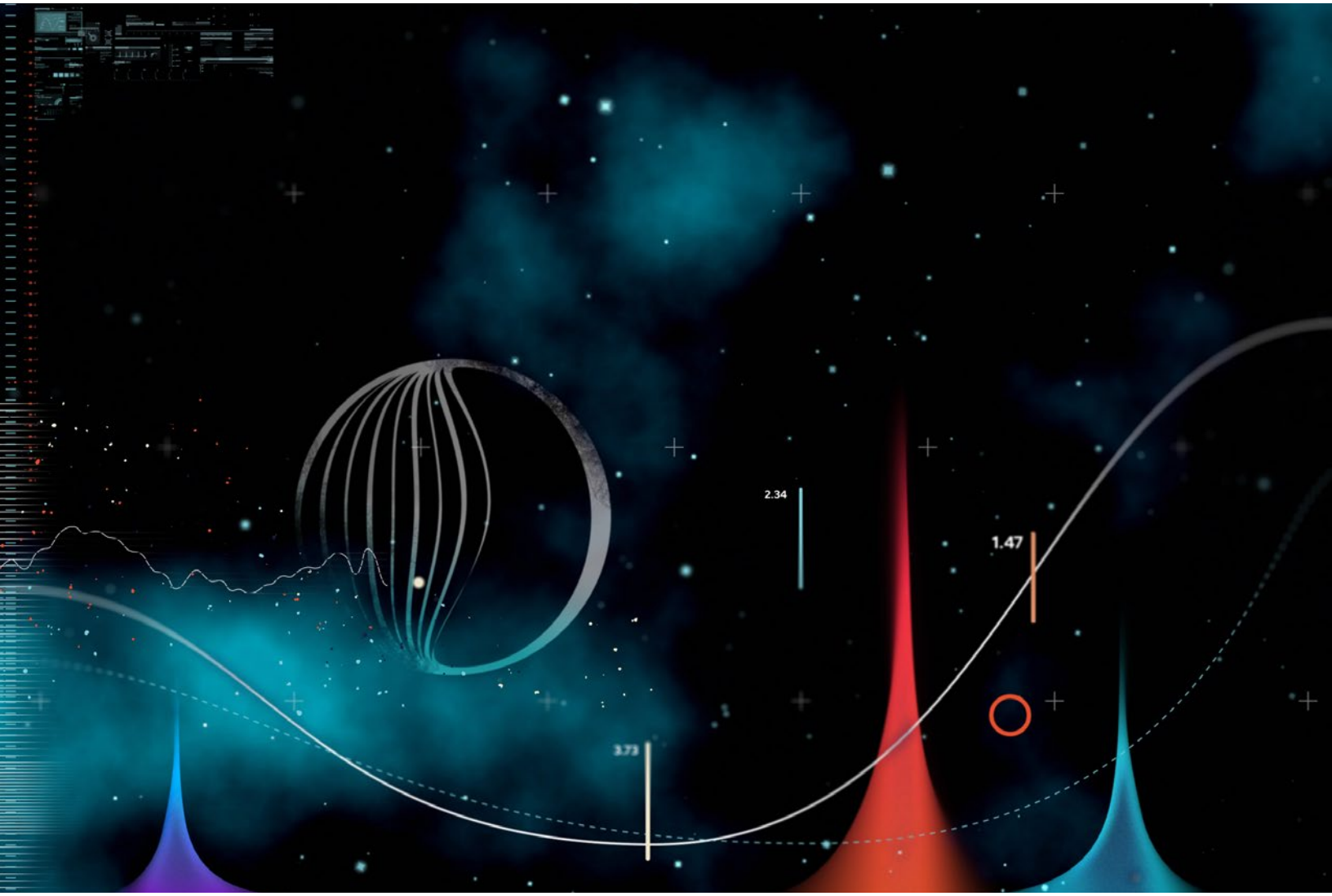
signal65
PINNACLE



Signal65 PINNACLE
Measuring Correct Work

A Methodology for Benchmarking Agentic AI in the Enterprise

v1.1

AUTHORS

Ryan Shrout
President & GM | Signal65

JV Roig
Senior Generative AI Platform Engineer | Kamiwaza

IN PARTNERSHIP WITH

KAMI WAZA

AUGUST 2026

1. Executive Summary

Enterprises are deploying agents that run for 30 to 80 tool use rounds, read policy documents, apply governing rules, query databases, and produce a set of deliverables that someone downstream depends on. The metrics available to evaluate those deployments each answer half the question. Capability leaderboards measure whether a model can answer questions. Infrastructure benchmarks measure how many tokens a platform can move. Neither tells a buyer whether the work came out right.

Signal65 PINNACLE measures correct work. It runs multi step enterprise jobs inside environments generated fresh for every run, grades the result in code against a generated answer key at the moment the environment was built, and reports quality and capacity as separate numbers that a reader can interpret independently and recombine deliberately.

1.1 What the method does differently

- **Generated environments**. Every run regenerates the sandbox. Entities, values, dates, file layouts, database contents, document text, and distractors are all drawn independently, and the resulting space runs to roughly 10 to the 80th power distinct instantiations. There is nothing static to memorize and nothing fixed to optimize toward, so contamination resistance comes from how the suite is built rather than from a disclosure policy.

- **Deterministic scoring, despite that randomness.** No model judges the output and no human rates it. The generator knows the correct answer because it generated it at the moment it built the environment, and the grader is code. A freshly drawn sandbox and a repeatable score are usually in tension. Generating the answer alongside the environment gets both, which converts benchmarking from a claim about a panel into a procedure a third party can run.

- **Real agents against a real tool surface.** Nothing here is a scripted replay or a reasoning question about work. Agents execute against a live filesystem, a Python runtime, and a live database, choosing their own path across 30 to 80 tool use rounds and deciding for themselves when the job is done. A separate retrieval component measures grounding, aggregation, and fabrication over long context, because agentic execution and faithful retrieval fail in different ways and enterprise work needs both.

- **Occupational grounding.** Task design follows an external occupational taxonomy published by the US Department of Labor rather than a set of scenarios Signal65 invented. What the work consists of is inherited. Which parts of that work can be simulated and scored is a Signal65 determination, made through two stated gates that chapter 6 sets out in full.

- **Two environments, not a difficulty dial.** Every job runs twice, once against data as an enterprise is likely to have accumulated it and once against data an enterprise has already organized. The difference between the two scores is a very useful number, because it tells a buyer whether a model can be trusted with the estate they have or only with the estate they wish they had.

© 2026 Signal65 PINNACLE. All rights reserved.

- **A quantified fabrication rate.** Every model carries a published rate for how often it answers a question the source material does not answer. Measuring that requires knowing with certainty that a fact is absent, which is only possible for a party that generated the corpus, so the axis is unavailable to any benchmark built on collected or recorded material.
- **Three lenses on one measurement set.** Model intelligence, silicon capacity, and solution economics come from the same runs rather than from three unrelated exercises, which is what makes cost per correct task a measured quantity rather than a modelled one.
- **A separate test for coding agents.** Coding agents run their own jobs, engineering a generator inside a codebase they have never seen, and are graded by running what they build on seeds chosen after the attempt ends. The result is a separate PINNACLE Coding Agent Score that sits beside the Model Score and does not change it. Chapter 21 sets it out in full.

1.2 The headline numbers

Two figures carry the headline. The PINNACLE Model Score reports quality, as a persona weighted composite of separately measured capabilities indexed so the reference model scores 1000. PINNACLE Intelligence Throughput reports productivity, combining that quality score with sustained platform speed weighted three to one, and compounding per task success across workflow depth.

A third figure reports coding. The PINNACLE Coding Agent Score is indexed to the same reference model at 1,000 and measured on its own jobs, and it is published beside the other two and folded into neither.

Everything underneath both is published as well. Tasks per hour, agents sustained at service level, token efficiency, fabrication rate, cost per correct task, and the utilization point where owning infrastructure becomes cheaper than renting it. Any reader who rejects the weighting can rebuild either headline from the parts.

1.3 What it is not for

Signal65 PINNACLE measures the work enterprises are actually deploying agents to do. That focus is the source of its value, because whether an agent finishes a real job correctly on real data is the measurement that decides whether an agentic program returns anything at all, and it is the measurement the industry has least of.

It is not built to rank depth of world knowledge, craft of prose, or progress toward General Artificial Intelligence. Other benchmarks attempt to answer those questions and a buyer with those questions should use them. One consequence follows and is worth knowing in advance. Rankings here will not match rankings elsewhere, and a smaller model can lead this table while sitting well down a general capability leaderboard. Section 2.5 sets out the boundary in full.

1.4 The finding this method makes available

Every job in the suite is run in two different enterprise conditions, sampled many times in each. Governed data means an estate already organized, classified, and documented. As-found data means duplicated records, half finished migrations, naming conventions that changed twice, and rules that contradict each other. The result that does not exist anywhere else is the difference between those two scores. A model can hold up on organized data and fall apart on the same job when the data is messy. That gap is a per model property, it is wide, and it maps directly onto a procurement decision.

> **A model that scores well under controlled conditions and collapses on messy data has a scope problem you can engineer around. A model that scores poorly under both has a capability problem you cannot.**

© 2026 Signal65 PINNACLE. All rights reserved.

2. The Measurement Gap

Two families of numbers dominate how agentic AI gets evaluated today. Both are legitimate. Both are carefully produced. Neither was built to answer the question an enterprise is actually asking, and the enterprise is left doing the translation itself.

2.1 What capability leaderboards measure

Capability leaderboards measure whether a model can produce a correct answer to a question posed in isolation. They are useful for tracking the frontier and they are the reason anyone can compare model families at all. Three properties limit what they can say about a deployment.

They are static, which means a fixed question set exists in the world and eventually finds its way into training data. Contamination is not hypothetical. When Microsoft Research rebuilt a widely used academic benchmark on contamination free lines, a leading proprietary model scored 73.4% against 88.0% on the original, a gap of 14.6 points between what a model remembers and what it can do. Zhao et al., MMLU-CF, arXiv 2412.15194.

They are single shot, which means they measure the answer rather than the work. An enterprise job is not one question. It is thirty to eighty rounds of reading, tool use, revision, and output, where an error in round nine propagates silently into every deliverable after it and the model never sees the correction.

And they are graded by judgment, either a human panel or another model. Judgment introduces a rater that has to be trusted, calibrated, and paid for, and the agreement rates published by the benchmarks that use it are lower than most readers assume.

2.2 What infrastructure benchmarks measure

Infrastructure benchmarks measure how much a serving stack can move. Throughput, interactivity, time to first token, performance per dollar, tokens per watt. These are real engineering measurements and the discipline behind the best of them is genuine. They are also the correct instrument for the question they answer, which is how expensive it is to run a given amount of traffic.

A correctness dimension is absent from all of them, and it is absent for a structural reason rather than an oversight. A benchmark built on recorded sessions can measure throughput because throughput is observable in a replay. It cannot measure quality, because a recording contains what happened rather than what should have happened. There is no answer key inside a trace to check an output against. That argument is developed in full in chapter 16.

2.3 The translation problem

The question an enterprise is actually asking is narrow and concrete. Which model and hardware stack will run my workload correctly, at the volume I need, within my budget. Every benchmark hands that buyer a score and leaves them to work out what it means, and a buyer holding a capability rank and a throughput curve still has to answer three questions that neither number addresses.

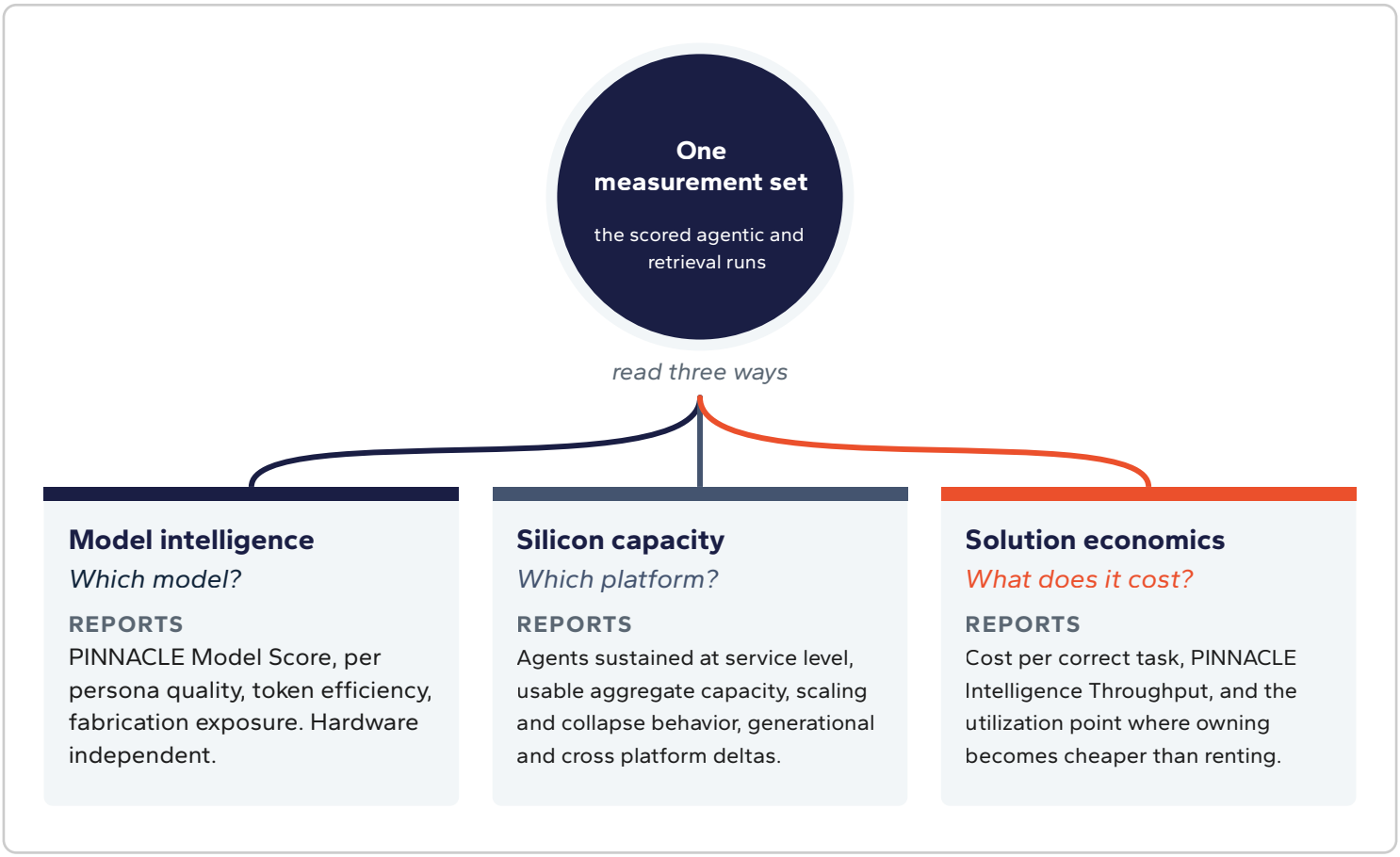
1. Will this model complete our jobs correctly on our data, which is not clean.
2. How many agents can this platform actually run at once while every one of them stays useful.
3. What does one correct outcome cost, which is a different quantity from what one million tokens cost.

© 2026 Signal65 PINNACLE. All rights reserved.

The third question is where the translation breaks down completely. Cost per token is knowable from a serving benchmark. Cost per correct task requires a correctness measurement taken on the same runs, and if quality and throughput are measured by two different parties on two different workloads, no honest arithmetic connects them.

2.4 The three lenses

PINNACLE is organized around the three questions above rather than around the instruments that produce the numbers. The same measurement set is read through three lenses.



One measurement set, read three ways. The lenses share an origin, which is what makes cost per correct task a measured quantity rather than a modelled one.

2.5 What this benchmark does not measure

A score is only interpretable if a reader knows what question the instrument was built to answer. Signal65 PINNACLE was built to measure whether an agent completes enterprise work correctly, at what capacity, and at what cost per correct outcome. It was not built to track progress toward General Artificial Intelligence. That is a scope decision made at the outset.

© 2026 Signal65 PINNACLE. All rights reserved.

Five things sit outside it.

- **Depth of general or specialist knowledge.** Whether a model knows molecular biology, international law, or quantum chemistry to a research standard is a question other benchmarks attempt to answer, and answer well. Scenario data here is generated, so industry specific knowledge of the world outside the sandbox earns a model nothing.
- **Quality of written prose.** Style, voice, structure, and persuasiveness are not scored. Fidelity to fact is scored, and the two should not be confused. A model that writes plainly and reports the correct figure outscores one that writes beautifully and reports the wrong one.
- **Creative and multimodal generation.** No image or video generation, no audio, and no assessment of visual design. Agents in the suite work with documents and structured data and are judged on the artifacts they produce from them.
- **Progress toward General Artificial Intelligence.** Nothing in this suite is calibrated to that question. A model does not score higher for reasoning like a doctoral candidate. It scores higher for finishing an enterprise job correctly.
- **Conversational range and open dialogue behavior.** Agents here are given a job and judged on the artifacts they produce, not on how they converse along the way.

The same argument runs through recent work from NVIDIA Research, Small Language Models are the Future of Agentic AI. Agentic systems put models to work performing a small number of specialized tasks repeatedly and with little variation, and for many of those invocations a small model is sufficiently capable and materially more economical, with systems calling several different models where broad conversational ability genuinely matters. An enterprise decomposing a workflow into orchestrated sub tasks does not need one model that is simultaneously the best physician, attorney, and physicist available. It needs each step done correctly, at a cost and a latency it can carry.

