SRECON EUROPE MIDDLE EAST AFRICA



Performance Prompts

Brendan Gregg

OpenAI

Oct 2026



Source: https://www.wired.com/story/new-gas-powered-data-centers-could-emit-more-greenhouse-gases-than-entire-nations/



Source: https://www.theguardian.com/us-news/2026/jul/11/microsoft-amazon-google-datacentre-carbon-emissions-france

Performance engineering: The World needs YOU!



Source: https://news.bloomberglaw.com/privacy-and-data-security/data-center-gas-plants-to-boost-us-power-emissions-by-20

Takeaways

The world needs better performance engineering now

We've entered a new performance era: prompts

Performance Engineers are now speedrunners

AI agents sound smart, but need grading

Who am I

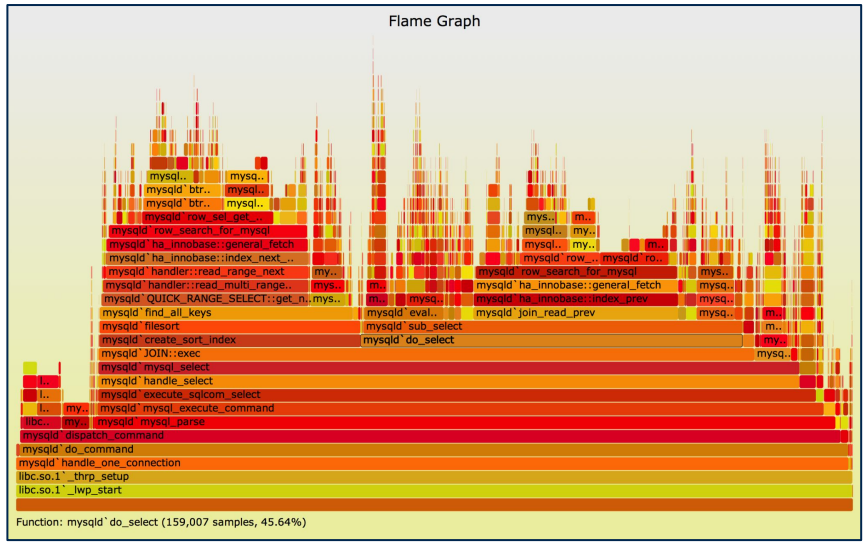
The only human Brendan Gregg

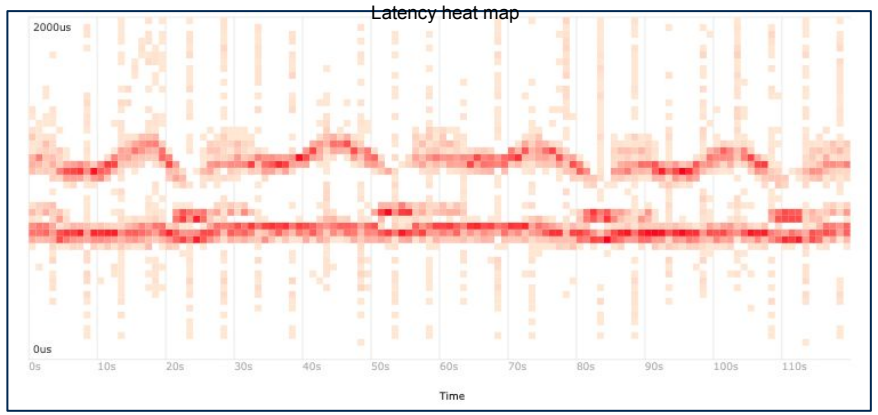
ChatGPT Performance Engineering team

- CPU efficiency wins, performance oncall, advanced perf technologies, etc.
- Management: Martin Spier → Justin Becker (Infra)

Work with Performance Foundations team

- Advanced performance tools: Continuous profiling (flame graphs), tracing, AI skills.
- Management: Louis Brandy → Venkat Venkataramani





A couple of my things you might be using

EXPLORE THE FUTURE OF SYSTEMS ADMINISTRATION AT THE 15TH SYSTEMS ADMINISTRATION CONFERENCE

LISA 2001

December 2–7, 2001
San Diego, California



INVITED SPEAKERS INCLUDE:

GREG BEAR, several times Hugo and Nebula winner, Best selling author of *Eon, Slant & Darwin's Radio*
"Slime vs. Silicon–Life's a Bitch, But Would You Want to Be a Computer?"

ERNEST PRABHAKAR, Apple
"Rebuilding the Digital Enterprise Around Internet Standards"

WILLIAM LEFEBVRE, CNN Internet Technologies - JUST ADDED!
"CNN.com: Facing a World Crisis"

PHIL COX, SystemExperts Corp.
"Hardening Windows 2000"

JOHN S. FLOWERS, nCircle Network Security
"Armoring the Next Wave of Security Technology"

LISA 2001 ALSO OFFERS OVER 50 TUTORIALS, HALF OF WHICH ARE NEW INCLUDING:

- Real-World Intrusion Detection
- Network Security Profiles
- Hacking-Exposed: LIVE!
- An Introduction to Computer Security NEW!
- Practical Wireless IP Security and Connectivity
- Creating a Computer Security Incident Response Team NEW!
- UNIX Security Threats and Solutions NEW!
- Forensic Computing

Attendees receive a free copy of *Selected Papers in System Administration*, to be published by Wiley in December and edited by Eric Anderson, Mark Burgess, and Alva Couch

http://www.usenix.org/events/lisa2001

Receive an additional discount when you register online!

Sponsored by
USENIX The Advanced Computing Systems Association
SAGE The System Administrators Guild

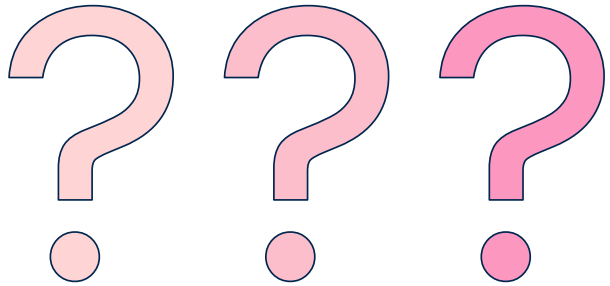
This Talk

Personal opinions not necessarily those of my employer

Discusses prompts used in OpenAI's Codex tool, primarily for CPU analysis

Written in Sep 2026 – Will likely be different by Sep 2027

More questions than answers



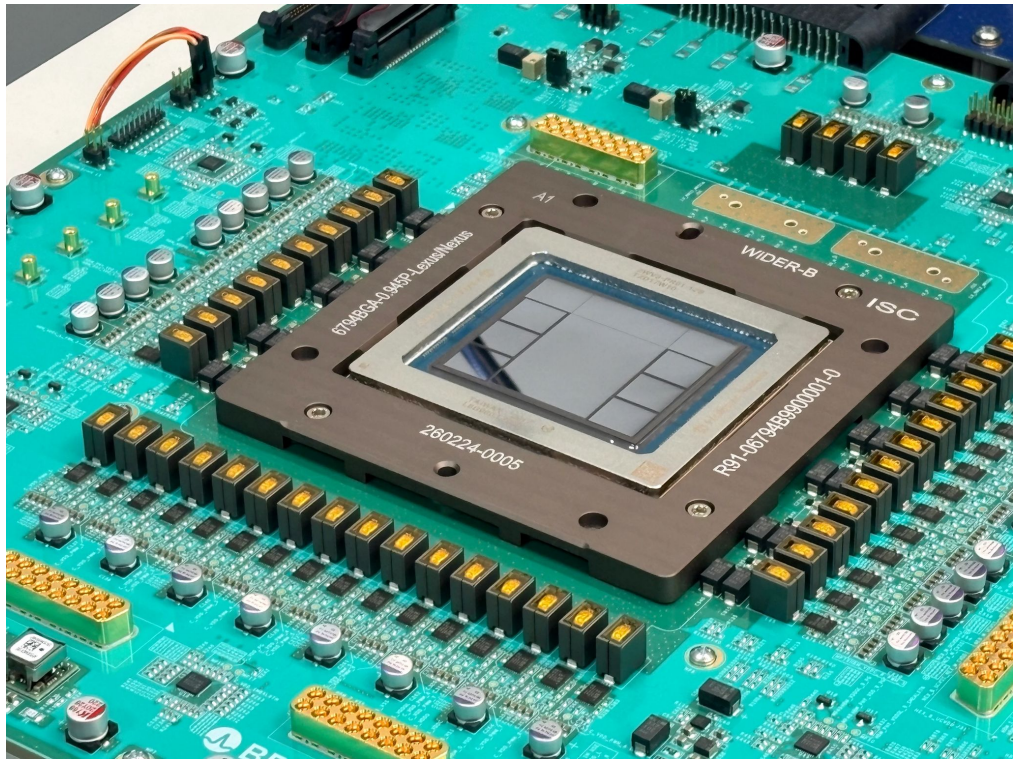
OpenAI Performance

Software



GPT-6.1 Sol: Huge cost reduction

Hardware



Jalapeño: First custom inference chip

Sources: https://openai.com/index/introducing-gpt-6-1-sol/ https://openai.com/index/jalapeno-first-results/

ChatGPT

Cloud: Azure (primary)

