How much Nasiko's compression saves, and what it costs in accuracy

Two studies · A: real system logs (LogHub), run 22 Oct 2026 · B: public agent test data, run 2 Nov 2026

AI agents read tool output (logs, search results, lists of records). All of it goes to the AI model, which bills by the **token** (a word piece; a model's unit of reading and billing). Nasiko **compresses** that output first. We asked the same questions with compression off and on, to measure the tokens saved and whether answers stay right. **It keeps the prompt cache working:** since 27 Oct 2026 the same tool output always compresses to the same text, so the provider's discount for repeated (cached) input still applies.[C1] These studies did not measure that discount.

-42% tokens

Logs (Study A). 50 questions about error lines: input tokens fell 42% and cost 40%. Correct answers stayed at 44 of 50.

-33% tokens

Records (Study B). On public agent test data, every value was kept. gpt-4o got 96 of 102 right against 100 uncompressed; gpt-4o-mini tied at 68.

Varies by content

Input tokens saved: logs 17–69% by system, incident debugging 42%, codebase exploration 30%, issue triage 28%, code search 26%; unrecognised formats (Linux logs) about 1%.

STUDY A · REAL SYSTEM LOGS

Data. Real logs from five systems, taken from **LogHub**, a public research collection, at a fixed version.[A1,A2]

Method. A log-reading agent answered 50 questions such as "what is the last error in lines 400–600?" A script wrote them and took each answer from the log. Each question ran in a new conversation, once per setting, in alternating order, with the same agent, model and instructions. Token counts are those OpenAI reported.[A3]

50 error questions · gpt-4o	Off	On	Change
Correct answers	**44 / 50**	**44 / 50**	**none**
Input tokens	**617,202**	**357,958**	**−42%**
Output tokens	6,831	6,525	−4%
Cost at list price[C2]	**$1.61**	**$0.96**	**−40%**
Cost per correct answer	$0.037	$0.022	−40%

95% range for the saving (from resampling the 50 questions 20,000 times): tokens 34–49%, cost 33–47%. Four answers went wrong to right and four right to wrong. No question was left out.

Figure 1 · How the test ran

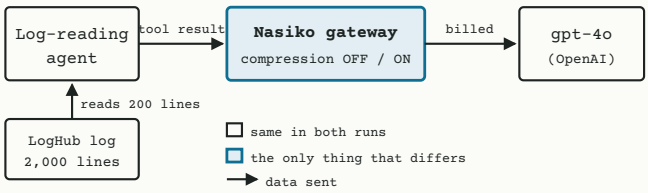
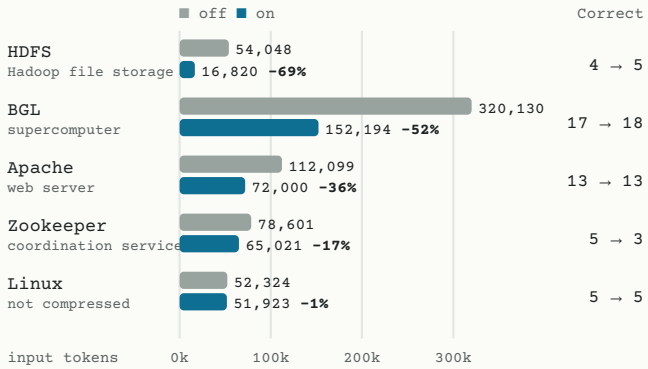


Figure 2 · Input tokens per log, off vs on (50 questions)



Analysis

- **The saving follows how routine a log is.** Per question, the saving ranged from 0% (Linux) to 82% (a BGL question). The compressor removed 79% of the text of HDFS reads, 62% for BGL, 50% for Apache and 20% for Zookeeper.[A3] Token savings are smaller because the agent's instructions and question are sent whole.
- **Zookeeper is the weak spot:** mostly warnings, so little was removed, and on 3 questions the model missed error lines that were there (1 other answer improved).
- **Linux logs were left whole:** their format was not recognised, so they were sent as they were. No saving, no harm.

Study A sources

A1. LogHub, commit dd61d09, files <System>_2k.log: github.com/logpai/loghub. Research use, with citation. A2. J. Zhu, S. He, P. He, J. Liu, M. R. Lyu, "Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics", ISSRE 2023, arxiv.org/abs/2008.06448. A3. Nasiko run records, 22 Oct 2026: 112 runs with OpenAI-reported tokens and bytes compressed; all figures recomputed from them. On request.

