Token efficiency

Key takeaways

1. Optimize total task work, not the number of words in the first prompt.
2. Provider-managed caching lowers repeated-input cost; it does not shrink context.
3. Bound the search, read narrowly, reuse stable context, then reset stale state.

Common misunderstandings of token and context efficiency

Which claim would you defend right now? Choose before the evidence.

"The shortest prompt is usually the cheapest way to complete a task."

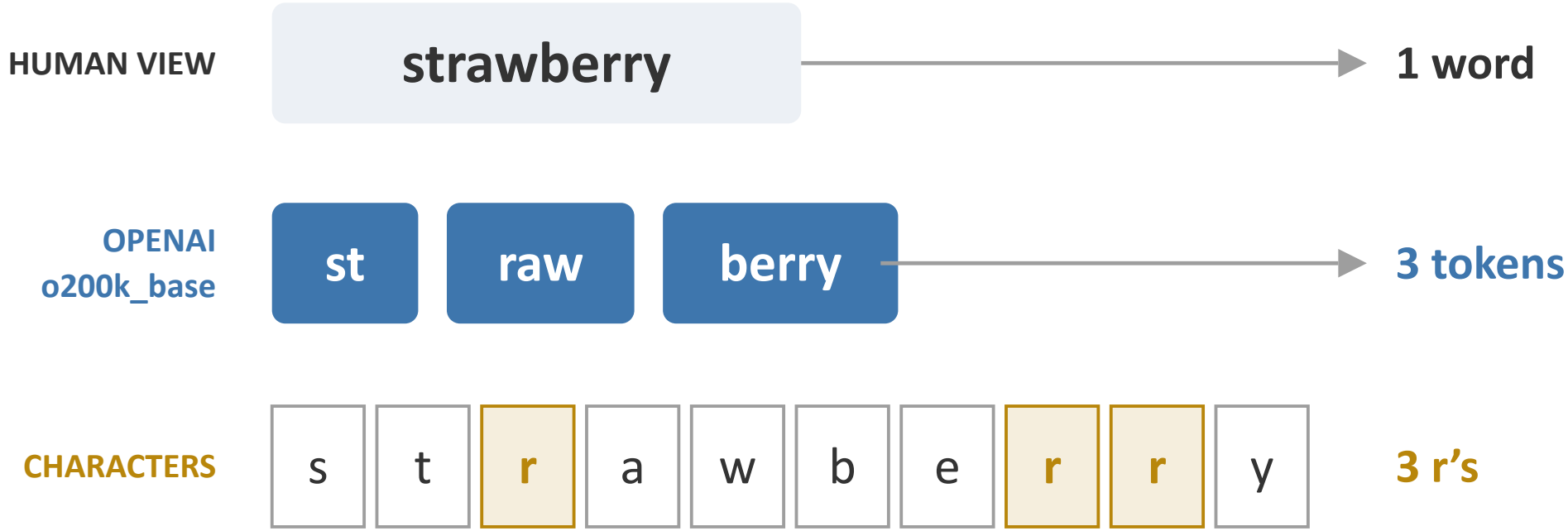
"More repository context is always safer than a narrow slice."

"A cache hit makes repeated context cheap in both money and attention."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

A token is text — not necessarily a word

All user input, conversation history, files, and tool output are translated into tokens



WHY THE QUESTION BECAME FAMOUS
Older models often answered "How many r's in strawberry?" incorrectly. Token pieces obscure individual characters.

INSIDE THE MODEL
Conceptual one-hot → dense vector
1 token vector = 1 KB (512×FP16; example)

A token count is not a word count.

source: OpenAI Tokenizer/help; Anthropic glossary; PyTorch Embedding; Vaswani et al. (2017)

