

A SYNTHESIS OF 1,110 CONFERENCE TALKS

The AI Stack

2026

Contents

Preface	7
HOW THIS WAS MADE	
I The Centre of Gravity	11
ORIENTATION	
II Not One Model, a Portfolio	17
MODELS & TRAINING	
III More Context Is Not Better Context	23
CONTEXT & RETRIEVAL	
IV The Harness Is the Product	31
AGENTS	
V Benchmarks Measure the Wrong Thing	37
EVALUATION	
VI The Unit of Work Changed	43
SOFTWARE ENGINEERING	
VII Serving, Voice and Security	49
INFRASTRUCTURE	
VIII MCP and the Interoperability Argument	55
PROTOCOLS	
IX What Leaders Actually Ask	61
ECONOMICS & ORGANISATION	
X Where Speakers Disagree	65
OPEN PROBLEMS	
XI Nobody Has Solved These	69
OPEN PROBLEMS	
The Twenty-Four Clusters	75
APPENDIX A	
The Nine Missing Transcripts	77
APPENDIX B	
How the Numbers Were Counted	79
APPENDIX C	
Notes	81
SOURCES FOR EVERY QUOTATION	

ORIENTATION

I

The Centre of Gravity

Start with the shape of the corpus, because the shape is the first finding and it is the one most likely to be surprising. Across 1,110 talks and 4.96 million words, the word **agent** and its variants appear 24,296 times, in 968 of the 1,110 talks. That is 87 per cent of the corpus. Whatever else is true about the state of AI engineering, speakers have stopped discussing whether agents are a real category and moved on to arguing about how to build them.

But the second-most common theme is not technical at all. Mentions of teams, engineers, hiring and productivity outnumber every model-side topic in the corpus, appearing across 936 of 1,110 talks. The third is evaluation, at 587 talks. Context and retrieval, which absorbed most of the engineering conversation in 2024, now appears in 474 talks — still enormous, no longer where the gravity is. Coding sits at 615 talks by reach, though with a lower mention count than its reach suggests, which tells you something about how speakers use the word: they describe software work constantly and the word **coding** comparatively rarely.

CONTEXT & RETRIEVAL

III

More Context Is Not Better Context

This is the most repeated practical finding in the corpus, and it is the one most likely to be counter-intuitive, because it runs against a decade of scaling intuitions. The finding is not that large context windows are useless. It is that the useful quantity is not the size of the window but the quality of what is in it, and that adding material to a window has costs that do not appear until you measure them.

Two hundred and five talks discuss context windows explicitly. When they do, they overwhelmingly describe a degradation effect rather than a capability one. The mechanism described is accumulation: old tool outputs, stale retrieved chunks, superseded conversation turns. Nothing is removed, so everything is still present, and the model is left managing a pile of history that increasingly resembles an archive of every mistake made so far.

EVALUATION



Benchmarks Measure the Wrong Thing

Five hundred and eighty-seven talks discuss evaluation or benchmarking. That is more than half the corpus, and the density is recent: the vocabulary of **trace**, **judge**, **regression** and **golden set** belongs to a discipline that barely existed as a named thing two years ago. The finding that organises this chapter is not that evaluation is hard. It is that the dominant evaluation methodology of the last several years measures something systematically different from what practitioners need.

The gap

“¹⁴ Scores 88% on Swebench Pro, but only 29% on the hardest slice of, uh, Frontier code. So, the same model on the same surface with the same job does 51 points less uh, less well if you're asking not does it pass the tests, but whether or not uh, you would actually merge the result.

Laurie Voss & Arize AI, “The Death of the Code Review: What the Data Actually Says”, 8:37

INFRASTRUCTURE

VII

Serving, Voice and Security

The unglamorous layer decides whether the rest of the stack works, and the corpus is blunt about this: every problem above it becomes a latency and cost problem at this layer, and none of it is visible from the application layer. Three themes dominate — the economics of serving, the engineering of real-time voice, and security as a discipline that had to be invented for this problem rather than inherited from it.

Serving economics

The argument in Chapter II about inference cost has a structural consequence that speakers return to repeatedly: the difference between an application that works and one that has margins is usually a dedicated inference endpoint rather than a shared API, and the decision of when to cross that line is a business judgement dressed as an engineering one.

“
19 *If you're doing a coding agent, you want to make sure it's able to write good code, it's able to debug, and it's able to recover from its errors. If you do have bottlenecks, that's probably not going to be fatal, but they will multiply your cost and latency. So, in that case, we normally just reduce the scope, simplify the task, and try again.*

“How We Build Effective Agents: Barry Zhang, Anthropic”, 4:09

IX

What Leaders Actually Ask

Seven hundred and thirteen mentions of teams, engineers, hiring and productivity, across 936 talks. By reach this is the second-largest theme in the corpus, and it is the one least connected to the engineering chapters. A distinct cluster of 52 talks and seventeen hours sits apart from the rest, with top terms **business, roi, impact, teams, organization, productivity, customer**. What that cluster discusses is not how to build the thing but whether it was worth building.

Perception is not measurement

The most repeated finding in the organisational cluster is that self-reported productivity and measured productivity diverge sharply, and that the divergence is not random noise — it runs in a consistent direction. The people closest to the work feel the speedup most acutely; the people downstream of it, reading the output, see less. The corpus's recommendation is unambiguous and unpopular: measure, do not survey.

OPEN PROBLEMS

XI

Nobody Has Solved These

The final chapter is the shortest and the least comfortable. A corpus of a thousand talks on a subject this new contains, in aggregate, a clear and repeated account of what is not solved. These are not speculative research questions; they are the things speakers say, with some frequency, that they cannot do.

Long-horizon autonomy

“
26 *The two big takeaways are first, long horizons we is still unsolved. The best agents only at 26%. There's plenty of headroom left. Second, the big bottleneck is robust verification.*

Rishi Desai & Abundant AI, “SWE-Marathon: Evaluating Coding Agents at Billion-Token Scale”, 11:48

Read that as a measurement rather than an opinion. The best agents on the hardest long-horizon coding tasks reach roughly a quarter success, and the two bottlenecks named are long horizons themselves and robust verification. Neither is a scaling problem. A model that is twice as capable does not automatically hold a thread twice as long, and the failure of long tasks is rarely a sudden collapse — it is a gradual loss of the thread somewhere in the middle, after which the remaining steps are technically executed and semantically pointless.