STUDY B · PUBLIC AGENT TEST DATA

B1 · Four agent workloads. 10 tool outputs (178,024 tokens, 15,309 values): code search, incident debugging, codebase exploration and GitHub issue triage. A published script made them with a fixed seed (the start value for its random choices).[B1] We wrote 102 questions that each look up one field of one record. The model saw each output whole or compressed, 15 questions per call, at temperature 0 (the most repeatable setting).

B2 · BFCL, the Berkeley Function Calling Leaderboard: the first 100 "simple" cases, each a set of tool definitions and a request.[B2] **B3 · 8 sample tool outputs** (JSON, logs, a traceback).[B3] These two ran in a published test harness; an answer counted as correct when it contained the expected answer. Our records hold every output, question, call and token count for all three.[B4]

B1 · 102 lookups	Off	On	Change
Correct, gpt-4o	**100**	**96**	**−4**
Correct, gpt-4o-mini	68	68	none
Input tokens (gpt-4o)	**343,212**	**229,049**	**−33%**
Values still in the output	15,309	15,309	all kept
B2, B3 · model Flint	**Off**	**On**	**Change**
BFCL correct (100 cases)	94	92	−2
BFCL input tokens	18,051	12,373	−31%
Samples correct (8 cases)	8	8	none
Samples input tokens	1,390	1,009	−27%

Flint is an open model (3-bit version) we ran on our own machine. Token counts are those OpenAI reported (B1) and the harness's own counts (B2, B3). **Note:** all three data sets are published by Headroom, an open-source compressor; we used them only as test data.[B1–B3]

Figure 3 · How the tests ran

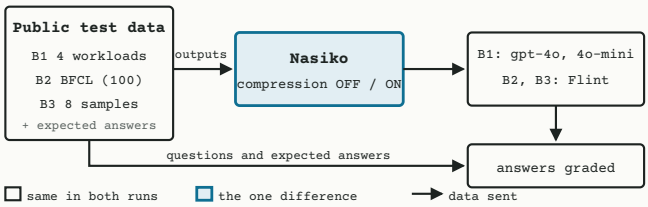
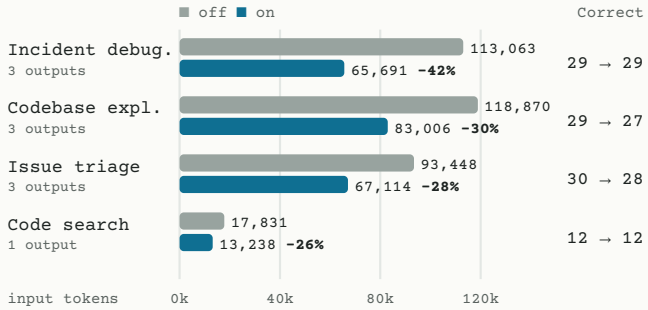


Figure 4 · B1: input tokens per output type, off vs on (gpt-4o)



Analysis

- **Nothing is lost on paper.** Decoding the compressed outputs gives back all 15,309 values.[B4]
- **A new kind of mistake.** Of the 5 answers gpt-4o lost (it gained 1), 4 were a value from a different record and 1 was a file path that does not exist. In a table a value no longer sits next to its field name, so the model must match row and column itself. gpt-4o-mini tied at 68 but changed 20 answers.
- **BFCL saves only layout:** compression removes extra spaces in the tool definitions. The answer changes (6 lost, 4 gained) are within the small model's noise.
- **Samples:** all 8 answers stayed right. 3 of the 8 (events, a traceback, a commit log) were left unchanged.

Study B sources

B1. Workloads: benchmarks/real_world_agent_benchmark.py, seed 20260902, in github.com/headroomlabs-ai/headroom, commit a2d4fe7 (Apache-2.0).
B2. BFCL v3 "simple", cases simple_0–simple_99: huggingface.co/datasets/gorilla-llm/Berkeley-Function-Calling-Leaderboard (Apache-2.0). Patil et al., ICML 2025, PMLR 267.
B3. load_tool_output_samples(), headroom/evals/datasets.py, B1's commit.
B4. Nasiko records, 2 Nov 2026 (outputs, questions, calls, tokens), and the harness in headroom/evals at B1's commit. On request.

ALSO IN NASIKO · SHORTER CHAT HISTORY (PACMS)

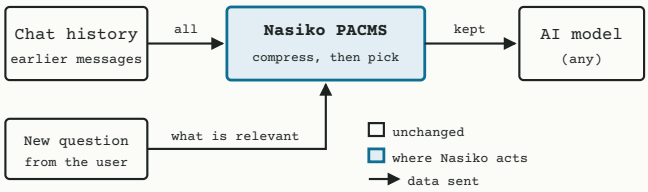
Agents resend the earlier conversation on every turn, and it grows with each message. Nasiko trims it two ways, both on by default:[C3]

- **PACMS** (Pluggable Agent Context Management System) keeps the earlier messages most relevant to the new question, within a budget each user chooses: 500, 1,000 (the default) or 5,000 tokens. The 3 newest messages are always kept.
- **History compression** shrinks older messages before they are chosen.

