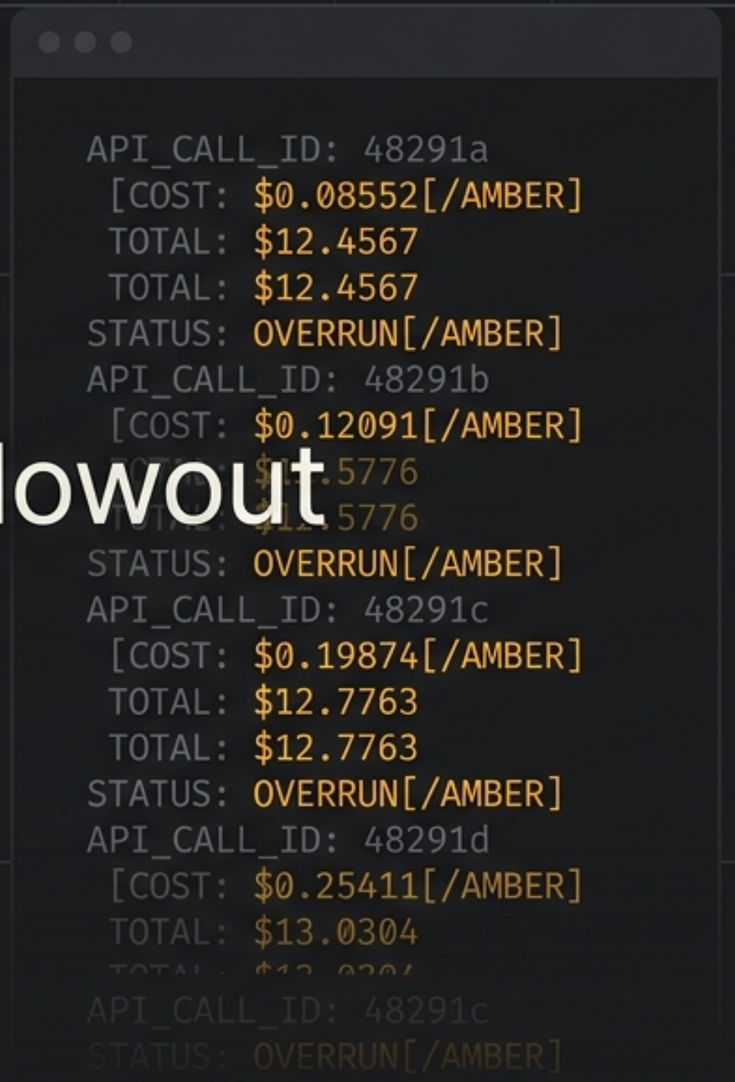
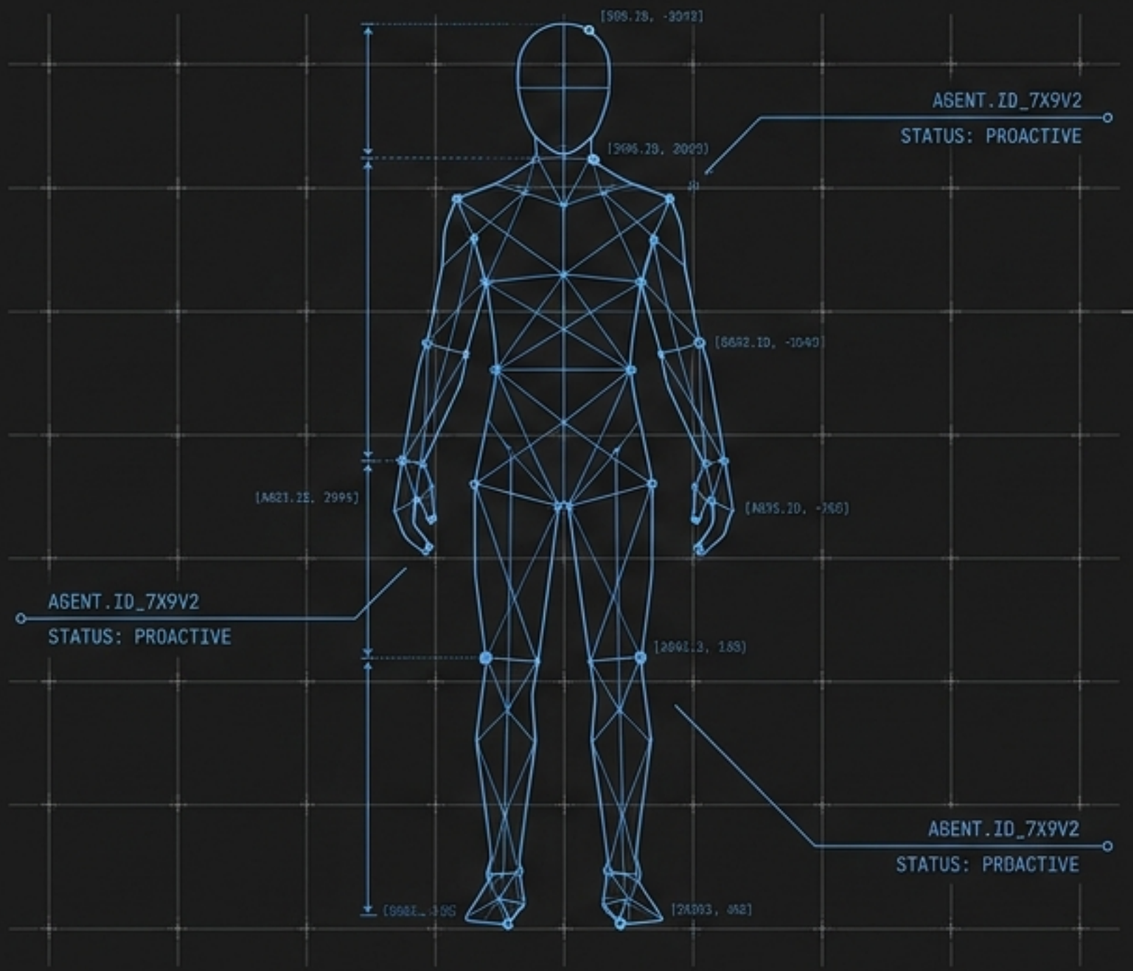
Autopsy of an LLM Cost Blowout

