

# The Always-On Agent

## How Persistent AI Workloads Expand Compute Demand

Research Paper | September 30, 2026

**Abstract.** AI agents change the scale of inference demand by turning a request for an answer into a delegated program of work. A single objective can generate repeated model calls, growing context, tool execution, verification, and parallel attempts; scheduled and event-driven execution can also increase how often that work occurs. This paper synthesizes production telemetry, agent benchmarks, and serving-system research to explain how those changes translate into future infrastructure requirements. It distinguishes accelerator work from elapsed task duration, and separates three physical accounts: inference flow, retained-state capacity, and state-transfer bandwidth. A worked example shows why context replay can grow much faster than the number of steps, while also showing how effective caching sharply reduces that burden without eliminating decoding or attention to prior context. Transparent sensitivity cases identify the condition under which growth in delegated work exceeds improvements in execution efficiency. The central conclusion is that persistent agents create a credible path to substantially higher aggregate compute demand, but not because every agent occupies a GPU continuously. Demand expands when more objectives become economically worth delegating and when the resources spent to complete those objectives grow faster than efficiency improves. The resulting pressure falls on accelerators, memory, networks, and supporting systems in different proportions. Capacity planning therefore needs to move from tokens per request to verified outcomes, resource use per outcome, and peak concurrent demand.

---

**Keywords:** agentic inference; compute demand; test-time computation; context replay; KV cache; memory hierarchy; concurrency; outcome economics.

**Research scope.** A primary-source synthesis with original analytical examples, not a new production experiment or a market-size forecast. Evidence is reviewed through September 30, 2026. Numerical scenarios are explicitly illustrative.

**Central proposition** The long-run demand driver is the expansion of delegated work, not the permanent reservation of hardware for every persistent agent.

# The compute-demand thesis in one page

*Agents loosen two constraints on AI consumption: the amount of work inside one request, and the need for a person to initiate every subsequent step.*

---

## 40.6

Mean LLM calls per **session** in a large Copilot trace study; the median was 15. First-party production telemetry. [1]

---

## $\sim 15\times$

Token use of multi-agent research relative to chat in Anthropic’s internal observations. Not a universal workload multiplier. [2]

---

## 33

Median turns in 610 Codex / SWE-bench Pro traces analyzed by the vLLM and Mooncake teams. A coding trace set, not a fleet average. [3]

## The strongest case for higher demand

The relevant shift is from answering questions to completing work. An answer may finish after one model response. A delegated task can continue through investigation, execution, testing, correction, and delivery. Once work can be initiated by a schedule or an event, aggregate usage need not remain proportional to the number of prompts people type. This is a demand mechanism, not proof of a particular adoption rate.

Three margins determine the outcome. **Volume** is the number of accepted outcomes users and organizations seek. **Intensity** is the work needed to deliver each outcome, including unsuccessful paths. **Efficiency** is how effectively models and infrastructure execute that work. The forward demand case becomes strong when the first two grow faster, in combination, than the third.

## Why better efficiency need not end the infrastructure cycle

Caching, smaller models, and better scheduling can reduce resources for a fixed task. Their success can also make previously uneconomic tasks affordable. These are different effects and must be measured separately. Neither “tokens are getting cheaper” nor “agents use more tokens” is sufficient to establish the direction of physical compute demand.

## What changes even when compute is saved

A cached agent can require less repeated prompt processing while retaining more state and transferring that state between machines. A parallel agent can finish earlier while imposing a higher instantaneous load. A waiting agent can release its accelerator while preserving a sandbox or checkpoint. The infrastructure mix changes with the execution pattern; it does not follow a single token-volume multiplier.

---

The decisive question is not whether agents are always running. It is how much additional work becomes worth running, and which resources are required to finish it reliably.

