OPEN SOURCE SUMMIT · NORTH AMERICA 2026

Headroom.

The context compression layer
for AI agents.

SPEAKER

Tejas Chopra

github.com/chopratejas/headroom

HELLO

I'm Tejas. I build infrastructure for AI agents.

Headroom is Apache 2.0, runs locally, and has saved >200 billion tokens for the community so far.

github.com/chopratejas/headroom

headroomlabs.ai/dashboard

discord.gg/yRmaUNpsPJ

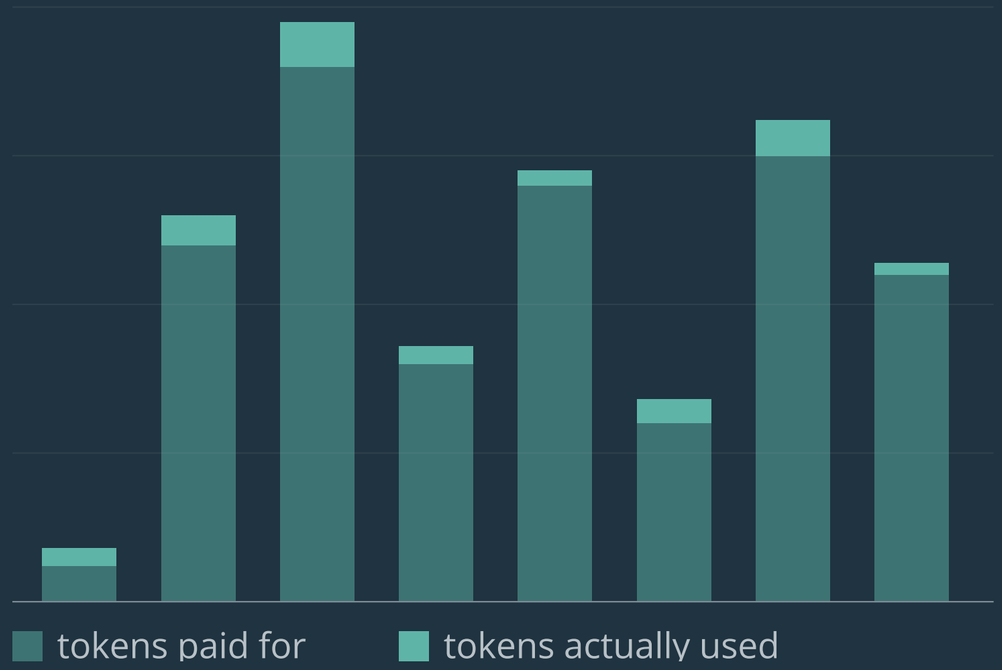
01

The token tax.

A coding agent's day, in tokens.

A NORMAL AGENT DAY

The agent reads 12,000 tokens to use three fields.



90%+

of an agent's token budget is noise.

EXISTING OPTIONS EACH COVER ONE SLICE

We needed all of it.

TOOL	COVERS	WHERE IT RUNS	LOCAL	REVERSIBLE
OpenAI compaction	Conversation history	Provider-side	—	—
RTK	Shell command output	CLI wrapper	●	—
lean-ctx	CLI · MCP · editor rules	CLI wrapper · MCP	●	—
Compresr · Token Co.	Text sent to their API	Hosted API	—	—
Headroom	All context — tools · RAG · logs · files · history	Proxy · library · middleware · MCP	●	●

02

What Headroom is.

IN ONE SENTENCE

A local compression layer between your agent and the LLM, shrinking everything in between.