Where a buyer needs general conversational range or frontier reasoning, a general benchmark is the right instrument and this one is not. Where a buyer needs to know whether an agent finishes the job correctly against their data, at what concurrency, and at what cost per correct outcome, this is the instrument built for that question.



3. Foundations

Four published papers already specify most of what a reader needs in order to check the mechanics of the suite. Each is summarized below and linked in full.

3.1 The execution harness

PICARD is the harness underneath both component benchmarks. It generates the sandbox, creates the ground truth alongside it and holds it outside the sandbox, exposes the tool surface, and grades the result. The contamination resistance argument and the deterministic scoring path are specified in the published framework documentation, PICARD, Testing What Models Can Do, Not What They Have Seen, and are summarized rather than re-derived in chapter 5.

© 2026 Signal65 PINNACLE. All rights reserved.

3.2 The agentic suite

KAMI is the agentic component. Two published rounds of results cover more than seventy models across open and proprietary families, round one on open models and round two on proprietary models. The formal specification and the full first round dataset are published separately, arXiv 2511.08042. The second round established a finding that motivates this entire program. Rank on traditional benchmarks predicts rank on agentic work poorly. Newer models within a family regularly scored below their predecessors on agentic tasks while scoring higher on conventional evaluations, which is what a field optimizing against the wrong instrument looks like from the outside.

3.3 The retrieval work

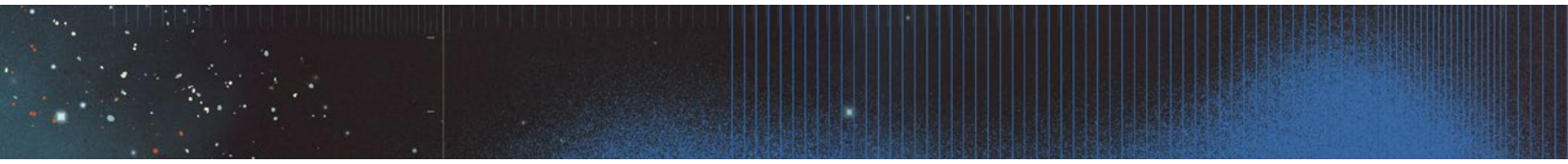
RIKER is the retrieval component. The published work defines grounding, aggregation, and fabrication resistance, measures each at 32K, 128K, and 200K context, and covers ninety one models at the shortest context length. Signal65 PINNACLE takes all three measures at 128K, for the reasons set out in section 10.2. Two findings from that work carry directly into this paper. Advertised context length is not a proxy for usable retrieval, and aggregation across documents degrades roughly twice as fast as single document extraction as context grows. The metric definitions used in chapter 10 are the published ones.

3.4 Why two component benchmarks rather than one

The two measure different faculties with non-overlapping failure modes, and enterprise work requires both. Almost every deployed agent both acts across many steps against a live tool surface and reasons over retrieved material it did not choose, so a model strong at one and weak at the other fails in production while looking competent on any benchmark that measures only its stronger half. The published retrieval data makes the point, since grounding and fabrication resistance turn out to be independent. The strongest single document retriever in the published roster at 128K context invents an answer on better than one unanswerable question in four, and no ranking built on retrieval accuracy would show it.

3.5 The coding work

The Coding Agent test is built on PICARD and developed in close partnership with Kamiwaza. It uses the engine in a different role, as the codebase a coding agent has to learn and extend, where the enterprise agents use it as their working environment, and it inherits the deterministic grading path described in chapter 5. Chapter 21 describes the test, its two kinds of job and its score.



4. Design Principles

Four commitments were fixed before any implementation detail, and every architectural decision in the chapters that follow traces back to one of them.

4.1 Deterministic scoring

No model judges, no human raters, no scoring drift. Grading is code comparing an output against a ground truth value that the generator wrote before the agent started. This commitment is expensive in coverage rather than in money. Anything whose correctness is a matter of opinion falls outside the suite entirely, which rules out grading open ended prose and everything else that needs a judgment call, and chapter 19 sets out what that excludes. What it buys is a score that does not move when the grader is replaced, the reviewer changes, or the vendor disagrees. A result produced this way cannot be argued into a better number.

© 2026 Signal65 PINNACLE. All rights reserved.

4.2 Contamination and optimization immunity by construction

Nothing is fixed, so there is nothing to train on and nothing to tune toward. Entities, values, file layouts, database contents, document text, and the distractors are all regenerated per run from a configuration space large enough that no two runs repeat. A model that trained on last week results gains nothing this week. This is a structural property of the generator rather than a promise about disclosure practices, which matters because the promise version has repeatedly failed in this field.

4.3 Occupational grounding

The work is defined by an external taxonomy rather than invented by the benchmark author. Task design follows the Generalized Work Activities published in the US Department of Labor occupational database, which means the answer to why these tasks and not others does not rest on Signal65 taste. Chapter 6 sets out exactly how much of that taxonomy is covered, how the covered set was chosen, and which parts are excluded and why.

4.4 Gradable live simulation

Agents run live against a real tool surface and choose their own path, and the answer key exists before the agent starts. Both halves matter. Live execution is what allows a model to take more rounds, recover from its own mistakes, or fail in a way a scripted replay would never surface. A generated answer key is what makes the outcome gradable at all. The design principle is not simulation for its own sake. It is that measuring correct work requires generating the work.



5. PICARD, the Execution Harness

PICARD builds the environment, generates the correct answer alongside it and holds it outside the agent sandbox, hands an agent a live tool surface, and verifies what comes back.

5.1 Sandbox generation and the configuration space

A scenario specification describes the shape of a job rather than an instance of it. At run time the harness instantiates that specification into a working sandbox. Entity names, numeric values, dates, directory structures, file counts, document bodies, database schemas and contents, and the specific distractors are all drawn per run.

A space that size is reachable because those draws are independent, so the dimensions multiply rather than add. No individual dimension has to be large. A few dozen independently drawn fields, each with a few thousand possible values, produces a space on the order of 10 to the 80th distinct instantiations without any one of them being remarkable.

The number that matters operationally is not the size of the space but the probability that any two runs land on the same instance. For a space of size N and r draws, that probability is approximately r squared over 2N. If every organization evaluating models ran a trillion scenarios between them, the chance that any two of those runs drew the same sandbox would sit below 10 to the negative 56th. Repetition is not unlikely in the ordinary sense. It does not occur.

Two things follow from that. Memorization has nothing to attach to, because the answer key for a given instance did not exist before that instance was generated, so a model cannot have encountered it during training regardless of

© 2026 Signal65 PINNACLE. All rights reserved.

what it trained on. And the object being estimated changes. A fixed test set yields a point score that can be overfit and that carries no sampling error, since running it again returns the same questions. Drawing from a distribution yields an estimate of expected performance across the scenario population, with a sampling error that is quantifiable and that narrows as samples accumulate. Overfitting is not merely discouraged in that setting. There is no fixed target to overfit against.

Two organizations running the same scenario specification on the same model will see the same score to within sampling variance, and neither will have seen the same sandbox. Reproducibility and non-repeatability are usually in tension. Generating from a specification gets both.

5.2 Ground truth generation

The correct answer is created at the same moment as the environment, by the same process, from the same draw. When the generator writes a database containing four hundred orders across three regions, it already holds the value that a correct regional revenue query returns, because it computed that value in order to write the rows. When it writes a policy handbook with a governing rule and then scatters records that the rule applies to, it holds the correct disposition of every record.

This is the inversion that makes deterministic grading possible. A conventional benchmark starts with documents and works backward to answers, which requires annotation and eventually requires judgment. Starting from the answer and generating the environment around it means the answer key is a byproduct of construction rather than a separate artifact that has to be produced and trusted.

5.3 Verification, and why no model appears in the grading path

Grading is code. A scoring type is attached to every deliverable and describes exactly what correct means for that artifact. File existence at a specified path. JSON key presence and value match. Numeric equality against a computed target. Structured content match against a generated reference. The grader reads the artifact the agent produced, applies the check, and returns a boolean.

No language model appears anywhere in that path, which removes an entire category of dispute. There is no rater agreement rate to report, no judge model version to disclose, and no drift when the judge is upgraded. It also removes a category of capability, since anything requiring qualitative assessment is out of scope by construction. Chapter 19 treats that as the limitation it is.

5.4 The tool surface and the round cap

An agent working a scenario has filesystem operations, Python execution, and SQL query access against the generated sandbox. It reads what it wants, in the order it wants, and decides for itself when the job is finished. Nothing constrains the path.

Sessions run against a cap of 100 tool use rounds. Observed usage on the current scenario set runs approximately 30 to 80 rounds per engagement, so the cap functions as a termination guard rather than as a binding constraint on well behaved runs. A session that hits the cap is scored on what it produced, which is normally a failure, and the round count is retained because a model that needs eighty rounds where another needs thirty is telling you something a pass or fail cannot.

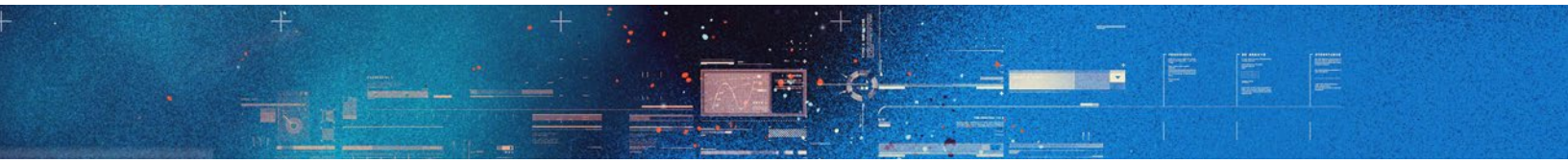
© 2026 Signal65 PINNACLE. All rights reserved.

5.5 Run to run variation

Because each run draws a fresh instance, individual runs vary. That variance is handled by sampling rather than by suppression. Every scenario is sampled 40 times per model, organized as 8 independent runs of 5 samples each. The split into 8 runs is a parallelization for compute efficiency and carries no methodological weight, since every sample regenerates its own environment from the specification and a sample drawn in one run is equivalent to a sample drawn in another.

Across the 7 scenarios in the suite that gives 280 scored samples per model. Those counts do not vary. Every model in every published result is measured on the identical sample count and the identical test suite, so no comparison in this paper rests on one model having been given more attempts than another.

Variance is also informative in its own right. A model whose score swings widely across draws of the same scenario is not equivalent to a model that scores the same mean consistently, and the two should not be presented as though they were. Where the spread is material it is reported alongside the mean.



6. Occupational Grounding

Every benchmark has to answer why these tasks and not others. Most answer it with taste. Signal65 PINNACLE answers it by inheritance.

6.1 The taxonomy

The US Department of Labor maintains an occupational database that decomposes work into a hierarchy. At the top sit 41 Generalized Work Activities, broad categories such as analyzing data, documenting information, and interacting with computers. Beneath them are Intermediate Work Activities, and beneath those, Detailed Work Activities that describe concrete acts performed on the job. Occupational analysts built and maintain that structure.

Task design starts from the Generalized Work Activities and reads downward through the detailed activities beneath them to understand what an activity actually consists of in practice. That reading is what a scenario is designed against.

6.2 The two gates

Two gates were applied to all 41, in order, on a plain reading of each activity and its detailed components.

Gate	The question it asks, and what it removes
Gate one	Can an agent do it at all. Removes activities requiring a physical body, and activities that depend on standing conferred by other people rather than on any act the agent performs.
Gate two	Can Signal65 PINNACLE score it deterministically. Removes activities whose output cannot be simulated and graded in code, because assessing whether the work was done correctly would require a judgment call rather than a comparison against a generated answer key.

Figure 1: *The two selection gates.*

© 2026 Signal65 PINNACLE. All rights reserved.

The second gate is a statement about the instrument rather than about agents, and separating the two is the point of running them separately. An activity removed by gate two is one a current agent may well perform competently. The suite simply cannot prove it either way without introducing a judge, and introducing a judge would cost the property that makes every other number in this paper checkable.

6.3 The result

Twenty-six of the 41 Generalized Work Activities clear both gates and are modeled into the suite. That covered set is named the **Signal65 Agent-Testable Work Activities**. Within it, eighteen are exercised directly by scenario design and eight are exercised indirectly, appearing as accompanying work products that a scenario requires in order to be complete rather than as the object of the scenario itself.

Group	Count	Disposition
Modeled into the suite	26	The Signal65 Agent-Testable Work Activities. Eighteen exercised directly, eight through accompanying work products.
Agent cannot perform	11	Seven require embodiment. Four require standing conferred by other people.
Signal65 cannot score	4	An agent can perform these. The outcome lives in another person response, so there is no deterministic answer key.
Total	41	The full set of Generalized Work Activities.

Figure 2: *Disposition of all 41 Generalized Work Activities. Measured against the published taxonomy, classified by Signal65.*

6.4 What the excluded activities have in common

The fifteen excluded activities fall into three groups. Some need a physical body. Some need actual interpersonal standing, meaning a role conferred by other people rather than any act the agent performs, which is why a model can advise a chief executive and cannot be one. The rest are activities our current infrastructure cannot yet simulate and score.

Only the third group is waiting on engineering. The first two are not capability gaps that scale will close, which matters for anyone sizing agentic deployment against a workforce.

6.5 What the four unscoreable activities cost us

Four activities pass gate one and fail gate two. An agent can perform them. The outcome lives in how another person responds, which means there is no artifact to check against a generated answer key. Persuading, negotiating, resolving conflict, and providing consultation all have this shape, and their absence from the covered set is a limitation of the instrument that Signal65 chose to accept rather than paper over with a model judge.

A benchmark that reports only the activities it happens to cover well, and lets a reader infer that the rest were unimportant, has told a true set of facts and left a false impression.

© 2026 Signal65 PINNACLE. All rights reserved.

7. Scenario Architecture

Every metric in this paper is computed over the artifact a scenario produces. What follows is a single job from generation through agent execution to grading.

7.1 Anatomy of a scenario

A scenario is not a question. It is a working session with four components.

- **Policy handbooks to interpret.** Documents stating how the organization does things, written the way internal documents are actually written, which is to say at length and with the operative rule not necessarily in the obvious place.
- **Governing rules to apply.** Explicit constraints that determine correct disposition, which the agent has to locate, read correctly, and apply to every record it touches.
- **Scattered source data.** The facts needed to do the job, distributed across files, directories, and databases rather than handed over in one place.
- **A full set of deliverables.** Several artifacts, each with a specified location and format, which together constitute the job being done.

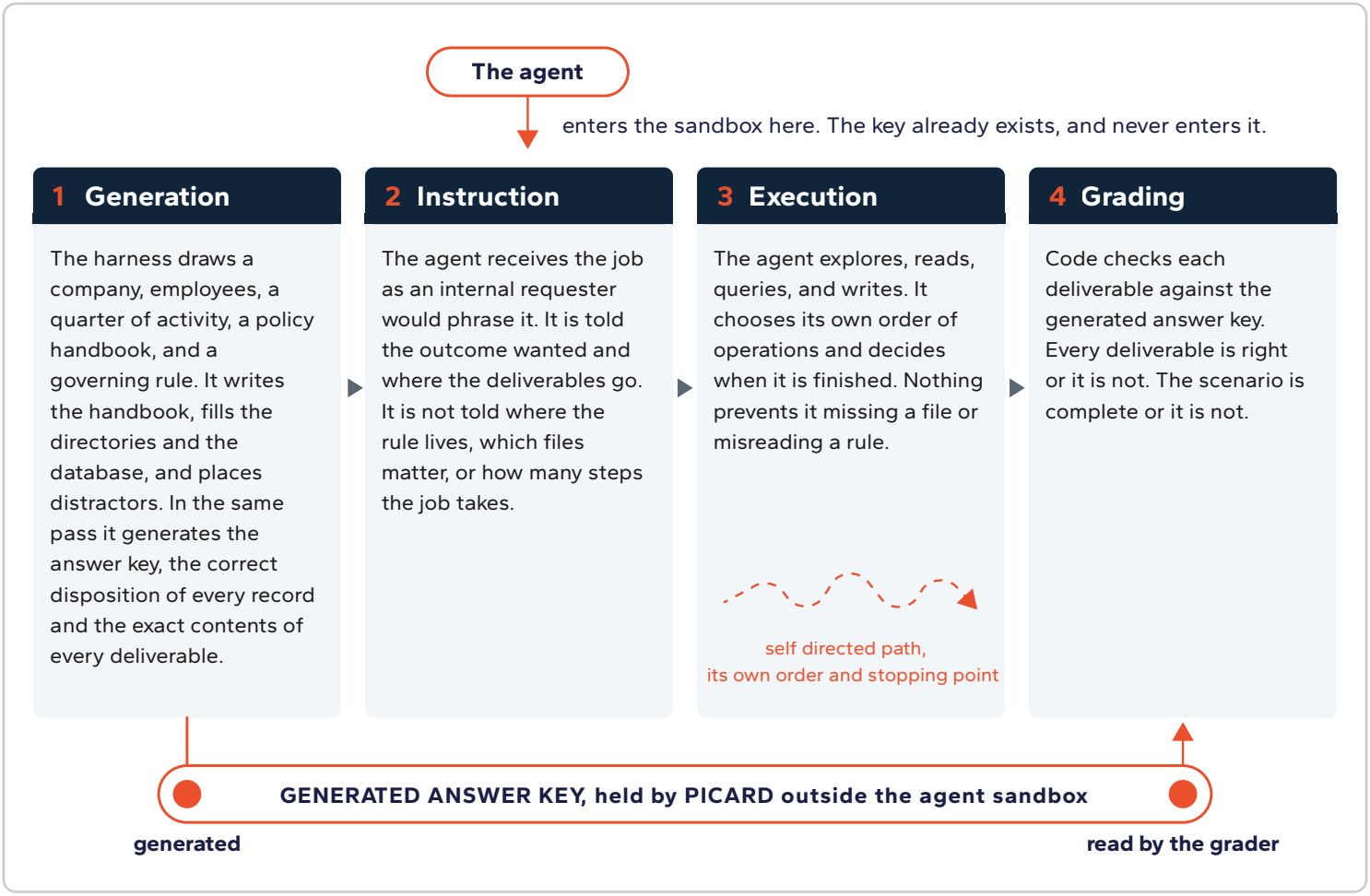
7.2 Round and context profile

Scenarios run against a 100 round cap and accumulate substantial context as they go. Each round appends the prior exchange plus whatever the agent read, so a session that begins with a short instruction is operating over tens of thousands of tokens of accumulated state by the time it produces its final deliverable. This is what makes the workload agentic in the sense that matters for infrastructure. Context grows monotonically, prefix reuse is high, and the tail of the session is where both quality and serving behavior are decided.