---

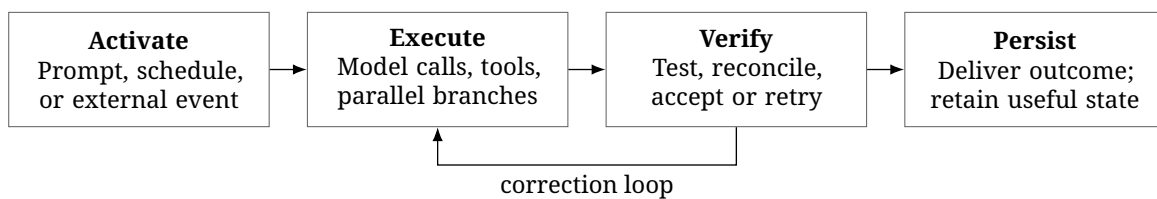
**Reading guide.** Sections 1–4 explain the workload change. Sections 5–7 connect it to physical resources. Sections 8–11 examine future demand and capacity. Sections 12–13 state the countercase and the measurements that would resolve it.

# 1. From answers to delegated work

*Persistence increases the opportunity to consume compute; autonomy determines how much of that opportunity becomes actual work.*

For this paper, an *agent* is a system that uses a model to select or execute steps toward an objective, usually with tools and feedback. A *trajectory* is the complete execution path, including branches, waits, retries, and verification. A *persistent* agent can preserve enough state to continue beyond a single synchronous exchange. None of these definitions requires uninterrupted inference.

An illustrative reporting workflow makes the distinction concrete. A chat assistant explains a discrepancy when asked. A delegated workflow retrieves source records, reconciles them, identifies an exception, requests missing information, resumes after the reply, and produces a checked report. Its final visible output might still be a page of text. The work behind that page is a program, not a longer answer.



*Exhibit 1. A delegated outcome can contain repeated model and non-model work. Schematic analysis; widths do not represent time or cost.*

## Three distinct routes to more demand

**More work per activation.** A request can trigger a chain of actions rather than a single response. The chain may expand because the objective is harder, because verification is more thorough, or because execution is inefficient. These explanations have different economic value even when they consume similar resources.

**More activations per user or business process.** A schedule or event can initiate work without a new typed prompt. OpenAI’s documentation describes recurring and supported event-triggered tasks, while its background API mode supports asynchronous long-running responses. These establish available execution modes, not their prevalence. [7][8]

**A wider set of delegable objectives.** Better reliability can make longer or more consequential tasks worth attempting. METR’s work measures task difficulty using the time a human expert would need; that capability horizon is not the model’s own runtime or a measure of deployed demand. It provides a way to study expanding task scope without confusing capability with consumption. [9]

## The demand frontier moves before the fleet mix is known

The forward implication is conditional but substantial: a system that can economically complete work has a larger potential consumption base than a system used only to explain work. Human attention, organizational budgets, authorization, and verification remain constraints. Persistence removes neither those constraints nor the need to earn value from each activation. It does, however, weaken the one-prompt, one-response ceiling on resource use.

## 2. What the evidence actually establishes

*The strongest evidence concerns multi-step execution. Evidence about universal adoption, always-on utilization, and aggregate retained memory remains incomplete.*

### SELECTED PRIMARY-SOURCE OBSERVATIONS

| Source and setting                 | Observed workload characteristic                                                                                                                    | Interpretation boundary                                                                           |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Copilot production traces [1]      | Table 4 reports 40.6 mean calls per session, but 6.6 per user-turn. Sessions have median duration 4.2 minutes and mean 62.6 minutes.                | A session can contain multiple objectives. These are not calls or GPU-minutes per completed task. |
| Anthropic research system [2]      | Internal observations put agent token use near 4× chat and multi-agent use near 15×.                                                                | Research-oriented architecture; no quality-matched, cross-industry demand multiplier.             |
| vLLM / Mooncake trace analysis [3] | Across 610 coding traces, context approaches 80,000 tokens by turn 30; median inter-turn delay is 5.2 seconds; the 99th percentile is 81.4 seconds. | Demonstrates context growth and gaps between calls; not continuously occupied accelerator time.   |
| SWE-Serve benchmark [4]            | Across its top-per-model configurations, mean task runtimes range from 17.6 to 99.9 minutes.                                                        | Engineering benchmark on 53 tasks; runtime includes tool and environment work.                    |
| Bai et al., coding study [5]       | Repeated runs on the same task vary by as much as 30× in token consumption across the study's evaluated settings.                                   | A long cost tail is not necessarily a long useful-computation tail.                               |
| AgentSysBench [6]                  | Non-LLM components dominate latency in 5 of 10 applications; sandbox working sets peak at 28 GB per session.                                        | Agent infrastructure extends beyond model inference; peaks are not averages.                      |

*Exhibit 2.* A selected evidence ledger, not a representative census. Results remain local to the stated systems, tasks, models, and measurement definitions.