20–40%
FEWER TOKENS

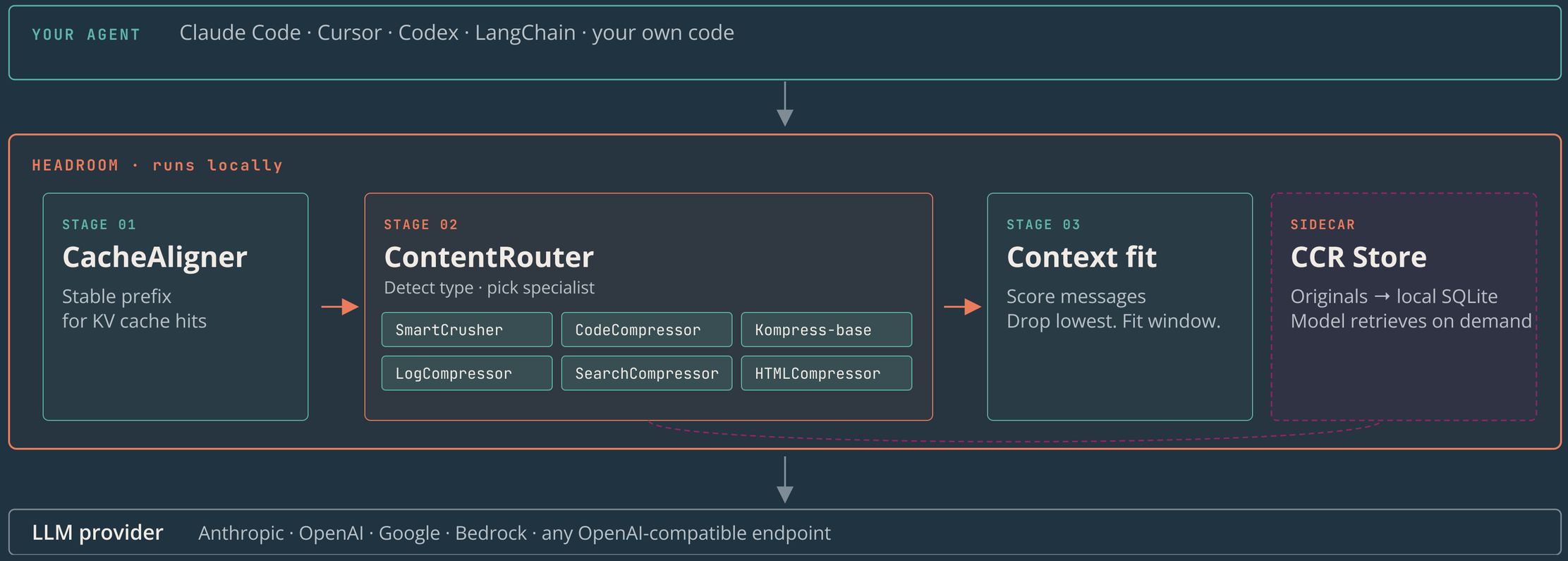
4
DEPLOY MODES

6
COMPRESSORS

0
DATA LEAVES THE BOX

ARCHITECTURE

One local box. Three stages. Originals never deleted.



FOUR WAYS IN

Same pipeline. Pick the least change.

