

למה ה-Harness חשוב יותר מהמודל | YC Paper Club

ערב על harnesses ב-YC: איך השכבה שמסביב למודל לוקחת את אותם משקלים
מ-30% ל-95% ב-ARC-AGI-3, ושלוש מערכות שנבנו מהרעיון הזה

Francois Chaubard, Seth Karten, Jon Saad-Falcon, Josh France, Regan Bell · Y Combinator · 7
בספטמבר 2026

משך: כ-60 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הקלטה של ערב YC Paper Club שהוקדש ל-harnesses, כלומר לשכבת הקוד שעוטפת LLM ומוסיפה לו לולאה, כלים, זיכרון, subagents וסביבת הרצה. את הערב פותח Francois Chabard (דוקטורנט ו-Visiting Partner ב-YC, כך לפי השקף), ואחריו מציגים שלושה פרויקטים: Seth Karten מ-Prime Intellect ו-Princeton מציג את Prime Agent, Jon Saad-Falcon מ-Stanford מציג את OpenJarvis, ו-Josh France ו-Regan Bell מ-YC מציגים את QM, ה-harness הפנימי ש-YC בנתה לעובדים שלה.

הטענה שחוזרת לאורך כל הערב היא שה-harness הוא לא "סתר wrapper" או prompt engineering, אלא מנוף מחקרי והנדסי מרכזי. לפי ההצגה, על אותם משקלים בדיוק harness טוב לוקח את ARC-AGI-3 מ-30% ל-95%, מאפשר להריץ מודלים מקומיים בעלות נמוכה פי 800 מהענן, ומאפשר לארגון לתת agent אישי לכל עובד בלי לנהל צי של מכונות. הכיוון שבולט בכל ההרצאות: harness דק ככל האפשר, שנותן למודל כמה שיותר יכולת ביטוי (expressiveness) ושמשפר את עצמו עם הזמן.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר.
- **2. למה harness:** הפתיחה של Francois, ה-auto-researcher שבנה כמעט במקרה, וההיסטוריה של harnesses בחמש דקות.
- **3. Prime Agent:** מודל הזיכרון בשכבות, subagents מתמידים, ותוצאות ARC-AGI-3 ו-long-horizon.
- **4. OpenJarvis: personal AI:** על מכשיר מקומי, חמשת ה-primitives ומודל ענן שמבצע אופטימיזציה ל-stack המקומי.
- **5. QM:** ה-harness של YC לעבודה: המוח מחוץ ל-harness, sandbox דק, ומה עדיין לא עובד.
- **6. מה לעשות עכשיו:** צעדים מעשיים וטבלת שאלות פתוחות.
- **7. מילון מונחים:** המונחים באנגלית שמופיעים במסמך.

הערת מקור המסמך הופק אוטומטית מהתמלול של ההקלטה ומהשקפים שהוצגו בה. ייתכן שזיהוי הדיבור (ASR) שיבש שמות ומונחים, ובמקומות כאלה הסתמכנו על השקפים וסימנו אי-ודאות. הציטוטים מועתקים מהתמלול מילה במילה ובבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

הערב בנוי כטיעון אחד שמגיע משלושה כיוונים. במחקר, harness שנותן למודל יכולת ביטוי משיג תוצאות שהמודל לבד לא משיג. ב-harness, edge שמודל ענן ביצע עליו אופטימיזציה מקרב מודל מקומי קטן לביצועי ענן. בארגון, harness דק שמוציא את ה"מוח" מה-sandbox ומחבר אותו לכל המשאבים של החברה הופך agent לכלי עבודה יומיומי. בשלושת המקרים המשקלים לא משתנים, וכל השיפור מגיע מהשכבה שמסביבם.

- **אותם משקלים, 30% מול 95%:** לפי Francois, Claude Opus הגיע ל-30% ב-ARC-AGI-3 על ה-holdout הפרטי, ועם harness הגיע ל-95%. בשקף: Prime Agent עם RHAЕ 95.5% ופתרון של NVIDIA עם 100% RHAЕ [3:02].

- **ה-harness לבד שווה 18%:** שקף הפתיחה מציג ב-Terminal Bench 2 שיפור של 18% במעבר מ-Claude Code ל-Meta-Harness על אותו מודל [0:00].

- **השלב הנוכחי הוא harness שמשפר את עצמו:** DSPy לומד את ה-Darwin Gödel system prompt, Machines משנים את קוד ה-harness עצמו, ו-Continual Harness מעדכן גם משקלים בזמן ריצה [13:53].

- **Expressiveness היא העיקר, לא התבנית:** Seth טוען שאין צורך לכפות לולאת plan-act-critique, אבל צריך לתת למודל state-compaction, REPL, subagents, כי "if you remove one of those, you're actually removing a capability" [24:40].

- **יחס עלות-ביצועים הוא חלק מהתוצאה:** עם Hermes Agent הוציאו כ-5,000 דולר על ARC בלי שיפור משמעותי ועצרו. עבודה פרוגרמטית עם הקונטקסט היא מה שחוסך כסף [32:36].

- **Long horizon עובד, אבל קשה לייחס אותו ל-harness:** ריצת Factorio של שבעה ימים עם 633 agents ו-23 מיליון output tokens המשיכה להתקדם עד הסוף [35:39], אבל ב-nanoGPT speedrun נאמר במפורש שאי אפשר לדעת אם השיפור בא מה-harness או מהמודל [34:08].