Forensic analysis of a 24/7 AI agent that burned a staggering budget in under two weeks.



The 24/7 Cyber Companion Experiment

A fully autonomous companion agent with an emotion system, memory, and proactive messaging, running on Telegram.



● [FRAMEWORK / ENGINE]

[FRAMEWORK]: OpenClaw
[ENGINE]: Gemini Pro (1M Context Limit)

PANEL.ID: 001 DATA.STREAM: ACTIVE

● [SYSTEM PERFORMANCE]

[UPTIME]: 2.5 Days (Current Session)
[OUTPUT]: 537 Turns

PANEL.ID: G01 LOAD: 98%

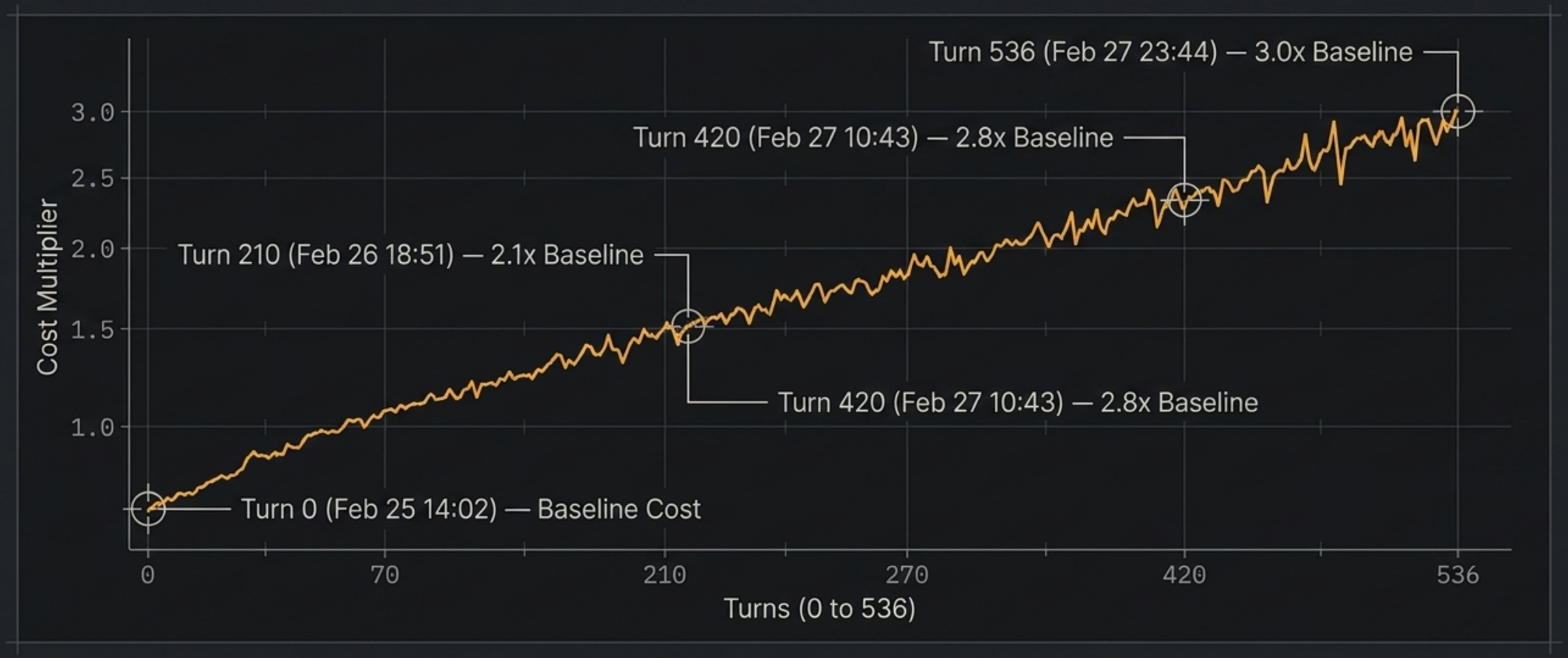
⚠ [CRITICAL STATUS]

[STATUS]: Monotonically increasing cost-per-turn. Budget exceeded.

PANEL.ID: 001 ALERT.CODE: C-101

The Context Was Growing Unboundedly

A script analysis of the session's `.jsonl` transcript revealed a relentless upward trend in token consumption per turn.



Isolating the Source of the Token Burn

An investigation initiated via Claude Code quickly mapped the turn distribution. Proactive and background tasks were largely innocent.