### Method and evidence quality

This paper uses disclosed production measurements to establish workload behavior, controlled evaluations to examine mechanisms, and product documentation only to establish capability. Provider-authored measurements are identified as such; they are not independent audits. Agency projections are treated as forecasts. The analytical examples in later sections are deliberately separated from all three.

No average across this table is meaningful: a session, a user-turn, a benchmark task, and an accepted business outcome are different denominators. The cited sources also do not jointly establish a representative fleet distribution of context length, task success, reuse intervals, or scheduled activation volume. The demand model therefore uses explicit scenarios rather than a point estimate of a global agent market.

### 3. Why tasks require more inference

*Extra work comes from repeated decisions, accumulated evidence, deliberate reasoning, and attempts to obtain a reliable result—not just from a longer final answer.*

#### Repeated decisions and growing context

An agent often alternates between model calls and tools. Each return from a tool introduces new evidence: files, search results, test failures, records, or observations. Without reduction, subsequent calls carry a larger history. The vLLM / Mooncake trace analysis reports average context growth of 2,242 tokens per turn in its examined coding traces. That pattern creates two burdens: new information must be processed, and earlier information may be processed again when reusable state is unavailable. [3]

A useful distinction is *prefill* versus *decode*. Prefill processes the input context to prepare a response. Decode generates new tokens step by step. Reusing a prefix can avoid rebuilding much of its stored computation, but new tokens still have to interact with relevant prior context. A cache hit is therefore not equivalent to free inference. PagedAttention and disaggregated serving research address different parts of this resource problem. [12][17]

#### Reasoning can be larger than the visible response

A model may spend computation before emitting its final answer. OpenAI's reasoning documentation explains that reasoning tokens are not shown as ordinary response text but still consume output-token budget; its earlier o1 results demonstrate performance improvements with additional test-time computation in the reported evaluations. Neither fact implies unlimited returns from longer reasoning. [10][11]

For infrastructure measurement, the implication is simple: counting only the words delivered to a user understates work when internal reasoning or discarded candidates are present. The relevant record includes all model-generated tokens exposed by usage telemetry, and should identify any unobserved component rather than estimate it from answer length.

#### Parallel search, verification, and retries

Parallel branches can explore alternatives, divide research, or test candidate solutions. They may reduce time to a result while increasing total work, since several branches can duplicate setup and context. Conversely, parallel exploration can save total work when it avoids a long unsuccessful serial path. The graph and the stopping rule determine which effect wins.

Verification creates a similar trade-off. More checking costs resources, but can prevent defective outcomes and expensive retries. A model that appears cheaper per attempt may be worse per accepted outcome. In SWE-Serve, four configurations with the same reported 64% pass@1 range from \$0.95 to \$7.24 per task. The comparison shows why price, duration, and success must be observed together; API cost is still not a direct measurement of physical GPU work. [4]

---

The infrastructure unit is the whole trajectory: every branch that ran, every failed attempt, every tool action, and the verification needed to decide that the work is finished.

---

## 4. Context replay: a multiplier, not destiny

*A growing history can make submitted input rise roughly quadratically with step count. Reuse can remove much of that replay, but it does not remove the task.*

Consider an illustrative 30-call agent. It begins with 12,000 input tokens. Every call generates 500 tokens; after each call except the last, a tool adds 2,000 new input tokens. All prior messages remain in context. There is no compaction, branching, or model change. These are assumptions, not measured workload averages.