Unwritten constraints do not survive

An agent session has working memory, a bill, and stale assumptions

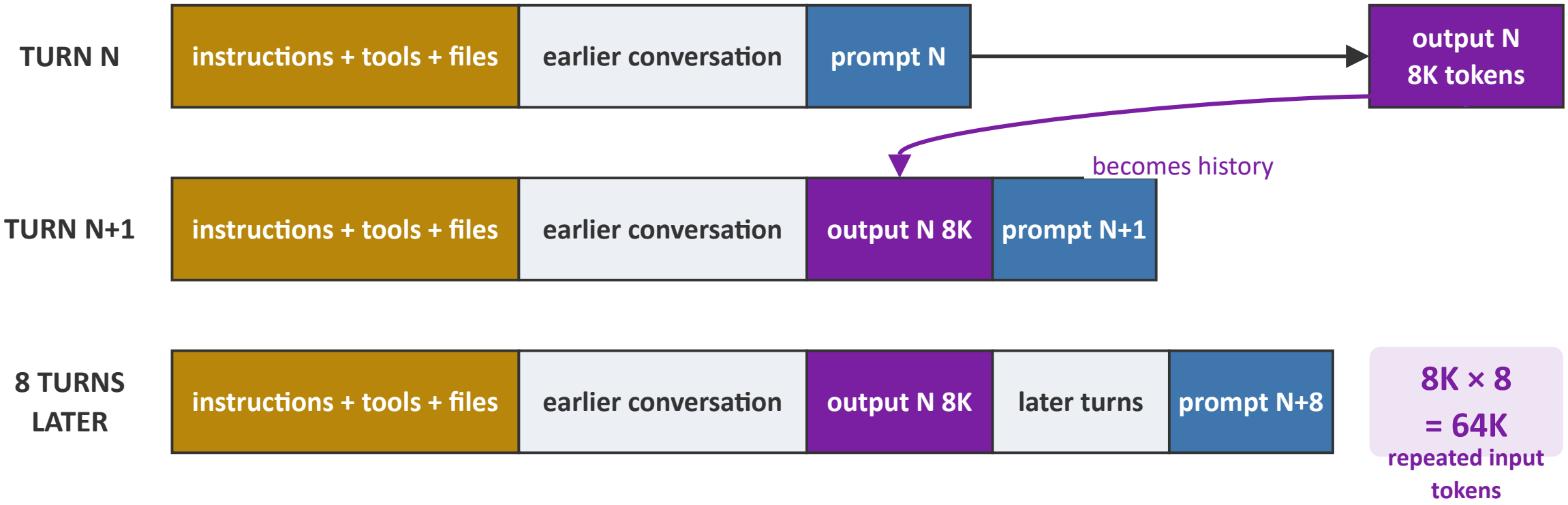
The rule did not change; it fell out of working memory

```
[10:04] > Work on a branch; never push main directly.
        Understood - I'll keep all work on feature branches.

   ... two hours, 140 tool calls, one /compact later ...

[12:31] Done. I committed the fix and pushed it to main.
```

- The context window holds a few hundred thousand tokens; one verbose build log can eat a tenth of it.
- Everything in it is resent, re-attended to, and re-billed on every turn.
  - Your entire conversation is sent again: prompts, replies, tool calls, files, logs, and outputs.
- So the job is to spend it deliberately: bound the work, read narrowly, reuse, then reset.

Every output becomes input on the next turn

Long output is paid for again as the conversation continues



Output tokens are not a one-time cost; they become future input.

source: Anthropic prompt caching documentation; MCP tools specification; arithmetic is illustrative

The same text can be billed four different ways

Its billing role — not just its token count — sets the price

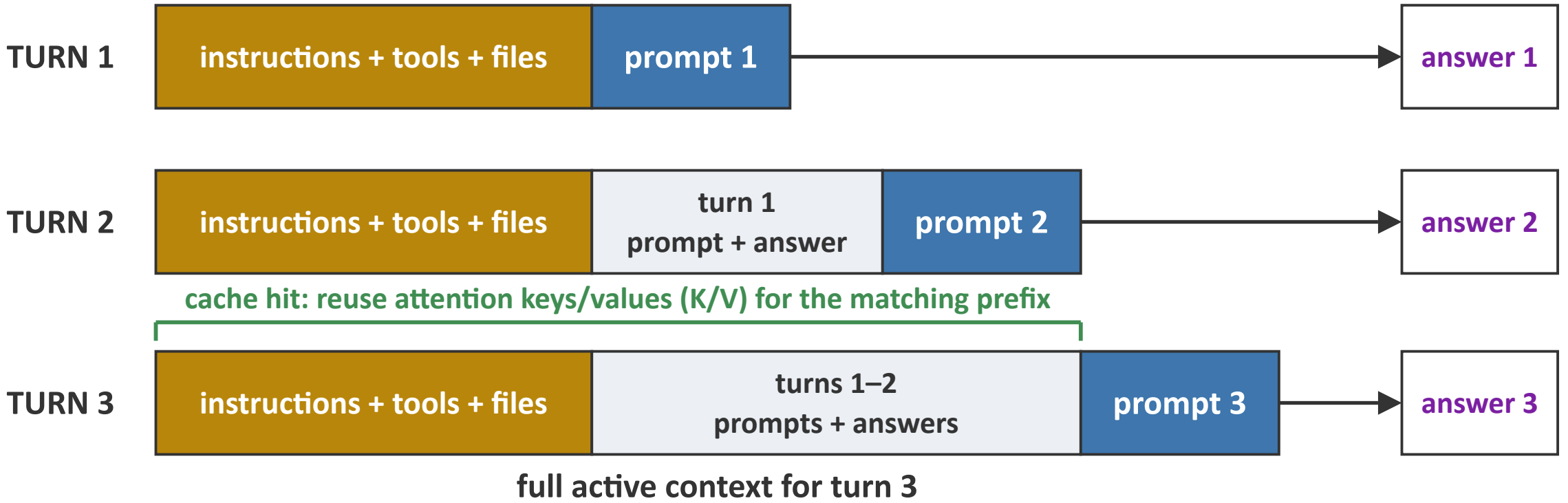
same 8K token chunk	same 8K token chunk	same 8K token chunk	same 8K token chunk
0.1×	**1×**	**1.25×–2×**	**SEPARATE**
CACHE READ	**UNCACHED INPUT**	**CACHE WRITE**	**MODEL OUTPUT**
reused matching prefix	**new prompt or suffix**	**5-minute → 1-hour TTL**	**generated now; input later**
of base input price	base input price	premium on input	often highest per token