© 2026 Signal65 PINNACLE. All rights reserved.

7.3 One job, end to end

The specifics below are illustrative of a scenario shape rather than a transcript of a particular run.



The answer key is generated with the environment and held by PICARD, outside the sandbox the agent works in. It exists before the agent sees anything, which is what makes deterministic grading possible.

7.4 The calibration loop

A scenario enters the suite only after it has survived a calibration pass in which traces are read rather than scores tallied. The pass asks one question about every failure. Did the model fail the job, or did the job fail the model. Calibration has caught cases of the second kind, where a strong model reached a defensible reading of a rule that the answer key did not encode. The correct response in that case is to rewrite the rule so that it admits one interpretation, not to adjust the key.

© 2026 Signal65 PINNACLE. All rights reserved.

8. The Two Environments

Jobs run in two environments that differ in how much work the enterprise did before the agent arrived. This is not a difficulty dial and it should not be read as one. The two environments describe two real deployment shapes, and most readers will recognize immediately which one describes their own estate.

The suite holds 7 scenarios. Three run in the as-found environment, three are their governed counterparts, and one runs governed only, with no as-found version at present. The paired six are what the score gap in chapter 9 is computed from.

8.1 As-found

The as-found environment is data as an organization is likely to have accumulated it. Files that were migrated halfway and then abandoned. Two copies of the same record with different values and no indication of which is current. Naming conventions that changed twice. Directories organized by a scheme nobody documented. Records that appear to be in scope and are not, and records that do not appear to be in scope and are.

The governing rule still has to be applied correctly to all of it. The clutter is not decoration. It is constructed so that a plausible shortcut produces a wrong answer, which is exactly what happens in production when an agent is pointed at a real share drive.

8.2 Governed

The governed environment is the same job after the cleanup. Data is organized and classified. There is an explicit procedure. There is written guidance on how the agent is expected to behave. Ambiguity has been removed.

What has not been removed is the work. The governed scenarios run tens of tool use rounds, accumulate substantial context, and require the same multi step reasoning and the same complete set of correct deliverables. No reader should take the governed environment for a warm up, and the round counts and context figures are published precisely so that nobody does.

© 2026 Signal65 PINNACLE. All rights reserved.

THE SAME JOB ON BOTH SIDES

Apply the credit policy in the handbook to every Q2 account and report total exposure.
Correct total on both sides, 388,400.

GOVERNED
data an enterprise has already organized

policies/	
credit_policy_handbook.md	*the governing rule*
procedure.md	*explicit procedure*
accounts/	*one directory scheme*
ACC-0001.json	
ACC-0002.json	*one record per account*
...	
ACC-0140.json	*one layout throughout*
data/	
orders.db	*one table, one schema*
deliverables/	
q2_exposure.json	*where the answer goes*

The obvious path returns 388,400, the correct total.

AS-FOUND
data as an enterprise is likely to have accumulated it

policies/		
handbook_v2_FINAL.docx	*the governing rule*	
archive/		
handbook_2024.docx	*superseded*	
accounts/		
2025/	*CSV per region*	3
west.csv		
east.csv		
2026/	*JSON per account*	3
ACC-0001.json		
ACC-0031.json	*status closed*	2
...		
legacy_export/		
ACC-0007 (copy).json	*older values*	1
data/		
orders.db		
orders_backup_old.db	*second copy*	
deliverables/		
q2_exposure.json	*where the answer goes*	

Each shortcut returns a plausible wrong total. The correct total is still 388,400.

THREE CONSTRUCTIONS THAT PUNISH A PLAUSIBLE SHORTCUT

1. **Duplication with divergence**
   ACC-0007 appears twice. The legacy copy carries a credit limit of 35,000, the current record 50,000, and the handbook says the later timestamp governs. Taking the first match understates exposure by 15,000.
2. **Scope distractors**
   Twelve accounts match the obvious attribute, region West, and fail the condition stated in section 3, status active. Filtering on region alone returns 412,900, a plausible and wrong total.
3. **Structural drift**
   Half the accounts sit in 2025/ as CSV by region and half in 2026/ as JSON per account, because the migration stopped partway. Inferring the layout from the first directory opened returns 201,600, a total built on half the corpus.

Illustrative example. Sandboxes are generated per run, not customer data. File names and values are placeholders that show the shape of each trap.

As-found is not harder data. It is data with specific structural pathologies, each of which turns a plausible shortcut into a specific wrong answer.

© 2026 Signal65 PINNACLE. All rights reserved.

8.3 How a trap is built

Ambiguity in the as-found environment is generated rather than authored, which keeps it contamination resistant along with everything else. Three constructions do most of the work.

- **Duplication with divergence.** The same entity appears twice with conflicting values, and the governing rule or a timestamp determines which is authoritative. An agent that takes the first match is wrong.
- **Scope distractors.** Records that match on an obvious attribute but fail a condition stated elsewhere in the handbook. An agent that filters on the obvious attribute alone returns a plausible and incorrect total.
- **Structural drift.** Part of the corpus follows one layout and part follows another, because a migration stopped partway. An agent that infers the layout from the first directory it opens misses everything in the second.

Fabrication traps run in both environments. Some questions have no supported answer, and the correct output states that the data does not support a conclusion. A model that produces a confident number instead has failed, and it has failed in the specific way that does the most damage in a deployed system.

8.4 How the two compose

Both environments contribute to the headline quality score. The as-found result carries 60% of the weight and the governed result carries 40%. As-found is the realistic case, and a score dominated by the governed environment would overstate what a buyer should expect on day one. Governed keeps substantial weight because a large share of enterprises are actively doing the cleanup, and for them the governed number describes where they are heading.

That weighting runs against the sample counts rather than with them. The three as-found scenarios contribute 120 samples per model and take 60% of the weight. The four governed scenarios contribute 160 samples and take 40%. The weighting is an editorial judgment about which condition matters more to a buyer, applied on top of the measurement rather than derived from it, and it is stated here so that nobody has to reverse engineer it from the scores.



9. The Gap Between Environments

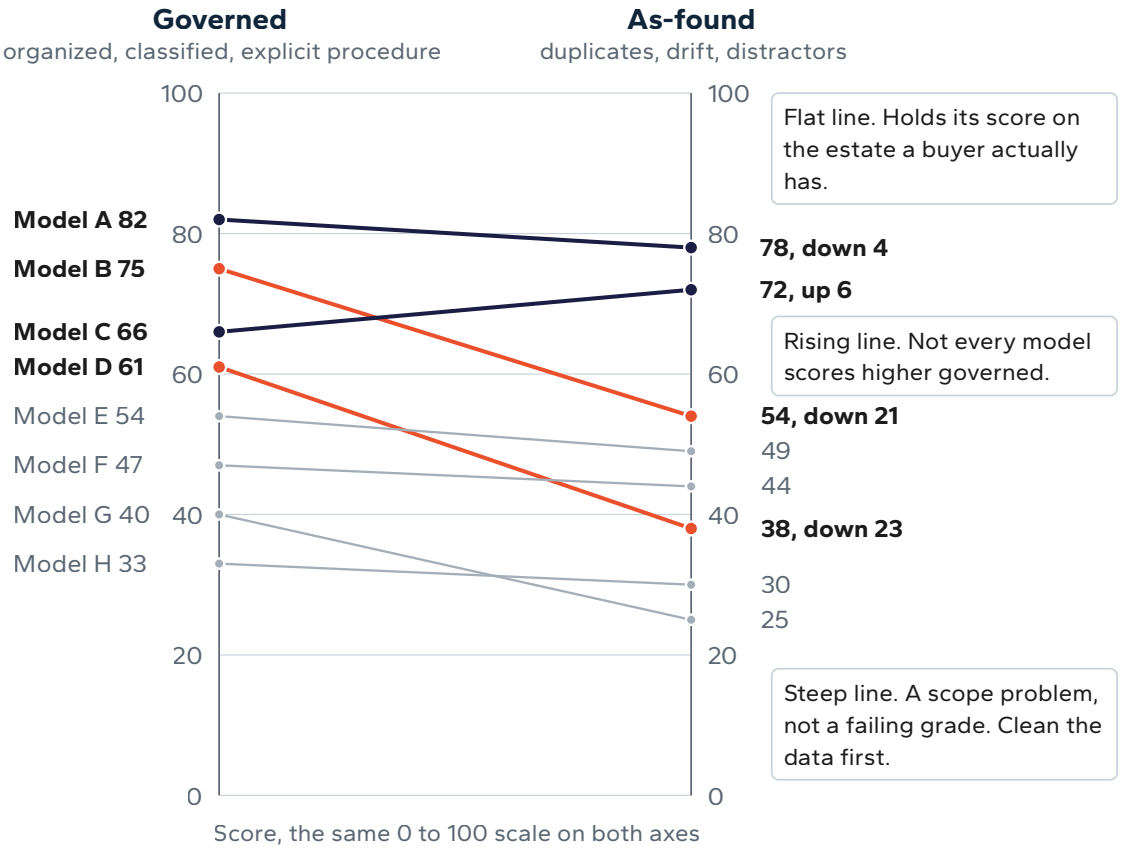
The difference between a model score in the two environments is a very useful number, and it is not available from any other instrument, because no other instrument runs the same job against two deliberately constructed data conditions.

9.1 What the gap is

The gap is a per model property. Some models hold their score almost flat across the two environments. Others post a strong governed score and lose a large fraction of it when the same job is run against messy data. The spread across the roster is wide enough to reorder rankings depending on which environment a reader looks at, which is precisely why publishing only one of them would be misleading.

© 2026 Signal65 PINNACLE. All rights reserved.

ILLUSTRATIVE. Placeholder values only.
Real per model governed and as-found scores follow the roster close.



A model that holds its score from governed to as-found data can be trusted with the estate a buyer actually has. A steep drop is a scope problem, not a capability problem.

9.2 What it tells you about a model

A model that holds up in the governed environment and falls apart in the as-found one is telling you that it needs the data cleaned before it can be trusted with the job. It reasons correctly when the inputs are unambiguous and does not reliably detect ambiguity when it is present.

A model that scores poorly in both is a different case entirely. It is not failing to handle mess. It is failing to do the job.

A model with a large gap has a scope problem you can engineer around. A model that scores poorly under both has a capability problem you cannot.

© 2026 Signal65 PINNACLE. All rights reserved.

9.3 The procurement decision it informs

An enterprise with organized and classified data, written agent rules, and bounded processes can safely deploy a model with a large gap, because it has already built the conditions that model needs. An enterprise pointing an agent at an estate as it accumulated cannot, and no amount of prompt engineering substitutes for the difference.

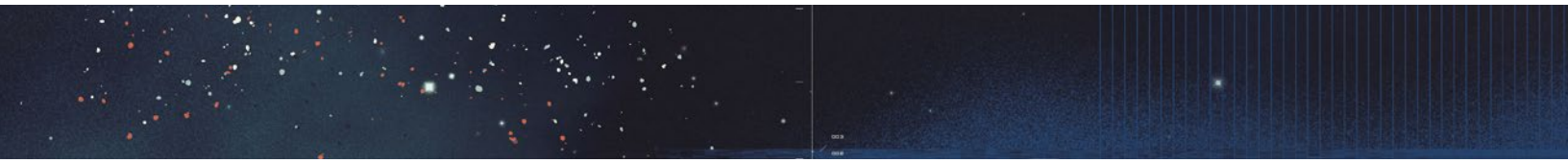
Read the other direction, the gap is also a budget argument. It converts data governance work from an act of hygiene into a quantity with a measured return, since the same model on the same job returns a different completion rate on either side of the cleanup.

9.4 How strongly the gap can be attributed

The governed scenarios are curated versions of the same underlying jobs as the as-found scenarios. Three as-found scenarios have governed counterparts built from the same specification, differing in the condition of the data and in nothing else.

That construction supports the strong reading. Because the job, the deliverables, the governing rule, and the answer key are identical on both sides, the difference between the two scores isolates the data condition rather than merely correlating with it. A model that drops twenty points crossing from governed to as-found dropped them to duplication, drift, and ambiguity, because there was nothing else to drop them to.

One scenario in the governed set has no as-found counterpart at present. It contributes to the governed score and to the headline, and it is excluded from the gap calculation, which is computed only across the paired scenarios.



10. Retrieval, Aggregation, and Fabrication Resistance

RIKER is the retrieval component of the suite. Documents are placed in context, no tools are available, and the model has to answer questions about what it was given. Three measures come out of it, and all three are defined in full in the published RIKER work.

10.1 The three measures

Measure	What it asks
Grounding	Can the model find and extract a fact from a document that definitely contains it. Includes facts stated outright, facts requiring minimal inference, facts in optional sections, and facts requiring cross referencing within a single document.
Aggregation	Can the model synthesize across documents. Counting, summing, comparing, enumerating, multi hop lookup, and temporal filtering over a corpus rather than a page.
Fabrication resistance	Will the model decline to answer when the answer is not there. Reported on the leaderboard as its complement, the fabrication rate.

Figure 3: *The three retrieval measures. Definitions are the published ones.*

© 2026 Signal65 PINNACLE. All rights reserved.

10.2 The retrieval context

All three measures are taken at 128K context. A single context length is used so that model comparisons hold one variable fixed and so that a persona score means the same thing for every model in the table.

The published retrieval work measured the same tasks at 32K, 128K, and 200K across ninety one models, and that work is what makes 128K the defensible choice rather than an arbitrary one. At 32K the field is compressed. Forty five of the ninety one models tested cleared 90% and twenty seven cleared 95%, which leaves too little separation to rank against. At 200K the roster thins out for a different reason, since only fifty four models were testable at that length at all and three cleared 95%, so the measurement increasingly describes which models can operate at that scale rather than how well they do the work. At 128K the field has separated enough to rank and the length still reflects a context that enterprises actually deploy.

10.3 Measuring fabrication requires generating the absence

Fabrication is the failure where a model produces a confident, specific, well formed answer to a question the source material does not answer. It is the failure enterprises ask about most and the one almost nobody quantifies, and the reason it goes unquantified is structural rather than a matter of effort.

To score fabrication, the grader has to know with certainty that the answer is absent. That certainty is only available to a party that built the corpus. A benchmark assembled from scraped documents, recorded sessions, or customer traces cannot certify that a fact appears nowhere in its material, because proving absence across a corpus nobody constructed is not something a benchmark can do. A model judge does not solve it either, since the judge has no more access to ground truth about absence than the model under test does.

Inverted generation solves it. The corpus is generated from a ground truth database rather than annotated after the fact, so the generator knows exactly which entities exist, which fields were populated, and which were deliberately left empty. Absence is generated with the same confidence as presence. Consistent entity relationships across the corpus are what make the guarantee hold at corpus scale, since an entity that is absent is absent everywhere and an omitted field is not quietly stated in a neighboring document.

Two probe types come out of that construction, and they correspond to two different real failures.

- **Non-existent entities.** Questions about an entity that appears nowhere in the corpus. The deployment analogue is a user asking about a contract, a policy, or an account that does not exist. The correct answer says so.
- **Absent information.** Questions about an optional field deliberately omitted from a document that does exist. The deployment analogue is a real contract with no early termination clause, and an agent that supplies one. This is the more dangerous of the two, because everything surrounding the invented value is genuine, so the answer looks well grounded to anyone checking it quickly.

