Long-Context Systems

Deep Dive into DeepSeek — Chapter 33

Yin Yang

Foundation Books Project

The one rule of this chapter

A context-window number is an admission claim.
It is not a useful-context claim.

- Admission says how many tokens a model may accept.
- Useful context says whether the answer actually depends on the right ones, once the request also shares memory, compute, and a scheduler with everyone else.
- A request can be admitted, cached correctly, and scheduled fairly, and still fail the one check that was the point: did the answer use the required facts?

Goal: build one systems envelope — admission, state, scheduling, accounting, and evaluation — and hold it fixed while one mechanism at a time changes, on one running fixture, *AtlasLedger*.

Roadmap

1. Admission is not usefulness

Roadmap

1. Admission is not usefulness
2. The long-context system boundary

Roadmap

1. Admission is not usefulness
2. The long-context system boundary
3. Measuring one axis at a time

Roadmap

1. Admission is not usefulness
2. The long-context system boundary
3. Measuring one axis at a time
4. Context caching and prefix reuse

Roadmap

1. Admission is not usefulness
2. The long-context system boundary
3. Measuring one axis at a time
4. Context caching and prefix reuse
5. Admission, scheduling, and fairness

Roadmap

1. Admission is not usefulness
2. The long-context system boundary
3. Measuring one axis at a time
4. Context caching and prefix reuse
5. Admission, scheduling, and fairness
6. Benchmarks that separate retrieval from synthesis

Roadmap

1. Admission is not usefulness
2. The long-context system boundary
3. Measuring one axis at a time
4. Context caching and prefix reuse
5. Admission, scheduling, and fairness
6. Benchmarks that separate retrieval from synthesis
7. What the campaigns actually showed

Admission is not usefulness

DeepSeek-V3's reported ladder, and what it does not settle

$$4K \rightarrow 32K \rightarrow 128K$$

[REPORTED: DeepSeek-V3 technical report; DeepSeek-V3 `README.md`]

- That ladder is a capacity story: it establishes admission.
- It does not establish that content near token 100,000 is retrievable, or that the request fits local hardware.

Those are separate tests. This chapter runs them, on purpose, instead of assuming the ladder answers them.

The running fixture: AtlasLedger

How many calibrated sensors were shipped to Cedar in 2026-Q2?

- `ATLAS-CEDAR-17`: 37 sealed crates. `ATLAS-PACK-04`: 12 sensors per crate. Required answer: $37 \times 12 = 444$, citing both.

The running fixture: AtlasLedger

How many calibrated sensors were shipped to Cedar in 2026-Q2?

- `ATLAS-CEDAR-17`: 37 sealed crates. `ATLAS-PACK-04`: 12 sensors per crate. Required answer: $37 \times 12 = 444$, citing both.
- Deliberate distractors: a wrong-quarter Cedar row, a wrong-product packing rule. A fluent answer built from either is wrong.

The running fixture: AtlasLedger

How many calibrated sensors were shipped to Cedar in 2026-Q2?

- `ATLAS-CEDAR-17`: 37 sealed crates. `ATLAS-PACK-04`: 12 sensors per crate. Required answer: $37 \times 12 = 444$, citing both.
- Deliberate distractors: a wrong-quarter Cedar row, a wrong-product packing rule. A fluent answer built from either is wrong.
- A 14-token teaching view labels the two answer-bearing entries at positions $E = \{5, 11\}$, scaled so the arithmetic stays hand-checkable.

The running fixture: AtlasLedger

How many calibrated sensors were shipped to Cedar in 2026-Q2?

- `ATLAS-CEDAR-17`: 37 sealed crates. `ATLAS-PACK-04`: 12 sensors per crate. Required answer: $37 \times 12 = 444$, citing both.
- Deliberate distractors: a wrong-quarter Cedar row, a wrong-product packing rule. A fluent answer built from either is wrong.
- A 14-token teaching view labels the two answer-bearing entries at positions $E = \{5, 11\}$, scaled so the arithmetic stays hand-checkable.

Fluency is not what is scored. Citing the right two records and doing the right multiplication is.

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$
State	Which cache/representation/reuse path is active?	cache state

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$
State	Which cache/representation/reuse path is active?	cache state
Scheduling	How does the request share memory and compute?	Λ

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$
State	Which cache/representation/reuse path is active?	cache state
Scheduling	How does the request share memory and compute?	Λ
Accounting	Which positions were reused or recomputed?	U, C

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$
State	Which cache/representation/reuse path is active?	cache state
Scheduling	How does the request share memory and compute?	Λ
Accounting	Which positions were reused or recomputed?	U, C
Evaluation	Did required facts change the answer?	A

Five ways the fixture can fail

Failure risk	Question	Recorded field
Admission	Can prompt plus output reserve enter?	$L_{\mathrm{ctx}}, B_{\mathrm{in}}, B_{\mathrm{out}}$
State	Which cache/representation/reuse path is active?	cache state
Scheduling	How does the request share memory and compute?	Λ
Accounting	Which positions were reused or recomputed?	U, C
Evaluation	Did required facts change the answer?	A

One passing row does not imply the others passed. This table organizes every result in the rest of the chapter.