[TURN DISTRIBUTION] DATA.STREAN: MAPPED

Regular Chat

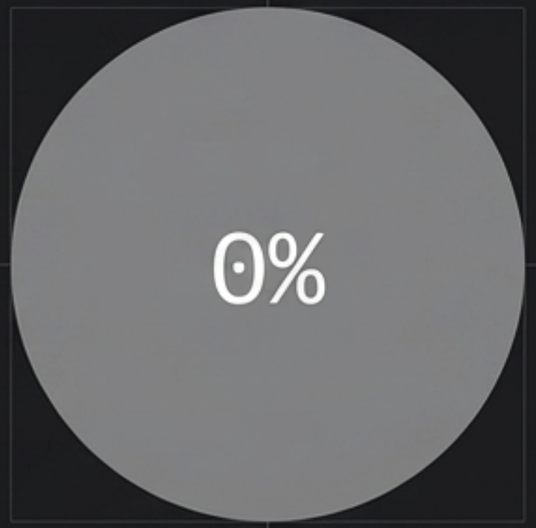
The undisputed culprit.

[BACKGROUND TASKS] DATA.STREAN: MAPPED

Heartbeat

Suspiciously high per-ping cost for a simple check-in, but mathematically not the primary driver of the blowout.

[PROACTIVE TASKS] DATA.STREAM: ISOLATED



Cron Jobs

Cleared. Configured with sessionTarget: 'isolated', they run entirely separate from the main conversation.

Dead Code: The Silent Context Leak

The framework's `cache-ttl` context pruning mode was enabled in the configuration. However, the logic was completely dead for this specific deployment.

The Config	The Source
`contextPruning: 'cache-ttl'`	`isCacheTtlEligibleProvider()` ← (Returns true ONLY for Anthropic)

The Impact: The agent runs Gemini Pro. Pruning was silently bypassed. No crashes, no errors—just money quietly leaking.