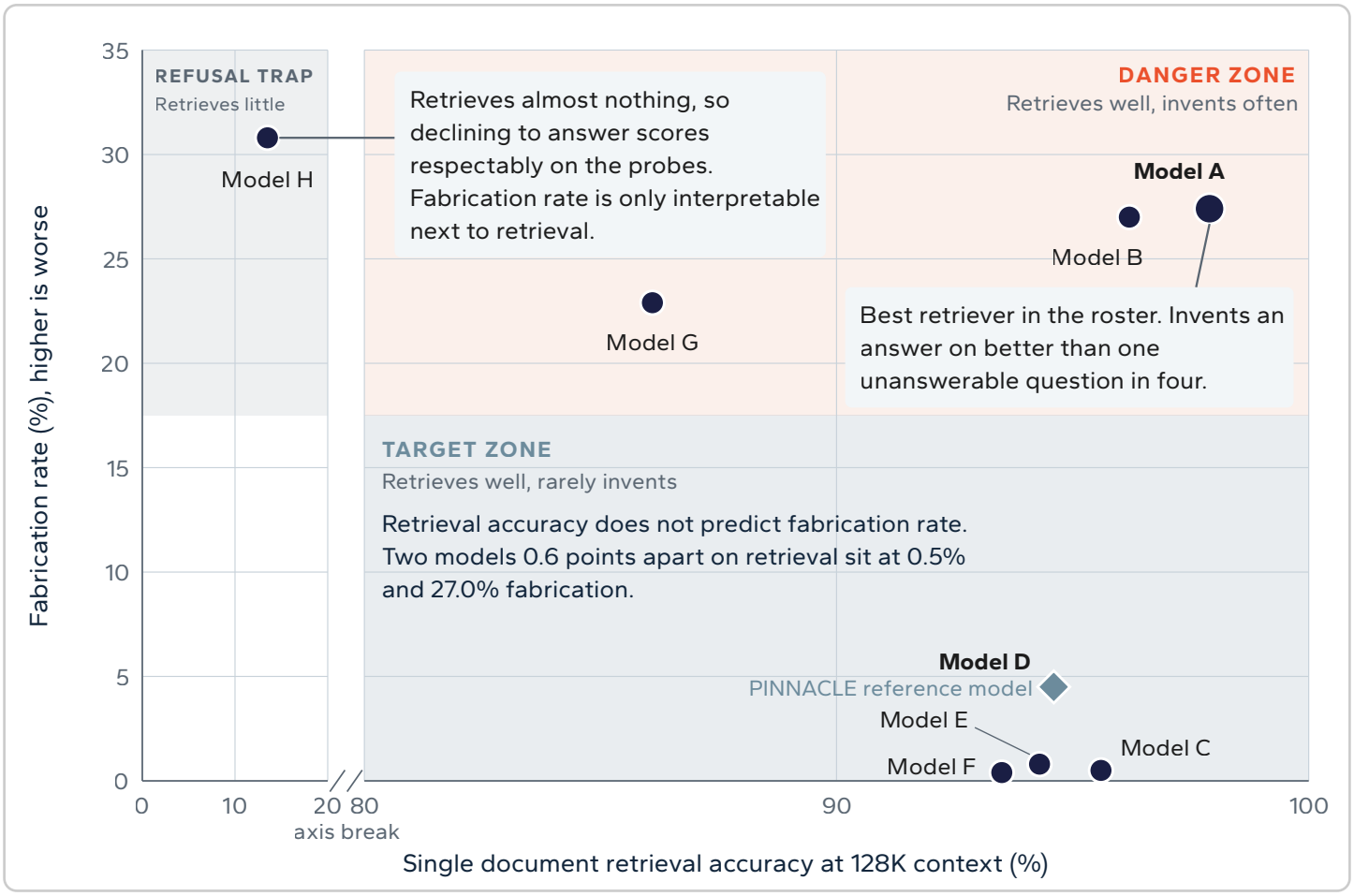
10.4 The fabrication rate

Fabrication resistance is measured as accuracy on those probes, and the leaderboard publishes its complement as a plain rate. A fabrication rate of 12% means that on 12 of every 100 questions whose answer was not present, the model produced one anyway.

Stating it as a rate matters because the number is not small and it is not evenly distributed. In the published work at 128K context, fabrication rates across the tested roster run from under 1% to more than 80%. No capability leaderboard reports that axis, and nothing else a buyer can consult would separate the two ends of it.

© 2026 Signal65 PINNACLE. All rights reserved.

The more useful finding is that fabrication rate does not follow retrieval accuracy. A model can be excellent at finding facts and poor at admitting when there are none.



Retrieval accuracy does not predict fabrication rate. The best single document retriever in the roster at 128K invents an answer on better than one unanswerable question in four.

Model A is the finding. The model with the highest single document retrieval accuracy in the entire roster at this context length also fabricates on better than one in four unanswerable questions. A buyer selecting on retrieval accuracy alone would pick it, and would be selecting the most confidently wrong model available to them.

Model H is the trap running the other way, and it is the reason fabrication rate is never published on its own. A model that answers almost nothing scores respectably on probes designed to reward saying nothing. Read in isolation the number flatters it. Read next to a retrieval accuracy of 13.5% it is exposed as a model that is not answering rather than a model that is declining well. Fabrication rate and retrieval accuracy are only interpretable together, which is why both are published and neither is folded into the other.

Configuration moves this axis further than most readers expect. In the published data, enabling reasoning on one model family took the fabrication rate at 128K from above 80% to near 3% on the same weights, a larger swing than any change of model in the same size class. Reasoning configuration belongs in the deployment decision alongside the choice of model.

© 2026 Signal65 PINNACLE. All rights reserved.

10.5 Why fabrication costs more than an empty answer

A retrieval miss and a fabrication are not the same failure and they do not cost the same to absorb.

A retrieval miss announces itself. The output says the information could not be found, a human follows up, and the process continues at the cost of a delay. A fabrication announces nothing. It produces a specific value in the expected format, in the expected place, indistinguishable from a correct answer to anyone who does not already know the correct answer. It flows into the next step of the workflow, into the deliverable, and into whatever decision the deliverable supports.

The asymmetry is in detection cost. Catching a fabrication requires going back to the source material and confirming the absence, which is the work the agent was deployed to remove. One invented value can therefore erase the savings from a large number of correct ones, and the deeper the workflow and the more autonomous the deployment, the worse the arithmetic gets.

This is why the Customer Operations persona weights fabrication resistance highest of any persona in the suite. An autonomous agent answering policy questions without a human in the path converts every fabrication directly into an exposure, and for that deployment a refusal is not a degraded answer but the correct one.

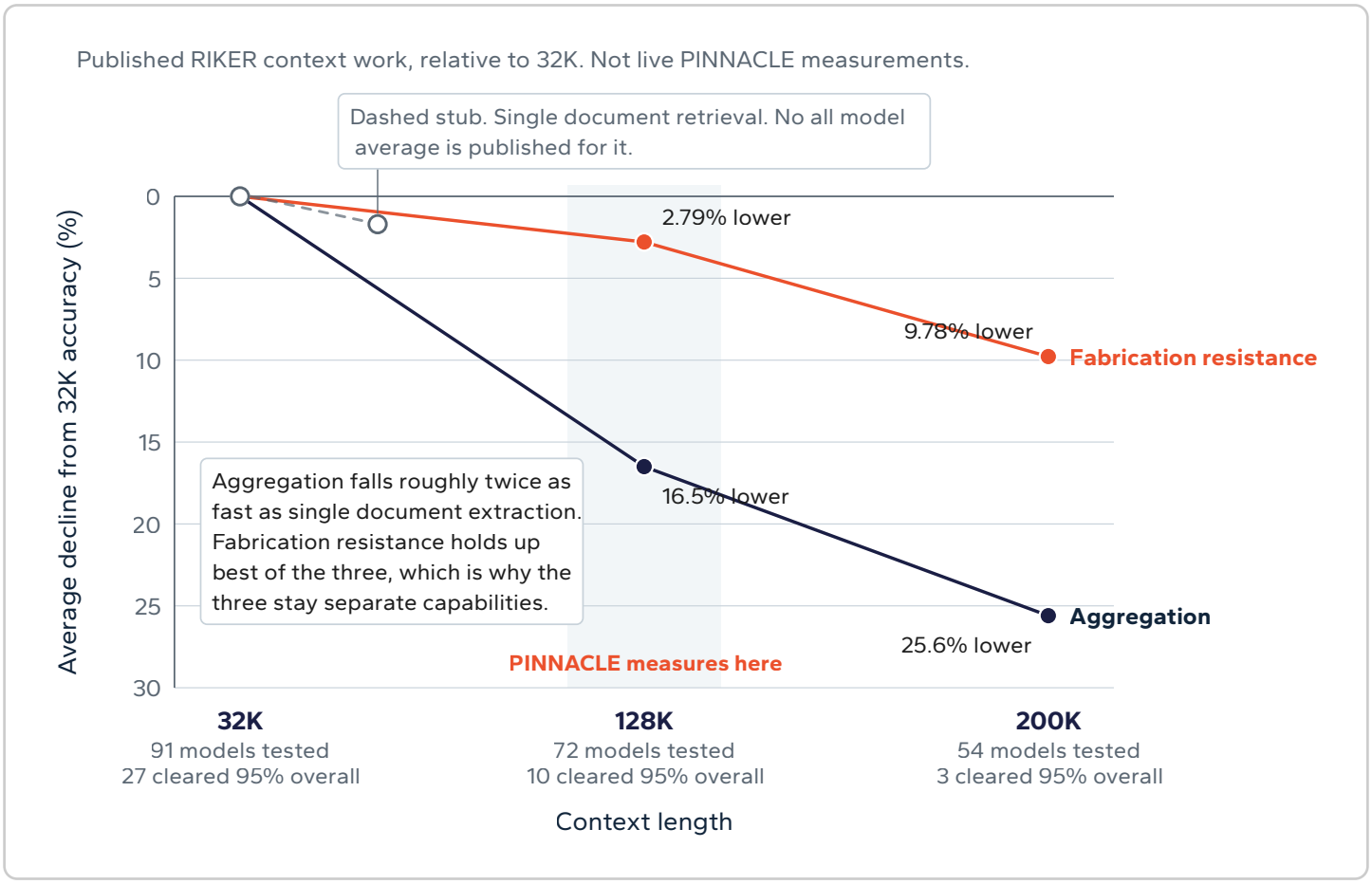
10.6 What the context work established

Two results from the published work carry into how the suite is read, beyond fixing the measurement at 128K.

Aggregation degrades roughly twice as fast as single document extraction as context grows. Across all models tested, aggregation accuracy fell by an average of 16.5% at 128K and 25.6% at 200K relative to 32K. Since most enterprise retrieval work is synthesis across sources rather than extraction from one page, the harder of the two measures is also the more representative one.

Fabrication resistance degrades more slowly than retrieval does. Probe accuracy fell by an average of 2.79% at 128K and 9.78% at 200K, materially less than either retrieval measure over the same range, and several models held flat or improved. Retrieval failure and fabrication are therefore partially distinct failure modes rather than two symptoms of one weakness, which is the measured basis for keeping them as separate capabilities in chapter 11 rather than combining them.

© 2026 Signal65 PINNACLE. All rights reserved.



The three retrieval measures do not degrade at the same rate. Aggregation falls roughly twice as fast as single document extraction, and fabrication resistance holds up best.



11. The Enterprise Personas

A single aggregated score is useful and it is not sufficient on its own. Signal65 PINNACLE publishes one, because a buyer comparing a field of models needs a place to start, and the standing objection to any such score is that aggregation hides the thing a particular buyer actually cares about. That objection is fair. What makes it hard to answer is that nobody outside a benchmark can take a published score apart again, because nobody outside controls the structure underneath it.

The persona layer is the answer. It works because five genuinely different capabilities are measured separately and kept separate, so the aggregate can be rebuilt in whatever proportions a reader needs rather than being the only view on offer.

© 2026 Signal65 PINNACLE. All rights reserved.

11.1 The five measured capabilities

Capability	Source
Workflow completion, as-found	Agentic jobs run against data as an enterprise accumulated it.
Workflow completion, governed	The same jobs run against organized and classified data with an explicit procedure.
Retrieval grounding	Fact extraction from documents, at 128K context.
Aggregation	Synthesis across documents, at 128K context.
Fabrication resistance	Refusal to invent when the answer is absent, at 128K context. Published as a fabrication rate.

Figure 4: *The five separately measured inputs to every persona score.*

11.2 The five roles

Each persona weights those five by what the role actually fails on. The weighting is not cosmetic. Because the five capabilities are empirically independent of one another, changing the proportions changes the answer, and the same roster of models produces materially different orderings depending on which role is asking.

Persona	Weighted toward, and the failure mode that motivates it
Knowledge Worker	Grounding. The work is finding and restating facts accurately, and the damage from a wrong fact is that it flows into a report nobody re-checks.
Data Analyst	Workflow completion and aggregation. Structured analysis across sources, where the failure is a query that returns a plausible number computed the wrong way.
IT Professional	Workflow completion. Long tool chains carried to a finished state, where the failure is a job abandoned partway with artifacts half written.
Customer Operations	Fabrication resistance, weighted highest of any persona. An autonomous agent that invents a policy answer creates direct liability, and a confident wrong answer is worse than a refusal. This is the persona where the fabrication rate in chapter 10 does the most work.
Executive	Aggregation. Synthesis across many sources, where the failure is a briefing built on a total that was never actually in the data.

Figure 5: *The five personas and what each is weighted toward.*

© 2026 Signal65 PINNACLE. All rights reserved.

11.3 What the weighting changes

Reordering happens, and it is the point. A model that leads on aggregate quality can drop several places under the Customer Operations weighting if its fabrication resistance is weak, and a mid-ranked model with strong refusal behavior can lead that persona outright. On the published leaderboard this is exposed as a single control that re-ranks the table by persona, which converts a leaderboard into an answer to a specific buying question.

11.4 What the persona layer is and is not

The persona layer is a reweighting. Personas do not run distinct scenario suites, and all five read retrieval measured at a single context length of 128K. Every persona reads the same five measurements, taken under the same conditions, and recombines them in different proportions. No material anywhere should imply that a role runs its own tasks or its own retrieval conditions, because it does not. The Coding Agent test in chapter 21 is not a persona for the same reason. It runs jobs of its own, so it is reported as a separate measurement and not as a weighting of these five.

The claim does not need it. A reweighting can be either a genuine analytical instrument or a cosmetic relabeling, and the difference between the two is testable.

A reweighting is cosmetic when the underlying measurements move together. If a model that is strong on one capability is reliably strong on all five, then any set of weights produces approximately the same ordering, the personas differ by a few decimal places, and the layer is decoration. That is the failure mode a skeptical reader should be checking for.

It is not what the measurements show. The five capabilities are substantially independent of one another, and two of those independences are large enough to reorder a leaderboard on their own.

- **Grounding does not predict fabrication resistance.** At 128K in the published work, the model with the highest single document retrieval accuracy in the roster carries a fabrication rate of 27.4%, while models scoring several points lower on retrieval sit below 1%. Any composite that averaged the two would conceal exactly the failure that an autonomous service deployment cannot tolerate, and the Customer Operations weighting exists to surface it.
- **Governed performance does not predict as-found performance.** The score gap in chapter 9 is a per model property and it is wide. A model can be near the top of the field on organized data and mid pack on the same job against data an enterprise is likely to have accumulated.

Because those independences are real, the reweighting does real work. A model can lead the overall PINNACLE Model Score and rank several places lower under the Customer Operations weighting, and a model that never leads anything overall can lead a single persona outright. Those movements are not noise around a true ranking. They are the consequence of five measurements that genuinely disagree, and a buyer who only ever sees the aggregate never learns that they disagree.

Three properties let a reader check the layer instead of taking it on trust. The weights are published per persona, so a reader can recompute any persona score from the five underlying capability numbers. The capability numbers are published, so a reader who rejects the weights entirely can build their own. And the layer is falsifiable in the direction that matters, because if the personas were decoration the rankings would barely shift when the weights change.

A benchmark that aggregates before publishing cannot offer this at any weighting, because the information was destroyed at the point of aggregation. The argument rests on measuring separately rather than on running separate suites.

© 2026 Signal65 PINNACLE. All rights reserved.

For a buyer the practical consequence is that the right table is rarely the default one. An enterprise deploying autonomous service agents and an enterprise deploying analyst copilots are not choosing between the same models, even though a single aggregate score would hand them the same ranking and the same recommendation. The persona layer is what lets one measurement set answer both questions honestly instead of answering one and approximating the other.

11.5 From personas to a workforce

Personas aggregate. An enterprise that knows roughly what share of its people work in each mode can combine the five into a company profile, and from there into token demand, sustained throughput, and infrastructure sizing. That extension is how the model quality layer connects to the capacity layer for a specific buyer rather than in the abstract.



12. Scoring

Every agentic run is scored two ways, only one of the two enters the headline, and the composite is built from there.

12.1 Two scoring modes

Mode	What it asks
Workflow completion	Did the whole job come out right. Binary per scenario. Every deliverable correct, or the run counts for nothing.
Per-deliverable accuracy	What share of the individual output artifacts were right. Partial credit within the scenario. A deliverable is a required output, not a tool use round, so a scenario running 60 rounds may produce seven deliverables.

Figure 6: *The two scoring modes.*

Every reader has met this as a grading policy. All or nothing per question against partial credit per question is a choice a teacher makes, and the two produce very different rankings over the same set of papers.

Partial credit earns its place because it measures potential rather than delivery. A model producing six correct deliverables out of seven has shipped nothing usable, and it has also demonstrated that it understood the job, located the governing rule, and executed most of it. That is a materially different proposition from a model that produced two of seven, and the difference describes what an integration team would be working with.

12.2 Why only workflow completion enters the headline

The headline PINNACLE Model Score is built from workflow completion in both environments. A deployed workflow that produces six of seven correct outputs has not shipped anything usable. Somebody still has to find the wrong one, and finding it requires redoing the work that was supposed to be delegated. Binary scoring is the view that matches what deployment requires.

© 2026 Signal65 PINNACLE. All rights reserved.