- **מודל מקומי עם spec שבוצעה עליו אופטימיזציה מתקרב לענן:** OpenJarvis מגיע לעלות נמוכה פי 800 [43:18]. לפי השקף התוצאות נמצאות בטווח של 3.2 נקודות אחוז מהמודל הטוב בענן, משתוות לו או עוקפות אותו ב-4 מתוך 8 benchmarks, עם latency נמוך בערך פי 4.

- **ב-QM המוח יוצא מה-sandbox:** כל השיחות נשמרות ב-Postgres, ה-sandbox הופך למשאב שה-agent בוחר, וה-harness נשאר דק, עם שלושה כלי ליבה בלבד [52:41], [55:50].

- **המגבלות כיום התנהגותיות:** agents מוותרים מוקדם מדי, ולכן QM מוסיף "grind tool" עם תקציב לכל מטרה [57:22]. בנוסף הם לא מבינים הקשר חברתי, ולכן כמות הידע שאפשר לתת להם מוגבלת לפי איכות מערכת ההרשאות [58:57].

מה לא נאמר

- **איכות המאמרים של ה-Francois auto-researcher:** אומר שהם "actually good", אבל לא מציג מדד, ביקורת חיצונית או שיעור קבלה.

- **Claude Code ב-Seth: ARC-AGI-3:** אומר שהריץ אותו והתוצאות "weren't very good", אבל לא הציג אותן. במקומן הציג את התוצאות המקוריות של Claude Code.

- **הפתרון של NVIDIA שמגיע ל-100%:** מופיע בשקף ובמשפט אחד בלבד, בלי פרטים על השיטה.

- **עלות ה-95.5%:** גרף ה-cost scaling מופיע בשקף, אבל עלות לכל משימה או עלות כוללת לא נאמרה בעל-פה.

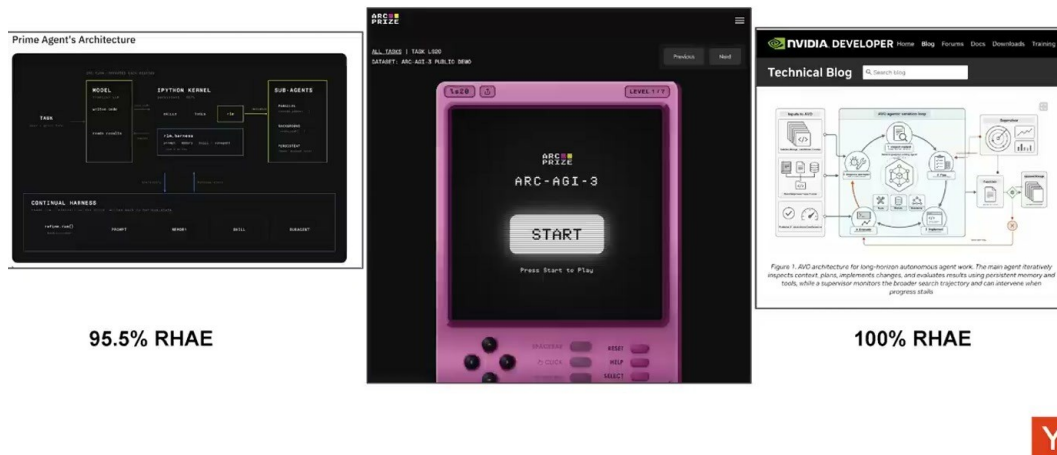
- **OpenJarvis:** לא פורט באילו משימות המודלים המקומיים נכשלים, מול איזה baseline בדיוק נמדד ה-800x, ועל איזו חומרה.

- **QM:** לא הוצגו מספרי שימוש, אימוץ או איכות. לולאת השיפור העצמי תוארה כבעלת "mixed results", בלי נתונים.

- **החלפת מודל כדי לעקוף סירובים:** QM מאפשר ל-agent לעבור למודל הנוכחי מסרב. ההשלכות על מדיניות, compliance ובקרה לא נדונו.

2. למה harness: הפתיחה וההיסטוריה

Harness ARC-AGI3 Saturated (Aug 2026)



השקף "Harness ARC-AGI3 Saturated (Aug 2026)": הארכיטקטורה של Prime Agent עם 95.5% RHAЕ, משחק ARC-AGI-3, והבלוג הטכני של NVIDIA עם 100% RHAЕ

Francois פותח עם הביקורת המוכרת: ציטוטים מ-Reddit שלפיהם prompt engineering לא שייך לכנסים מובילים ו-context engineering הוא "not a research problem". מולם הוא מציג נתונים. ב-Terminal Bench 2, המעבר מ-Claude Code ל-Meta-Harness מוסיף 18% (לפי השקף). בגרף של METR, שמראה כמה זמן agent מסוגל לעבוד, הוא טוען שחלק גדול מההתקדמות מגיע מ-harnesses, ומחלק את התקופה ל-static harness era ולעידן של self-improving harnesses בחצי השנה האחרונה.

הטיעון המרכזי שלו הוא שאנחנו כמעט לא לומדים מ-ICL. test-time experience נרווה אחרי 40-50 דוגמאות, ומשם צריך לעבור ל-LoRA בדרגה נמוכה, ל-LoRA בדרגה גבוהה ול-ARC-AGI-3. SFT מודד בדיוק את היכולת להסתגל מהר להתפלגות חדשה. לדבריו, Claude Opus היה מהראשונים שאומתו על ה-holdout הפרטי, וקיבל 30% (בשקף: 30.2%).

