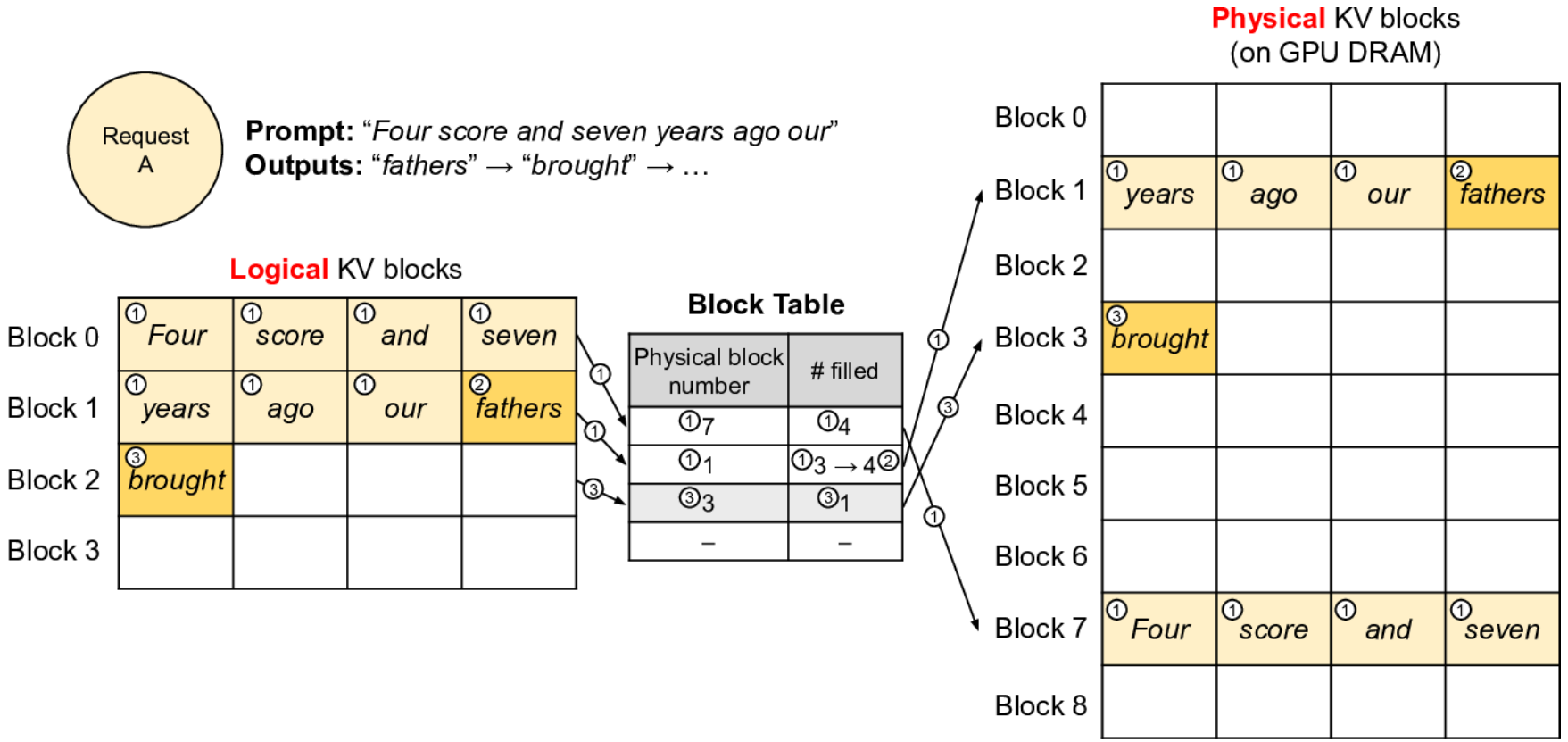
Cached-token API pricing

Model	Cached input	Regular input	Output
DeepSeek V4 Flash	**$0.007–0.014**	$0.22–0.44	$0.66–1.32
GLM-5.2	**$0.26**	$1.40	$4.40
Kimi K3	**$0.30**	$3.00	$15.00
GPT-5.6 Luna	**$0.02**	$0.20	$1.20
GPT-5.6 Sol	**$0.40**	$4.00	$20.00
Claude Opus 5	**$0.50**	$5.00	$25.00

USD per 1M tokens · prices checked Aug. 30, 2026
“Cached” means a cache read/hit; cache creation may be billed separately.
DeepSeek range: off-peak–peak.