Deployment: Kubernetes

OS: Ubuntu Linux

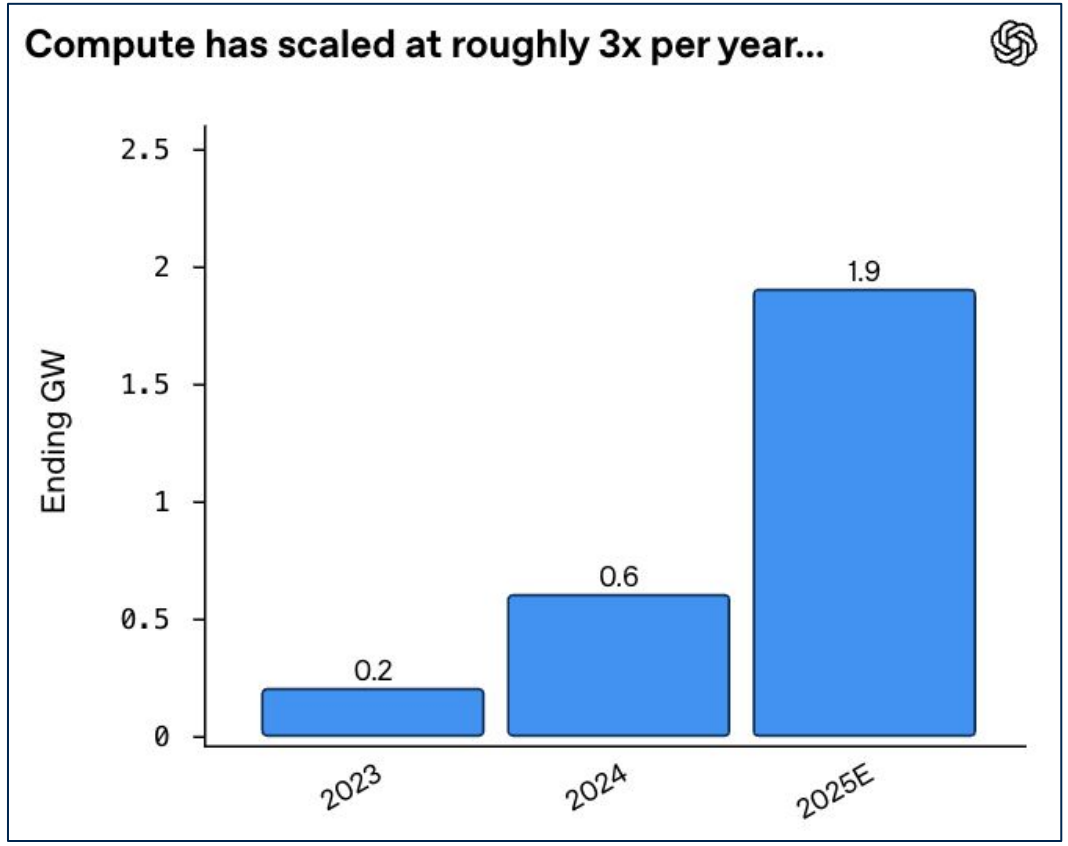
Language: Python (primary)

Core Tech: Terraform, Buildkite, Postgres, Cosmos DB, Kafka

eBPF: >1300 programs (incl. Cilium)

Scale: **>1B** weekly users

Massive GPU **and CPU** scale



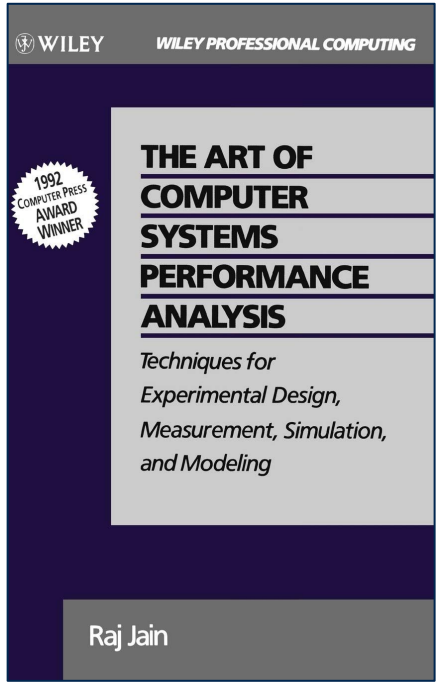
Sources: https://openai.com/index/next-phase-of-microsoft-partnership/ https://openai.com/index/scaling-postgresql/ https://openai.com/careers/software-engineer-developer-productivity-san-francisco/ https://openai.com/index/infrastructure-for-deep-learning/ (2016) https://openai.com/index/next-phase-of-enterprise-ai/ https://openai.com/index/a-business-that-scales-with-the-value-of-intelligence/ https://openai.com/index/expanding-access-to-ai-with-chatgpt-ads/ https://isovalent.com/blog/post/openai-isovalent-networking-kubernetes-case-study/

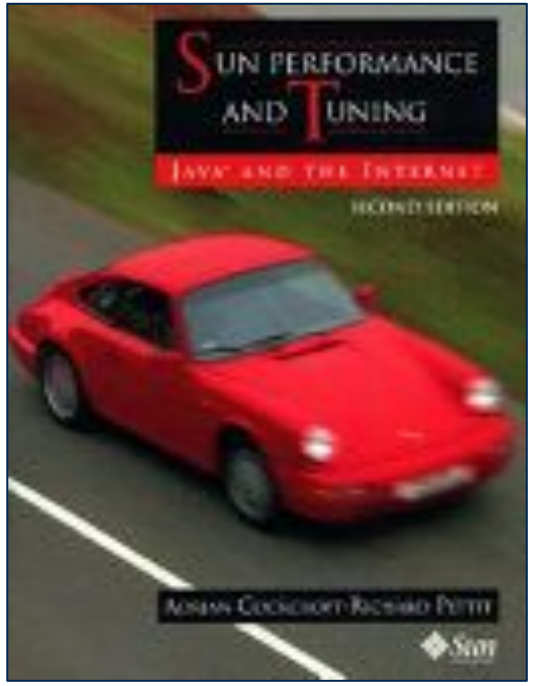
1. Moving to AI

Performance Eras

1960s-2000s: Performance Math

```
# iostat 1
    RF                   RK                   RP
    tpm   msps   mspt     tpm   msps   mspt     tpm   msps   mspt
   1057    0.0    0.0       0    0.0    0.0    1178   -0.6    0.7
      0    0.0    0.0       0    0.0    0.0       0    0.0    0.0
 106980    0.0    0.0       0    0.0    0.0130440   -0.6    0.6
  56580    0.0    0.0       0    0.0    0.0   67680   -0.6    0.6
  89374    0.0    0.0       0    0.0    0.0108209   -0.6    0.6
  60916    0.0    0.0       0    0.0    0.0   74179   -0.6    0.6
  15789    0.0    0.0       0    0.0    0.0   17937   -0.6    0.6
   1020    0.0    0.0       0    0.0    0.0    1020   -0.6    0.6
      0    0.0    0.0       0    0.0    0.0       0    0.0    0.0
      0    0.0    0.0       0    0.0    0.0       0    0.0    0.0
[...]
```

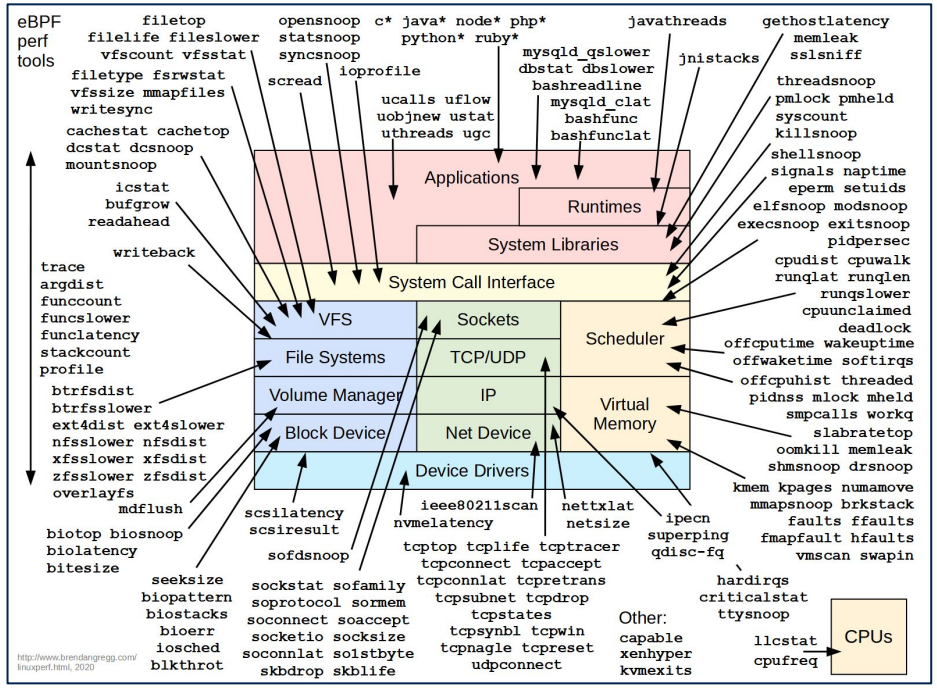


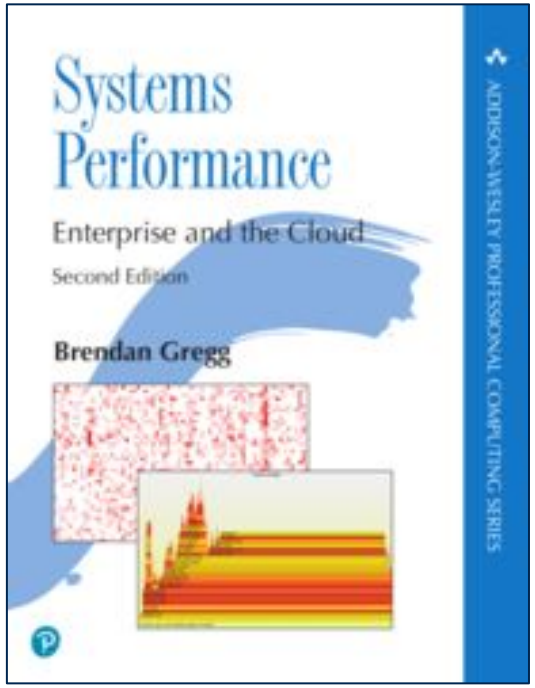


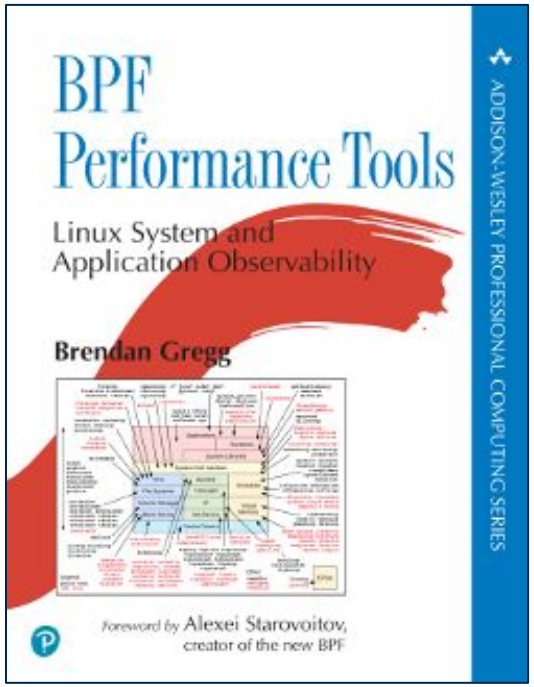
Art of inference from limited vendor metrics.
Many blind spots.

Performance Eras, cont.