At call  $s$ , the input length is  $P_s = 12,000 + (s - 1) \times 2,500$ . Summing the 30 calls gives:

$$\underbrace{30 \times 12,000}_{\text{initial context replayed}} + \underbrace{2,500 \times \frac{30 \times 29}{2}}_{\text{accumulating history}} = 1,447,500 \text{ submitted input tokens.}$$

With perfect reuse of the key-value (KV) cache—the model’s stored attention state—only the initial input and newly arriving tool tokens need fresh input processing:  $12,000 + 29 \times 2,000 = 70,000$  tokens. This idealized accounting treats prior output state as already computed; message-formatting and response-boundary overhead are excluded.

| History-reuse assumption                | Fresh input positions | Generated tokens |
|-----------------------------------------|-----------------------|------------------|
| No reusable history                     | 1,447,500             | 15,000           |
| 90% of eligible repeated history reused | 207,750               | 15,000           |
| All eligible repeated history reused    | 70,000                | 15,000           |

*Exhibit 3.* Author calculation. “Fresh input positions” counts input tokens processed without reusable prior state, not floating-point operations (FLOPs). All three cases submit the same logical history; the final call sees 84,500 input tokens.

### What the arithmetic demonstrates

First, raw submitted tokens can dramatically exceed genuinely new input. A forecast that treats every submitted token as a full-price, uncached computation will overstate work when prefixes are reusable.

Second, even perfect history reuse leaves the entire 15,000-token generation process and all newly arriving evidence. It also leaves attention to stored context, cache access, orchestration, and tools. The final context is much larger than the initial one, so the per-token serving burden need not remain constant.

Third, the difference between the rows is a systems variable. Cache survival, model compatibility, routing, and compaction determine realized reuse. It is not appropriate to choose either zero reuse or perfect reuse as a universal agent assumption.

The durable compute-demand case should rest on additional useful work and additional activations. Recomputing the same history is a real burden, but it is also an optimization target.

**Calculation note.** Eligible repeated history is  $1,447,500 - 70,000 = 1,377,500$  token positions. At reuse fraction  $r$ , fresh input is  $70,000 + (1 - r) \times 1,377,500$ . This is token accounting, not an assumed linear mapping to dollars or accelerator time.

## 5. A physical model of agent demand

*A useful demand model keeps quantities with different units in separate accounts. Tokens, gigabytes, seconds, and dollars are not interchangeable.*

### Account 1: inference work over the entire execution graph

For an execution  $e$ , let  $G_e$  contain every model-call node that actually runs. Define:

$$W_e = \sum_{s \in G_e} [F_m(u_s, c_s) + D_m(o_s, \ell_s)].$$

Here  $u_s$  is uncached input,  $c_s$  is reused prefix length,  $o_s$  is generated output, and  $\ell_s$  is attended context length.  $F_m$  and  $D_m$  represent measured prefill and decode service demand for the routed model  $m$ . They should be calibrated on a declared hardware and serving configuration, rather than replaced with a single token price. Cached context still matters to the work performed.

An accelerator-second represents one second of attributed device execution; ten devices working for one second can consume ten accelerator-seconds. This measure is often more useful for capacity planning than a theoretical operation count alone. The measurement must state how batched device time is allocated; independently timed calls cannot simply be added as though each owned the whole device. CPU, tool, and storage work remain separate resource records.

### Account 2: physical state retained in each tier

$$M_k(t) = \sum_b \text{size}(b) n_{b,k}(t).$$

$M_k(t)$  is bytes physically stored in tier  $k$  at time  $t$ . Each  $b$  is a distinct cache block, and  $n_{b,k}$  is the number of physical copies. Shared blocks are counted once per stored copy; replicas are counted separately. This is a capacity measure, not bytes multiplied by a probability of reuse. Reuse probability determines whether storing a block is worthwhile, not how many bytes the block occupies.

### Account 3: traffic through a link or storage tier

$$\overline{B}_k = \frac{\text{bytes transported through resource } k \text{ in a window}}{\text{window duration}}.$$

Average bandwidth is traffic divided by time. Peak requirements must be measured over shorter windows that capture overlapping restores and writes. Adding the transfer rates of operations that occur at different times would overstate simultaneous demand.