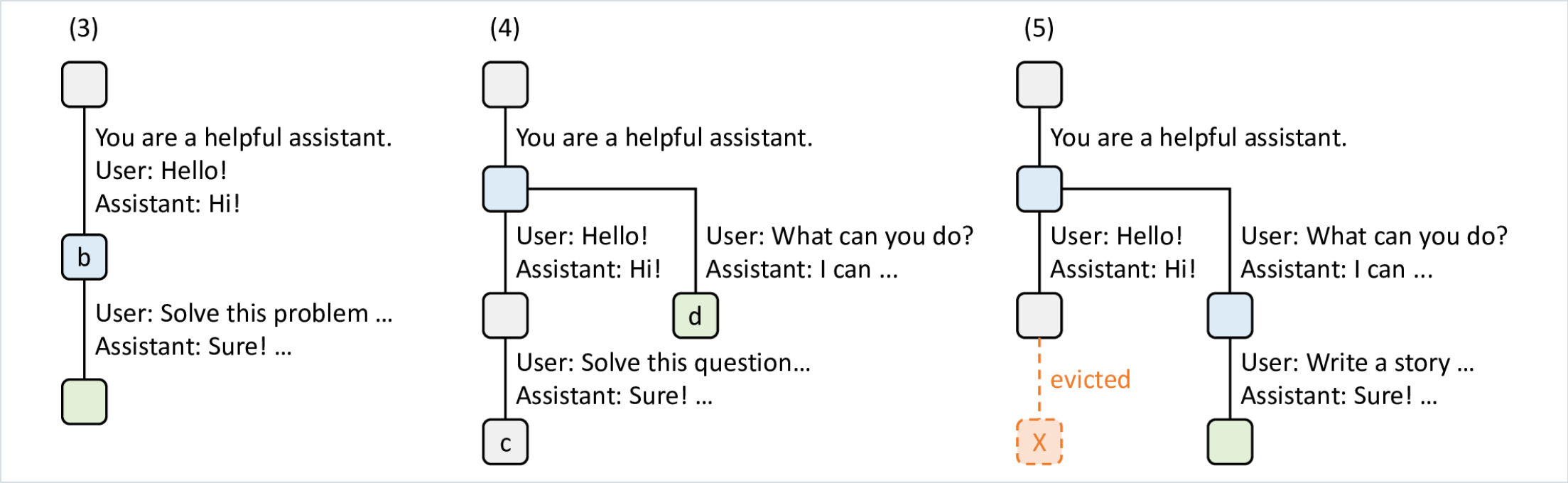
DeepSeek ↗ Z.AI ↗ Kimi ↗ OpenAI ↗ Anthropic ↗