Per-deliverable accuracy is measured and published alongside, never folded in. The distance between the two carries a great deal of the useful signal, which is the subject of the next section.

A run that produces seven deliverables and gets six right scores 0 on workflow completion and roughly 86% on per-deliverable accuracy. A run that gets three right scores 0 and roughly 43%. Under the headline, those two runs are identical, and under the companion measure they are not close. Both readings are true and they answer different questions.

12.3 The distance between the two modes

The gap between workflow completion and per-deliverable accuracy is an estimate of integration effort, and it is the second thing the de-aggregated structure makes available.

A model that completes few whole jobs while producing mostly correct individual deliverables is telling a buyer that the remaining distance is closable with engineering. Orchestration, validation steps, retry logic, and tighter prompting all attack exactly that gap, and a model in this position is a reasonable bet for a team willing to build around it. A model scoring poorly on both is telling them something else, because no scaffolding manufactures deliverables the model never got right.

Both numbers publish. Neither is folded into the other, and the distance between them is reported as a figure in its own right.

12.4 How the composite is built

Persona scores are a weighted geometric mean of the five measured capabilities using that persona weights. The overall PINNACLE Model Score is an unweighted geometric mean across the five personas.

$$\text{persona score}(P) = K \cdot \prod_i \text{capability}_i^{w_i}$$

$$\text{PINNACLE Model Score} = (S_{KW} \cdot S_{DA} \cdot S_{TO} \cdot S_{SA} \cdot S_{EX})^{1/5}$$

w_i	weight on capability i, published per persona
K	set so the reference model scores 1000

The geometric mean is the standard aggregation for composite benchmarks and it is used here for the same reasons it is used elsewhere. It treats percentage improvements symmetrically, it prevents one high scoring component from dominating, and it preserves relative rankings regardless of the reference chosen. Leaving the outer mean unweighted means a model has to be competent across all five personas, since catastrophic weakness in one cannot be bought back by excellence in another.

12.5 The reference model

K is set so that the reference model scores exactly 1000 on every persona and 1000 overall. The reference is Gemma-4-31B-it with thinking enabled, chosen on three grounds. It is a widely used open model with an established following. It comes from a major research lab rather than an obscure one. And during data collection it landed near the midpoint of the agentic score distribution, which puts the anchor in the middle of the field rather than at either end.

© 2026 Signal65 PINNACLE. All rights reserved.

The last of those matters most for a scale. An anchor at the frontier compresses every other model into a narrow band below 1000, and an anchor at the bottom does the reverse. A midpoint anchor spreads the field in both directions and keeps 1000 meaningful as a reference point rather than as a ceiling.

The reference is held constant across releases within a major version. Moving it silently would invalidate every published comparison, so the conditions under which it moves are stated in the governance rules and prior results are restated rather than quietly replaced when it does.

12.6 Token efficiency

Token consumption is a property of the model rather than the hardware, and it is reported alongside the quality score rather than folded into it.

Two measures publish, because the plain reading and the useful reading are different numbers.

Output tokens per task. Median output tokens the model spends on one sample, regardless of whether it got the answer right. This is verbosity, and it is what most readers mean by token efficiency.

Output tokens per correct task. The first measure divided by the success rate. This is what a correct outcome actually costs, and it is the number that belongs in a budget.

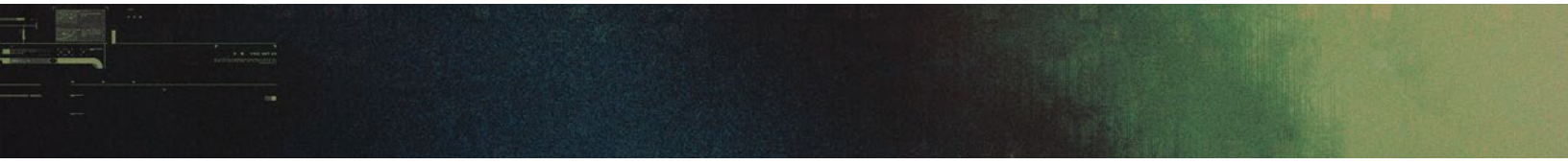
The second is not a restatement of the first. Take two models that each spend 5,000 output tokens on a median sample, so their verbosity is identical. One succeeds every time and the other succeeds half the time. The first costs 5,000 output tokens per correct task. The second costs 10,000, because on average it takes two attempts worth of tokens to arrive at one correct result. Identical on the plain measure, twice the cost on the one that matters.

Hidden reasoning tokens count in both, because an enterprise pays for them in hosted and owned deployments alike.

Neither measure is folded into the quality score. A model that spends five times the tokens to reach the same score is a different proposition to deploy, and combining the two would make a score of 1150 ambiguous between accurate and efficient, and mediocre at both.

12.7 The coding score

Coding results are scored separately, on their own pass rates, and indexed to the same reference model. Section 21.6 gives the formula, the reasons for its weighting and the ceiling rule, which applies to both scores. Nothing in the coding score enters the PINNACLE Model Score in this version.



13. Performance Methodology

Speed in Signal65 PINNACLE is the usable capacity of a fully loaded platform, joined to the scored workload through a token recipe measured on the same runs. No published figure rests on a wall clock.

© 2026 Signal65 PINNACLE. All rights reserved.

13.1 Full box saturation and the concurrency sweep

Speed is measured by loading a box rather than by timing one request. A saturation instrument sweeps concurrency, running N agents against the platform simultaneously and recording per agent decode rate, first token latency, and aggregate throughput at each rung.

The sweep does not stop when aggregate throughput peaks. It continues until a rung fails one of the two service level gates, and it stops as soon as either gate is breached rather than waiting for both. Past the aggregate peak a platform is less throughput efficient per agent, and it is frequently still serving every agent acceptably, so the concurrency it can carry can keep rising after the throughput it can deliver has begun to fall. Stopping at the peak would discard that range.

The search strategy is a coarse scan followed by refinement around each transition. Rungs are spaced widely, then narrowed to locate both the aggregate peak and the last rung that clears the gates. Every rung run is published, including the ones that failed and which gate each one failed on.

Single stream speed is deliberately not the measurement. One lucky fast request tells a buyer nothing about how many capable agents a platform can serve at once, and a productivity number that rewarded it would reward the wrong engineering. Loaded aggregate capacity rewards scalability, which single stream measurement cannot see at all.

13.2 Why no published figure rests on a wall clock

Any duration recorded in any trace is contaminated by the box it ran on, the load that box was under, the network in the path, and thermal state. Durations do not port between environments and they are not reproducible by a third party, which makes them unsuitable as the basis of a published comparison.

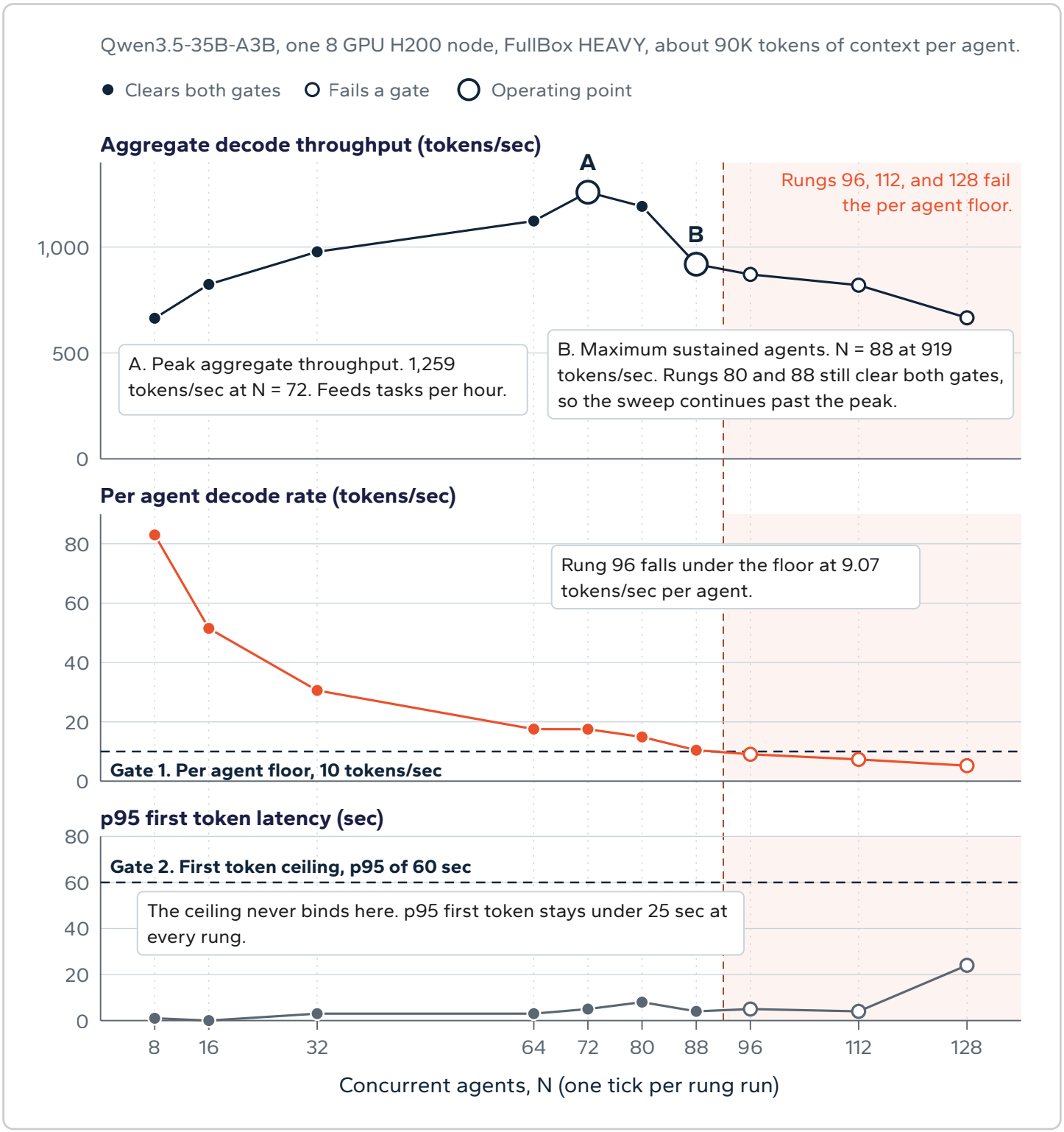
What survives is two portable quantities. A token recipe from the scored traces, which is what the intelligent behavior cost in tokens. And rates from the saturation instrument, which is how fast a platform emits tokens under load. These are multiplied at the end. A wall clock is never multiplied by anything.

13.3 The decode bound argument

Carrying a single rate is only legitimate if one phase dominates task time, and on this workload it does. Prefill runs approximately 500 times faster than decode, measured in our testing at roughly 19,000 tokens per second against roughly 40 tokens per second per agent on the largest models in the roster. Prefix reuse across rounds is high, which makes the prefill tail cheaper still.

So the portable number is decode aggregate throughput, applied to the completion token count from the scored trace. The naive alternative of dividing total tokens by one blended rate is wrong by construction, because prefill and decode differ by two orders of magnitude and growing context has to be charged on the uncached delta per round rather than on the full context every time.

© 2026 Signal65 PINNACLE. All rights reserved.



Capacity is not one number. The sweep continues past peak throughput, two operating points come out of it, and a rung counts only if it clears both service level gates.

© 2026 Signal65 PINNACLE. All rights reserved.

13.4 The gates, and the two operating points

Raw peak aggregate is gameable. A platform can admit an enormous number of agents and slow every one of them to a crawl, and the aggregate number will look excellent while no agent is doing useful work. Two gates close that, and a rung has to clear both to count.

Gate	Threshold	What it catches
Per agent decode floor	10 tokens per second per agent	Usefulness. Every agent must decode at a rate a human or a downstream system can actually consume, which is a time per output token of 100 ms.
First token ceiling	p95 latency to first token across all admitted agents, 60 seconds	Admission honesty. A serving stack can accept agents it is not actually running, and per agent decode looks compliant for the subset being served. The ceiling catches the queue.

Figure 7: *The two service level gates. A rung counts only if it clears both.*

Two operating points come out of the sweep and both publish, because they answer different questions.

Operating point	What it reports
Peak aggregate throughput	The rung where total useful decode output is highest, and the concurrency at which it occurred. This is the efficiency point, and it feeds tasks per hour.
Maximum sustained agents	The highest concurrency that still clears both gates. This is the capacity point. It sometimes coincides with the concurrency at peak throughput and sometimes sits above it.

Figure 8: *The two reported operating points. Where they diverge, the platform is trading per agent efficiency for headroom.*

The first token ceiling exists for a reason that is only obvious after it has caught something. A stack under pressure can admit sessions into a queue and report healthy decode rates for the subset it is actively serving. Per agent decode is therefore not sufficient on its own. Measuring the latency to the first token across all admitted agents exposes the difference between agents being served and agents being held.

Sixty seconds is deliberately permissive. It is a liveness threshold rather than an interactivity one, because agentic work is dispatched and collected later rather than watched. Under normal operation the gate never binds, and it binds sharply once a stack begins queueing, which is exactly the behavior it exists to catch.

Signal65 Comment: *Agent count is a capacity number and a quality of service axis, and it is reported on the silicon lens. It is deliberately not a term in the productivity composite, because concurrency cancels out of tasks per hour. At equal in service aggregate throughput, tasks per hour is identical however many agents share it.*

© 2026 Signal65 PINNACLE. All rights reserved.

13.5 The bridge to tasks per hour

The rate comes from the box and the token cost comes from the task. They are joined per token.

$$\text{tasks per hour} = \frac{3600 \cdot R}{C_\pi}$$

R	peak aggregate decode tok/s, clearing both gates
C_π	completion tokens per task, from the scored trace
3600	seconds per hour

Every term is either measured or explicitly derived. *R* is measured on the reference platform under load. *C* is measured on the scored runs. The 3600 converts seconds to hours. Nothing in the chain is estimated and nothing is a duration.

This construction is also hard to game in the direction vendors are usually accused of gaming. A configuration tuned for throughput at the expense of answer quality still owes the quality term in the composite, and a configuration tuned for a single fast stream never reaches a competitive *R* because *R* is measured under load.

13.6 The measurement honesty check

The saturation instrument runs a single task, and the entire speed methodology depends on that task loading a model the way the scored suite does. That is an assumption, and it was tested rather than asserted. The result below includes the imperfect part.

The tokenizer carries the argument. A model uses the same content to token mapping in every instrument, so a saturation task that holds a fixed volume of content rather than a fixed token count loads each model to its own representative point automatically. A token efficient model legitimately sits lighter, which is a real efficiency to credit rather than an artifact to normalize away. The check is whether the saturation context lands where the scored workload actually lives, and it is run on models that appear in both instruments with the same tokenizer.

Model family	Tokenizer	Saturation context	Own p95	Ratio
Qwen family	Heavier	90K to 91K	82K	1.10
MiniMax, MXFP8	Lighter	58K	61K	0.95

Figure 9: *Saturation context against each model own scored p95 context. Measured. The median scored sample sits at 46K, so the saturation anchor is the loaded normal point rather than the average.*

Both land within roughly 10% of the scored p95 for that model, on its own ruler. The cross-model relationship is directionally consistent as well, with the lighter tokenizer sitting at approximately 0.64 of the heavier one in the saturation instrument against 0.74 at scored p95.

A residual remains. Token accumulation across rounds depends on what the agent reads and how it proceeds, both of which vary run to run, so a small mismatch is inherent to measuring agentic work rather than a sign that something is wrong. The cross check covers two tokenizer families, which is a small sample for an assumption this load bearing.

© 2026 Signal65 PINNACLE. All rights reserved.

13.7 Publishing the sweep

The full concurrency sweep is published rung by rung, including the levels that failed and which gate each one failed on. A capacity claim of the form this platform sustains N agents is only checkable if a reader can see the rung above N failing, and can see which of the two thresholds it failed.

This also disciplines the claim internally. Publishing the failing rung makes it impossible to quietly report a peak that was never sustainable, and it gives a vendor engineer something specific to argue with, which is a better outcome than a headline number they can only disagree with in general terms.



14. PINNACLE Intelligence Throughput

Quality and speed are reported independently and can be read that way. They are also combined into one figure, because a buyer comparing platforms eventually has to trade them off and pretending otherwise pushes the judgment onto the reader without giving them the tools to make it.

14.1 The composite

$$\text{Intelligence Throughput} = \frac{\text{score}^{\,2w} \cdot \text{speed}^{\,2-2w} \cdot p^{\,N}}{1000}$$

$w = 0.75$	quality exponent 1.5, speed 0.5
score	PINNACLE Model Score, that persona
speed	tasks per hour, section 13.5
p	per task success rate
N	workflow depth, that persona

At *w* equal to 0.75 the speed term is a square root. Tripling the usable capacity of a platform raises Intelligence Throughput by a factor of roughly 1.73 rather than 3. That is the weighting doing its job. Scalability is rewarded and raw speed cannot buy quality cheaply.

14.2 The weighting is judgment, and it is owned

Three to one is an editorial judgment by Signal65. It is not derived from data and it is not the output of an optimization. It is stated as judgment everywhere it appears, it is the same weighting for every vendor and every result, and the reasoning behind it comes from advisory work rather than from the benchmark.