Published results. A paper by Nasiko's researchers tested PACMS on 100 questions from LongMemEval, a public test of long-term chat memory with long multi-session histories, keeping 45% of each history's tokens.[C4,C5]

Inside Nasiko, at its budgets of 500, 1,000 (default) and 5,000 tokens, 500 LongMemEval questions each (the 'oracle' version: only the sessions that hold the answer). OpenCode answered and Qwen3-32B judged.[C6]

Figure 5 · How history is chosen each turn



Correct · 500 questions	PACMS	Top-k	MMR	Newest
Low (500)	**63%**	**34%**	**34%**	**20%**
Medium (1,000)	**77%**	**64%**	**68%**	**31%**
High (5,000)	80%	78%	82%	64%

Newest (medium) is out of 484 and top-k and newest (low) out of 499, after failed calls were excluded; details in C6.

Keeping 45% of the history · 100 questions	Evidence kept	Correct, GPT-5-mini	Correct, GPT-5.4-mini
PACMS	**90.9%**	**52%**	**68%**
Most similar messages (top-k)	93.3%	50%	64%
Similar but varied messages (MMR, LangChain)	89.5%	44%	56%
Newest messages only	22.1%	44%	42%

"Evidence kept" = share of the history rounds that hold the answer and survived selection, with near-duplicate filler rounds added (redundancy level R = 2 in the paper). Correct answers were judged by GPT-4o-mini with LongMemEval's official prompt. Source: paper Tables 1 and 2.[C4]

Analysis

- **Most right answers.** PACMS led every other selector under both readers: +2 and +4 points over top-k, +8 and +12 over MMR.
- **Keeping evidence is not enough.** Top-k kept slightly more answer-bearing rounds (93.3% vs 90.9%) yet got fewer answers right: how the kept context reads matters too.
- **Keeping only the newest messages fails** when the history repeats itself: it kept 22% of the evidence and did not improve with the stronger reader (44% → 42%).
- **Inside Nasiko, PACMS matters most when the budget is tight:** it leads clearly at the low and medium budgets (63% vs 34%; 77% vs 68%). At the high budget, where almost all the evidence fits, MMR edges it (82% vs 80%).

ABOUT THESE RESULTS

Both studies

1. **Focused runs.** Each study is one run of 50 to 102 questions; "no change" means none was seen in these runs.
2. **Run in-house.** Nasiko designed and ran the tests. Full records are on request; independent repeats are welcome.
3. **Cost basis.** Costs use public list prices for model calls, before any provider discount for repeated (cached) input.

Study A

4. **Error-focused questions.** They ask about error lines, the compressor's main use. Its settings (what it keeps and compresses, on or off per agent) can be tuned to a team's own logs.
5. **Routine-line questions are harder.** In 6 extra lookups about ordinary lines, compression got 4 right against 6 without, and the agent re-read the log, using more tokens.
6. **Log samples.** Each log is a public 2,000-line sample, read 200 lines at a time.

Study B

7. **Generated data.** B1 and B3 use generated records in realistic formats; B3 is a small set of 8 cases. Repeat runs of B1 vary by about 2 answers.
8. **Single reading.** Each output was read once, without follow-up turns.
9. **Model for B2 and B3.** These ran on a small open model on our own hardware, with a pre-release Nasiko build.

Chat history

10. **Two sources.** The paper (one budget, one duplicate level, 100 questions) is by Nasiko's researchers. The in-Nasiko run used another model, judge and data version, so the two are not directly comparable. Studies A and B started every question in a new conversation, so history selection played no part there.

Prices, code and papers

C1. Nasiko change "keep compressed prompts cache-stable and lossless when unrecoverable" (27 Oct 2026). Not yet in the public repository.

C2. gpt-4o list price: $2.50 per million input tokens and $10 per million output tokens. openai.com/api/pricing

C3. History defaults: config/src/lib.rs and migrations 0038–0039 in github.com/Nasiko-Labs/nasiko.

C4. M. Ghulyani et al., "PACMS: Submodular Context Selection as a Pluggable Engine for LLM Agents", arXiv:2606.20047 v2 (3 Sep 2026), Tables 1–2.

C5. D. Wu et al., "LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory", ICLR 2025.

C6. Nasiko three-tier PACMS report (LongMemEval oracle, 500 questions per budget; OpenCode on openai.gpt-5.6-terra via Bedrock), 13 Nov 2026. On request.