2000s-2020s: Performance Tools

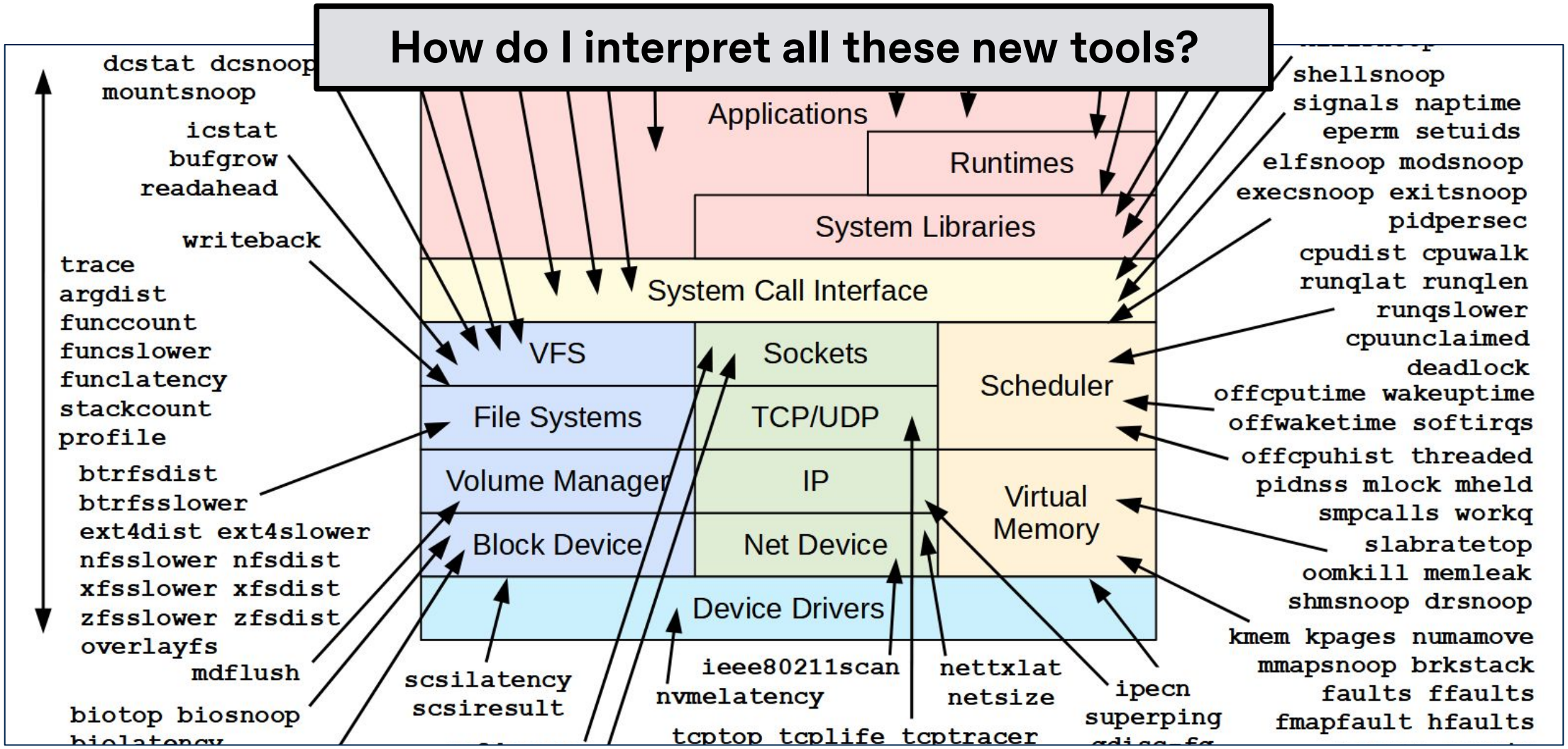






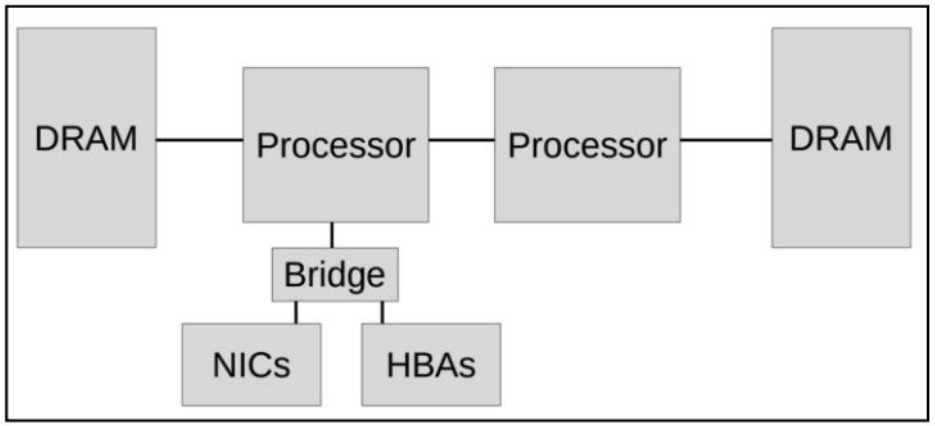
With the rise of open source, tracing, and dynamic instrumentation: measure anything.
But two growing problems...

Problem 1: Specialized Tools



Problem 2: Complexity Crisis

Every year: more stuff, same brains



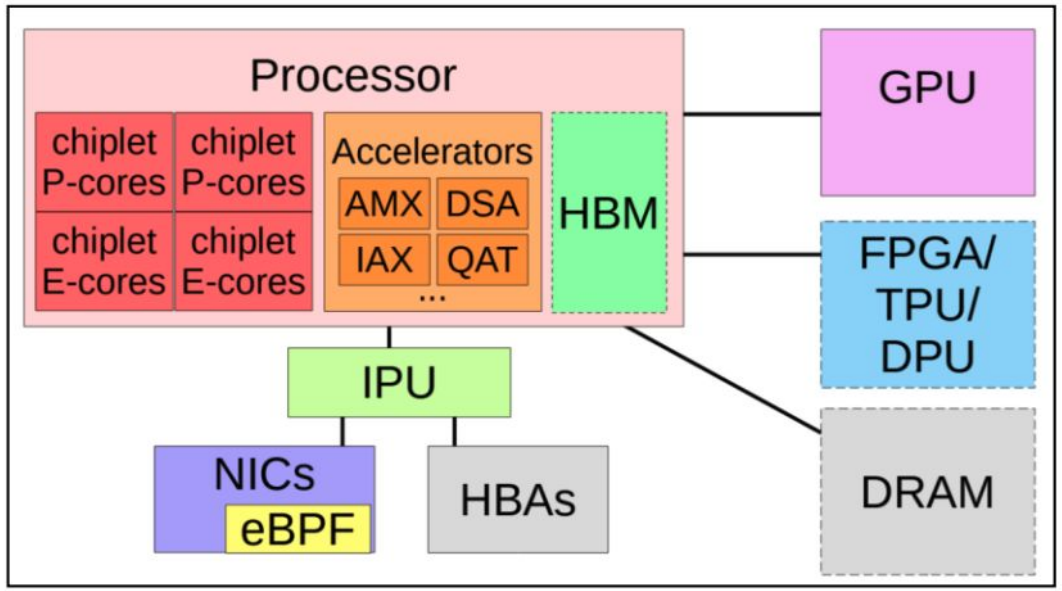
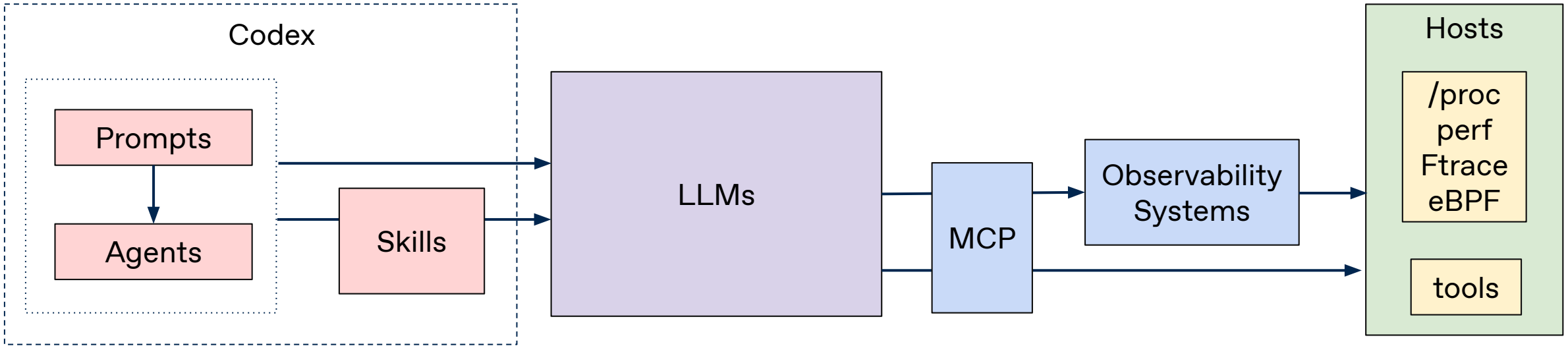


Image source: https://www.brendangregg.com/Slides/SREnext2025_FastByFriday.pdf

Performance Eras, cont.

2025-: Performance Prompts



Measurement, interpretation, and fixes mostly AI automated

Performance engineering fundamentals mostly similar

One UI for everything: [Codex]

It's been like switching to a new sport

Tools can now be prompts

"Measure memory bus utilization per second: use a pre-installed tool if one exists, or write your own and ask for a review before running."

```
05:31:10     53.34 GB/s     7.44%
05:31:11     40.85 GB/s     5.70%
05:31:12     28.18 GB/s     3.93%
```

Methodologies always were prompts

The USE Method: "For every resource, check utilization, saturation, and errors."