Francois Chaubard [3:02] this thing that doesn't deserve any research just some wrapper and — some scaffolding we can get to 95

AutoResearcherUI Harness: Francois' researcher loop



☒ Run the agent with --dangerously-skip-permissions

RESEARCH

Purpose — what are we researching, and why?

Prove that an ensemble of n diffusion language models (n large) is a more efficient learner than autoregressive LLMs. Start with classic GPT-2 on GSM8K as the AR baseline. Then train 1 diffusion LLM on *GPT-2* then 2 with different initial batch sizes, ensemble their

Seed ideas

Vary n (ensemble size) 1,2,3,...; different inits per member; different batch orderings per member; logit averaging vs other ensembling.

Evaluation function / validation set

GSM8K test split; metric = exact-match test accuracy.

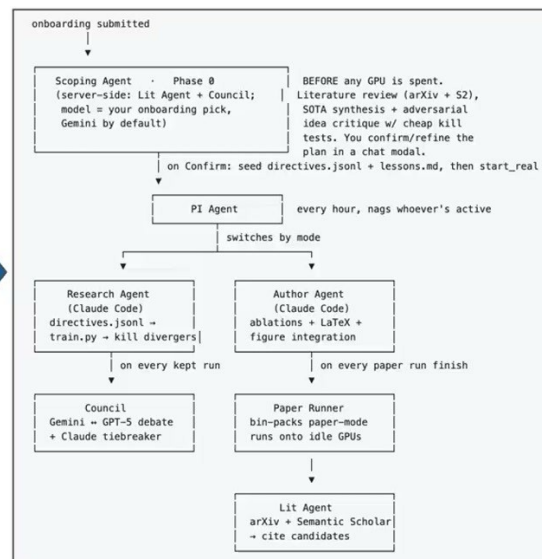
Validation metric

val_loss

Baseline method(s) to run first

GPT-2 trained with autoregressive cross-entropy loss on GSM8K.

AGENT (ADVANCED — LEAVE THE TEXTAREA ALONE FOR SENSIBLE DEFAULTS)



השקף "AutoResearcherUI Harness: Francois' researcher loop": טופס שמגדיר purpose, eval metric, ו-seed ideas, ותרישים זרימה של scoping agent, PI agent, research agent, council, ו-author agent

הדוגמה האישית של Francois: הוא עשה fork ל-auto-researcher של Karpathy רק כדי לבנות ממשק מעקב, ובדבריו "[3:02] I ended up building a harness by accident". המערכת מקבלת מטרה (למשל לבדוק אם sharding של diffusion LLM ל-ensemble מנצח מודל AR), seed ideas, ו-eval metric כמו GSM8K. משם המשימה עוברת ל-scoping agent שמחפש מאמרים ו-repos, ל-PI agent שמנדנד כל שעה, ל-research agent, ל-council שבו Francois עצמו נותן משוב, ולבסוף ל-author agent שמקפא את הרעיון וכותב ablations ומאמר. כיום הוא נותן שמונה רעיונות לשמונה nodes של 8xH100 ומקבל בחזרה מאמרים. בלשונו, "the things that we can now do just because of harnesses on the same exact weight file is just wild" [6:08].

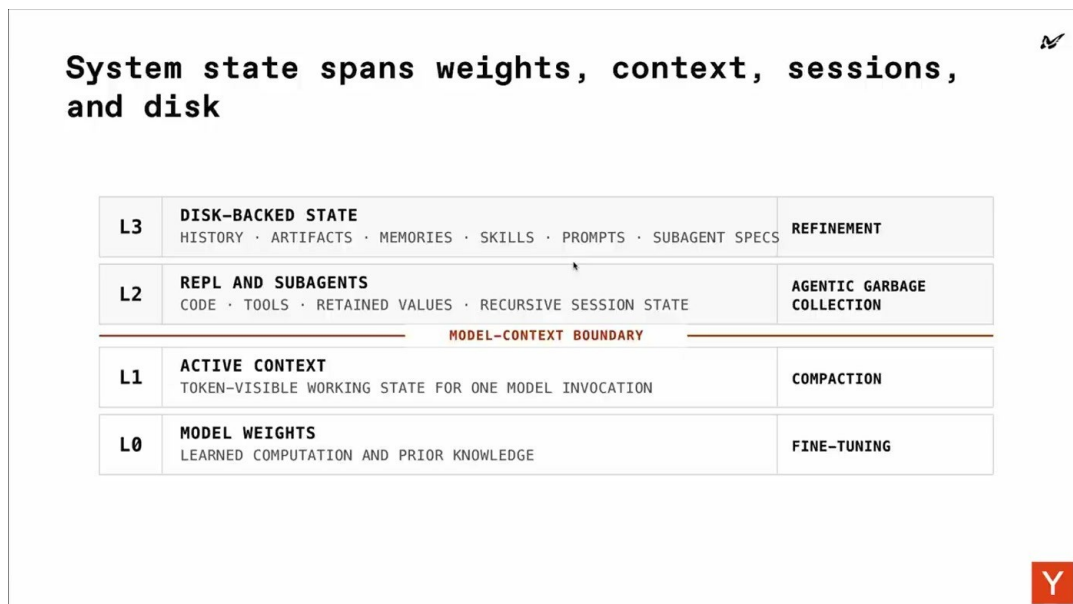
בחלק הבא Francois עובר על ההיסטוריה של harnesses. כל שלב בה מתואר כהרחבה של אחד משלושה מרכיבים: active context, action space, או learning signal. הסדר לא כרונולוגי, וזה מכון.