01 · LIBRARY

Inline call.

```
from headroom import compress
r = compress(messages)
```

02 · PROXY

Drop-in URL.

```
$ headroom proxy
  --port 8787
```

03 · AGENT WRAP

One command.

```
$ headroom wrap claude
$ headroom wrap codex
```

04 · MCP SERVER

Three tools.

```
$ headroom mcp install
→ compress / retrieve / stats
```

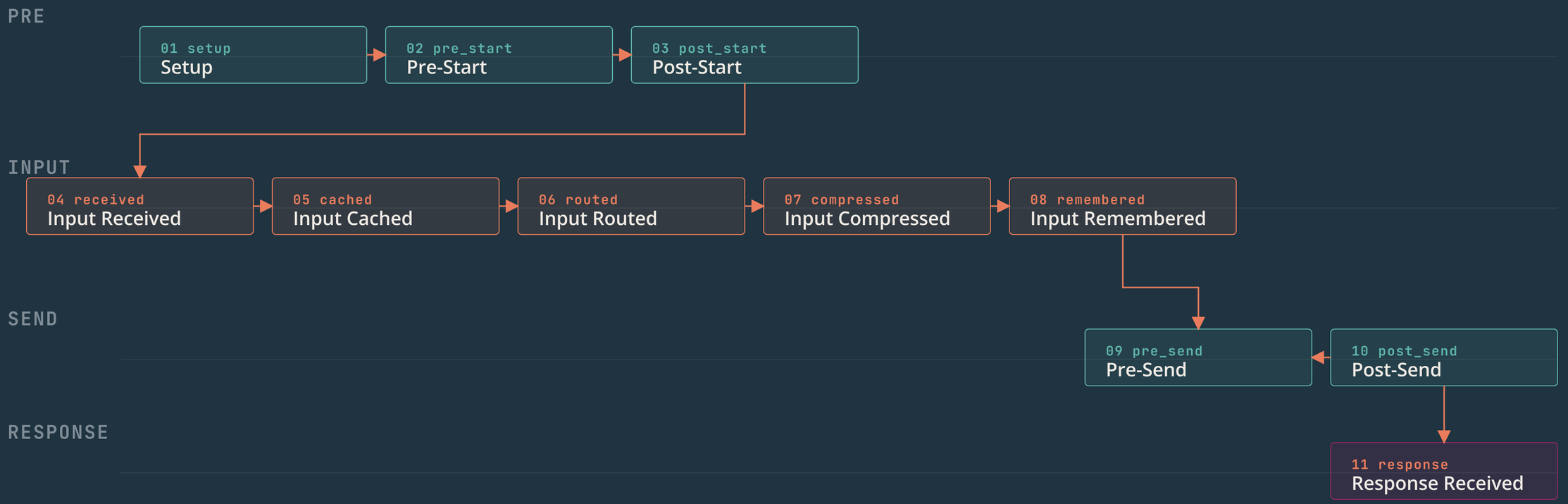
03

Under the hood.

One slide per moving part.

REQUEST LIFECYCLE

Eleven hooks. Every entry point uses them.



Detect what it is. Pick the specialist.

DETECTED AS	SPECIALIST	TYPICAL SAVINGS
JSON array of dicts	`SmartCrusher`	83–95%
Source code	`CodeCompressor`	40–70%
Search results	`SearchCompressor`	80–95%
Build / test logs	`LogCompressor`	85–95%
Unified diff	`DiffCompressor`	60–80%
HTML	`HTMLCompressor`	50–95%
Plain text · fallback	`Kompress-base`	60–80%

Score every item on five axes.

INPUT · 1,000 items

01	First & last N	100%
02	Errors	100%
03	Anomalies > 2σ	100%
04	Query match · BM25	top-K
05	Change points	all

1,000 → 50

items kept · 90% reduction

retention split — 30% start · 15% end · 55% by score

CODECOMPRESSOR · TREE-SITTER AST

Signatures stay. Bodies fold. Output always parses.

BEFORE · 240 TOKENS

```python
import os
from typing import List, Dict

@cached(ttl=60)
def process_items(items: List[str])
        -> Dict[str, int]:
    """Count occurrences."""
    result = {}
    for item in items:
        item = item.strip().lower()
        if item in result:
            result[item] += 1
        else:
            result[item] = 1
    return result
```

AFTER · 96 TOKENS · 60% OFF

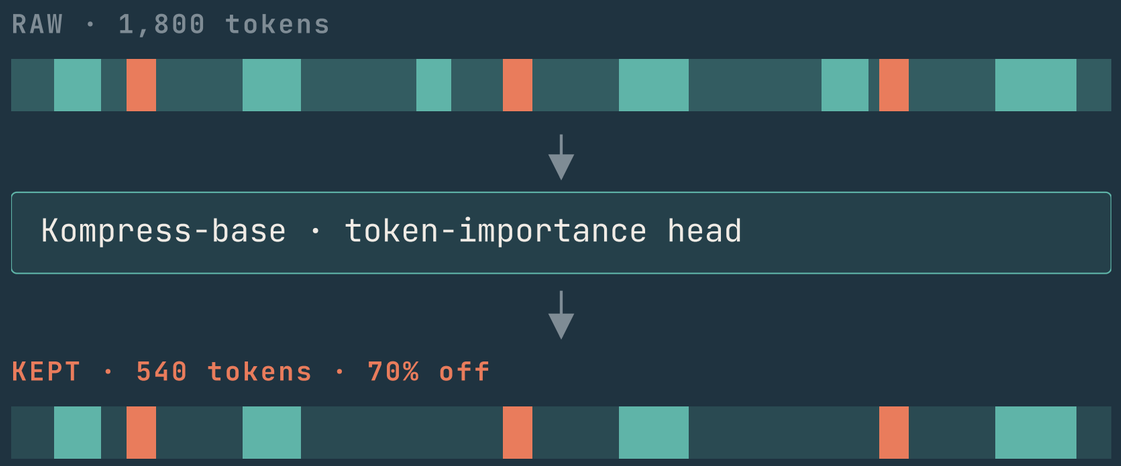
```python
import os
from typing import List, Dict

@cached(ttl=60)
def process_items(items: List[str])
        -> Dict[str, int]:
    """Count occurrences."""
    # ... body folded
    pass
```

Tier 1 · Python · JS · TS — full AST

Tier 2 · Go · Rust · Java · C · C++ — bodies

KOMPRESS-BASE · THE MODEL

Trained on agentic traces, not Wikipedia.



SPECS

Family	encoder · importance head
Params	~110 M
License	Apache 2.0
Hosted	hf · chopratejas/kompress-base

Stable prefix in. 50–90% off on cached reads.