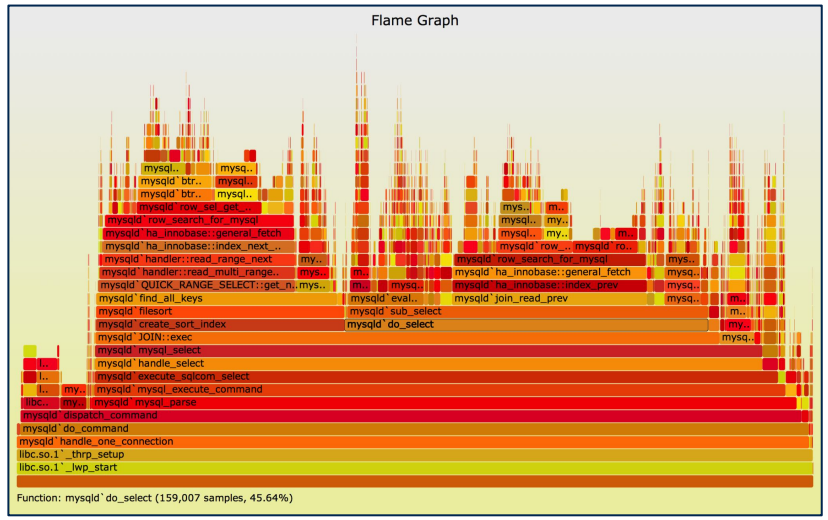
https://www.brendangregg.com/usemethod.html https://www.brendangregg.com/methodology.html

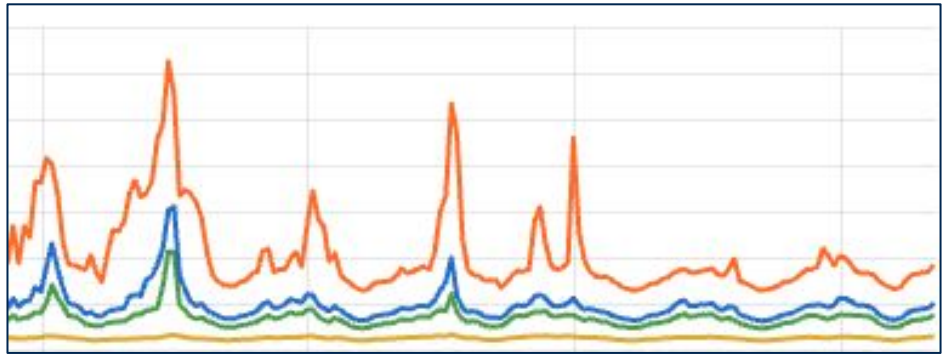
AI Performance Engineering Prerequisites

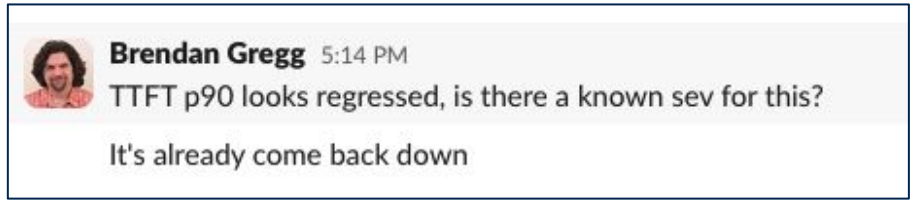
AI connected to all data sources (MCP): code, profiles, observability, chats, etc.

```
while (1):
    try:
        sleep(int(args.interval))
    except KeyboardInterrupt:
        exiting = 1

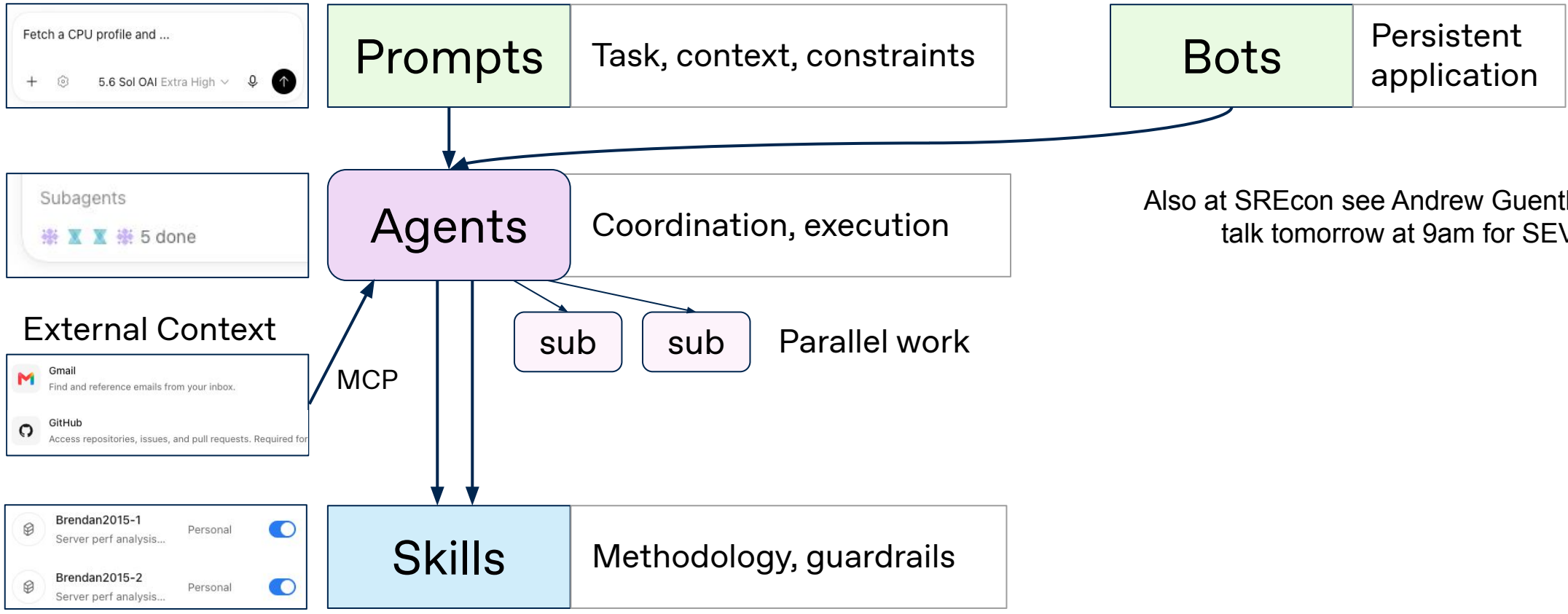
    print()
    if args.json:
        if args.timestamp:
            print("%-8s\n" % strftime("%H:%M:%S"), end="")
```







Prompts, Skills, Agents, Bots



OpenAI has many skills for performance & SRE. With cloud instances, agents can run 24x7.

Prompt Sizes

Level	Prompt Size	Contains	Good for	Shared
1	**Sentence**	Single task	Learning AI	Many public examples
2	**Paragraph**	Input sources, analysis direction, actions, some context and constraints	Early work	Ad hoc internally
3	**Page**	High-level goal, common analysis directions, caveats, constraints, prior history, success criteria, best practices (do's, dont's), verification	Mature work	Engineer's private tool
4	**Multiple pages**	Detailed description of performance engineering behavior	Skills	Company wide

The following examples are mostly 1 & 2 to fit on a slide

2. Prompts

#1 Short Examples

"**Optimize this loop** for memory efficiency and explain why your version is faster."

"**Find expensive operations** repeated within this request handler and identify caching opportunities."

"**Explain why [metric] changed** during [time window], break the movement down by relevant dimensions, and separate confirmed causes from unverified hypotheses."

"Establish a baseline, **make one focused optimization**, rerun the evaluation after each change, record the results, and continue until the target is met."

"**Draft a monitoring dashboard** specification with metric definitions, chart requirements, data-quality checks, ownership, alerting strategy, and publication risks."

"Profile one user-visible iOS application flow, ensure application symbols are available, preserve the flame-graph JSON, and **report the highest-cost application-owned stacks.**"

Sources: https://openai.com/business/guides-and-resources/how-openai-uses-codex/ https://learn.chatgpt.com/use-cases/kpi-root-cause-analysis
https://learn.chatgpt.com/use-cases/iterate-on-difficult-problems https://learn.chatgpt.com/use-cases/dashboard-builder-monitor
https://github.com/openai/plugins/blob/main/plugins/build-ios-apps/skills/ios-ettrace-performance/SKILL.md

#2: Optimization

“Collect a CPU profile for <app> from <continuous profiler> for the last hour and find 10 areas that can be optimized. Don’t write code yet, produce an enumerated table of optimizations and the expected speed up based on the CPU profile, sorted by speed up, and include a column for trade-offs/downsides. Search <internally> and don't include proposals if someone else is working on them or recently pushed a fix. Use subagents.”

#2: Optimization

"Collect a CPU profile for <app> from <continuous profiler> for the last hour and find 10 areas that can be optimized. Don't write code yet, produce an enumerated table of optimizations and the expected speed up based on the CPU profile, sorted by speed up, and include a column for trade-offs/downsides. Search <internally> and don't include proposals if someone else is working on them or recently pushed a fix. Use subagents."

#	Area	Possible optimization	Expected speedup	Trade-offs / downsides
1	<redacted>	Decode each stored block once, reuse trusted decoded payloads, and avoid repeated construction.	**0.30 pp CPU**	Bypassing validation risks malformed nodes, mutation aliasing, and subtle compatibility bugs.
2	<redacted>	Cache canonical serialized bytes and the resulting hash by immutable config or bundle version.	**0.25 pp CPU**	Cache invalidation must be exact; stale hashes could prevent config refreshes.
3	<redacted>	Coalesce timer wakeups and only call socket_action for sockets with due work.	**0.17 pp CPU**	Timeout and cancellation timing may become slightly less precise.

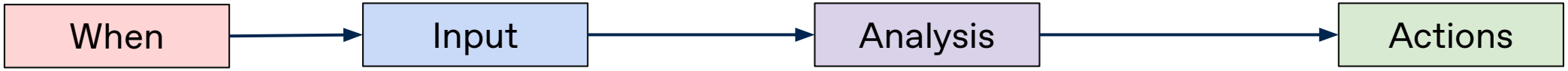
...

"Ok, go ahead with 2, 3, and 7. Use subagents, test on <my dev instance> and fix problems as they arise, then usual prod testing, and if ok create PRs. Finish with a list of PR links."

Codex turns profiles into PRs

Optimization Prompt Anatomy

"(now) Take this flame graph, look for performance issues, and suggest PRs to fix them"



Now
On alerts
Monitoring
24×7 analysis

Monitoring
Traces
/proc, sysstat
perf (incl. PMCs)
Ftrace
eBPF tools
Flame graphs
Obscure stuff
etc.