מה נוסף ל-harness	עבודה (כפי שמופיעה בשקף)
harness v0: לולאה עד EOS, top-k sampling, וסביבה שמחזירה reward. אין כלים ואין skills.	GPT-2 (Feb 2019)
דוגמאות פתורות בתוך הקונטקסט.	Few-shot / ICL (Jul 2020)
פריסת החישוב על יותר טוקנים במקום לחזות את התשובה ישר.	Chain of thought
tool calling: אובייקט JSON שמפעיל קוד חיצוני.	WebGPT / Toolformer
CRUD על אזור זיכרון בקונטקסט במקום append בלבד.	MemGPT (Oct 2023)
skills: שרשור כלים ושמירת הרצף כ-skill (הבסיס ל-skills.md של היום).	Voyager (Oct 2023)
קוד כ-action: כלים ו-skills שנוצרים on the fly.	InterCode (May 2023)
evaluator פנימי ו-reflection שמתקן את הפלט.	ReAct / Self-Refine / Reflexion
subagents ב-REPL מתמשך, ו-rm_query רקורסיבי. Francois קורא לזה harness v1.	Multi-agent → RLM (Dec 2025)
לימוד ה-system prompt באמצעות genetic programming על train set קטן.	DSPy
ה-agent משנה את קוד ה-harness עצמו, ויש archive של וריאנטים ו-fitness function.	Darwin Gödel Machines (Mar 2026)
harness שתפקידו לייצר harnesses.	Meta-Harness (Mar 2026)
online adaptation, סיווג עשיר יותר של זיכרון ועדכון משקלים בסגנון DAgger.	Continual Harness (May 2026)



השקף "Static Harness v1": קוד ה-loop מימין לשמאל, כלומר LLM call, context compiler, session management, ו-tool invocation, לצד agent spec עם allowed skills, allowed tools, ו-max turns, ו-subagents

3. Prime Agent: harness רקורסיבי שמשפר את עצמו



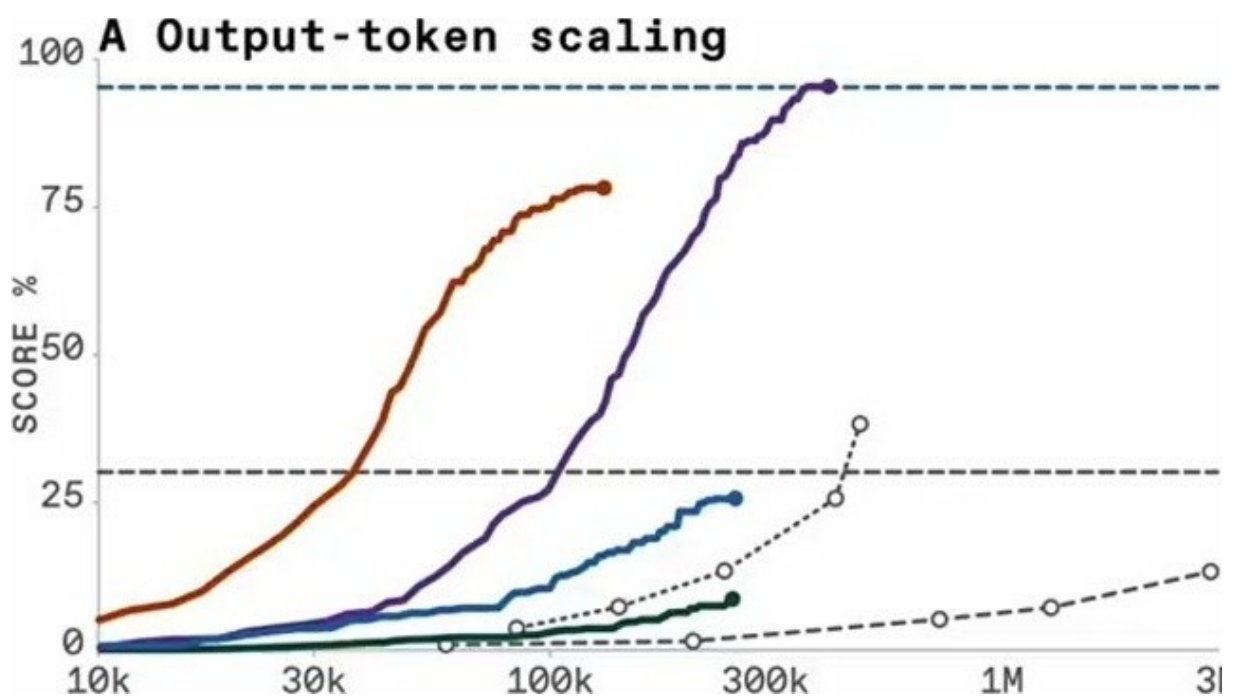
השקף "System state spans weights, context, sessions, and disk": L0 model weights (fine-tuning), L1 active context (compaction), L2 REPL (agentic garbage collection), ו-L3 disk-backed state (refinement)

Seth מתחיל מהיסודות. LLM הוא מעבד סדרתי עם משקלים קבועים שמקבל טוקנים ומוציא טוקנים, ו-"The harness itself is the layer between the LLM and the world that adds things like this persistent state, tools, and compute" [18:36]. הוא מתאר את הזיכרון כהיררכיה של cache. המשקלים הם השכבה המהירה ביותר אבל גם היקרה ביותר לעדכון. ה-active context מנוהל באמצעות compaction. בשכבת L2 יש REPL של

IPython, שבו משתנים נשמרים ב-RAM, ה-agent מעבד אותם בקוד במקום לטעון אותם לקונטקסט וחוסך כך טוקנים. בשכבת L3 נמצאים הדיסק, ה-skills, הזיכרונות וה-prompts. לכל שכבה יש מנגנון ניקוי: agentic garbage collection ב-REPL ו-refinement בדיסק.