The False Suspect: Tool History Was Already Stripped

The transcript showed 72 thinking blocks and 257 tool calls. This seemed like the obvious source of context bloat—until the configuration revealed they were never sent to the API.

[LOG]: 72 thinking blocks + 257 tool calls exist in `.jsonl` transcript for debugging.

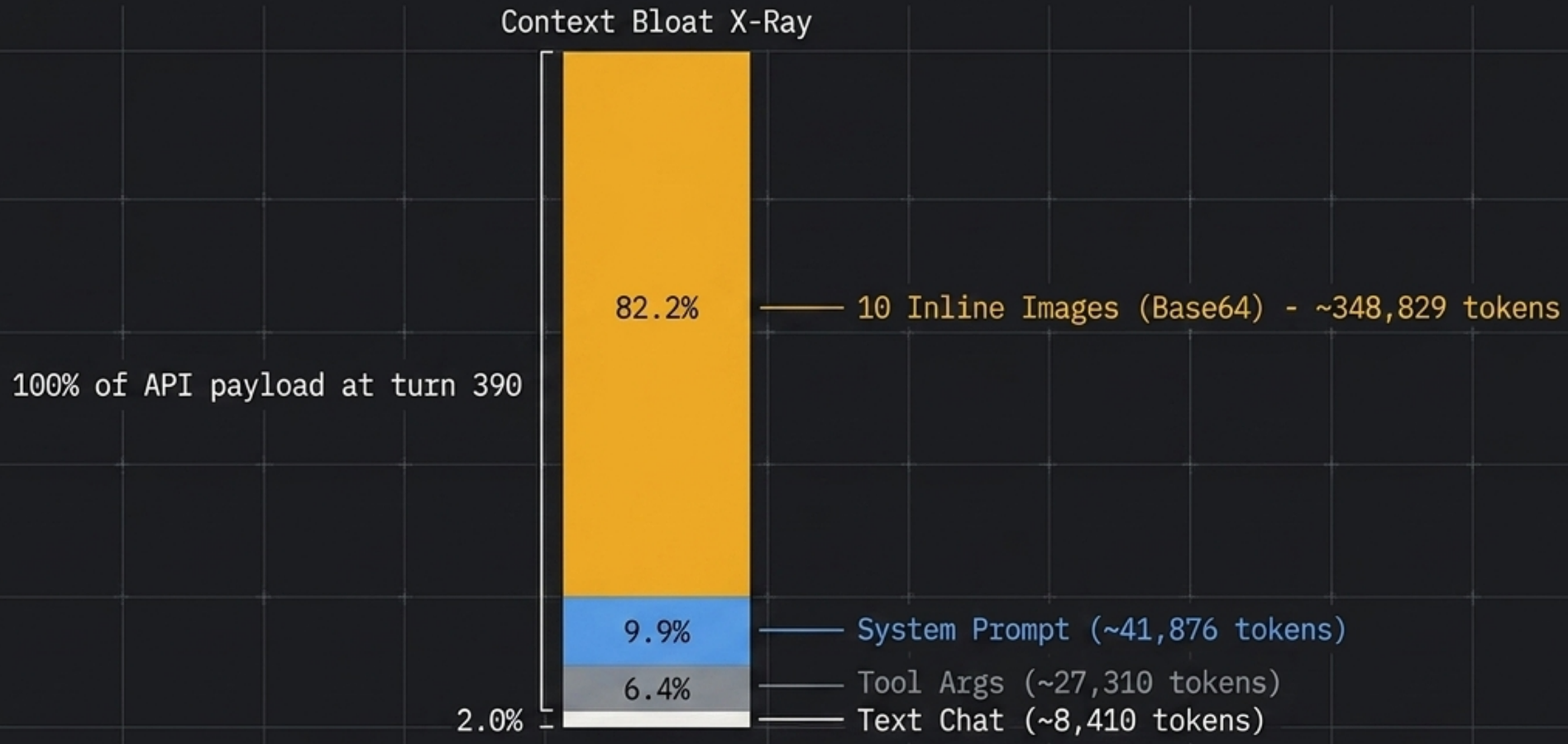
dmStripToolHistory: true

[API]: Stripped prior to LLM transmission.