### Join the accounts at the accepted outcome

For a defined cohort, let  $N$  be accepted outcomes and  $g$  the total accelerator service spent across *all* its attempts divided by  $N$ . Then total inference work is exactly  $W = Ng$ . This ratio-of-totals identity requires no assumption that difficulty, context, branch count, and failure are independent. Retained bytes, traffic, and non-model work should be reported alongside  $g$ , not folded into a unitless multiplier.

**Analytical scope.** These equations are an accounting framework. They become an empirical capacity model only after the service functions, workload mix, shared-resource attribution, and acceptance criteria are measured.

## 6. Persistent agents and the memory hierarchy

*An agent can remember an objective without keeping its entire computational history on an accelerator. The distinction is central to credible infrastructure forecasts.*

|                   | Durable semantic state                         | Computational KV state                                         |
|-------------------|------------------------------------------------|----------------------------------------------------------------|
| What it retains   | Facts, plans, results, provenance, checkpoints | Model-specific key and value tensors for processed context     |
| Value             | Continuity and recovery                        | Avoided recomputation of compatible context                    |
| Reuse condition   | Information remains relevant and trustworthy   | Model, prefix, positions, and implementation remain compatible |
| Typical placement | Database, object store, document or index      | Accelerator memory, host memory, SSD, shared cache             |

*Exhibit 4.* Analytical distinction informed by KV management and disaggregated-serving research. Durable meaning and cached computation are complementary, not equivalent. [12][16]

### Why long context can be a large physical object

For a conventional full-attention model with grouped-query attention, an uncompressed KV payload is approximately:

$$B_{KV} = 2 \times L \times H_{KV} \times d \times b \times T.$$

The factors are layers  $L$ , key/value heads  $H_{KV}$ , head dimension  $d$ , bytes per element  $b$ , and context tokens  $T$ . The factor two represents keys and values. Grouped-query attention uses fewer KV heads than query heads, so substituting the query-head count would misstate memory. [13]

For an *illustrative*, not model-specific, configuration with 80 layers, 8 KV heads, a head dimension of 128, and two-byte elements, each token requires 327,680 bytes of raw KV. A 100,000-token context therefore occupies about **32.8 GB** (30.5 GiB), before metadata or allocator overhead. Ten thousand distinct, unshared contexts at that size would occupy roughly 328 TB across whatever tiers retain them. This does not imply ten thousand dedicated GPUs.

The example is architecture-sensitive. One-byte elements halve the raw payload, while different attention designs can change the formula's application. DeepSeek-V3's multi-head latent attention and research such as KIVI demonstrate that KV representation is an active efficiency lever, not a fixed bytes-per-token constant. [14][15]

### The economic effect is an allocation problem

HBM—high-bandwidth memory attached to an accelerator—must accommodate weights, active context, and execution workspace. Retaining an idle context can crowd out useful active work. A cheaper tier may preserve reuse value without consuming scarce HBM, while a compact semantic checkpoint may be preferable when exact computational reuse is unlikely. More persistent agents can therefore expand host-memory or storage demand even when they do not expand resident accelerator state proportionally.

## 7. Efficiency changes where demand lands

*Better infrastructure does not merely shrink a workload. It can move its limiting resource from arithmetic to memory, bandwidth, or coordination.*

### Cache reuse trades recomputation for retention and transfer

Mooncake organizes serving around reusable KV state and uses pooled memory and storage beyond individual accelerators. PagedAttention attacks fragmentation and duplication in KV allocation. These approaches address different costs: finding useful state, holding it efficiently, and making it available where the next computation will run. [12][16]

That last step has a physical floor. A single 400 Gb/s link carries at most 50 GB/s before overhead, so transferring 100 GB takes at least two seconds of serialization. It does *not* take 0.25 seconds; that would confuse bits with bytes. Real restore time also includes storage reads, setup, contention, and placement. A cache is beneficial only when the whole restore path is preferable to recomputation under the required latency target.