Three observations from that work point the same direction. Enterprises abandon agentic pilots over accuracy far more often than over latency, and a pilot that produces wrong output does not get a second round regardless of how fast it produced it. A wrong deliverable also costs more than a slow one, because someone has to find it, which means redoing the work that was delegated in the first place and absorbing whatever the error touched downstream. And latency has a floor below which additional speed stops being worth anything, since work handed to an agent is typically collected later rather than watched in real time, while accuracy has no such ceiling.

© 2026 Signal65 PINNACLE. All rights reserved.

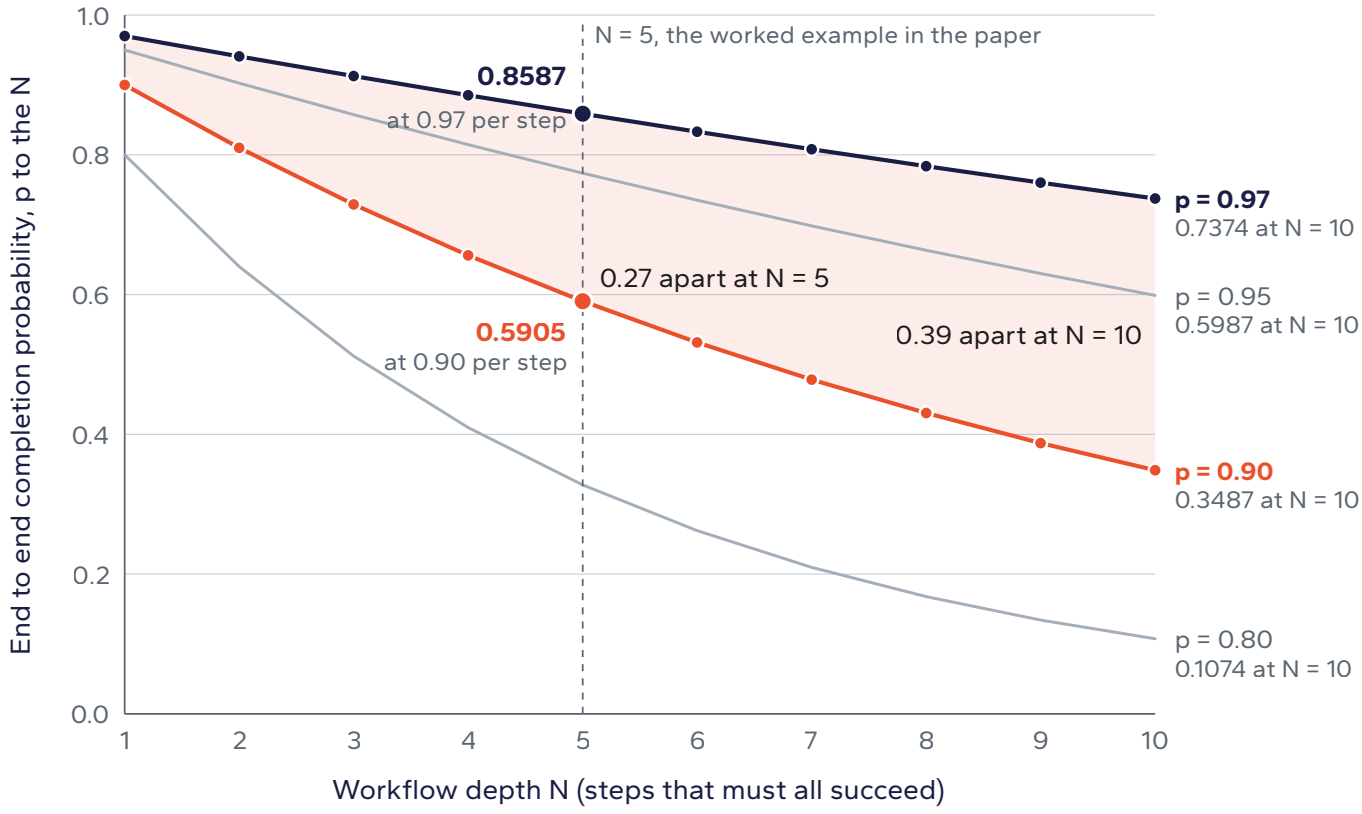
Weighting quality three times as heavily as speed reflects that asymmetry. It is a judgment about what enterprises are observed to do rather than a claim about what they should do, and a buyer whose economics differ can recompute the composite from the published inputs.

One weighting publishes, and it is the same one for every vendor and every result. Every input to the composite publishes alongside it, so a buyer whose economics differ can recompute it at whatever ratio their own experience supports.

14.3 Why per task success compounds

The p to the N term is what separates this composite from a throughput number with a quality adjustment bolted on. A workflow of depth N completes only if every step in it completes, so a per step success rate of 0.97 across a depth of five yields roughly 0.86 end to end, and a rate of 0.90 across the same depth yields roughly 0.59. Small differences in per step reliability become large differences in delivered work, and the deeper the workflow the more brutal the arithmetic gets.

This is the honest reason autonomy is hard. It is also why a model that is slightly less accurate and considerably faster does not automatically win, since the compounding term erodes faster than the speed term grows.



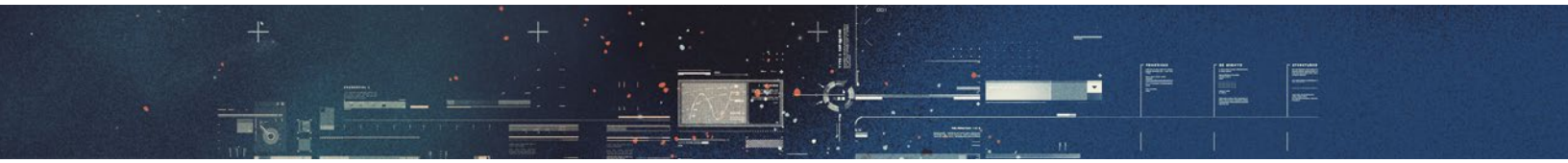
Small differences in per step reliability become large differences in delivered work. At depth five, 0.97 per step delivers 0.8587 end to end and 0.90 delivers 0.5905.

© 2026 Signal65 PINNACLE. All rights reserved.

14.4 Break even failure cost

Alongside the composite, each model carries a break-even failure cost. It answers a single question. How expensive would one failure have to be before a cheaper, less accurate model stops being the better buy. A model with high accuracy and high token cost has a low break-even, meaning it pays off as soon as failures cost anything meaningful. A fast, cheap, less reliable model has a high break-even, meaning it only makes sense where failures are close to free.

Expressed that way, the number tells a reader something about their own tolerance for autonomy rather than something about the model in isolation, which is what makes it useful for a procurement conversation.



15. Precision and Quality Parity

Reduced precision buys throughput, and the question a buyer needs answered is what it costs in return. Signal65 PINNACLE measures the answer as well as the rate, on the same runs, which turns that question into something we can measure directly.

15.1 Why a cross precision throughput number is not a comparison

Reduced precision produces more tokens per second. That is the point of it. A comparison that reports tokens per second at one precision against tokens per second at another has measured two different workloads and presented the result as though it measured one, and the direction of the error always favors whichever side reduced precision further.

The comparison is only meaningful if the answers are equivalent. Establishing that requires a quality measurement taken on the same configuration, which is exactly what a throughput only benchmark cannot supply.

15.2 The quality parity gate

Matched precision is the published default on the leaderboard. A cross platform capacity comparison is reported at equivalent precision unless the paired quality runs demonstrate that the answers hold, and where a vendor recommends a reduced precision configuration for production, the quality run at that precision is published alongside the capacity number rather than assumed.

The practical shape of the gate is a paired run. The same model, the same scenarios, the same harness, at two precisions, scored the same way. If the scores are within measurement noise, the precision reduction is free and the throughput gain is real. If they are not, the throughput gain has been partly paid for in answer quality and the comparison has to say so.

15.3 Quantized models are separate entries

A quantization changes the weights, so a quantized model is not the same model as the one it was derived from and it is never merged with it. Every quantized configuration appears in published results as its own entry, with its precision in the model identifier.

Coverage here is deliberately narrow at launch. Signal65 has not systematically swept quantizations for every model, so where multiple precisions of one model exist in the results they exist because that pairing was worth testing, not because the roster is comprehensive on this axis.

© 2026 Signal65 PINNACLE. All rights reserved.

15.4 Where reduced precision stops holding

Precision effects are not uniform across task length. Reduced precision configurations that hold quality on short tasks do not necessarily hold it on long context work, where small numerical differences accumulate across many rounds and many tokens of context. A parity check run only on short tasks can clear a configuration that will not hold in a deployment.

This is why the parity check runs on the same agentic and long context workloads as the rest of the suite rather than on a short proxy.

15.5 The serving stack is part of the configuration

Model scores in this suite do not change with the accelerator, and we know that because we tested for it. Scores were originally computed separately on each hardware platform. Across MI300X, H200, and B300 the differences in both component benchmarks sat inside run to run noise, so the practice was dropped and quality is now measured once per model. The reference platform for PINNACLE Speed is B300, which puts every model on the same box for its throughput terms. Multiple platforms enter the suite only in the silicon capacity lens, where throughput and sustained agent count are the quantities being compared.

The serving stack is a different matter, and it is the one configuration axis that has visibly moved answer quality in our testing. The mechanism is rarely intentional. Kernel bugs, version regressions, and misapplied optimization flags all surface the same way, as a configuration that runs fast and returns wrong output. Multi token prediction is the clearest example encountered so far, since it is supposed to leave outputs unchanged and simply arrive sooner. In practice, specific model and stack combinations have produced degraded scores and, in some cases, garbage tokens on a fraction of requests while the throughput number looked excellent.

This is why a quality check runs ahead of a capacity sweep rather than alongside it. A configuration that produces wrong output is not a fast configuration, it is a broken one, and no throughput figure derived from it is worth publishing. A vendor recommended flag that reduces answer quality is exactly the finding a buyer following that vendor cookbook needs to see.

Current results are overwhelmingly vLLM. A small number of models have been measured on SGLang as well, and those comparisons publish as a supplementary view rather than as a claim of broad cross stack coverage. Expanding that coverage across the roster is planned for subsequent releases, and until it lands, no result in this suite should be read as a general statement about how a model behaves on a stack it was not measured on.

Hardware support for numeric formats is the remaining axis. Platforms do not offer the same set, so a like for like comparison at a format only one platform supports is unavailable. Where that happens the comparison is reported at the highest format both platforms support, and the reduced precision result publishes separately with its own quality run attached.

15.6 What this means for a cost per token figure

A cost per token won at reduced precision is a cost per token for a different answer distribution. Where the parity check passes, the cost figure stands and the platform deserves the credit. Where it does not, the honest quantity is cost per correct task rather than cost per token, and the two can point in opposite directions.

© 2026 Signal65 PINNACLE. All rights reserved.

16. Live Simulation and Gradability

There is a real and interesting comparison to be drawn between measuring recorded work and measuring generated work, and the two are often treated as competitors when they are better understood as instruments with different reach.

A recording carries something a generated scenario cannot, which is the actual shape of production traffic. A generated scenario carries something a recording cannot, which is a correct answer. Both are worth having. Only one of them supports a statement about whether the work came out right.

> **You cannot grade a recording. A trace contains what happened, not what should have happened.**

16.1 What a live agentic session is

An agent receives a job, explores an environment it has not seen, chooses its own path, and produces a set of artifacts. It does not know how many rounds the job should take. It does not know which files matter. It does not know that a second copy of a record exists three directories away, or that the rule it needs sits in the fourth section of a handbook rather than the first.

What it does have is the freedom to be wrong in its own way, and that freedom is the measurement. A model that takes eighty rounds where another takes thirty, or that abandons a job partway, or that reconciles a contradiction incorrectly, has revealed something that no constrained replay would surface.

16.2 Why a recording cannot be graded

No ground truth was ever created for a recorded session. The data was whatever the data was, and no process wrote down the correct outcome before the work started. So there is no answer key inside the trace to check an output against.

This is why benchmarks built on recordings report throughput and cost and never report quality. It is not an oversight in those benchmarks and it is not a criticism of them. It is a property of the artifact they are built on. A replay can tell you exactly what a stack costs to serve a given shape of traffic, which is a genuinely useful thing to know.

16.3 Why measuring correct work requires generating it

The correct answer has to be created in the same moment as the environment. That is the only construction in which a deterministic grader has something to compare against, and it is the reason generation is a requirement rather than a convenience.

Three layers carry the argument.

1. **The work is inherited.** What the job consists of comes from an external occupational taxonomy, so the choice of tasks is not a Signal65 invention.
2. **The instance is fresh.** Every run regenerates, so there is nothing static to memorize or optimize toward.
3. **The answer is a generated answer key.** The correct outcome exists before the agent starts, which is what makes deterministic grading possible at all.

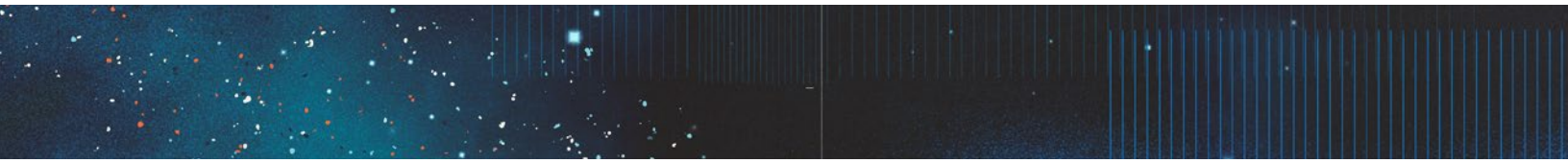
© 2026 Signal65 PINNACLE. All rights reserved.

Generation is the mechanism. Gradability is the point. The distinction that matters when comparing instruments is not whether a workload was recorded or generated, but whether the resulting measurement can be checked against a known correct outcome.

16.4 The consequence for capacity measurement

There is a second consequence that matters specifically for platform comparison. A model prevented from choosing its own path cannot show that it would have taken more rounds. A replay fixes the request schedule, so the traffic shape is identical regardless of which model is behind it, and the difference between an efficient model and a wasteful one disappears from the measurement entirely.

Under live execution that difference is preserved. A model that needs more rounds and more tokens to reach the same outcome consumes more of the platform, and the capacity number reflects it. That is the correct behavior, because it is what the platform will actually experience in production.



17. Reading the Results

The full result set publishes alongside this document. The three lenses below carry their own reading rules, with a small number of worked examples chosen to show the shape of an answer rather than to carry a headline.

17.1 Model intelligence

The model lens reports the PINNACLE Model Score, the five persona scores beneath it, token efficiency, and fabrication rate. It is hardware independent, which means it answers the first of the three sequential procurement questions and can be read before any infrastructure decision has been made.

The default presentation is a two axis view with quality on one axis and token efficiency on the other, because the two are independent and a single ranking hides the trade. A model can lead on quality and sit at the bottom on efficiency, and for a high volume deployment that combination is often the wrong buy. Persona selection re-ranks the table, which is where a leaderboard becomes an answer to a specific question rather than a general one.

17.2 Silicon capacity

The silicon lens reports what a platform sustains under load. Usable aggregate capacity at the gates, the concurrency at which it was reached, first token behavior at that rung, and the scaling and collapse profile across the sweep. Configuration is disclosed inline, including serving framework, precision, node shape, and software versions, because a capacity number without its configuration is not a measurement anyone can act on.

The example below shows the shape of a generational within vendor comparison across a model roster, which is the most straightforward reading the lens supports.

© 2026 Signal65 PINNACLE. All rights reserved.

Model	Prior gen usable agg	at N	Current gen usable agg	at N	Generational gain
Qwen3.5-9B	902	64	1701	192	1.89x
Qwen3.5-27B dense	552	32	883	104	1.60x
Gemma-4-31B dense	292	16	327	32	1.12x
Qwen3.5-35B-A3B	534	96	1220	128	2.28x
Qwen3.5-122B-A10B	348	32	909	128	2.61x
Qwen3.5-397B-A17B	195	20	501	64	2.58x

Figure 10: *Generational capacity comparison within one accelerator vendor. Measured. Usable aggregate is decode tokens per second at the highest concurrency rung clearing both gates, and at N is that concurrency. Gains are derived as the ratio of the two measured aggregates.*

Three things in that table drive the reading rules for this lens.

- **The gain is not uniform and the spread is a finding.** A generational comparison reported as one number would have collapsed a range from 1.12x to 2.61x into an average that describes none of the six cases. In our testing, sparse mixture models and larger models gained most, and the smallest gain landed on a dense mid size model.
- **Larger and sparser models gained most.** The two mixture of experts entries and the largest dense model post the biggest generational improvements, while the smallest gain lands on a mid size dense model. Architecture predicts how much a generational step is worth to a given deployment more than parameter count does.
- **Concurrency moved more than throughput in several rows.** Aggregate capacity and the concurrency at which it was reached are different numbers, and a platform that reaches a similar aggregate at three times the agent count is a materially different proposition for a service deployment.

17.3 Solution economics

The solution lens reports cost per correct task and the utilization crossover between renting and owning. Hosted models are priced at their full API bill, which counts input tokens, cached input at the vendor's discounted cache-read rate, and output tokens, all at published list rates. Open models are priced as the share of a rented node-hour the work consumes, at a stated utilization. Both sides count every token the job consumed, and the tokens spent on failed attempts are billed to the tasks that finished, which is what makes the two directly comparable. Pricing hosted models on output tokens alone understates their cost per correct task several times over on agentic work, because the input an agent re-reads on every round dominates the hosted bill, and an earlier draft of the tooling made exactly that error before it was corrected.