[RESULT]: The true cost driver was still hiding in the payload.

The 82% Payload Dominance

Images were never evicted. Over 2.5 days, 10 base64-encoded images accumulated in the session.
Every single API call resent all 10 of them.



Roughly 350,000 tokens per turn were spent just keeping old images in memory.
The actual conversation was statistically invisible.

The 25x Tool-Use Multiplier

One session ballooned from 30 user messages into 750 assistant turns.
The framework encourages tool use but has no mechanism to control cost amplification.

You think you are chatting with an AI companion. In reality, the AI is chatting with its own tools, and you are paying to upload the massive context window on every internal loop.

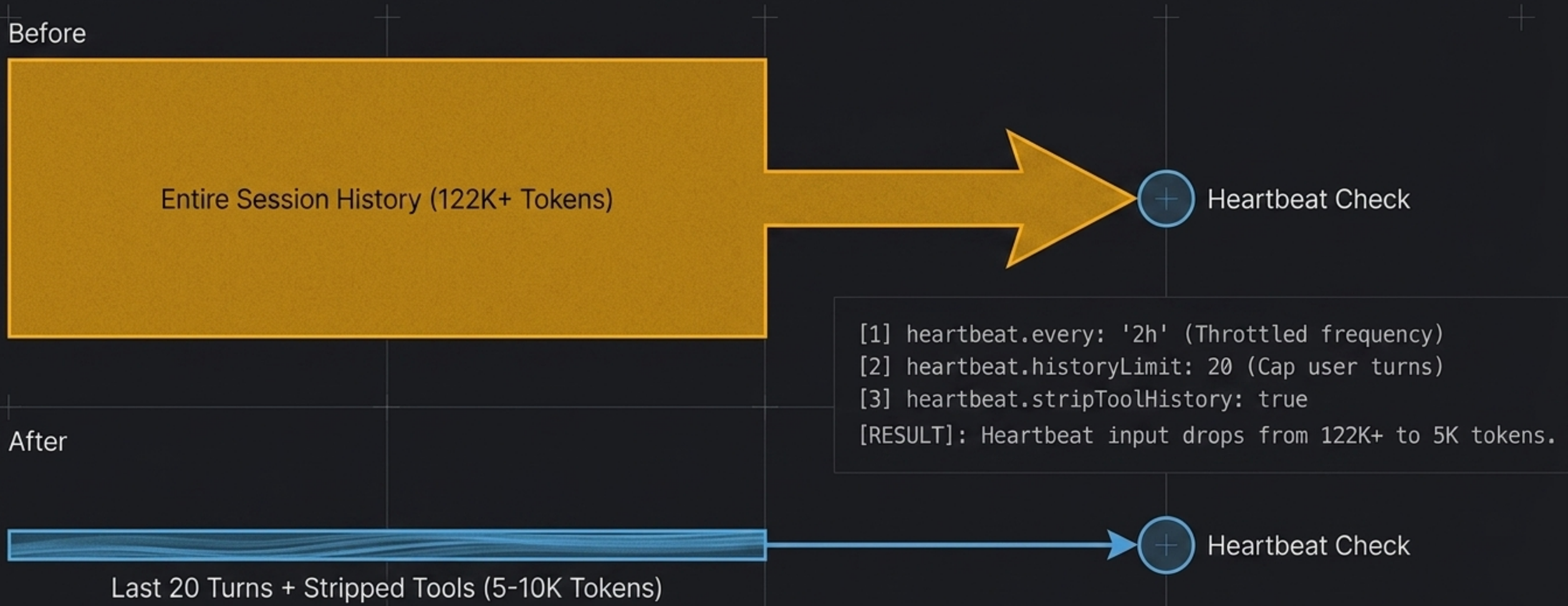
The Safeguard Trap: When 1M Context is a Footgun

In 537 turns over 2.5 days, the session automatically compacted zero times. The model's massive capacity actively prevented the safety net from firing.

With `compaction.mode: 'safeguard'`, cleanup only triggers when approaching the hard limit. At the current growth rate, it would take another week to hit the 500K trigger, ensuring an astronomical bill.