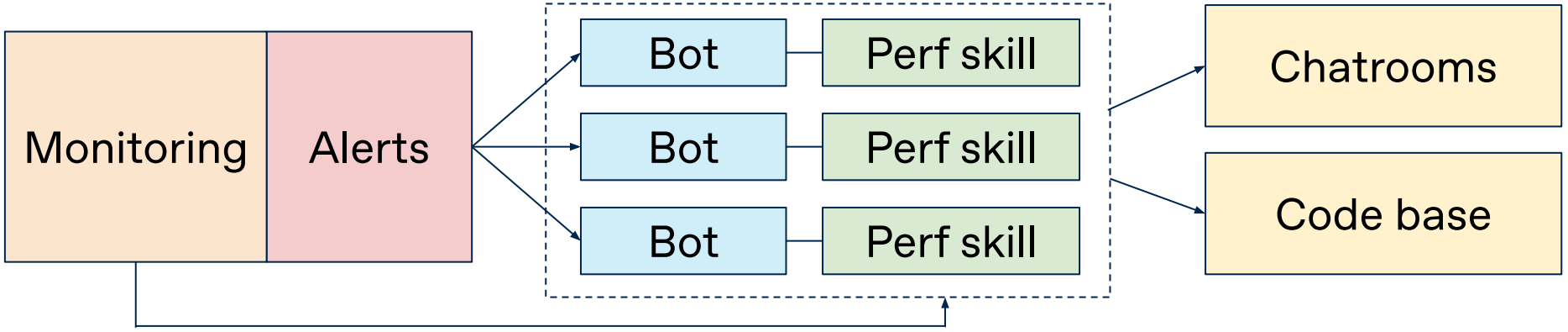
Find perf issues
Methodologies:
- Time correlations
- The USE Method
- Latency drilldowns
- 5 Whys
- etc.

Develop code fixes (PRs)
Make tunable changes
Propose design changes
Alert

#3 Regression Analysis

"Drill down to the root causes of the rising p90 latency shown here <dashboard link>. Suggest a fix. Check <internally> if other teams are working on it, and if there's already a SEV. If not, and it meets SEV criteria, tell me the SEV level and summary description."

This is heavy automated at OpenAI. Monitoring alerts trigger multiple bots to begin analysis and post summaries with suggested fixes to alert chatrooms. These bots use multiple perf analysis skills.



This is awesome work by Cyrus Forbes, Ben Romano, Okay Zed, etc.

All AI bots are oncall 24×7

#4 Interpretation

“Summarize what's interesting and changing in /proc/stat, and include recommended actions for a performance engineer, as though you are a pirate.”

#4 Interpretation

"Summarize what's interesting and changing in /proc/stat, and include recommended actions for a performance engineer, as though you are a pirate."

Arrr, mark these clues well:

- NET_RX be roaring at ~338k/sec. Inspect NIC queues, RSS/RPS, IRQ affinity, and dropped packets.
- Some CPU decks toil at ~66%, while others loaf below 2%. Rebalance IRQs and make irqbalance earn its rum.
- Context switches be ~586k/sec. Hunt the thread-happy culprits with perf, pidstat, or scheduler traces.
- Runnable hands rose 40 → 59, yet none be blocked. Watch run-queue latency, but blame not the disks.
- Iowait and steal be near zero. Leave storage and the hypervisor in peace, unless fresh evidence sails in.

Verdict, ye scurvy machine: network receive work be the chief suspect, concentrated on a few CPUs. But the vessel is not overloaded—merely hauling a lively cargo.

Interpretation: PMCs

"Run "perf stat -d -a -- sleep 10" and summarize the output for dummies, and make recommended fixes, as though you are perf. Be brief. Include the perf output."

```
Performance counter stats for 'system wide':
[...]
           2856301      page-faults                 #    1.464 K/sec
     1715236966809      cycles                      #    0.879 GHz
     1832099565053      instructions                #    1.07  insn per cycle
      361069901722      branches                    #  185.027 M/sec
        8882889565      branch-misses               #    2.46% of all branches
     8453916333888      slots                       #    4.332 G/sec
     2246509798374      topdown-retiring            #     26.2% retiring
      967038463296      topdown-bad-spec            #     11.3% bad speculation
     2574144570873      topdown-fe-bound            #     30.0% frontend bound
     2790610199887      topdown-be-bound            #     32.5% backend bound
      149813225493      topdown-heavy-ops           #      1.7% heavy operations
      920229870630      topdown-br-mispredict       #     10.7% branch mispredict
     1769623284972      topdown-fetch-lat           #     20.6% fetch latency
     2131160301980      topdown-mem-bound           #     24.8% memory bound
      488128805371      L1-dcache-loads             #  250.138 M/sec
       21902578572      L1-dcache-load-misses       #    4.49% of all L1-dcache accesses
[...]
```

Interpretation: PMCs

"Run "perf stat -d -a -- sleep 10" and summarize the output for dummies, and make recommended fixes, as though you are perf. Be brief. Include the perf output."

I am perf stat. I watched the whole node for 10 seconds.

- I counted ~1.8T instructions and ~1.7T cycles: IPC was 1.07, meaning about one useful instruction per CPU tick. Mediocre for this Xeon.
- Only 26% of CPU slots did useful work.
- 30% waited for instructions, 32.5% waited for memory/backend resources, and 11.3% was wasted on bad branch guesses.
- Branch misses were 2.46%; L1 cache misses were 4.49%.

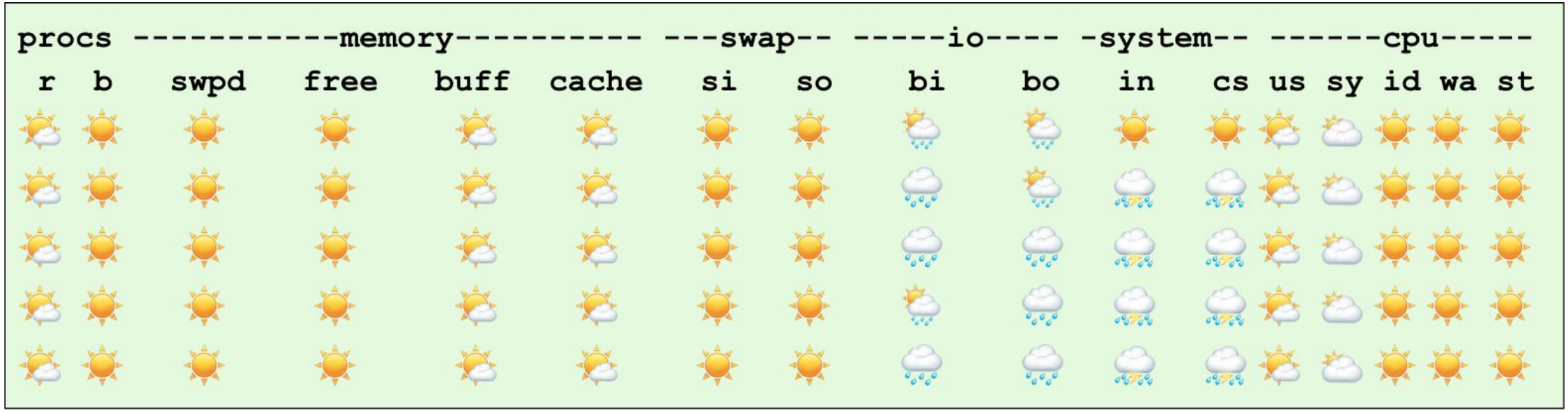
Recommended fixes:

- Measure a specific PID or cgroup next; system-wide data mixes everything.
- Profile branch misses and frontend stalls with perf record.
- Investigate memory locality if backend/memory stalls persist.

Tools can now talk

Interpretation: Metrics

“Run "vmstat 1 5" and turn the output fields into weather emoticons”



Footnote: I previously used weather icons (seriously) in a Sun Microsystems performance dashboard: https://www.brendangregg.com/blog/2008-11-10/status-dashboard.html

Interpretation: eBPF and Ftrace

"Run klockstat (eBPF tool in /usr/share/bcc/tools) for 10 seconds and then interpret the output: explain if, why, and how much findings matter. Also include the raw output."

- generic_fadvise is the most suspicious: ~1.16s cumulative wait, 1.13ms average, 92ms worst-case. Could hurt tail latency if request-path code calls posix_fadvise.
- Rerun with stacks/PID filtering for generic_fadvise, vfs_open, and __fput.

"Check that there is no Ftrace tracer active, and abort if there is. Enable timerlat for five seconds, and interpret the output: explain if, why, and how much the findings matter, and make recommendations to improve performance. Include the raw output (save it locally)."

- Worst event was CPU 98: IRQ arrived 6.60 ms late, and the thread ran only ~32 µs later. That suggests the delay happened before the IRQ handler
- Run osnoise next to classify IRQ, softirq, thread, NMI, or hardware noise.

Output trimmed to include one example observation and one actionable item

Interpretation: Config

"Summarize /proc/cpuinfo as though you are the CPU, and tell me how close to the latest offerings for this CPU vendor you are, in terms of years/months, and expected performance differences."

I am an Intel Xeon Platinum ...
Age and distance from latest:

- My 5th Gen Xeon family launched December 2023: **about 32 months old**.
- Xeon 6 P-core launched September 2024: one comparable generation newer, about 9 months after me.
- Xeon 6+ E-core arrived Q2 2026: about 2.5 years newer, but aimed at high-density cloud throughput.

Expected performance gap:

- Normal server workloads: **newer Xeon 6 P-core may be ~1.3–1.5x faster**.
- Compute-heavy AI/HPC/database work: up to ~2x, with Intel claiming up to 2.5x for selected HPC workloads.

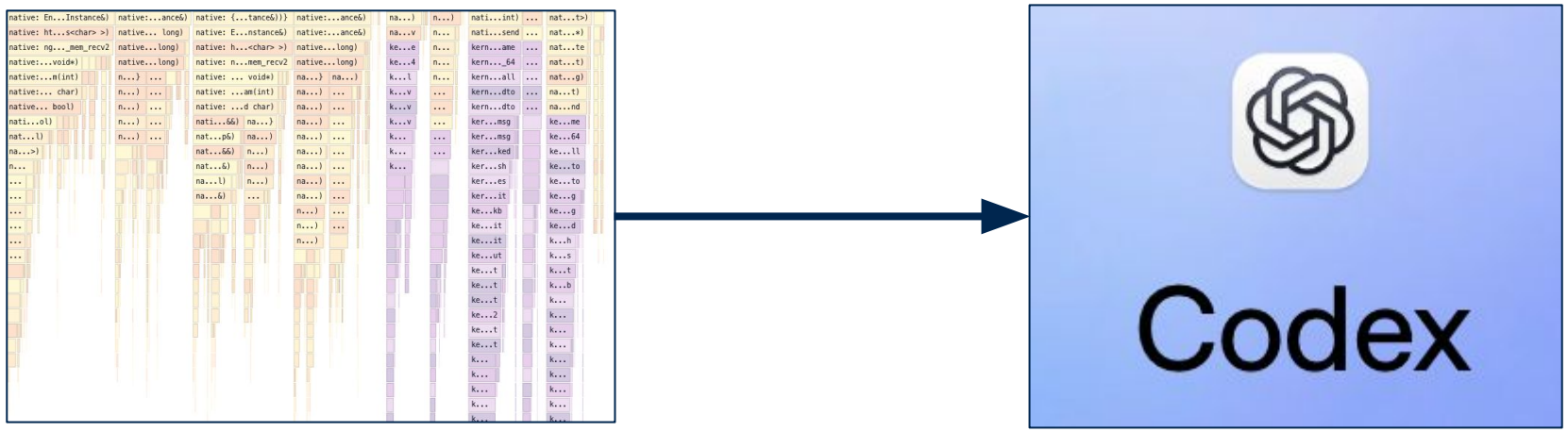
Verdict: I am still modern, but roughly one P-core generation and 9 months behind comparable Intel server CPUs; the newest density-focused Intel CPUs are about 2.5 years newer.