המטפורה שלו היא שמודל לבד דומה למכונת Turing עם סרט סדרתי, ואילו harness הופך אותו למחשב von Neumann עם זיכרון חיצוני שאפשר לקרוא ממנו ולכתוב אליו. מכאן נובעת עיקרון התכנון: harness צריך להיות expressive ככל האפשר. לולאת plan-act-critique כבר לא נחוצה, כי המודלים למדו לעשות אותה בעצמם. מה שחשוב הוא לחשוף להם יכולות. ב-Prime Agent כל קריאה ל-rlm() פותחת subagent session מתמשך, שעובר בין המצבים inactive, admitted, running, idle. אפשר לשלוח אליו הודעה ולחדש אותו עם כל הקונטקסט שצבר. בנוסף כל agent יכול לשלוח הודעות ל-agents אחרים ב"משפחה הגרעינית" (הורה, ילדים ואחים). יש daemon שממשיך לרוץ גם אחרי שסוגרים את הפטופ, ו-refinement שמעדכן prompts, skills, memory ו-subagent specs מתוך trajectories קודמות.

Seth Karten [24:40] And if you remove one of those, you're actually removing a capability that it won't be able to do otherwise

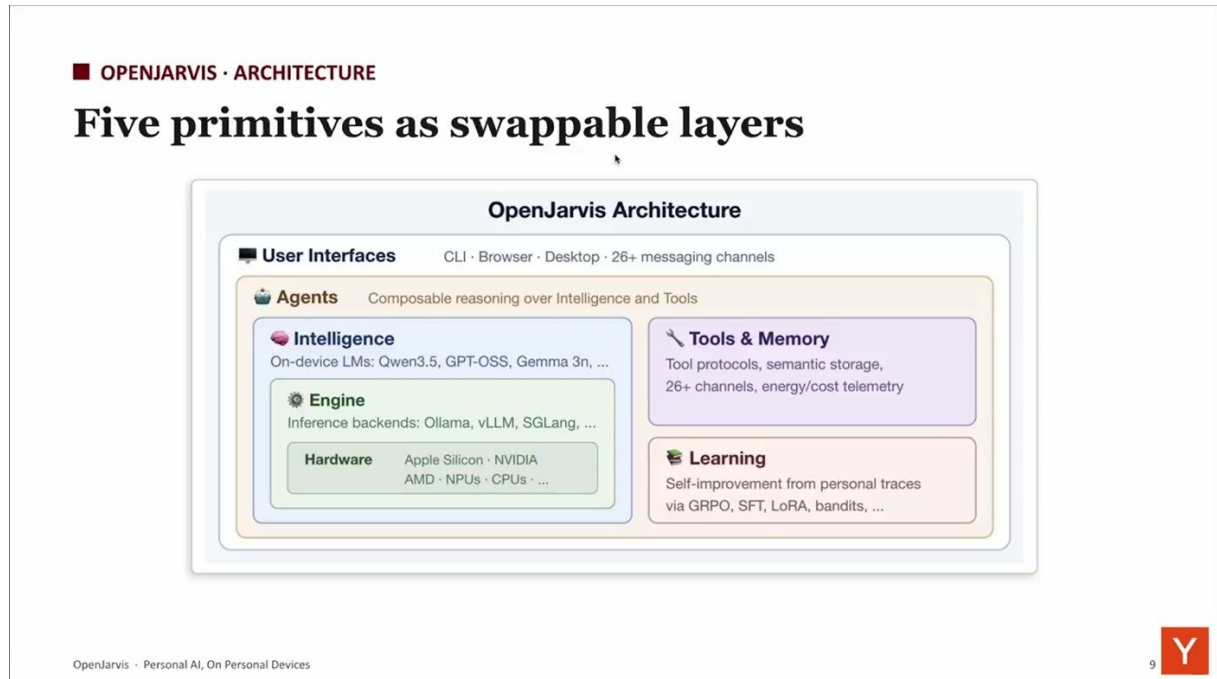


השקף "Prime Agent scales to 95.5% with more output tokens and API cost": שני גרפים בסקאלה לוגריתמית (output tokens ועלות API) ומקרא: Opus 5 עם 95.5%, GPT-5.6 Sol עם 78.3%, Terra עם 25.7%, GLM 5.2 עם 8.8% ו-Hermes Agent עם 5.8%

הסיפור של ARC-AGI-3 מוצג בכנות. Seth לקח system prompt מ-leaderboard קהילתי, הכניס אותו ל-Prime Agent, וקיבל בהרצה הראשונה 99.9%. אז בדק את הלוגים, ובמילותיו: "I looked at the logs and I was cheating" [30:58]. אחרי יום של sandboxing תקין הגיע ל-78% עם GPT Sol, ל-25.7% עם Terra ול-95.5% עם Opus. מבדיקת ה-traces עולה שהמודל כתב הרבה קוד ב-REPL כדי לבדוק תרחישים ולעבד תמונות. הוא מדגיש שגם ההשוואה ל-harnesses אחרים היא חלק מהתוצאה: עם Hermes Agent "i spent like five thousand dollars without making much performance" [32:36].