The long-context system boundary

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact
Reuse and eviction state	Invalid reuse, stale state, memory thrashing

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact
Reuse and eviction state	Invalid reuse, stale state, memory thrashing
Scheduler state	Good capacity, unacceptable queueing

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact
Reuse and eviction state	Invalid reuse, stale state, memory thrashing
Scheduler state	Good capacity, unacceptable queueing
Evaluation predicate	Completes successfully, fails retrieval or synthesis

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact
Reuse and eviction state	Invalid reuse, stale state, memory thrashing
Scheduler state	Good capacity, unacceptable queueing
Evaluation predicate	Completes successfully, fails retrieval or synthesis

- Four mechanisms can move resource use along this path: MLA changes the stored representation, context caching reuses eligible prefix state, sparse attention changes which positions are scored, and token-wise compression changes stored bytes.
- They may coexist, and each one needs its own measurement.

One path, five objects that can independently fail

System object	Failure made visible
Token and position map	Truncation, duplicated positions, invalid extension
Attention state	Cache-shape mismatch, missed distant fact
Reuse and eviction state	Invalid reuse, stale state, memory thrashing
Scheduler state	Good capacity, unacceptable queueing
Evaluation predicate	Completes successfully, fails retrieval or synthesis

- Four mechanisms can move resource use along this path: MLA changes the stored representation, context caching reuses eligible prefix state, sparse attention changes which positions are scored, and token-wise compression changes stored bytes.
- They may coexist, and each one needs its own measurement.

A request can complete nominally while a failure in any one state object invalidates its answer.

Measuring one axis at a time

The counter that makes recomputation concrete

$$U = (T_{\text{prompt}}, T_{\text{reused}}, T_{\text{prefill}}, T_{\text{completion}}), \qquad T_{\text{prompt}} = T_{\text{reused}} + T_{\text{prefill}}$$

- Example: $U = (14, 4, 10, 3)$ — four reused prefix tokens, ten measured-prefill positions, three generated tokens.

The counter that makes recomputation concrete

$$U = (T_{\text{prompt}}, T_{\text{reused}}, T_{\text{prefill}}, T_{\text{completion}}), \qquad T_{\text{prompt}} = T_{\text{reused}} + T_{\text{prefill}}$$

- Example: $U = (14, 4, 10, 3)$ — four reused prefix tokens, ten measured-prefill positions, three generated tokens.
- Cache construction stays in workload accounting even when the request-latency window excludes it.

The counter that makes recomputation concrete

$$U = (T_{\text{prompt}}, T_{\text{reused}}, T_{\text{prefill}}, T_{\text{completion}}), \qquad T_{\text{prompt}} = T_{\text{reused}} + T_{\text{prefill}}$$

- Example: $U = (14, 4, 10, 3)$ — four reused prefix tokens, ten measured-prefill positions, three generated tokens.
- Cache construction stays in workload accounting even when the request-latency window excludes it.
- Prerequisite mechanisms — MLA, DeepSeek Sparse Attention, paged KV-cache allocation, the FlashMLA kernel boundary, precision — become independent measurement axes here, not new definitions.

The counter that makes recomputation concrete

$$U = (T_{\text{prompt}}, T_{\text{reused}}, T_{\text{prefill}}, T_{\text{completion}}), \qquad T_{\text{prompt}} = T_{\text{reused}} + T_{\text{prefill}}$$

- Example: $U = (14, 4, 10, 3)$ — four reused prefix tokens, ten measured-prefill positions, three generated tokens.
- Cache construction stays in workload accounting even when the request-latency window excludes it.
- Prerequisite mechanisms — MLA, DeepSeek Sparse Attention, paged KV-cache allocation, the FlashMLA kernel boundary, precision — become independent measurement axes here, not new definitions.

A recomputation number that quietly drops cache-build work is not comparable to one that keeps it.

Context caching and prefix reuse

From concept to code: partitioning reused work

Ten requests share a 4,000-token stable prefix and each add a 1,000-token suffix. Cold: $10(4000 + 1000) = 50{,}000$ prefill positions.

from `code/chapter_long_context_systems/long_context_record_reference.py`

```python
def prefix_reuse_totals(prefix_tokens, suffix_tokens, requests):
    total = requests * (prefix_tokens + suffix_tokens)
    reused = (requests - 1) * prefix_tokens
    prefilled = prefix_tokens + requests * suffix_tokens
    ...
    return {"prompt": total, "reused": reused, "prefilled": prefilled}
```

From concept to code: partitioning reused work

Ten requests share a 4,000-token stable prefix and each add a 1,000-token suffix. Cold: $10(4000 + 1000) = 50{,}000$ prefill positions.

from `code/chapter_long_context_systems/long_context_record_reference.py`

```python
def prefix_reuse_totals(prefix_tokens, suffix_tokens, requests):
    total = requests * (prefix_tokens + suffix_tokens)
    reused = (requests - 1) * prefix_tokens
    prefilled = prefix_tokens + requests * suffix_tokens
    ...
    return {"prompt": total, "reused": reused, "prefilled": prefilled}
```

Idealized reuse: $T_{\text{prefill}} = 14{,}000$, $T_{\text{reused}} = 36{,}000$ — the two classes still sum to 50,000. Reuse requires exact token and interpretation identity, checked against a no-reuse oracle, not just a string match. [INSPECTED: companion code `long_context_record_reference.py`]

Admission, scheduling, and fairness

Fits alone is not fair alongside

$48\ \mathrm{GiB} + 20\ \mathrm{GiB} > 64\ \mathrm{GiB}$ cache budget $\Rightarrow$ delay, reject, preempt, evict, or compress.