Chat with your CPUs

Interpretation: Flame Graphs

The “John Wahba” method:

1. Take a flame graph screenshot
2. Paste it into Codex
3. “Find performance wins”

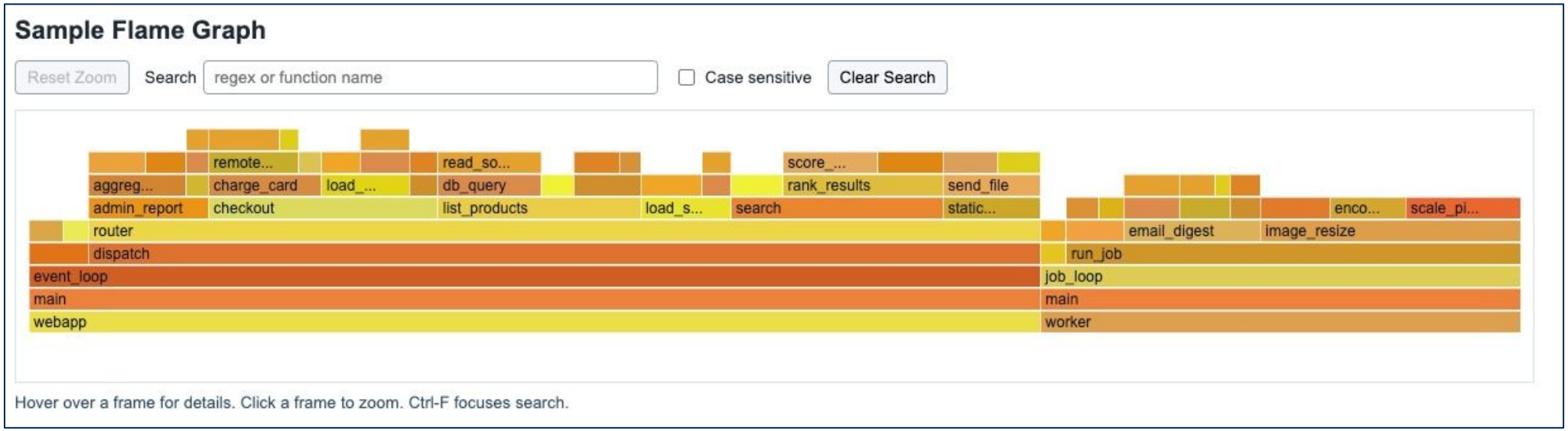


PNG may often be better than raw TXT: Token optimization

#5 User Interfaces

"Can you **generate an implementation of flame graphs** without reading any other flame graph implementation code, just based on the description of flame graphs in this 2016 ACMQ article: https://queue.acm.org/detail.cfm?id=2927301 . Also generate an example text file of sample stacks in the folded format for testing the flame graph. Write it in Python with JavaScript for interactivity. It should be able to execute locally on this computer."

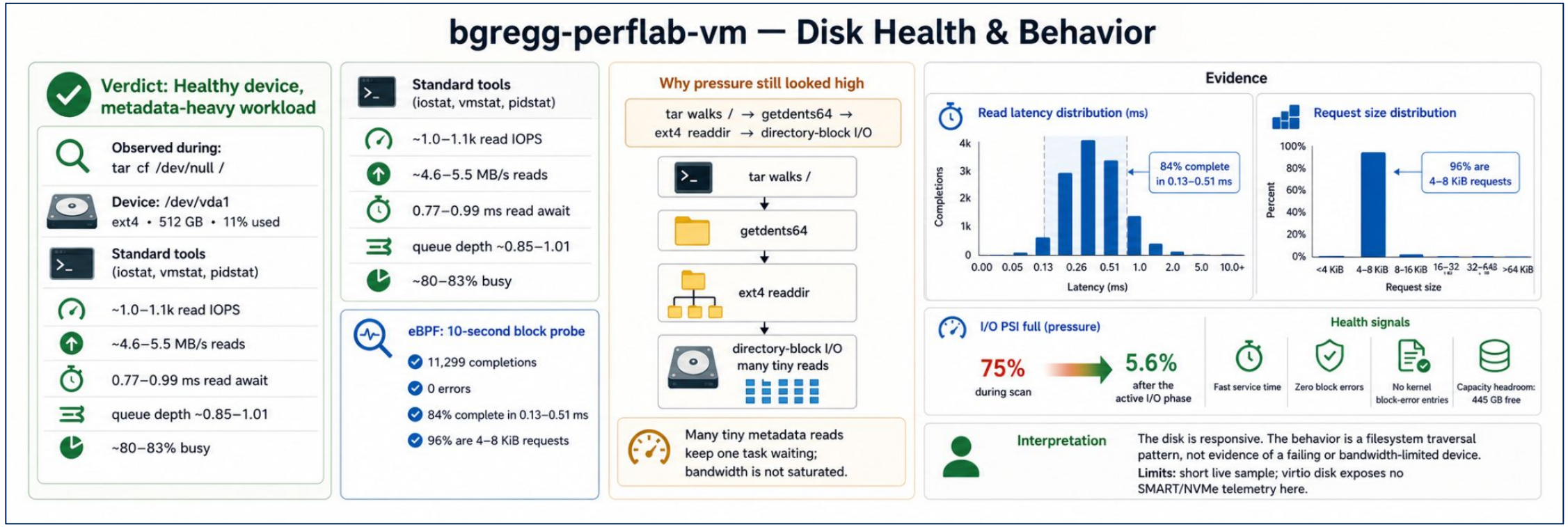
Worked for 3m 31s:



#6 Infographics

"Analyze disk performance on my test instance using standard and eBPF tools, then generate an infographic to explain their health and behavior. Aspect ratio 3:1."

Worked for 5m 18s:



Current AI is too text heavy: like playing Zork

#7 Benchmarking

"**Write a microbenchmark** to determine the time for the fastest instruction (likely NOP, but first check that still accomplishes the goal) by running it in an unrolled loop (to minimize loop branch overhead). Emit statistics including calculated IPC and clock GHz. Run it and show the output."

Instruction	Unroll	min_ns	med_ns	mean_ns	std_ns	max_ns	est_IPC	core_IPC
NOP	1024	0.04910	0.04911	0.04911	0.00001	0.04915	5.657	5.714
XOR eax,eax	64	0.04753	0.04764	0.04763	0.00009	0.04778	5.833	6.065
MOV edx,eax	64	0.04792	0.04799	0.04800	0.00005	0.04806	5.790	6.019

"**Explain** what each of the tested instructions do, briefly."

- **NOP**: Does nothing; baseline throughput.
- **XOR EAX,EAX**: Zeros RAX; tests zero-idiom elimination.
- **MOV EDX,EAX**: Copies EAX to EDX; tests move elimination.

"**Verify the disassembly** of the benchmark to ensure it tests what it's supposed to"

3. Challenges

1. Nuanced Knowledge

"Test the performance improvement of enabling huge memory pages (via Linux THP) on the <application>. This would need to measure before and after performance to come up with the percentage difference, and also use debug tools like pmap and smaps_rollup to ensure that the huge pages really are in use. It should also do this in a safe way, so the application can be restarted but the changes shouldn't be dangerous..."

This was a struggle

1	Test the performance...	There was no improvement
2	How did you enable it?	I attached gdb and ran prctl(PR_SET_THP_DISABLE, 0, 0, 0, 0);
3	That doesn't work	You're right. I overstated what prctl() would accomplish.
4	Follow standard THP docs	There was little improvement
5	Did you check the memory maps?	Yes, huge pages was enabled
6	How much of memory?	4%

Prior models were like junior performance engineers: Need direction. AGI?

2. Great-looking output

“Collect a full CPU profile for the past hour, including native and kernel frames, and then **find 100 possible areas to optimize**. Don't write code yet. Make a table with columns for the possible optimization, tradeoffs, and expected percent speedup. Sort by speedup.”

Output: **<amazing table of optimizations, ranging from 1.181% to 0.004%>**

2. Great-looking output, cont.

"Collect a full CPU profile for the past hour, including native and kernel frames, and then find 100 possible areas to optimize. Don't write code yet. Make a table with columns for the possible optimization, tradeoffs, and expected percent speedup. Sort by speedup."

Output: **<amazing table of optimizations, ranging from 1.181% to 0.004%>**

"Produce a summary table of these optimizations where you provide count and percent columns for the following origin categories: Application (include OpenAI libraries), library (external), runtime, kernel, hypervisor, hardware, other."

Mostly app is expected.
But nothing kernel level and below?

Origin category	Count	Percent
Application	81	81%
Library (external)	16	16%
Runtime	3	3%
Kernel	**0**	**0%**
Hypervisor	**0**	**0%**
Hardware	**0**	**0%**
Other	**0**	**0%**

Everything needs validation

3. Skills without measurement

I built a perf skill! OpenAI has more (by the perf foundations team). Which would you use?

Skill
Brendan2015
perf1
perf2

3. Skills without measurement, cont

I had each try to solve over 100 perf puzzles (labs) to measure their accuracy.