BEFORE · CACHE MISS DAILY

```
"You are a helpful assistant.  Date: 2026-05-12"
```

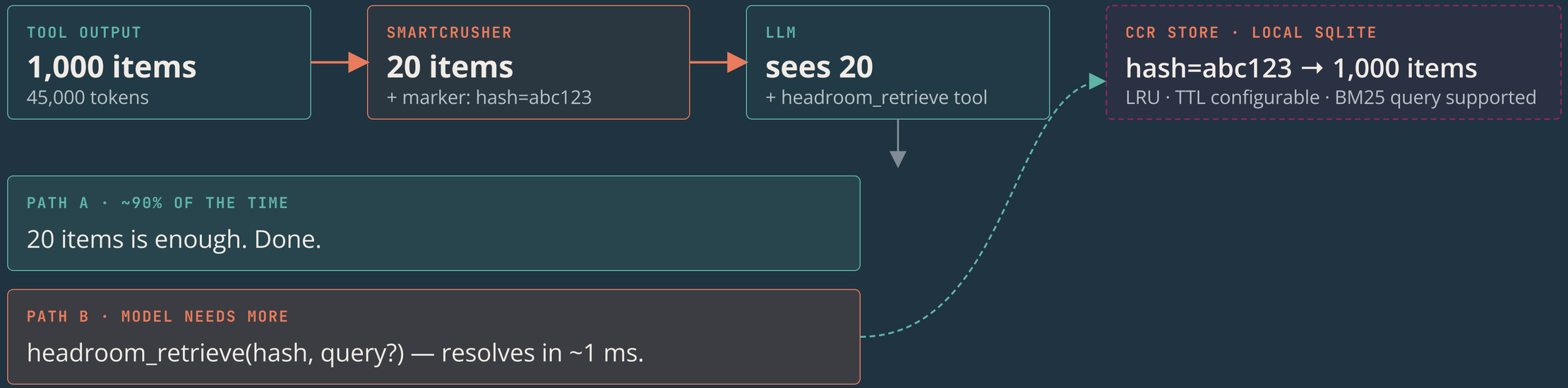
AFTER · PREFIX BYTE-STABLE

```
"You are a helpful assistant."
[Context: Date: 2026-05-12]
```

PROVIDER	MECHANISM	DISCOUNT
Anthropic	`cache_control`	**-90%**
OpenAI	auto prefix	**-50%**
Google	`CachedContent`	**-75%**

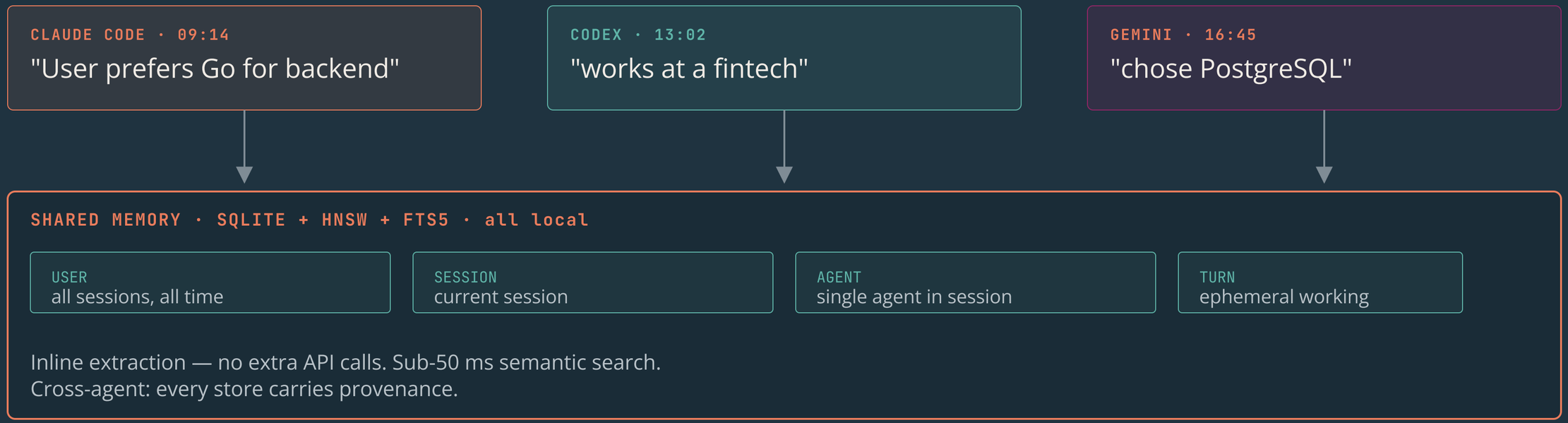
CCR · COMPRESS · CACHE · RETRIEVE

Lossy savings, lossless data.