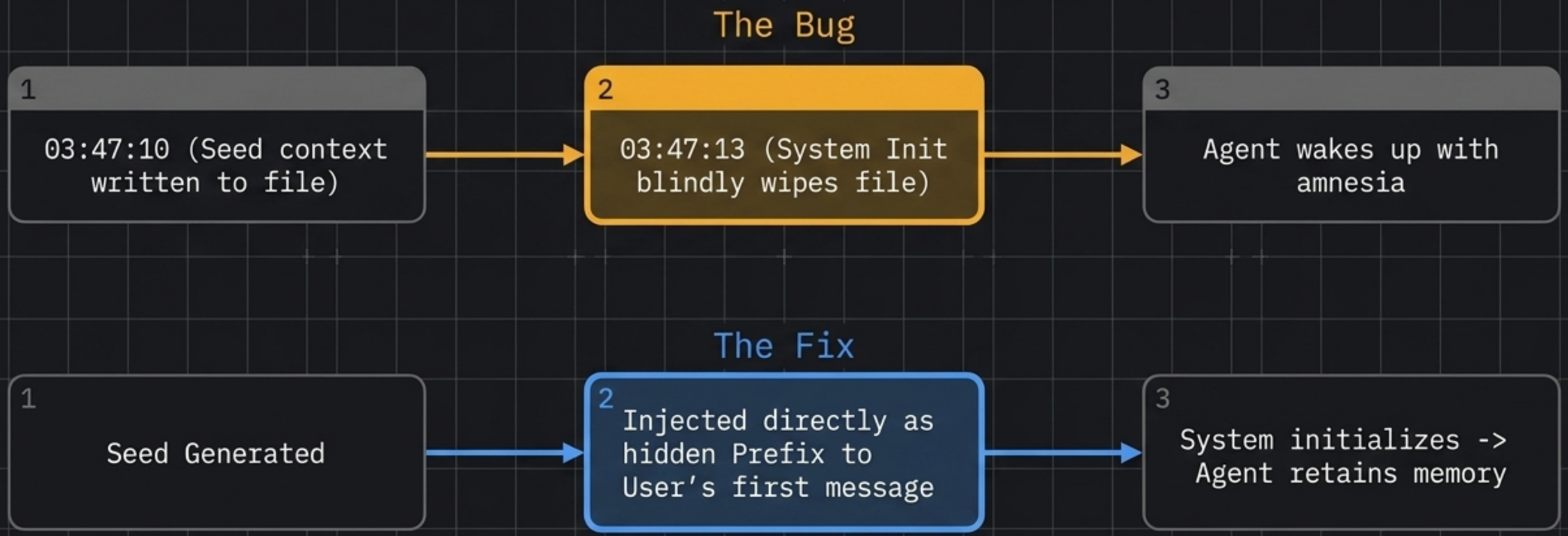
Quarantining the Heartbeat Context

Heartbeats are simple checks to decide if the agent should say “good morning.”
They do not need the main session’s unbounded context.



The Three-Second Amnesia Bug

Implementing a daily session reset surfaced a silent framework bug. The agent's memory seed existed for exactly **three seconds** before being overwritten.



By injecting the daily seed directly into the message persistence flow rather than relying on a fragile `.jsonl` overwrite, the companion successfully retains previous context.