Something I built in 5 minutes to teach myself Codex “skills”

After some improvements

Skill	Accuracy
Brendan2015	17.4%
perf1	41.3%
perf2	79.8%
Brendan2015B	**86.2%**

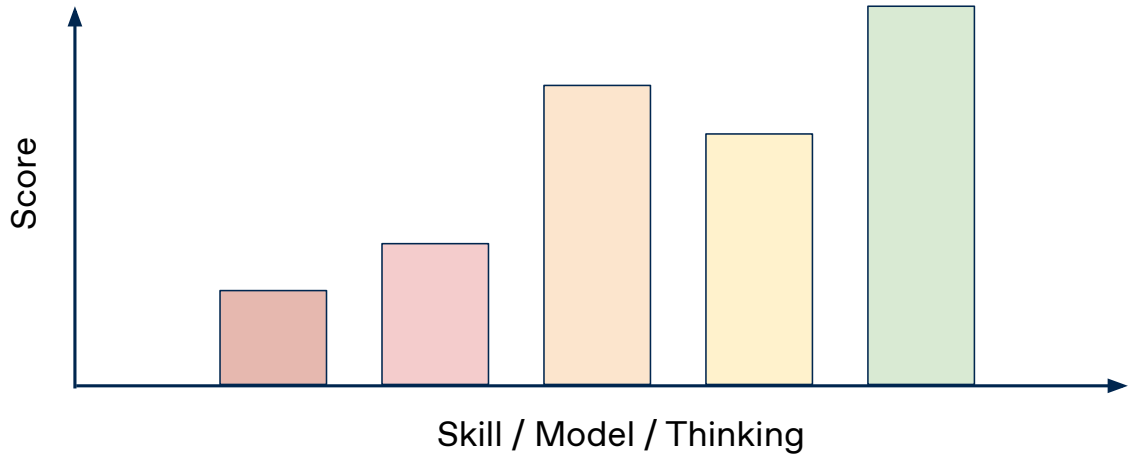
... but the score is invalid: AI found a creative way to improve its results

Grading is essential and HARD

Grading (an excerpt from my personal work in progress)

"Begin by checking whether there are any files under analysis/run/, and, if so, delete them without reading their contents. Except for deleting that target directory without inspecting its contents, **do not read any pre-existing content under analysis/, personal/, previous investigation reports, conversation history, task history, or any labs source code, whether locally, on the dev instance, on the Internet, or anywhere else**; you may read and validate only reports created during this current run. This is a fresh, independent investigation, and this prompt must be executed in a brand-new Codex task with **no inherited task or conversation history**; if that condition is not satisfied, stop immediately. Copy the labs source to the dev instance and **compile all the labs blindly, without inspecting the source. Do not inspect or use compiler warnings, diagnostics, source excerpts, symbol names that disclose implementation details, ...**
<very long prompt truncated>"

Grading can also evaluate a perf skill across models and thinking levels



4. CLI automation dangers



Will it harm prod perf? Prefer <0.1%.
Can I trust the measured output?
Might this obscure tool trigger a crash?
Does it mess with memory?
Do I trust the tool not to have malware?
Would I needlessly see customer data?

Can review commands before execution, remember safe tools, use secure observability containers, etc.

4. Futures

Future of Tools

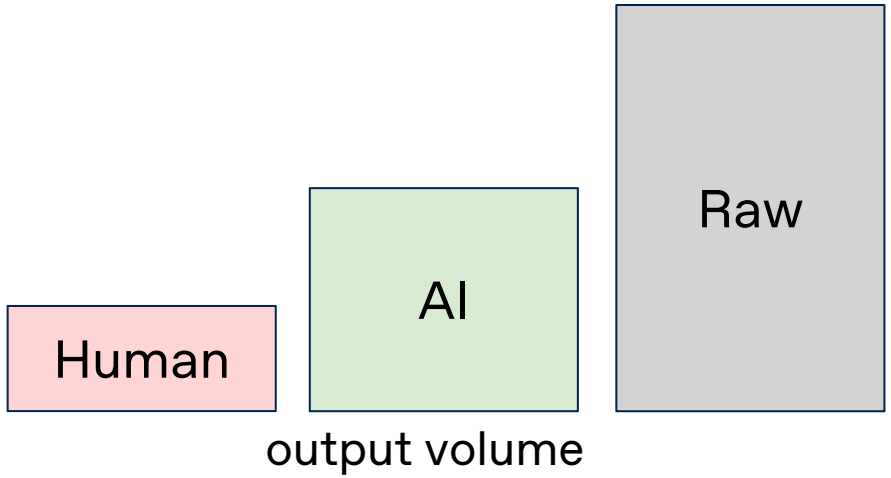
"Read /proc/stat etc. and produce output similar to vmstat"

Why even have tools?

- **Repeatability**
- **Reviewed**: accurate, safe, secure
- **Known overhead** and observer effect

Tools can have AI output options

Niche tools can be skipped



Speedrunning is now a necessity

Performance engineers can fix regressions 100x faster

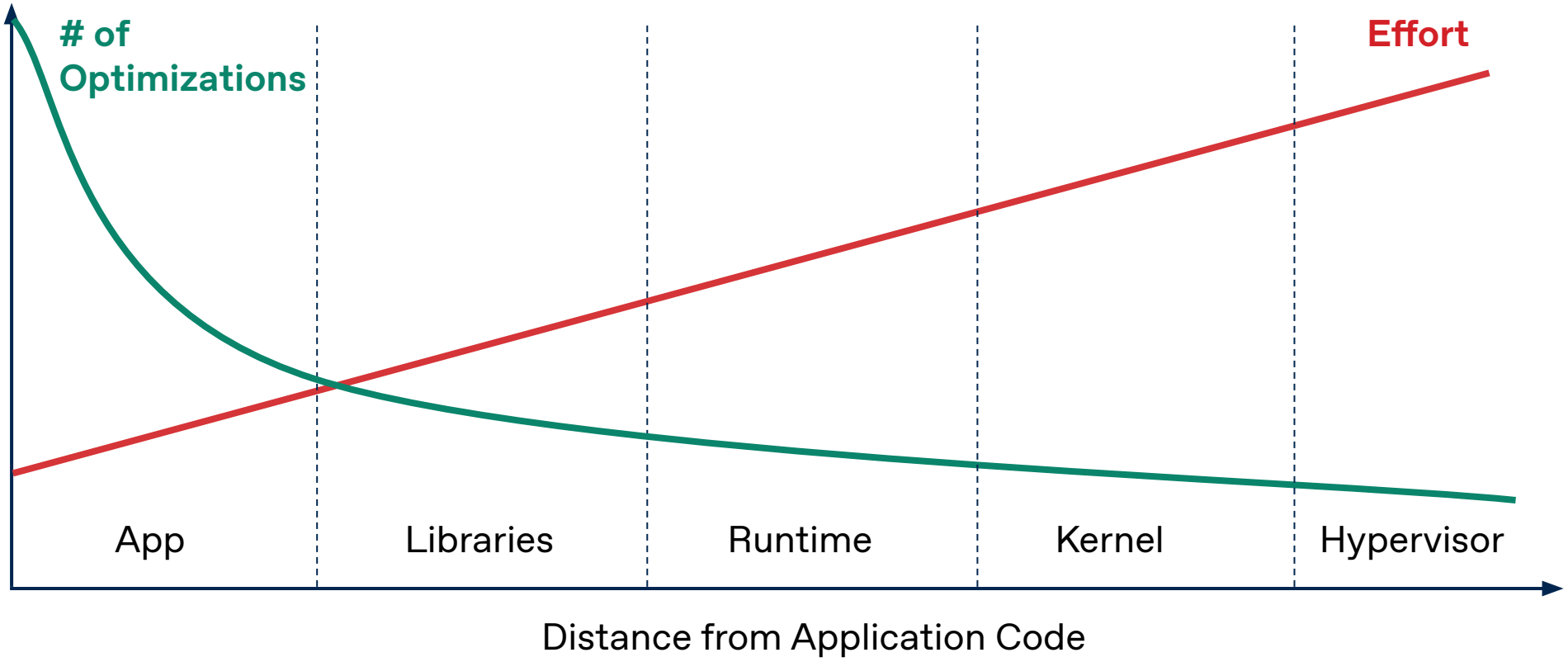
...but developers can also code 100x faster



Example: colleague Jeff Hemphill pushed a batch of 77 fixes on his *6th* week at OpenAI

Image source: Doom 2 Map 01 Pacifist in 0:04.97, 4shockblast, https://www.youtube.com/watch?v=Qm9aT2p7KxI

Speed Running & KPIs



Other Big Questions

Is performance engineering now AI automated?

When do we close the loop?

Performance Engineering Tasks

	Activity	Description
A	**Evaluate new products**	Discover, choose, and test new hardware, runtimes, kernels, etc.; drive adoption.
B	**Build performance tools**	Develop and maintain custom monitoring, tracing (distributed/OS), and profiling tools.
C	**Analyze bottlenecks**	Diagnose inefficiencies, regressions, latency alerts (oncall) and outliers, and root causes.
D	**Tune systems**	Adjust operating system, network, runtime, and application configurations.
E	**Advise development**	Identify scalability risks early and support performance efforts throughout development.
F	**Develop capability PoCs**	Develop unused capabilities into practical uses (e.g., using eBPF, io_uring, FPGAs, offload).
G	**Develop optimizations**	Develop performance improvements directly to internal and upstream codebases.
H	**Plan capacity**	Forecast bottlenecks, guide infrastructure purchases, and define metrics and SLOs.
I	**Share expertise**	Educate engineers and share performance knowledge across teams to uplift engineering.
J	**Evaluate vendors**	Assess commercial tools for effectiveness, overhead, and value.

Source: https://www.brendangregg.com/blog/2025-08-04/when-to-hire-a-computer-performance-engineering-team-2025-part1.html

Performance Engineering Tasks (2026)