בתוצאות הנוספות, לפי השקף, ל-Prime Agent יש point estimate גבוה יותר ב-20 מתוך 27 השוואות: 8/9 מול Pi-mono עם GLM-5.2, ו-6/9 מול Claude Code עם Opus 5 וגם מול Codex עם GPT-5.6 Sol. ב-Emulator Bench (בניית אמולטור Game Boy Color) הוא מנצח, וב-GPU kernels התוצאה בערך שוויונית. ב-nanoGPT speedrun על 8xH200 במשך שבוע הוא לא מייחס את השיפור ל-harness, אבל מציין התנהגות מעניינת: ניסוי out-of-loop על CPU לפני שמשקיעים בהרצות H200. יקרות. ריצת Factorio של שבעה ימים כללה 633 agents (לכל היותר 7 פעילים במקביל לפי השקף) והמשיכה להתקדם ב-tech tree עד הסוף. ההמלצות שלו לסיים: context management, swarms, RLMs עמוקים יותר, ו-evals סטנדרטיים (הכול ניתן להרצה עם ה-verifiers של Prime Intellect package).

4. OpenJarvis: personal AI על מכשיר אישי

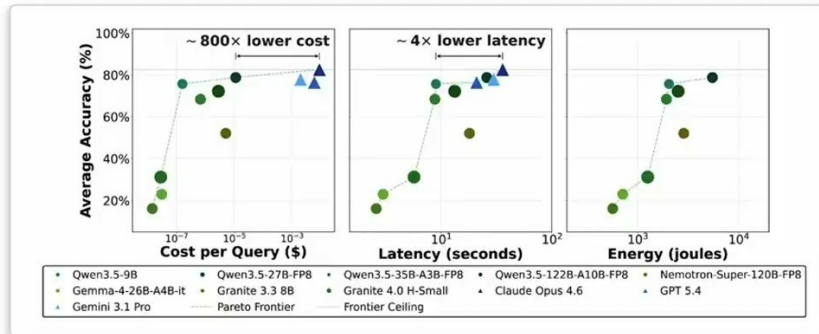


השקף "Five primitives as swappable layers": User Interfaces, Agents, Intelligence (Qwen3.5, GPT-OSS, Gemma 3n), השקף Learning (GRPO, SFT, LoRA) ו-Tools & Memory, Engine (Ollama, llama.cpp, vLLM, SGLang) מעל החומרה,

Jon Saad-Falcon (מ-Stanford), עם co-lead author שהשם שלו שובש בתמלול, ובהנחיית Azalia Mirhoseini ו-Christopher Ré (לפי ההקשר) פותח בבעיה: כלי personal AI כמו OpenClaw ו-Hermes Agent נשענים על הענן. זה יקר (אלפי דולרים בשנה), לא פרטי, מחייב "לשכור" אינטליגנציה ודורש יותר אנרגיה בכמה סדרי גודל. מנגד, לטענתו מודלים מקומיים מפגרים אחרי ה-frontier ב-6 עד 12 חודשים בלבד. בתמלול מוזכר מודל Qwen ב-27B שמשתווה ל-Claude 4.6 Opus. בשקף מוזכרות המשפחות Qwen3.5 ו-Gemma4 ומודלים בגודל 1B עד 128B על מאיצים צרכניים, ולכן שם המודל המדויק אינו ודאי.

OpenJarvis מצמצם כל harness לחמישה primitives: Intelligence, Engine, Agents, Tools & Memory ו-Learning. כולם מוגדרים ב-spec אחד מוקלד שאפשר לבצע עליו אופטימיזציה. החידוש המרכזי הוא LLM-guided spec search. מודל ענן חזק משמש כ-teacher: הוא מאבחן כשלים מתוך traces, מציע עריכות לכל ארבעת ה-primitives הניתנים לעריכה, מאשר רק עריכות שלא גורמות לרגרסיה, וה-spec הסופית רצה מקומית בלי עלות API. כך משלמים על יכולת הענן פעם אחת בזמן החיפוש, והעלות מתחלקת על כל השאילתות המקומיות. לפי השקפים, Qwen3.5-9B אחרי spec search מגיע ל-100% ב-PinchBench, ל-83% ב-LiveCodeBench ול-91% ב-LiveResearchBench. האופטימיזציה עצמה זולה פי 7 עד 11 מ-DSPy, LoRA, SFT או GEPA. כל teacher עזר, ו-Opus 5 ו-GPT 5.6 Sol היו הטובים ביותר.

On-device LM systems rival cloud LMs



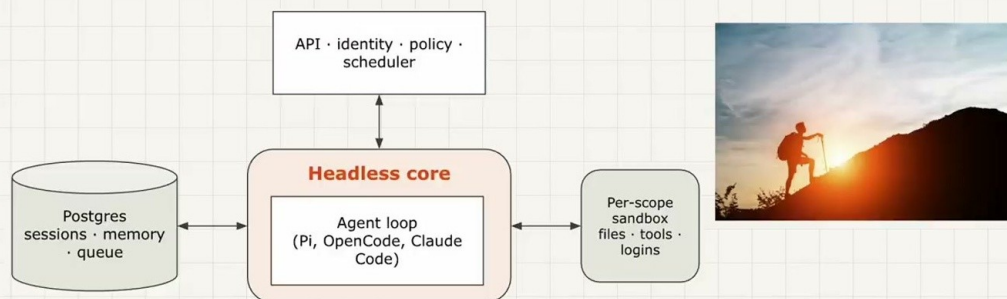
Within 3.2 pp of the best cloud model, matching or beating it on 4 of 8 benchmarks, at ~800× lower cost and ~4× lower latency.