- A mixed-workload fairness contract needs *both* sides: a short-request tail-latency bound while long requests run, and a long-request progress bound while short requests keep arriving.
- Protecting only one class is priority or isolation, not fairness.

A system can pass the single-request view — fits, completes, answers correctly — and still fail the mixed-workload view.

Benchmarks that separate retrieval from synthesis

Two predicates, two failure modes

- **Retrieval** $R_{\mathrm{ret}}(y, E)$: does y identify the required entries, independent of how well it is written?
- **Synthesis** $R_{\mathrm{syn}}(y, a^{\star})$: does y use the retrieved facts to produce the required answer?
- Citing the right entries but drawing the wrong conclusion passes retrieval and fails synthesis — and the reverse also happens.

A minimum suite reports needle retrieval, multi-evidence synthesis, lost-middle placement, adversarial distractors, and mixed-workload serving as separate rows, never pooled into one accuracy number.

What the campaigns actually showed

The reviewed evidence, kept on its own scope

Comparison	Reviewed outcome
64K vs. 131K	One H200, 32B model: fixed-suite joint score fell 26/27 $\rightarrow$ 9/27.

[OBSERVED: bounded *AtlasLedger* campaigns]

The reviewed evidence, kept on its own scope

Comparison	Reviewed outcome
64K vs. 131K	One H200, 32B model: fixed-suite joint score fell 26/27 $\rightarrow$ 9/27.
Guarded prefix reuse	131,072 of 655,360 positions reused; 30 logits tensors matched bit for bit.

[OBSERVED: bounded *AtlasLedger* campaigns]

The reviewed evidence, kept on its own scope

Comparison	Reviewed outcome
64K vs. 131K	One H200, 32B model: fixed-suite joint score fell 26/27 → 9/27.
Guarded prefix reuse	131,072 of 655,360 positions reused; 30 logits tensors matched bit for bit.
Native SGLang cache	All 2,048 prefix positions reused warm/restored; a position-1,024 mutation stopped reuse there.

[OBSERVED: bounded *AtlasLedger* campaigns]

The reviewed evidence, kept on its own scope

Comparison	Reviewed outcome
64K vs. 131K	One H200, 32B model: fixed-suite joint score fell 26/27 → 9/27.
Guarded prefix reuse	131,072 of 655,360 positions reused; 30 logits tensors matched bit for bit.
Native SGLang cache	All 2,048 prefix positions reused warm/restored; a position-1,024 mutation stopped reuse there.
Mixed traffic	p95 TTFT 7.82–11.92 s across offered rates 0.75–4.0 req/s; every point failed the 5 s and 2× bounds.

[OBSERVED: bounded *AtlasLedger* campaigns]

The reviewed evidence, kept on its own scope

Comparison	Reviewed outcome
64K vs. 131K	One H200, 32B model: fixed-suite joint score fell $26/27 \rightarrow 9/27$.
Guarded prefix reuse	131,072 of 655,360 positions reused; 30 logits tensors matched bit for bit.
Native SGLang cache	All 2,048 prefix positions reused warm/restored; a position-1,024 mutation stopped reuse there.
Mixed traffic	p95 TTFT 7.82–11.92 s across offered rates 0.75–4.0 req/s; every point failed the 5 s and $2\times$ bounds.

[OBSERVED: bounded *AtlasLedger* campaigns]

Answer quality and service fairness are independent axes: all mixed long answers still passed while short-request tails blew through the bound.

Execution evidence forms a scope ladder, not a subtraction

Input tokens	Endpoint TTFT p50	Eager device p50	Kernel-event p50
8,192	0.894 s	0.857 s	0.838 s
130,816	31.943 s	32.332 s	31.668 s

- The dominant retained kernel name shifts from NVJet TNN at short lengths to FlashAttention SM90 at long lengths.
- Client TTFT, eager device time, cumulative kernel-event duration, and direct decode rate answer different questions.

Only values paired under one declared configuration support subtraction or causal attribution.

Concept-to-code map for the chapter

- **AtlasLedger record validation and grading** → `code/chapter_long_context_systems/long_context_record_reference.py`
- **MLA cache buffers behind the representation axis** → `DeepSeek-V3/inference/model.py`
- **Sparse-attention kernel surface** → `FlashMLA/flash_mla/flash_mla_interface.py`
- **V3.2 sparse long-context mechanism** → `DeepSeek-V3.2-Exp/inference/model.py`

Hold the answer predicate fixed. Change one systems axis at a time.

Next: the precision chapter supplies payload and tolerance rules for compressed state; the System Correctness chapter turns these cache, scheduler, and replay obligations into explicit oracles.