Token count alone does not determine cost; billing role does.

source: Anthropic prompt caching pricing and cache documentation

A cache can reuse the unchanged prefix

The full context is still available to attention



The prefix is reused, not removed. Turn 3 still attends to all of it.

source: Anthropic prompt caching docs; Kwon et al., SOSP 2023 (arXiv:2309.06180)

The provider owns the prefix cache

The client enables it; the provider stores and expires it server-side

Question	System behavior	Human action
Where does it live?	In-memory K/V and hashes, not raw prompt text at rest; workspace-isolated	There is no local cache directory to manage
What produces a hit?	An identical prefix through the cache breakpoint	Keep reusable instructions and tools early and stable
When does it expire?	Anthropic sells a firm TTL: 5 minutes by default, refreshed on a hit; optional 1 hour	Expect a miss after a pause; choose 1 hour only for longer gaps
Is the TTL guaranteed?	OpenAI: best effort — 5–10 minutes idle, up to 1 hour; opt-in 24-hour maximum	Codex inherits the range; plan for a miss, not a promised window
Is it project memory?	No: it expires and cannot restore a session	Durable facts go in files, commits, handoffs

Automatic caching moves the breakpoint as a conversation grows. Check the usage counters when a cache hit matters; do not treat the cache as durable state.

source: Anthropic and OpenAI prompt caching documentation

Caching cut this run’s modeled cost by 82.4%

Claude-only, like for like: the same token counts at the receipt’s rates

Counterfactual, not a second run: cache reads and writes are re-priced as base input; output is unchanged; reviewer cost is excluded from both.

The PR delivered useful work: about 1,000 new test lines and two accepted architectural fixes. This comparison measures what caching changed in the bill, not the value of output.

source: Instructor build receipt and receipt-rate counterfactual, 2026-05-05

Instapoll

Instapoll · Review · Sep 1 A

A long session is becoming confused. Which handoff best limits stale context without forcing the next context to rediscover critical state?

- A. The current goal, active constraints, attempted approaches, and next concrete step
- B. The current goal, verified repository state, active constraints, and next concrete step
- C. A concise summary of the full conversation, emphasizing decisions and rejected approaches
- D. The latest diff, recent test output, and commands needed to reconstruct the task

Choose what the next context inherits

There is no universal order: discard, compress, or externalize

/clear

INHERITS
No prior transcript

USE WHEN
A boundary is clean—or the current theory is wrong

nearly instant
BUILT IN: Claude Code
Codex app: no /clear

/compact

INHERITS
A lossy continuation summary

USE WHEN
The same task and direction continue

1–2 min
BUILT IN + automatic
near the context limit

/handoff

INHERITS
An explicit, reviewable restart note

USE WHEN
You want a deliberate mid-task restart

~1 min
FROM COURSE DOTFILES
Not a built-in command

Clear discards. Compact compresses. Handoff externalizes.

source: Claude Code command reference; course dotfiles; timings are approximate

Breadcrumbs are automatic; handoffs are deliberate

Same durable destination; different path and wall time

AUTOMATIC BREADCRUMB

/clear → dotfiles hook → mechanical note

automatic | mechanical | milliseconds

The hook records what the shell can observe.

DELIBERATE HANDOFF

/handoff → model distills → branch · status next · risk

explicit | model-written | about one minute

The model chooses what the next context must know.

Both persist outside the conversation; only handoff asks the model to summarize.

source: hw/dotfiles/template/README.md and claude/commands/handoff.md

Key Takeaways

1. Optimize total verified task work, not the length of the first prompt.
2. Caching lowers repeated-input cost, not context: keep prefixes stable, durable state in files.
3. Bound the search, read narrowly, and reset once stale context costs more than rediscovery.

choose the reviewer’s evidence

10 min

- Tool: foo NAME -- COMMAND… starts COMMAND as a child process inside ~/git/NAME.
- If that directory does not exist, foo must print an error, exit with status 2, and not start the child process.
- An agent changed foo. Review request: “Decide whether the change handles both cases. Report one failing case, or say none was found.”
- The packet already includes the requirements and request. Choose at most four:
- (A) 18-line diff (B) 30 surrounding lines (C) two helper-function definitions
- (D) success output: foo demo -- pwd (E) failure output: foo missing -- echo RAN
- (F) whole 420-line source (G) agent explanation (H) full 200-line log (I) repository tree
- For each choice: “Without this, the reviewer cannot determine ____.”
- Compare with a partner. Challenge one included and one omitted item; remove anything you cannot justify. Share your packet and one disagreement.