### Specialize execution without assuming a universal gain

Prefill and decode have different operating characteristics. DistServe separates them to reduce interference and optimize the number of requests meeting latency objectives. This provides a reason to size serving pools by workload phase rather than treating all tokens identically; it does not establish that disaggregation wins under every context, network, or batch configuration. [17]

Program-aware scheduling goes further. ThunderAgent exposes the complete workflow, including KV state and tool assets, and reports  $1.5\text{--}3.6\times$  serving-throughput improvement in its evaluated workloads. This is evidence that orchestration can materially improve execution efficiency, not a factor to multiply by gains from unrelated studies. [18]

| Optimization             | Resource burden reduced             | Residual or shifted burden                   |
|--------------------------|-------------------------------------|----------------------------------------------|
| Prefix reuse             | Repeated context processing         | KV capacity, transfer, context attention     |
| Context reduction        | Input length and state size         | Selection work; possible loss or extra steps |
| Smaller-model routing    | Expensive model service             | Routing, escalation, quality verification    |
| Program-aware scheduling | Avoidable idle time and duplication | Shared-state management and queue control    |

*Exhibit 5.* Mechanism-level analysis. These benefits overlap and should not be multiplied as independent efficiency factors.

### Measure the complete system after optimization

If inference becomes faster but a sandbox remains on the critical path, end-to-end completion can improve much less than model throughput. AgentSysBench’s heterogeneous application results illustrate why tools, environments, and model service must be measured together. [6] The relevant outcome is not the fastest individual component, but more verified work completed within an acceptable time and resource budget.

## 8. Future demand: the break-even condition

*The demand thesis can be stated as a break-even condition without pretending that public data supports a precise market forecast.*

Start with accepted outcomes, not seats or nominal agent counts. Over comparable periods, write the ratio of required inference service as:

$$\frac{W_1}{W_0} = \frac{A \times I}{E}$$

$A$  is growth in accepted outcome volume.  $I$  is growth in gross work per outcome under a fixed reference execution method, reflecting broader tasks, more reasoning, or stricter verification.  $E$  is the combined execution-efficiency improvement applied to that new workload. These are scenario inputs, not independent random variables or separately estimated average multipliers.

The reference hardware and the definition of an accepted outcome must be held explicit. Here,  $E$  includes improvements in models, context policy, and serving on a common hardware basis. Hardware-generation changes can be modeled separately, once; these cases hold reference hardware constant. In a real study, estimate the workload-weighted efficiency jointly, rather than multiplying isolated benchmark speedups.

| Illustrative case    | Volume $A$ | Intensity $I$ | Efficiency $E$ | Work ratio   |
|----------------------|------------|---------------|----------------|--------------|
| Efficiency dominates | 4×         | 1.5×          | 8×             | <b>0.75×</b> |
| Broad delegation     | 12×        | 2×            | 8×             | <b>3.0×</b>  |
| High delegation      | 40×        | 3×            | 8×             | <b>15.0×</b> |

*Exhibit 6.* Sensitivity cases chosen to show the threshold, not to predict a year, assign probabilities, or estimate a market. All assume the same eightfold efficiency gain.

### The threshold is more useful than a heroic forecast

Demand rises when  $AI > E$ . With eightfold efficiency and a doubling of work intensity, outcome volume must grow more than fourfold for aggregate work to rise. Below that threshold, efficiency wins. Above it, cheaper execution and greater consumption coexist.

The high-delegation case is possible only if users actually fund and accept substantially more work. Scheduling features, benchmark progress, and long traces cannot establish that assumption by themselves. The framework makes the missing variable visible instead of disguising it as a token-growth forecast.

### Work growth is not automatically hardware growth

If required service triples while sustainable utilization improves from 65% to 80%, equivalent capacity rises by approximately  $3 \times 0.65/0.80 = 2.44$  times, before considering a change in hardware generation. Memory constraints, tail-latency targets, and deployment fragmentation can alter that result. The next planning step is to map service work to the achievable utilization of the actual workload, not simply purchase three times as many devices.