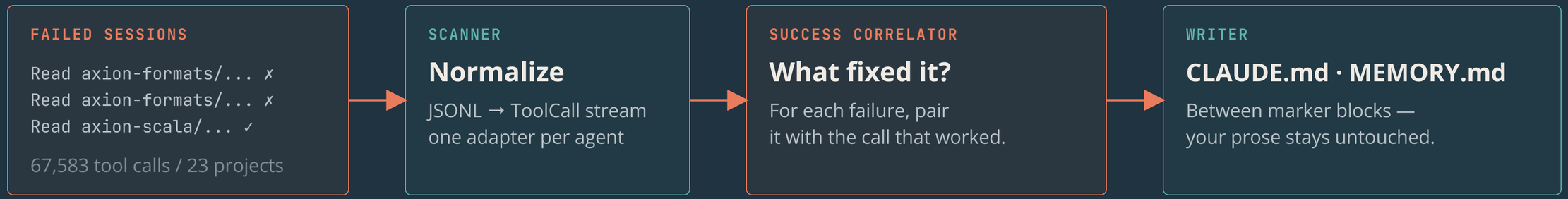


PERSISTENT MEMORY · CROSS-AGENT

10,000 tokens of history → 100 tokens of facts.



Pair every failure with what fixed it.



EXAMPLE CORRECTION

```
- FirstClassEntity lives at axion-scala-common/, not axion-formats/
```

04

Demo.

Switching to the terminal.

LIVE · ~3 MIN

Wrap an agent. Watch the token graph drop.

~/projects/axion · zsh

```
$ headroom wrap claude --memory

claude
› refactor every call to ProcessItems and stub the error path

⏎ tool grep "ProcessItems" — 142 hits
⏎ tool read 17 files · 84k tokens raw

▸ headroom: SmartCrusher · CodeCompressor · CacheAligner
▸ 84,231 → 9,418 tokens · 88.8% off

$ headroom stats --since session
tokens saved: 74,813 · cost saved: $0.41

▮
```

05

Numbers.

TOKEN SAVINGS

Four real agent workloads. Same model. Same answers.

WORKLOAD	BEFORE	AFTER	SAVED
Code search · 100 results	17,765	1,408	92%
SRE incident debugging	65,694	5,118	92%
GitHub issue triage	54,174	14,761	73%
Codebase exploration	78,502	41,254	47%

ACCURACY PRESERVED

Compression only matters if the model is still right.

GSM8K · MATH

0.870

baseline 0.870 · headroom 0.870

±0.000

TRUTHFULQA

0.560

baseline 0.530 · headroom 0.560

+0.030

SQUAD V2

97%

at 19% compression

n=100

BFCL · TOOL USE

97%

at 32% compression

n=100

WHAT'S NEXT

Three bets on where this is going.

01 · VERTICALS

Compressors per domain.

02 · MODALITIES

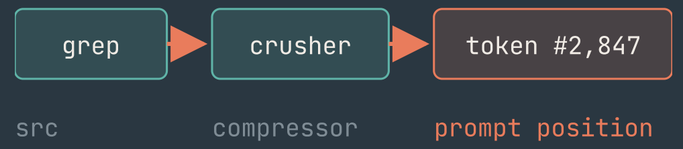
Voice agents · tool-calls.

Real-time voice agents call LLMs mid-conversation. Same token tax, much tighter latency budget. Streaming compression for audio transcripts and tool I/O.

03 · HEADLIGHT / SOON OPEN-SOURCE

Provenance for every token.

Where did this token come from? Which tool, which file, which retrieval? Headlight tracks the lineage end-to-end — debugging, auditability, attribution.



THANKS

Try it. Break it. Send PRs.

INSTALL

```
# Python
$ pip install "headroom-ai[all]"

# Node
$ npm install headroom-ai

# Docker
$ docker pull ghcr.io/chopratejas/headroom
```

REPO · APACHE 2.0

github.com/chopratejas/headroom

DOCS

headroom-docs.vercel.app

Tejas Chopra · @chopratejas

discord.gg/yRmaUNpsPJ