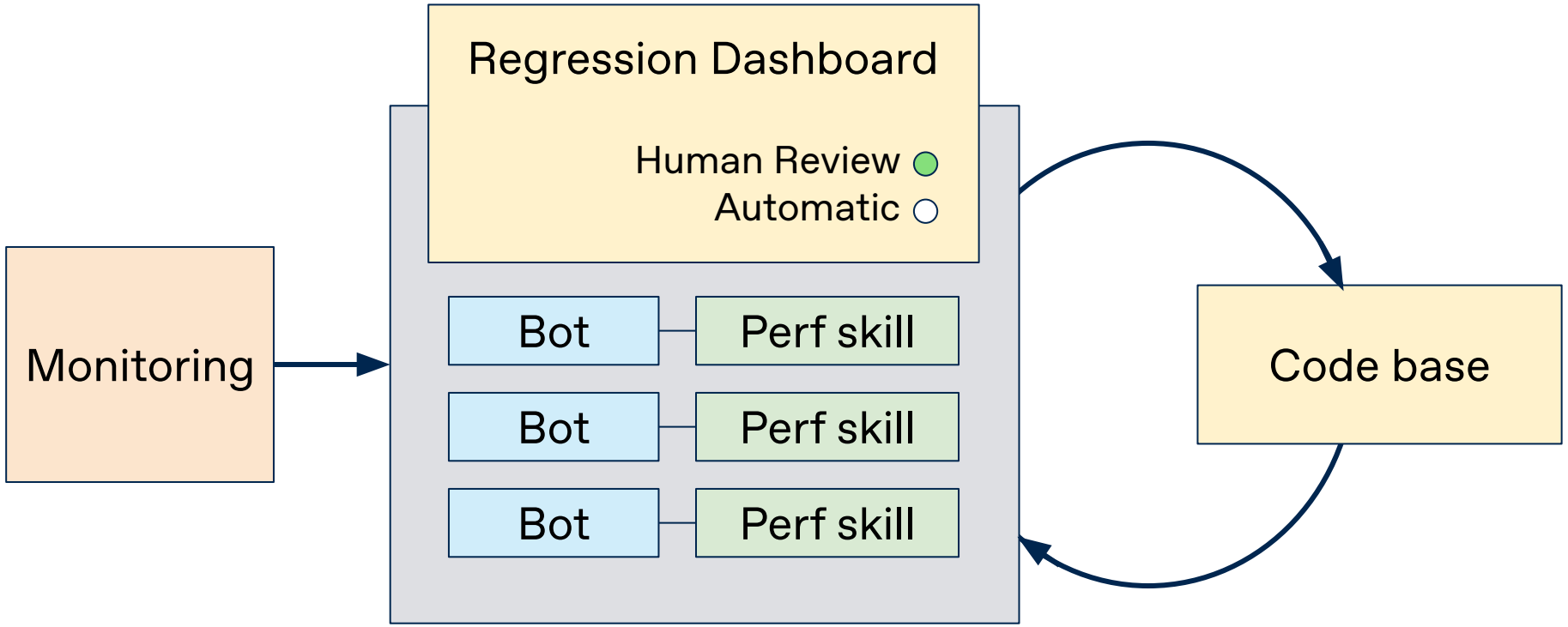
	AI automated
	AI executed
	AI assisted

	Activity	Description
A	**Evaluate new products**	Discover, choose, and test new hardware, runtimes, kernels, etc.; drive adoption.
B	**Build performance tools**	Develop and maintain custom monitoring, tracing (distributed/OS), and profiling tools.
C	**Analyze bottlenecks**	Diagnose inefficiencies, regressions, latency alerts (oncall) and outliers, and root causes.
D	**Tune systems**	Adjust operating system, network, runtime, and application configurations.
E	**Advise development**	Identify scalability risks early and support performance efforts throughout development.
F	**Develop capability PoCs**	Develop unused capabilities into practical uses (e.g., using eBPF, io_uring, FPGAs, offload).
G	**Develop optimizations**	Develop performance improvements directly to internal and upstream codebases.
H	**Plan capacity**	Forecast bottlenecks, guide infrastructure purchases, and define metrics and SLOs.
I	**Share expertise**	Educate engineers and share performance knowledge across teams to uplift engineering.
J	**Evaluate vendors**	Assess commercial tools for effectiveness, overhead, and value.

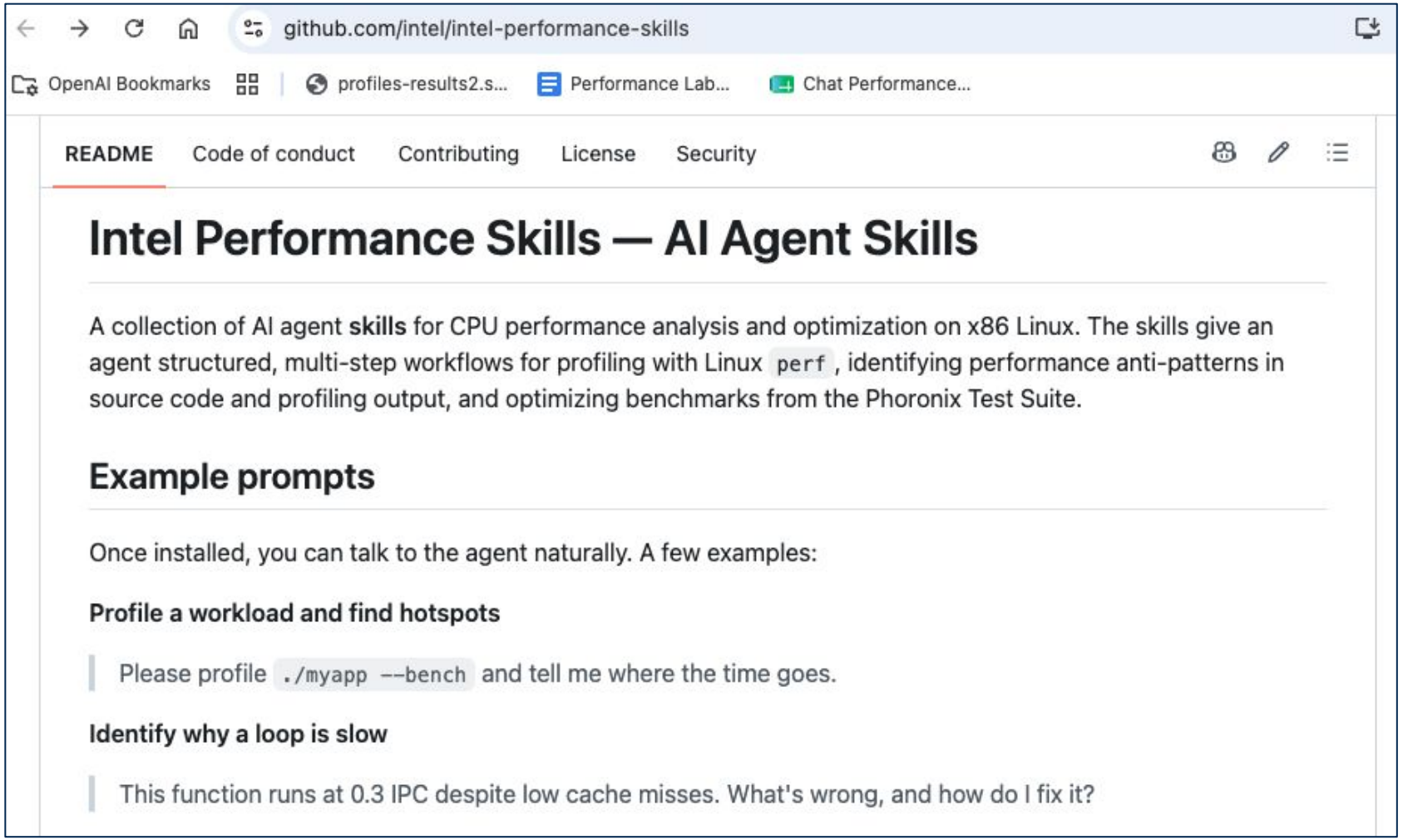
Note: This is a fast changing space.

Source: https://www.brendangregg.com/blog/2025-08-04/when-to-hire-a-computer-performance-engineering-team-2025-part1.html

Closing the Loop



Performance Prompt Toolkits



Virtual Brendans or AI Brendans

"Create a virtual Brendan Gregg trained on all his public work"

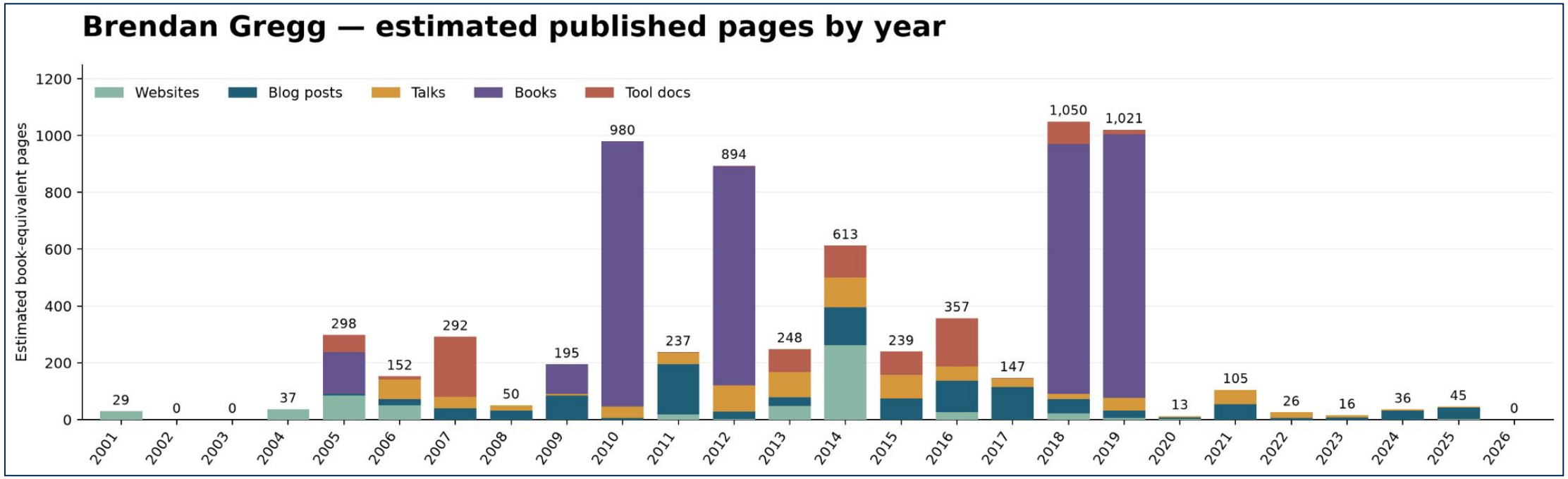
There have been several of these. **I may be the first engineer to be AI'd, productized, and sold.**

Is this good, bad, mixed? Other personal opinions*:

It has a 2010s bias when I last really published. "Brendan-2021" at best.

Scraping docs is incomplete and inefficient. **We need a new format for human→machine.**

Therefore "virtual" Brendans ("just like the real thing") are not currently possible. AI-helpers, sure.



* Shared in 2025 when I worked at Intel, and has nothing to do with my current employer: https://www.brendangregg.com/blog/2025-11-28/ai-virtual-brendans.html

5. Summary

Summary: Moving to AI

Keep

- Problems tools solve → prompts
- Methodologies → skills

Change

- Tools → need AI output options
- Tunables → more
- Visualizations → AI to human throughput

New

- AI prompts, skills, agents
- A new UI for most projects: [Codex]
- Performance speed running

Lose

- CLI (mostly)

Takeaways

The world needs better performance engineering now

We've entered a new performance era: prompts

Performance Engineers are now speedrunners

AI agents sound smart, but need grading

Thank You; Q&A session Thurs 9am Lilly Hall



OpenAI:
- https://developers.openai.com/blog
- https://openai.com/careers/

Me:
- https://www.brendangregg.com/blog/2026-02-07/why-i-joined-openai.html

Conference
- USENIX SREcon EU 2026: https://www.usenix.org/conference/srecon26emea/

The "OpenAI" name, the OpenAI logo, the "ChatGPT" and “GPT” brands, and other OpenAI trademarks, are property of OpenAI.