

The cheapest step is the one Claude never takes

Lab note 1 15 Oct 2026 4 min read
written by Claude from this repository's data

Ask most people what makes a Claude Code session expensive and they will say tokens: big files, long answers, a chatty model. That is true, but it points at the wrong lever. Our first useful result was learning where the money actually goes.

Every step re-reads everything

A Claude Code session is a loop. Claude gets the conversation so far, decides on one step (read a file, run a grep, make an edit), the tool answers, and the answer joins the conversation. Then the whole conversation goes back to the model for the next step.

The whole conversation, every time. Prompt caching makes the re-read cheap per token: a cached token costs \$0.20 per million instead of \$4 on Opus 5.5. It does not make it free, and it multiplies. In a session holding 220,000 tokens, every step costs about \$0.044 before Claude has thought about anything. A task that takes eight steps instead of five pays that three more times, plus the output of each step, plus whatever those steps add to the context, which every later step re-reads.

So the unit of cost is the **request**: a step of the main model. That shaped everything else. A mod that makes each tool answer 10% shorter saves a little. A mod that removes a step saves a whole re-read of everything.

What the pack actually saves

The pack's first measurements came out exactly along that line (SUITE.md). Tokens per request barely moved. Requests did: 5.2 per task with the pack against 7.2 without, on the same seven tasks, every run graded correct.

The release measurement of version 0.3.0, 116 runs, put numbers on it (SUITE.md):

	Change in cost per task, pack against plain Claude Code	95% interval
Long session (Claude has read the project)	-26.2%	-31.3% to -20.9%
Fresh session, multi-step tasks	-10.9%	-17.2% to -3.9%
Fresh session, quick questions	+5.5%	+1.2% to +10.2%

The long session is where the pack pays most, because that is where a removed step removes the most re-reading. In a long session Claude needed 3.5 requests per task with the pack instead of 5.4.

Which mod does it

An ablation takes one mod out at a time and measures what is lost (SUITE.md). The answer was not the mod we expected.

- **cells** lets Claude make several edits and run the tests in one step, through Claude Code's own edit tool. Taking it out made the pack 8.9% more expensive [+2.1%, +16.4%], and 16% on multi-step work. The feature task took 14.2 requests without it instead of 9.8.
- **lsp-plus**, which answers code searches from the language server, showed no net effect on the whole (-0.6% [-7.2%, +6.7%]). On the one task where it fires most, a rename, the pack without it needed 5.0 requests instead of 3.2. It helps exactly where Claude searches for a symbol, and is silent elsewhere.

The quick-question tax

The +5.5% on quick questions is the other side of the same arithmetic. The pack's tools, their descriptions and its guidance are about 4,000 tokens: 30.6k in the first request of a session instead of 26.7k. Cached, that is about \$0.001 per request. But a fresh session has to write it to the cache first, and a three-step lookup has nothing to win back.

We could have hidden this in an average. We put it in the README instead, because it is the honest shape of the trade: the pack costs a little where there is nothing to save, and saves a lot where there is.

Then it did not replicate. Measured again two days later with the next version of the pack (note 8), quick questions cost **3.6% less** [-8.2%, +1.3%], and on a TypeScript library 9.9% less (note 9). Both days pooled: +2.3% [-1.1%, +6.0%], an interval across zero. The arithmetic above is real, but it is about a cent, and a cent is inside the spread of a three-step lookup from one day to the next. The README now says "about even". The long session replicated in the other direction: -39.9% on the second day, -33.2% [-38.9%, -26.9%] pooled.

The biggest saving was not a mod

While measuring that tax, we found a bigger one we were paying without knowing it. The person running this lab had six claude.ai connectors enabled (mail, a whiteboard, file storage and others), called between 0 and 13 times each in 946 sessions. Turning them off took the deferred tools offered to every session from 378 to 15, about 6,000 tokens of names and instructions, and cut the cost of the four quick tasks by 34%, with the same number of requests (SUITE.md). That is more than any mod in this pack saves on small work. If you take one thing from these notes: look at what your sessions carry that you never use.

What this changed

- We count requests first, tokens second. A feature that adds tokens but removes a step is usually a win; one that saves tokens but adds a step usually is not.
- We measure in long sessions, not only fresh ones. A fresh session is where demos happen; a long session is where the money goes.
- A mod has to pay for its own prompt. Anything that adds text to every request has to save more than it costs on the tasks where it does nothing.

Method & limits

Click 8.5.0, Opus 5.5, headless Claude Code, costs from transcripts at list prices; 2–8 runs per task and condition; intervals are 95% bootstrap intervals over runs. The connector measurement compared runs from different hours, so the cache's state may differ a little; the size of the effect is not in doubt. All tasks were solved by every condition, so these numbers say nothing about quality on harder work. Data:

```
results-030.jsonl , results-ablation.jsonl , results-noconn.jsonl ; harness: bench.py .
```

Sources

- Anthropic, Prompt caching: how cached prompt prefixes are written and read, and what each costs.