## 9. Falling unit costs, expanding demand

*Efficiency lowers the cost of a fixed unit of work. It does not determine how many units become worth buying or how ambitious those units become.*

### Separate prices, resource efficiency, and customer value

An API price is a commercial schedule, not a physical measurement. It can change with model mix, discounts, competition, or provider economics. Epoch AI's historical inference-price study finds substantial declines at fixed benchmark-performance thresholds, but explicitly excludes reasoning models because additional inference tokens complicate comparisons. The study does not isolate the causes of price declines. Its result supports falling prices, not a one-for-one inference about falling GPU work per real-world outcome. [21]

A customer makes a different comparison: expected benefit versus total cost of delegation. That cost includes model usage, tools, human review, errors, and delay. A technically cheaper agent that creates more rework can have worse economics. A more expensive trajectory can be attractive when it replaces a larger amount of valuable manual effort. This paper's demand thesis depends on the latter margin expanding, not on maximizing token consumption.

### Two forms of demand expansion

**The extensive margin** is additional work: more records checked, more changes investigated, more tests generated, or more recurring processes monitored. Lower costs can make formerly ignored or sparsely sampled work worth performing.

**The intensive margin** is more effort on existing work: broader search, more candidate solutions, additional verification, or tighter service requirements. The budget for a useful outcome may increase even as cost per unit of inference falls. Both margins are analytical possibilities, not assumed behavioral laws.

An illustrative business that once checked one exception manually might delegate checks across all exceptions after automation becomes inexpensive. Conversely, a business with a fixed workload and no benefit from broader coverage may simply keep the savings. Rebound is therefore conditional on the supply of valuable unfinished work, willingness to pay, and the ability to trust the result. It is not an automatic consequence of cheaper compute.

### The counterforce is already technically credible

AgentDiet reports 39.9–59.7% fewer input tokens in its tested settings, with reported cost reductions and maintained performance. Observation-masking research finds that a comparatively simple context-management policy can compete with summarization while reducing cost. These results show that long trajectories need not retain their initial inefficiency. They do not establish how much new demand the savings subsequently induce. [19][20]

---

The bullish case is not that inefficiency persists. It is that economically useful delegation expands faster than the resources required to execute any fixed task decline.

---

**Measurement implication.** Publish cost per accepted outcome and accelerator service per accepted outcome separately. A price reduction and a physical efficiency gain are not the same event.

## 10. Concurrency and the capacity problem

*Elapsed lifetime determines how many tasks are in the system. Active service demand and workload shape determine how much accelerator capacity they need.*

### A long-running task is not a continuously busy GPU

Suppose tasks arrive at an average rate of 10 per second and remain in the system for five minutes. In a stable system, arrival rate multiplied by average lifetime implies about **3,000 live tasks**. This standard flow relationship counts tasks waiting on tools or external events as well as tasks using a model.

Now assume each arriving task requires 12 reference accelerator-seconds across its entire execution graph. The steady-state workload is 120 accelerator-seconds per second. At a planning utilization of 60%, that corresponds to about **200 equivalent accelerator units**, not 3,000. All numbers here are illustrative; they show why task count, elapsed duration, and occupied compute must remain separate.

### Parallelism can increase peaks without increasing useful output

If several branches launch together, their model and state requirements overlap. Parallel execution can shorten completion time, but a scheduler must decide whether the faster outcome justifies higher simultaneous demand. Branch budgets, cancellation of losing candidates, and limits on speculative work affect both total service and peak capacity.

The tail matters as much as the mean. A few unusually deep trajectories can consume a large share of work or hold state far longer than the typical task. Bai et al.'s run-to-run token variation motivates measuring these tails, but it does not by itself identify the peak load of a mixed production fleet. [5]

### Persistence can smooth demand as well as amplify it

Not every background task needs immediate service. Jobs with flexible deadlines can be batched or shifted into underused periods. That improves utilization and can lower installed capacity per unit of completed work. In contrast, many scheduled tasks released at the same time, or many agents reacting to one event, can create a synchronized burst.

Observed usage should not be described as continuously flat merely because agents support background execution. The Copilot study still reports pronounced daytime peaks and overnight declines in its sampled US traffic. [1] Whether future deployment smooths or intensifies peaks is a scheduling and adoption question, not a property implied by the word “agent.”