Paged KV-cache allocation



Block-table translation · vLLM / PagedAttention · Kwon et al., SOSP 2023 ↗

RadixAttention prefix sharing



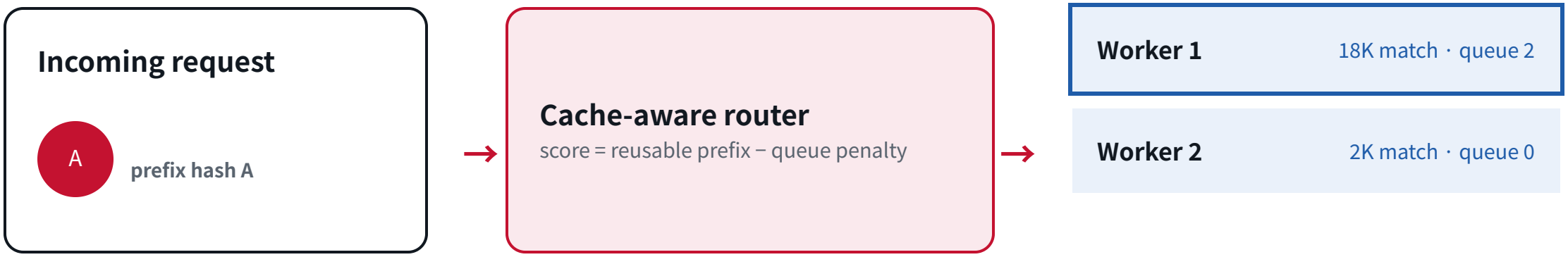
3 · Match
reuse prefix, append suffix

4 · Split
share the common prefix

5 · Evict
remove an LRU leaf

RadixAttention operations (steps 3–5) · SGLang · Zheng et al., 2023 ↗

Cache-aware request routing



SGLang v0.4 · cache-aware load balancing ↗

Prompt design for cache reuse

Stabilize the prefix

instructions · tools · examples

Append changing state

new user turn · observations

Avoid needless churn

timestamps · reordered schemas

Measure reuse

cached tokens · TTFT · eviction

Context compaction

Compaction example

Before

OBSERVATION	CI job times out after 30 minutes.
ACTION	Inspect the job log.
OBSERVATION	24K-line log; repeated “address already in use.”
ACTION	Set `--workers=1` and rerun.
OBSERVATION	Suite passes; log saved at `/tmp/ci.log`.

→ compact

After

OBSERVATION	**Goal:** fix CI timeout. **Cause:** port collision. **Verified:** `--workers=1` passes. **Evidence:** `/tmp/ci.log`.
NEXT ACTION	Open the PR with the tested worker setting.

Compaction as state estimation

Behavioral fidelity

$$p(A_{\text{future}} \mid H_t) \approx p(A_{\text{future}} \mid \hat{s}_t)$$

Bounded representation

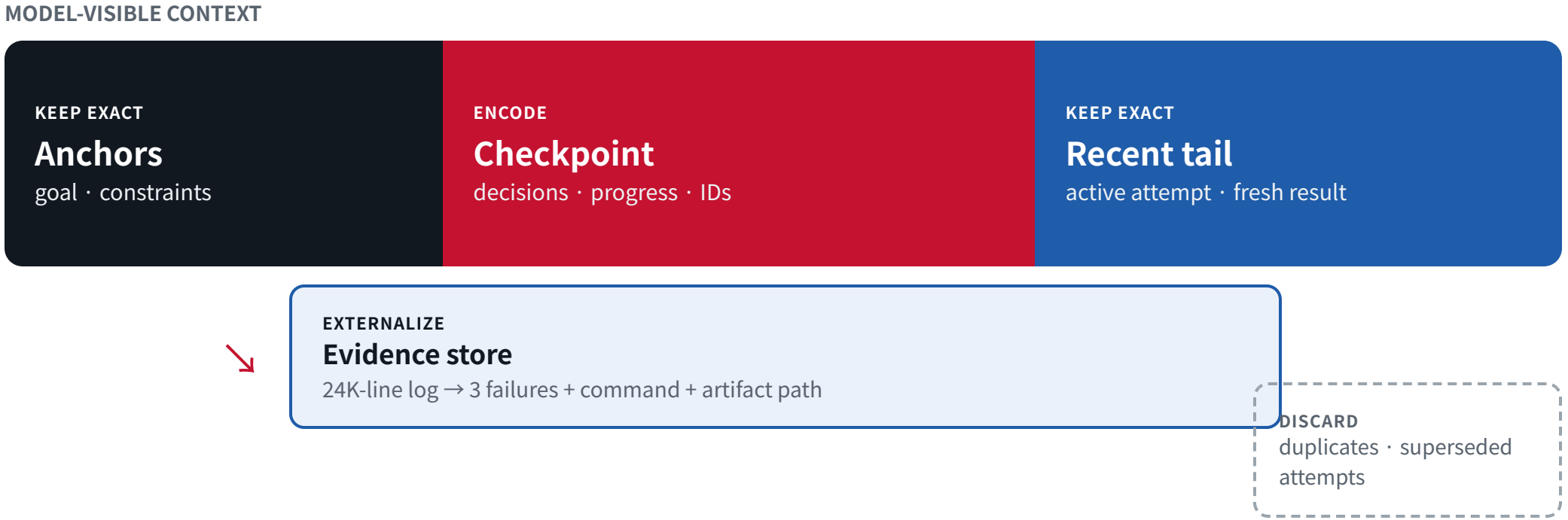
$$\hat{s}_t = \mathcal{C}(H_t; B), \qquad |\hat{s}_t| \leq B$$