השקף "On-device LM systems rival cloud LMs": שלושה גרפי פיזור של דיוק מול עלות לשאילתה, latency ואנרגיה. השורה התחתונה: within 3.2 pp of the best cloud model, ~800x lower cost, ~4x lower latency

5. QM: harness של YC לעבודה

1. Give them the ability to edit themselves



השקף "Headless core": Give them the ability to edit themselves" עם agent loop (Pi, OpenCode, Claude Code), שמחובר ל-Postgres (sessions, memory, queue) לשכבת API עם identity, policy ו-scheduler, ול-sandbox לכל scope עם files, tools ו-logins

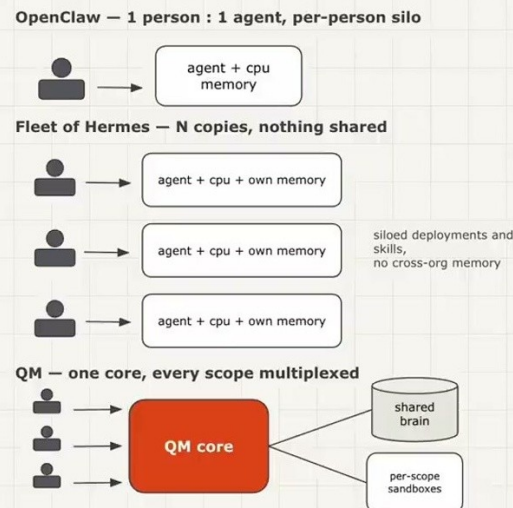
Regan Bell ו-Josh France מתארים איך התפתחו ה-agents הפנימיים של YC. בינואר 2025 היה "general agent" עם כלים בלולאה, אותו agent יכולם, וטוב במפתיע בשאלות על דאטה. ביוני 2025 הריצו system prompt ו-Claude Code ב-VM עם תיוג ב-Slack, ושיפרו את agents.md לפי הכשלים שראו. בינואר השנה ה-partners אימצו את OpenClaw, כי לראשונה היה ל-agent מחשב משלו. באפריל הקימו צי של יותר מ-50 Hermes agents ב-VMs. הם עזרו, אבל דרשו הרבה קונפיגורציה, והתחזוקה הייתה "whack-a-mole" של התחברות ב-SSH לכל instance בנפרד.

המסקנה הייתה לשלוק את ה"מוח" מתוך ה-sandbox. כל השיחות נשמרות ב-Postgres ונחשפות ל-agent עצמו, וה-sandbox הופך למשאב ש-agent משתמש בו לפי הצורך, ולא למקום שבו הוא חי. מזה נוצר גם eval set גדול

מתוך ה-traces. ניסיון לשיפור עצמי אוטומטי עליו נתן תוצאות מעורבות. Regan מתאר "main character syndrome where the agents are making fixes that are, you know, only seeing their piece of the elephant effectively", ולכן human in the loop עדיין הכרחי. [52:41]

3. Give them the ability to provision the compute that they need

- By default, QM uses a sandbox with durable disk on a per-channel basis. One for each user, one for each slack channel
- However, it can refer to named environments and create new ones on an as-needed basis (eg for heavier dev workloads that require more RAM)
- One YC company uses this to allow researchers to spin up jobs on GPU clusters



השקף "Give them the ability to provision the compute that they need" השוואה בין OpenClaw (agent אחד לאדם, silo), צי של Hermes (N עותקים בלי שיתוף) ו-QM core אחד שמשרת כל scope, עם shared brain ו-sandboxes לפי scope

- **לחבר את ה-agent לכל המשאבים:** CLI פנימי קיים, keychain ברמת הארגון ו-keychain אישי (כולל device-OAuth code עם רענון). קריאות מותרות לפי ACL, וכתובה למסד הנתונים עוברת דרך bulk upsert שאדם סוקר. הם מודים: "[54:19] we've started just kind of rubber stamping these".
- **לתת ל-agent לבחור compute:** כברירת מחדל יש sandbox עם דיסק קבוע לכל משתמש או ערוץ, אבל ה-agent יכול לבחור מכונה חזקה יותר לעבודת dev כבדה. חברה מ-YC משתמשת בזה להרצת jobs על GPU clusters.
- **לא לנעול ספק:** החלפה תוך כדי ריצה בין ספקי מודלים (בשקף: אם Fable מסרב, עוברים ל-GLM 5.3) ובין ספקי sandbox באמצעות routing table. הארגון הוא הבעלים של הדאטה, כלומר ההיסטוריה, ה-skills וה-connectors.
- **harness דק:** Pi ועוד שלושה כלים: הרצה ב-sandbox מרוחק, קריאה וכתובה ל-object storage ופרסום אפליקציות פנימיות. כלי memory ו-cron נחשבים scaffolding זמני. בלשונם, "keep the harness extremely thin". [55:50]
- **grind tool:** "they often give up way too early" [57:22]. הפתרון הוא תקציב למטרה, כלומר זמן ריצה (למשל כמה שעות) או הוצאת טוקנים, שלפניו ה-agent לא רשאי לוותר. לדבריהם זה משפר דוחות ותוצרי מחקר.
- **Post-training מתנגש עם ה-agents harness:** שוכחים מה נאמר להם על הסביבה (multiplayer ו-Slack) גם כשזה כתוב ב-system prompt, ולכן נדרשים רמזים מקומיים בסביבה עצמה.