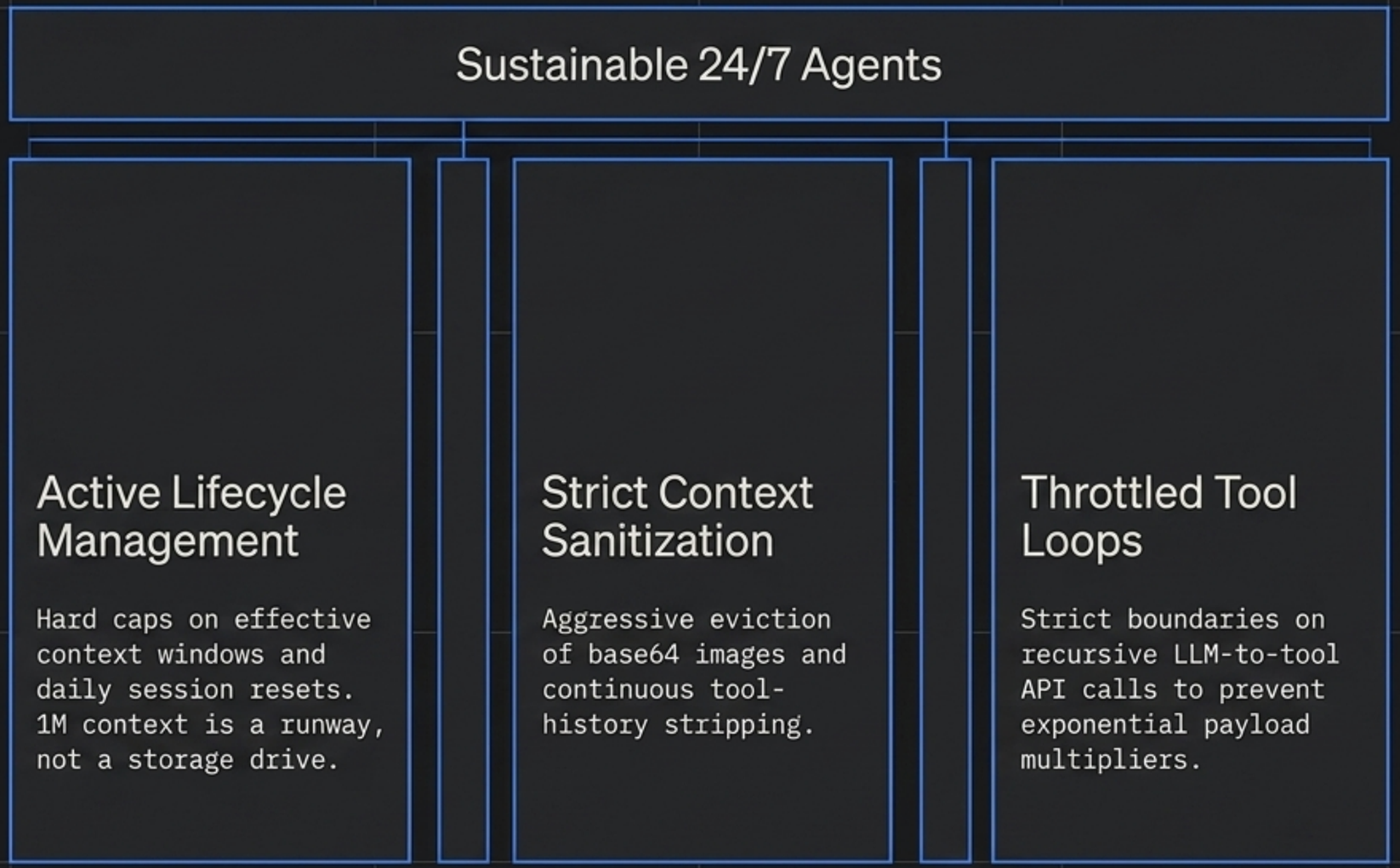
System Architecture Diagnostic

Rebuilding the configuration for sustainable 24/7 operations.

Component	Framework Default (Before)	Optimized Architecture (After)
[Pruning Provider]	`cache-ttl` (Anthropic Only)	‘always’ (Gemini Enabled + Image eviction)
[Session Lifecycle]	‘idle’ (3-day timeout)	‘daily’ (Hard reset at 4am PT)
[Context Cap]	Unset (Inherits 1M limit)	200000` (Compaction triggers at 100K)
[Thinking]	‘high’	`low’` (Reduces output tokens per turn)

The Hidden Architecture of Long-Running Agents

Token economy is a first-class architectural concern, not just an operational afterthought. You cannot rely on default framework configurations for continuous uptime.



AI Debugging AI: The Feedback Loop

The same patterns that caused the blowout—parallel tool-use loops and iterative scripting—were required to solve it.

```
[ALERT] Turn 537 execution cost threshold exceeded. Monotonically
increasing payload detected.
```

```
> Cost debugging is forensic work.
> You cannot guess at token totals based on uptime.
> Dead code leaks money silently.
> Investigation closed. New optimizations deployed in < 10 mins.
```