Recoverability

Copy exact anchors; retain pointers to source evidence.

Notation: H_t is the full history; $\hat{s}_t$ is compacted working state; B is its token budget; A_{future} denotes future actions.

What survives compaction



MemGPT · externalize and retrieve ↗ LongLLMLingua · selective compression ↗

Compaction policy decisions

1 · Trigger

token threshold
provider overflow
manual request

reserve room for the next output

→

2 · Select

protect beginning
compact older region
keep recent tail

cut at a valid turn / tool boundary

→

3 · Replace

readable summary
structured checkpoint
opaque provider item
fresh context

→

4 · Recover

leave history unchanged
retry or reset
retain searchable history

define behavior when compaction fails

Codex ↗ OpenCode ↗ Pi ↗ Hermes ↗ OpenHands ↗

Repeated compaction drift

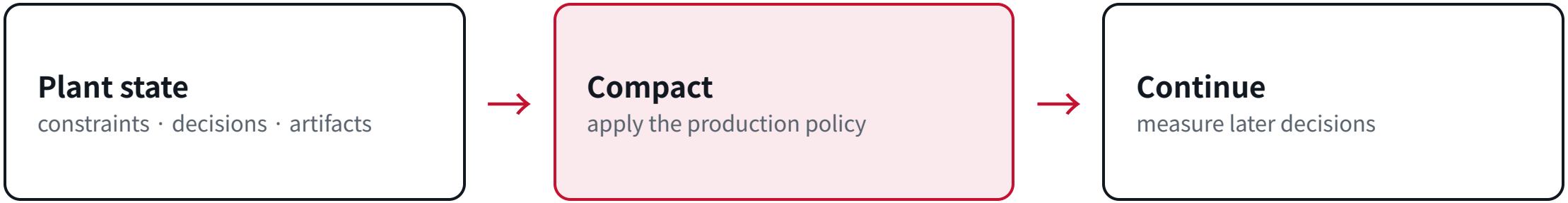
Each update inherits the previous checkpoint

$$\hat{s}_{k+1} = \mathcal{C}(\hat{s}_k \oplus R_k)$$

copy anchors exactly · keep the recent tail · retrieve source evidence

ReSum · summary-conditioned agent continuation ↗

Continuation-based evaluation



State recall constraints + identifiers

Task success correct continuation

Efficiency tokens + latency + cost

Stability repeated compactions

Takeaways

The long-context system stack

1. **Growth:** agent loops repeatedly ingest instructions, actions, and environment observations.
2. **Inference:** prefill, decode, dense attention, and KV memory create different bottlenecks.
3. **Architecture:** local, recurrent, sparse, and compressed mechanisms choose what history remains accessible.
4. **Caching:** stable prefixes avoid repeated prefill but do not reduce visible history.
5. **Compaction:** preserve intent, exact task state, recent work, and recovery paths—then evaluate continuation.

Questions

Next Class: Agent Capabilities 3 — Skills and Memory