The crossover is the point at which amortized owned infrastructure becomes cheaper than per token pricing for a given volume of correct work. It moves with model choice, with token efficiency, and with utilization, and it publishes as a curve so a reader can locate their own volume on it.

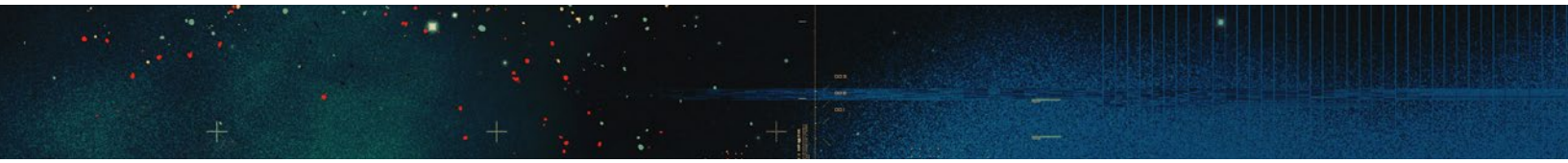
© 2026 Signal65 PINNACLE. All rights reserved.

17.4 Configuration disclosure

Every published figure carries the configuration that produced it. Model identifier and precision, serving framework and version, node shape and accelerator count, context length, concurrency rung, and any tuning applied along with where that tuning came from. This appears with the result rather than in an appendix, because a reader evaluating a number should not have to leave the number to find out what it means.

17.5 Coding agents

The coding results are published beside the enterprise results on the model and solution lenses, under the name Signal65 PINNACLE Coding Agent. The model lens reports the Coding Agent Score, a Work Breakdown of the two classes of work behind it, each class on its own with its interval, and a table with the share of samples answered right, token use and the Model Score for the same configuration. The solution lens reports a cost frontier against cost per attempt and cost per passed job, for Frontier Class Work and for Local Class Work and for hosted and open-weights models, with hosted cost from the provider's bill and open-weights cost from a rented B300 node. Section 21.9 sets out the reading rules, and the Coding Agent test FAQ answers the common questions.



18. Governance

Benchmark results are only as useful as the process that produced them, and a reader deserves to know that process before they are asked to rely on a number. Signal65 publishes its governance rules in advance for that reason, so that anyone can see how results are produced, refreshed, corrected, and disclosed, and can hold the published record against the stated rules.

Vendor involvement is part of that process rather than something worked around. Vendors know their platforms better than any outside party does, they are frequently the first to spot a misconfiguration, and results are stronger when they have had the opportunity to check the facts. The rules below set out where that involvement helps and where the line sits. The full document publishes alongside this paper.

Rule	What it states
Update cadence	How often results refresh, what triggers an out of cycle run, and how coverage of newly released models is handled.
Vendor review	That vendors may review results for factual accuracy before publication, the length of that window, and that the review carries no approval right and no veto over publication.
Configuration disclosure	That every published result carries its full configuration, software versions, and serving framework, and that any tuning applied is disclosed along with its source.
Re-run triggers	What causes a result to be re-measured, including software version changes, vendor supplied configuration corrections, and identified methodology errors.

© 2026 Signal65 PINNACLE. All rights reserved.

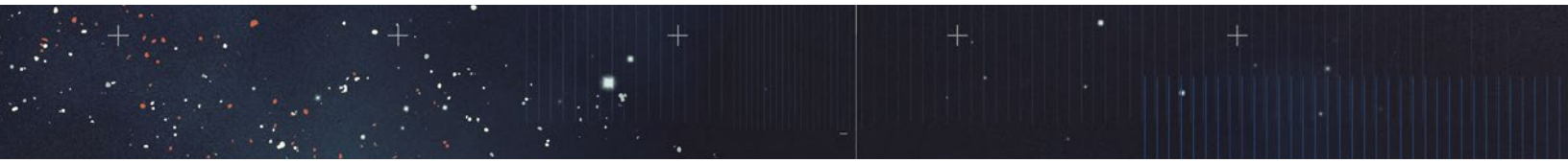
Corrections	How an error is corrected once published, where the correction appears, and that superseded figures stay visible rather than being silently replaced.
Reference changes	The conditions under which the reference model or reference platform moves, and what happens to prior results when it does.
Funding disclosure	Which results were commissioned and which Signal65 initiated, stated per result rather than as a blanket policy line.
Methodology versioning	What constitutes a breaking change, how versions are numbered, and how comparability across versions is described.
Test versions	That each Coding Agent test version is frozen at publication, that any change to a job or a grading rule ships as a new version with every model re-scored, and that prior results stay visible.
Assumed results	That a result resting on an assumption, such as a hosted model shown at 100% Local Class Work before its run, is disclosed on every view that carries it and replaced when the measurement exists.
Score ceiling	That a build with no failed work has no finite index, is listed as clearing every job with its pass rates shown, and ranks above every indexed build, for both the Model Score and the Coding Agent Score.

Figure 11: *What the governance document covers.*

Two of these carry more weight than the rest. The vendor review window exists because factual errors are worth catching before publication and vendors are frequently the party who can catch them. It carries no approval right, and a vendor disagreement about a result that is factually correct does not delay or alter publication. Funding disclosure is stated per result rather than as a policy line, because a blanket statement that Signal65 works with vendors tells a reader nothing about the specific number in front of them.

The coding results carry one disclosure of their own. Before publication, nine open models were first run with a per-request limit on output tokens that cut off long responses. Those runs were discarded and rerun without the limit, and only the rerun results are published. Grading rules and briefs were also revised during calibration, before any result was published, and the change log lists those revisions as calibration and not as corrections.

Underneath the rules sits the structural point. Scoring is deterministic and the methodology is published, so a result cannot be argued into a better number and any party with the hardware can reproduce it. Rules matter, and rules that rest on a reproducible measurement matter considerably more.



19. Limitations and What We Do Not Claim

Every item below is something a determined critic would otherwise find on their own.

© 2026 Signal65 PINNACLE. All rights reserved.

19.1 On coverage and grounding

- **The testable set is a Signal65 determination.** The US Department of Labor built the taxonomy. The judgment that 26 of the 41 activities are agent testable is ours, reached through the two gates in chapter 6, and no government body reviewed or endorsed it.
- **Coverage is breadth of activity type, not depth within an occupation.** No score in this suite certifies a model for a named job, and no coverage percentage should be read as a share of any occupation or of the economy.
- **Coverage across the 26 is not uniform.** Eighteen are exercised directly and eight appear only as accompanying work products.
- **Four activities are absent because we cannot score them.** Not because agents cannot perform them. The outcome lives in another person response and there is no deterministic key.
- **The activities ground task design and are not a scoring dimension.** No result is attributed to an individual activity, because the activities overlap and one scenario exercises several at once.
- **The suite measures enterprise agentic work only.** It does not measure depth of world knowledge, quality of prose, any modality beyond text, conversational range, or progress toward general artificial intelligence. Section 2.5 sets out the boundary in full, and rankings here should be expected to differ from rankings on general capability benchmarks.
- **The scenario suite is 7 items.** Three as-found, three governed counterparts, one governed only. That is a small suite, sampled deeply at 40 samples per scenario per model, and depth of sampling is not the same thing as breadth of scenario coverage.

19.2 On scoring

- **Deterministic scoring cannot grade open ended output.** Anything requiring qualitative assessment is out of scope by construction. That is the price of removing model judges and it is a real price.
- **Sandboxes are synthetic.** They are generated rather than drawn from real enterprise systems, which is the cost of contamination immunity and a generated answer key. The as-found environment is constructed to be realistic and it is still constructed.
- **The persona layer is a weighting scheme over shared measurements.** Personas do not run distinct scenarios and do not run at distinct context lengths. Every difference between two persona scores for the same model is a difference in weighting, applied to identical underlying measurements.
- **Retrieval is measured at one context length.** All retrieval, aggregation, and fabrication figures are taken at 128K. Models that hold accuracy well beyond that length and models that collapse just past it are not distinguished by any number in this suite, and the published context work in section 10.6 is the reference for that behavior.
- **Fabrication rate is measured on generated absences.** The probes test entities that were never generated and fields deliberately left empty. That is a controlled form of the failure. It does not capture fabrication arising from genuine ambiguity in source material, where two documents support different answers and neither is absent.
- **We do not claim to predict on the job performance.** We claim verified task completion under stated conditions.

© 2026 Signal65 PINNACLE. All rights reserved.

19.3 On performance

- **The three to one weighting is judgment.** A different buyer would reasonably choose differently, and the inputs are published so they can.
- **Tasks per hour is derived rather than timed.** The transform is stated in section 13.5 and no published figure rests on a wall clock.
- **Launch capacity results run one serving framework and one node shape.** A different framework or a different node shape can change the ranking, and results should not be read as a general property of the silicon.
- **Results reflect software versions at a moment in time.** Serving stacks move quickly, and a result is a measurement of a stack on a date rather than a permanent property of a platform.
- **The context cross check covers two tokenizer families.** That is a small sample for a load bearing assumption, and it is stated in section 13.6 rather than left to be discovered.

19.4 On independence

Signal65 is an independent research and analysis firm, and independence in this business is demonstrated rather than asserted. Ours rests on four things a reader can check directly.

- **The methodology is open.** Everything required to understand how a number was produced is in this document and the governance rules that accompany it.
- **The scoring is deterministic.** No result in this suite is a matter of opinion, ours or anyone else. A score cannot be argued into a better number, by a vendor or by us.
- **The configuration is disclosed with every result.** A reader who doubts a figure knows exactly what produced it and can say precisely what they would change.
- **Any party with the hardware can reproduce it.** That is the strongest guarantee available in benchmarking, and it is available here because the suite was built to make it available.

Signal65 works commercially with technology vendors, including vendors whose products appear in these results, and that funding relationship is disclosed per result rather than as a blanket statement. Reproducibility is what makes the disclosure meaningful.

19.5 On the coding test

- **One task family.** Every coding job asks the model to build a PICARD generator in Python. The jobs vary widely in business content and in what has to be built, and they are still one shape of engineering work. Debugging an existing service, reviewing a change and working across languages are not measured.
- **The engine is new to every model.** The test runs on PICARD v2, which has not been published. Version 1 is public, and a model that has seen it starts with the advantage experience of a related library gives an engineer, and no more, because every attempt is graded on seeds rendered after the attempt ends.
- **Attempt counts are modest.** Thirty attempts per hosted configuration leaves intervals of about 30 points near the top of the table, and the leading configurations read as a tier.

© 2026 Signal65 PINNACLE. All rights reserved.

- **Some Local Class results are not yet measured.** Hosted frontier configurations without a Local Class run are shown at 100% Local Class Work until measured, which raises their score most when their Frontier Class pass rate is low.
- **Open-model cost rests on enterprise-agent throughput.** It is a bridge until a coding capacity run replaces it, and open-weights models with no capacity run are not priced at all.
- **The structural limits are judgment.** The 70% repetition limit and the 2% limit on revealing details are stated thresholds chosen by the test designers, and a different threshold would move some verdicts near the line.



20. Roadmap

The suite described in this paper is the first release rather than the finished instrument, and the work below is where it goes next. Several of these are already underway and all of them are open to input from the partners reading this document, which is a large part of why the roadmap is published at all. Nothing here carries a date and nothing here ships at launch.

20.1 A sub-agent environment

A third environment, governed but much shorter, where a scenario is one or two steps rather than a full workflow. A large share of real deployments are built this way, with an orchestrator decomposing work and calling narrow agents to execute pieces of it, and the current suite does not measure that shape directly.

20.2 Broader coverage

More of the testable set exercised directly rather than through accompanying work products, a broader model roster, and coverage of new models closer to their release date.

20.3 Decomposed workflows

Measuring an orchestrator routing work to smaller models running narrow subtasks, which is a materially different economic proposition from running one large model end to end and is not comparable to it under the current scoring.

20.4 Solution and cluster validation

Validation across the full stack rather than a single node, including the difference between first light and steady state, which is where a meaningful share of deployed capacity is actually lost.

20.5 Cross stack coverage

Systematic measurement of the same models across serving frameworks, so that the effect of the stack on both throughput and answer quality becomes a published axis rather than a supplementary view. Early results already show that the choice of inference stack changes both, and the second of those is the one buyers do not expect.

© 2026 Signal65 PINNACLE. All rights reserved.

20.6 Energy

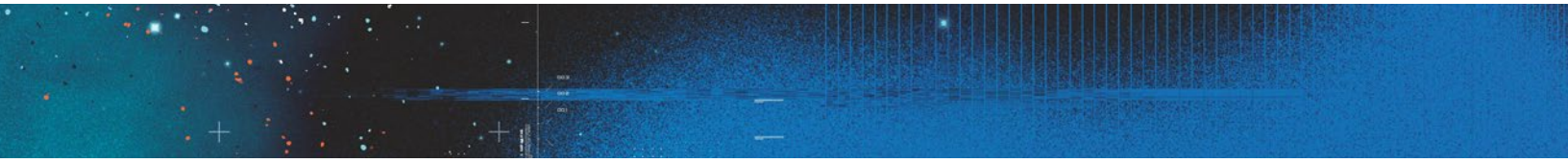
Performance per watt from measured device power rather than from thermal design power, which assumes full utilization and describes a cooling envelope rather than a workload. The instrumentation exists on both major accelerator families and the work is scoped rather than scheduled.

20.7 Distributed and CPU side measurement

Multi node and split inference shapes, and a companion instrument that isolates the CPU contribution to agentic performance. Early proof of concept work shows that the ranking between two CPUs can invert as agent concurrency rises, with peak single thread performance winning while cores are idle and core count and memory headroom deciding once every core is busy. That is a sensitivity curve rather than a single score, and it is the correct shape for the question.

20.8 Coding agents

A harder set of Frontier Class jobs before the leading configurations saturate the current one, more coding task shapes and languages beyond Python, and a capacity measurement with coding agents on GPU platforms, which also replaces the open-model cost basis with a measured one. At the next major version the coding score is designed to join the headline alongside the enterprise measurements.



21. The Coding Agent Test

Coding agents are among the most widely deployed agent workloads in the enterprise, and the agentic suite described in chapter 7 does not measure them. Its jobs call for routine Python and SQL in the course of business work, but none of them asks a model to engineer software inside a codebase it has never seen, which is what a coding agent is hired to do. The Signal65 PINNACLE Coding Agent test measures that work directly. It runs its own jobs, grades them by running what the model built, and reports a separate number, the PINNACLE Coding Agent Score, that sits beside the Model Score and does not enter it.

The test was developed in close partnership with Kamiwaza and is built on PICARD, the same harness that generates and grades the enterprise jobs, so the design principles in chapter 4 carry over unchanged. Scoring is deterministic, every instance is generated fresh, and no model and no human judges the output.

21.1 Why a separate test, and why it is not a persona

The persona layer in chapter 11 reweights five shared measurements and runs no jobs of its own, a point section 11.4 makes deliberately. Coding work cannot be expressed that way, because none of the five measurements observes it. The Coding Agent test therefore stands as its own measurement with its own score, and it is described as a test, not a sixth persona, so that the persona layer keeps the meaning section 11.4 gives it.

It also stays out of the PINNACLE Model Score for this major version. Folding a new measurement into the headline would re-rank every published result, and it would require every model on the enterprise board to be run on the coding jobs before any coding result could appear. A separate score adds information without moving anything already published. The two are designed to combine when the next major version restates the headline, and the coding score is built in the same units as the Model Score so that the combination does not require a new scale.

© 2026 Signal65 PINNACLE. All rights reserved.

The separation matters to a buyer because the two rankings do not match. In our testing they agree broadly across the field and part ways near the top, where most buying decisions sit. The two scores sit on different scales, so the comparison is one of position, and only configurations run at the same setting on both are compared. The leading hosted model on coding sits several places lower on enterprise work, a hosted model in the top three on enterprise work sits outside the top fifteen on coding, and an open-weights model from the lower third of the enterprise table sits in the coding top ten, passing every Local Class job and about a third of Frontier Class ones. A model chosen for governed business work is not, on that evidence alone, the right model for an engineering team.

21.2 The job

Every coding job is the same kind of engineering task. The model is given a business brief, the full source and documentation of PICARD v2, a version of the engine that has not been published, and a working environment, and it is asked to build the generator for a new PICARD job. Concretely it writes a Python program that emits a PICARD test configuration, which in turn generates fresh sandbox data for every sample and computes that sample's answer key from the same data. The prompt carries one instruction above the others, that the engine generates the data and the model does not. The documentation alone runs to about 74K tokens.

The task was chosen because it reproduces the position a coding agent is in on its first day inside an enterprise codebase. The engine is large and unfamiliar, the documentation is long, and the important rules live partly in the source. The output has to be exactly what the brief specifies, it has to be correct on inputs nobody has looked at yet, and it has to hold up as a design, not as a set of answers that happen to be right today. Figure 12 maps each demand of the job to the skill it exercises.

What the job demands	The skill it exercises
Find what is needed in a large engine's documentation and source	Working in a large, unfamiliar codebase
Produce files, schemas and lines exactly as the brief specifies	Following a specification precisely
Write a generator, then test it before finishing	Building tooling and verifying one's own work
Get boundary rules right ("more than", "on or before")	Edge-case correctness
Keep answer scaffolding out of what the agent under test can see	Clean interfaces that do not leak internals
Make data whose structure varies, not only its values	Design judgment from the principles of the system

Figure 12: *What the coding job demands, and the engineering skill behind each demand.*

© 2026 Signal65 PINNACLE. All rights reserved.