Josh France / Regan Bell [58:57] And so the information that you can put in the brain is — effectively bounded by how good your permission system is

6. מה לעשות עכשיו

- **לבדוק מה ה-harness מסתיר מהמודל:** לעבור על הרשימה של Seth (compaction, REPL, subagents) פרוגרמטיים, גישה ל-state, feedback) ולבדוק איזו יכולת חסרה ב-agent שלכם.
- **למדוד עקומה, לא נקודה:** להשוות harnesses בתקציב קבוע ולחפש את ה-practical plateau, ולא לעצור כשאחד מהם "נגמר".

- **sandboxing לפני שחוגגים:** תוצאה של 99.9% יכולה להיות רמאות. כדאי לקרוא את ה-traces בכל eval לפני שמסיקים מסקנות.
- **להוציא את ה-state מה-sandbox:** שיחות, זיכרון ו-queue צריכים להיות במאגר מרכזי, וה-sandbox צריך להיות משאב שה-agent בוחר. זה עדיף על צי של VMs שכל אחד מהם silo.
- **להשתמש בענן לאופטימיזציה ולהריץ מקומית:** עבור משימות חוזרות, כדאי לבדוק spec search עם מודל frontier כ-teacher ו-stack מקומי בזמן ההרצה.
- **להשקיע בהרשאות לפני שמשקיעים בידע:** לפי QM, כמות הקונטקסט שאפשר לתת ל-agent מוגבלת לפי מערכת ההרשאות. בלי ACL מפורט, ידע מסווג ידלוף.
- **להגדיר תקציבים למטרות:** זמן מינימלי או כמות טוקנים מינימלית לפני שמותר לוותר, במשימות מחקר ודוחות.

שאלה פתוחה	למה זה חשוב
כמה מהשיפור ב-ARC-AGI-3 נובע מה-harness וכמה מה-system prompt שנלקח מ-leaderboard קהילתי?	Seth לקח prompt ייעודי ל-ARC, ולכן הטענה ל" harness כלילי" חלקית.
האם self-improving harness עובד בפרודקשן בלי אדם בלולאה?	QM מדווחים על "mixed results" ועל main character syndrome.
מתי rubber stamping על כתיבות של agent הופך לסיכון?	הסקירה האנושית נשחקת עם הזמן, ו-YC אומרים שהם עוקבים אחרי זה.
מה המדיניות כש-agent עובר למודל אחר כדי לעקוף סירוב?	היכולת הוצגה כיתרון בלי דיון ב-governance.
על אילו משימות מודל מקומי עדיין לא מספיק?	OpenJarvis הציגו ממוצעים, ולא את הזנב של המשימות שנכשלות.
איך מודדים harness במשימות של כמה ימים כשהשונות גבוהה?	ב-nanoGPT לא ניתן היה לייחס את השיפור ל-harness.

7. מילון מונחים

מונח	משמעות
Harness	שכבת הקוד בין המודל לעולם: לולאה, כלים, זיכרון, subagents וסביבת הרצה.
Scaffolding	כינוי מזלזל ל-harness, כ"פיגום" זמני מסביב למודל.
ICL (In-Context Learning)	למידה מדוגמאות בתוך הקונטקסט, בלי לעדכן משקלים.
LoRA / SFT	שיטות fine-tuning: עדכון בדרגה נמוכה, ואימון מפקח מלא.
Skills	רצפי פעולות או הוראות שמורים (למשל skills.md) שה-agent טוען לפי הצורך.
RLM	Recursive Language Model: מודל שקורא לעצמו רקורסיבית מתוך REPL כדי לפרק בעיות.
REPL	סביבת הרצת קוד אינטראקטיבית (IPython) שבה משתנים נשמרים בין צעדים.
Subagent	agent משני שה-agent הראשי מפעיל למשימה ממוקדת. ב-Prime Agent הוא נשאר פעיל לאורך זמן.
Compaction	סיכום של היסטוריית הקונטקסט כדי לעבוד מעבר לגבול חלון הקונטקסט.
Refinement	עדכון ומחיקה של prompts, skills ו-memory מתוך trajectories קודמות.
Meta-harness	harness שתפקידו לייצר harnesses ולשפר אותם.
Long-horizon	משימות של שעות עד ימים שבהן ה-agent עובד בלי השגחה.
Spec search	חיפוש קונפיגורציה ל-stack מקומי שמודל ענן מנחה (OpenJarvis).
Sandbox	סביבת הרצה מבודדת שה-agent מריץ בה קוד ופקודות.
Bulk upsert	כתיבה מרוכזת למסד נתונים. ב-QM היא עוברת סקירה אנושית לפני הביצוע.
Unhobbling	הסרת מגבלות מלאכותיות כדי לנצל יכולת שכבר קיימת במודל.
RHAE	מדד הציון שמופיע בשקפי ARC-AGI-3. ההגדרה שלו לא הוסברה בהרצאה.