Building a test for a test engine is not circular. The model never sees the grader, the grader never consults a model, and the artifact the model produces is ordinary software, a program that generates structured data and computes answers from it. That the software happens to describe a PICARD job is what makes it gradable, because PICARD can render the program on seeds chosen after the attempt ends and check every result in code.

21.3 Two kinds of job

The test has two parts that share the task, the harness and the grading, and differ in how much the brief says. Frontier Class Work, the open-ended and more difficult half, gives a long business brief that states what to build and leaves the design to the model, the way a real ticket does. Local Class Work, the fully scoped and easier half, gives a short brief and writes every design requirement into it, including which items must vary between samples. The first separates the frontier. The second separates the large share of the field that passes no Frontier Class job, and it is where models sized for a local workstation show what they can do.

Frontier Class Work is the reason the test exists. Anyone who has used a coding agent has seen it produce a program that runs and still misses the point, a design no competent engineer would choose because it ignores how the system is meant to work. Those failures are not the kind a human engineer makes, and they are the ones that cost a team the most time. The open-ended jobs measure whether a model understands the principles behind the engine and applies them without being told, the design judgment and common sense that separate the right program from a working one.

	Frontier Class Work	Local Class Work
Briefs	10 business scenarios with many input files and several deliverables, written the way a real ticket is	10 scenarios with one to three kinds of input and one or a few deliverables, each one fully hand-held
Design requirements	Implicit. The model has to infer them from the principles of the system.	Explicit. A list of variation requirements ends every brief.
Samples judged per attempt	25 unseen seeds × 8 samples = 200	25 unseen seeds × 5 samples = 125, or 100 on two briefs. Easier jobs vary less, so fewer samples judge them reliably.
Per-sample checks	Identical. Exact files, schemas and pinned lines, guarantees as SQL counts, every answer recomputed from the rendered files.	Identical
Checks across samples	Repeated answers, counts that never vary, surface tells, realistic mixes	Only what the brief states. Named items must vary, and no deliverable may repeat in more than 70% of samples.
Attempts per model	30 for hosted models, 60 for most open models, 300 for three open models	60 per model, 300 for three open models
Pass	Every sample passes every check	Every sample passes every check

Figure 13: *The two kinds of coding job. Both run in the same harness and are graded by the same code.*

Both sets cover ten briefs. The Frontier Class briefs are business scenarios drawn from the same kinds of work the enterprise suite covers, and the Local Class briefs climb from one CSV and one number to several folders, a policy document and files the task does not use.

© 2026 Signal65 PINNACLE. All rights reserved.

Frontier Class briefs	Local Class briefs
Bonus round	Vehicles above a stated mileage, from three depot folders with different column names
Vendor payment audit	One department's total salary
Defect root cause	High-priority tickets older than a stated number of days, with each ticket's age
Adverse event reporting	Employees rated Exceeds, from a roster and review memos
Crew duty compliance	Large expenses across three department folders
Cycle count reconciliation	Refunded orders across three regional databases with clutter
Claims coverage	Large expenses with fields and files the task does not use
Payroll tax	SLA breaches against a policy threshold, with one outbox message per team
Access review	Invoice totals from pipe-delimited line items
Loan delinquency	Claims in two layouts that need director approval

Figure 14: *The ten briefs in each set.*

One Local Class brief, reproduced in full, shows what "every requirement stated" means in practice. Its last block is the part a Frontier Class brief never states. The data behind it regenerates on every run, so publishing the brief gives a model nothing to memorize.

Benchmark scenario, small. Data: staff.csv with columns name, department, salary; at least 20 rows, regenerated with different content per sample; every department should appear on several rows. Question: "What is the total salary of the "<department>" department?" The department is picked per sample from the departments that actually occur in that sample's CSV. Answer: the agent writes answer.json beside the CSV containing {"total": N} where N is the exact integer sum. Variation requirements: the department named in the question must differ between samples; the content of staff.csv must differ between samples; no deliverable may have the same content in more than 70% of all samples, pooled over many runs.

Figure 15: *A Local Class brief, abridged for layout. The Frontier Class version of the same scenario would leave the variation requirements for the model to work out.*

© 2026 Signal65 PINNACLE. All rights reserved.

21.4 The working environment

Every attempt runs in the same agent harness, a container with a shell, Python and the engine, driven by an open-source coding agent. The model chooses its own path, reads what it decides to read, runs and tests its own code, and decides for itself when the job is done. Credentials for hosted models never enter the container. A proxy outside it holds them and records which model answered every request, so a result cannot be attributed to the wrong model.

Setting	Value
Agent harness	opencode 1.18.32 in a container with a shell, Python and the PICARD v2 engine
Network	Available inside the container. Model credentials held by a proxy outside it.
Time limit	7,200 seconds per attempt. An attempt that reaches the limit is stopped and graded on the files it has written.
Context	No minimum context length. The harness compacts the session automatically when it nears the model's window.
Generation length	No limit on output tokens per request.
Effort	Set per configuration and published with every result, using the same labels as the enterprise views.
Hosted models	Provider APIs at the endpoint and effort shown with each result.
Open models	vLLM 0.28 to 0.29 builds on NVIDIA GB300, at native BF16 unless the result label says FP8, MXFP8, MXFP4 or NVFP4. Nodes, parallelism, served context length and speculative decoding publish per model.

Figure 16: *The coding agent environment, identical for every model.*

21.5 How an attempt is graded

Grading runs in four stages, all of them code. The configuration the model built is rendered under 25 seeds it never saw, producing 200 complete sandboxes on a Frontier Class job and 125 on a Local Class one, each with its own data and the model's own answer key. Every sample is then checked on its own, the samples are pooled and checked together, and the attempt is scored.

Per-sample checks. The answer sheet confirms that the files exist with the exact names and schemas the brief requires, that the pinned lines appear verbatim, that the question says what it must, and that every guarantee in the brief holds as a SQL count over the rendered data. The sheet then recomputes every answer independently from the rendered files and compares it with the model's key. If the two disagree, the model's key is wrong, whatever the reason.

The check catches errors nobody anticipated. In one sample a payment's invoice was authorized for 2,900 with a materiality of 38%, an allowed excess of exactly 1,102, and the payment was 4,002, exactly on the line. The brief defines over-authorized as more than the threshold, so the payment was clean. The model's key used "at least" and flagged it. Four of its 200 samples landed on that boundary, and all four were wrong.

© 2026 Signal65 PINNACLE. All rights reserved.

Checks across samples. A configuration can produce fresh values in every sample while keeping the same skeleton, and a skeleton can be memorized. The pooled checks ask questions no single sample can answer. Whether one value of a deliverable appears in more than 70% of samples. Whether the counts and parameters of the situation change from sample to sample instead of being "always exactly one". Whether a surface detail gives the answer away, measured as the share of uncertainty about a row's class that the detail removes beyond chance, with a limit of 2%. And, where the brief asks for a believable mix, whether the pooled rate holds within the brief's range. Local Class Work checks only the variation the brief states.

One Frontier Class configuration sent every planted breach to the same vendor. The values changed in every sample, but that vendor's payments were findings in 982 of 982 cases across all 200 samples. A model trained on leaked samples would learn that anything paid to that vendor is a finding, and the attempt failed.

Pass. An attempt passes only if every sample passes every check. The share of samples answered right is published beside the pass rate and never folded into it, and the distance between the two carries the same kind of signal as the distance between the two scoring modes in section 12.3. In our testing one hosted model answers 97% of Frontier Class samples correctly and passes none of its 30 attempts, because the data it builds never varies in structure.

Three commitments keep the grading fair as well as strict. Every brief has a reference solution that passes all of its samples with margin. Every check traces to the brief or to a principle stated in the engine documentation every model reads, so a sheet stricter than its brief is treated as a bug. And when nearly every model fails the same check, the strongest included, the sheet is assumed wrong until it is shown to be right. Every scoring rule ships with tests, and each test is proven by mutation, breaking the rule on purpose to confirm the test fails.

> *A generator whose design never changes passes its samples and fails the job, because the next engineer could predict every sample it will ever make.*

21.6 Scoring

Each part of the test reports a pass rate, the share of attempts that passed. The two combine into a composite on a 0 to 100 scale, and the composite is indexed to the reference model in the same way as the Model Score.

> *Composite C = 50 × Local Class pass rate + 50 × Frontier Class pass rate. PINNACLE Coding Agent Score = 1,000 × (100 − C reference) ÷ (100 − C model).*

The reference is Gemma-4-31B-it with thinking enabled, the same model that anchors the Model Score in section 12.5. In our testing it passes 21 of 60 Local Class attempts (35%) and none of 30 Frontier Class attempts, for a reference composite of 17.5. A Coding Agent Score of 2,000 therefore means half the weighted failed coding jobs of the reference, which is how the Model Score already reads. Figure 17 works the formula through five cases.

© 2026 Signal65 PINNACLE. All rights reserved.

Configuration	Local Class	Frontier Class	Composite	Score
Gemma-4-31B-it · thinking (reference)	35%	0%	17.5	1,000
A model that fails every job	0%	0%	0.0	825
A model that passes every Local Class job and no Frontier Class job	100%	0%	50.0	1,650
A local-size open model	93%	7%	49.8	1,645
An open model near the top of the open-weights field	100%	35%	67.5	2,538

Figure 17: *The Coding Agent Score worked through five cases. The last two rows are measured configurations, in our testing.*

The scale has two properties to note. A model that fails every job scores 825, not zero, because the reference itself fails most jobs and the index measures failed work relative to it. At the other end the score rises without bound as failed jobs approach zero, which is true of the Model Score as well, and it has no finite value for a build that passes every attempt on both classes of work. Such a build is listed as "clears every job" with both pass rates shown and no index, and it ranks above every indexed build. The rule is stated now, while no model is near it. The leading configurations in our testing pass about two thirds of Frontier Class jobs.

The weighting is an editorial choice. Half and half states that the everyday coding a local assistant does and the open-ended engineering a frontier agent does matter equally to the score. A different reader could weigh them differently, and the two pass rates are published so that anyone can. Two observations support the choice. Of the 49 configurations measured on both parts, 35 pass no Frontier Class job, so a score built on Frontier Class jobs alone would report most local and mid-size models as having no coding value at all, which the Local Class results contradict. And the ordering barely depends on the weights. Across splits from 30/70 to 70/30 the rank correlation with the 50/50 order stays at 0.998 or above, so the weighting sets the spread of the scale and leaves the ranking alone.

Why the score counts whole jobs. The same argument as section 12.2 applies. A configuration that is right on 199 samples and wrong on one has shipped a generator someone has to debug, so only a passed job counts, and the share of samples answered right is published beside the score, never blended into it. A geometric mean of the two pass rates was rejected because it falls to zero for any model with no Frontier Class pass, which is most of the field today.

Hosted models without a Local Class run. Some hosted frontier configurations have Frontier Class results and no Local Class run at publication. They are shown at 100% Local Class Work, the footer of every coding view says so, and the measured result replaces the assumption when the run completes. The assumption rests on the Local Class results in hand. Every configuration passing more than a quarter of Frontier Class jobs has passed 95% or more of Local Class ones, while the older hosted models measured on both parts pass 55% to 93%. It moves a score most for configurations with low Frontier Class pass rates, which are first in line for measured runs.

© 2026 Signal65 PINNACLE. All rights reserved.

21.7 Attempts and confidence

Attempt counts differ by configuration. Hosted models run three attempts per Frontier Class brief, 30 in all, most open models run six, and three open models ran 30 per brief to measure run-to-run spread directly. Every Local Class result in the main run uses six attempts per brief. Every pass rate publishes with its 95% interval, computed with the Wilson method and published beside every pass rate, so the precision of a row is visible next to the row.

The intervals are wide enough to change how the top of the table reads. In our testing 21 passes in 30 attempts spans 52% to 83%, 21 in 60 spans 24% to 48%, and 84 in 300 spans 23% to 33%. Configurations whose intervals overlap are presented as one tier. The reference carries the same uncertainty, and because every score is indexed to it, additional attempts on the reference narrow every score at once.

21.8 Token use and cost

Coding jobs are long. A passed Frontier Class attempt runs dozens of rounds, re-reads a context of 100K to 250K tokens on each one, and can generate more than 100K output tokens in the process. Token use is reported per attempt, as rounds, input tokens, output tokens with hidden reasoning tokens counted, and peak context, for each class of work.

Cost follows the rules of section 17.3 so the two cost views can be read side by side. Hosted models are priced at their full API bill at standard list rates, counting fresh input, cache writes where the provider bills them, cache reads at the discounted rate, and output. Open models are priced as a rented 8×B300 node at $70 an hour, the median on-demand price researched October 19th, 2026, for the node-hours an attempt's output tokens take at the throughput the PINNACLE agent harness measured for that model on that node. Cost per attempt is reported for each class of work separately. Cost per passed job is the cost of every attempt in that class divided by the number that passed, so the tokens spent on failed attempts are billed to the jobs that finished. It is published once a build has at least two passed jobs in the class, so that one pass cannot carry the bill for every failed attempt, and builds below that line show their cost per attempt only. The cost frontier plots the Coding Agent Score against cost per attempt and not cost per passed job, because the score already counts failed jobs and dividing the cost by the pass rate as well would count them twice.

The open-model throughput is a bridge at release, and the footer says so. It comes from capacity measurements taken with enterprise agents, and coding agents carry larger contexts per agent, so fewer of them fit on a node at once. A capacity run with coding agents on the reference box replaces it. An open-weights model that has no capacity run at all is scored on the model lens and not priced on the solution lens until one exists, and the footer names it. Every hosted cost comes from provider-reported usage with the fresh, cache-read and cache-write split.

21.9 Reading the coding results

- **Read the interval before the rank.** Rows whose intervals overlap are one tier.
- **Read the two parts together.** A model can pass no Frontier Class job and nearly every Local Class one, which describes a capable assistant for well-specified work that does not yet engineer from principles. The Work Breakdown view shows both halves of every score.
- **Compare matching configurations.** Effort settings change coding results substantially, and any comparison with enterprise results pairs only configurations run at the same setting on both.
- **Check the footer.** Hosted models shown at 100% Local Class Work without a measured run, open-weights models scored but not yet priced, and the enterprise-agent basis of open-model cost are all stated there.

The Coding Agent test FAQ at pinnacle.signal65.com/coding-agent-faq, answers the questions this chapter raises in plain terms, with worked examples from the published results.

© 2026 Signal65 PINNACLE. All rights reserved.

22. What This Benchmark Is For

An enterprise evaluating agentic AI is not buying a model. It is buying an outcome produced repeatedly, correctly, and at a cost it can forecast. That is a more specific question than the one the field usually asks, and it is the question this suite was built to answer.

That focus is what makes the numbers usable. Because the scope is fixed and stated, every number here is checkable. Scoring is code against a generated answer key before the agent started. Environments regenerate, so nothing static exists to memorize. Configuration travels with every published result. Any party with the hardware can reproduce the measurement and disagree with it specifically rather than in general terms.

What comes out of that is a set of numbers a buyer can act on. Whether a model finishes the job on data as they actually hold it. How far it falls when the data is not clean. How often it invents an answer that was never there. How many agents a platform sustains while all of them stay useful. What one correct outcome costs, and at what volume owning the infrastructure beats renting it.

Those answers will sometimes disagree with the general capability leaderboards, and the disagreement is informative rather than embarrassing. A model that finishes enterprise work reliably at a fraction of the token cost is the better buy for that work even when a larger model reasons more impressively about everything else. Signal65 PINNACLE exists to make that case in numbers rather than in argument.

© 2026 Signal65 PINNACLE. All rights reserved.

Important Information About this Report

CONTRIBUTORS

Ryan Shrout
President and GM | Signal65

JV Roig
Senior Generative AI Platform Engineer | Kamiwaza

PUBLISHER

Ryan Shrout
President and GM | Signal65

INQUIRIES

Contact us if you would like to discuss this document and Signal65 will respond promptly.

CITATIONS

This paper can be cited by accredited press and analysts, but must be cited in-context, displaying author name, author title, and Signal65. Non-press and non-analysts must receive prior written permission by Signal65 for any citations.

LICENSING

This document, including any supporting materials, is owned by Signal65. This publication may not be reproduced, distributed, or shared in any form without the prior written permission of Signal65.

DISCLOSURES

Signal65 provides research, analysis, advising, and consulting to many high-tech companies, including those mentioned in this paper. No employees at the firm hold any equity positions with any companies cited in this document. Funding status is disclosed per published result under the governance rules described in chapter 18.

ABOUT SIGNAL65

Signal65 is an independent research, analysis, and advisory firm, focused on digital innovation and market-disrupting technologies and trends. Every day our analysts, researchers, and advisors help business leaders from around the world anticipate tectonic shifts in their industries and leverage disruptive innovation to either gain or maintain a competitive advantage in their markets.

For more information, visit signal65.com
or contact research@signal65.com

signal65

IN PARTNERSHIP WITH

KAMI WAZA

Document Control

Version	Date	What changed
1.0	August 31, 2026	First published version
1.1	October 28, 2026	Adds chapter 21, the Coding Agent test, with amendments to chapters 1, 3, 11, 12, 17, 18, 19 and 20. Former chapter 21 becomes chapter 22.

signal65 PINNACLE

CONTACT INFORMATION
Signal65 PINNACLE I pinnacle.signal65.com

© 2026 Signal65 PINNACLE. All rights reserved.