### Provision by deadline and resource class

Separate interactive work, deadline-bound background work, and deferrable batch work. For each class, measure arrival bursts, concurrent branches, service demand, state residency, and the time allowed to finish. The same average work can require very different capacity when one class tolerates a queue and another does not.

---

The right capacity question is: how much work must finish within which deadlines, with what reusable state? Counting agents alone cannot answer it.

---

# 11. From compute to physical infrastructure

*Accelerator work, installed capacity, electricity, and capital spending are connected—but each link needs its own assumptions.*

An expanding agent workload can pressure several systems at once. More fresh context and generation increase inference service. Longer contexts compete for fast memory. More resumption increases cache traffic. More tool use requires CPU execution, databases, network access, and sandboxes. The balance depends on the task and the efficiency measures chosen; there is no fixed hardware bill of materials per agent.

| Infrastructure layer | Demand transmission                          | Planning variable                                   |
|----------------------|----------------------------------------------|-----------------------------------------------------|
| Accelerator capacity | More model service; stricter deadlines       | Accepted outcomes per device-hour at target latency |
| HBM and host memory  | More active or retained context              | Concurrent bytes, sharing, retention policy         |
| Storage and network  | Checkpoints, tool data, KV movement          | Traffic volume, peak restore rate, locality         |
| CPU and environments | Retrieval, execution, testing, orchestration | Working sets, sandbox lifetime, tool throughput     |
| Power and cooling    | Installed systems and their load profile     | Measured IT power, utilization, facility overhead   |

*Exhibit 7.* Analytical transmission map. A bottleneck in one layer can prevent spending in another from producing more completed work.

## Energy is an outcome of the deployment, not a token constant

Facility energy depends on the power drawn by the actual mix of accelerators, memory, storage, networking, and cooling over time. Even an accurate token forecast cannot supply that mix on its own. A higher cache-hit rate can reduce computation while increasing state traffic; a newer device can alter both throughput and power; better batching can change energy per accepted outcome.

The IEA’s April 2026 update projects global data-center electricity consumption rising from 485 TWh in 2025 to about 950 TWh in 2030. It also identifies constraints in electricity connections, equipment, and chip supply chains. This is an agency projection for *all* data centers, not an estimate of energy attributable to persistent agents. It provides physical context for the demand mechanism, not a calibration of this paper’s scenarios. [22]

## Installed capacity can lag economically desired work

If power, memory supply, or deployment timelines constrain expansion, demand can appear as queues, tighter usage limits, higher prices, or deferred projects rather than immediately higher delivered compute. Conversely, installed capacity can rise ahead of utilization when operators build for expected future load. Capital spending therefore cannot be read as a direct contemporaneous measure of agent usage.

The practical implication is to track the complete conversion chain: desired outcomes, affordable activations, delivered model service, occupied infrastructure, and facility energy. A claim about one stage should not silently become a claim about all the others.

## 12. The strongest countercase

*A credible expansion thesis must allow efficiency, substitution, and limited adoption to win. The central issue is whether demand intensity survives quality adjustment.*

### The efficiency-dominant path

Routine decisions may migrate to smaller models or deterministic code. Better tools can replace many conversational steps with one structured operation. Context reduction can flatten history growth. Improved stopping rules can eliminate unproductive exploration, and good memory policy can replace large dormant caches with compact checkpoints. The first scenario in Exhibit 6 shows that even substantial volume growth can coexist with lower total inference work.

These changes would not be surprising failures of the framework; they are its explicit counterforce. Studies of context reduction, KV management, and program-aware scheduling already show mechanisms that can lower the burden of a fixed workload. Their fleet-level combination remains an empirical question. [12][18][19][20]

### The adoption-limited path

Organizations may discover that oversight, permissions, integration, and error costs dominate the value of autonomous execution. Persistent systems can be technically capable of more work than customers are willing to authorize or fund. A scheduled check that rarely produces useful information can be throttled, replaced with a simple rule, or removed entirely.

This weakens the extensive-margin assumption. Agent counts, feature availability, and benchmark capability would then overstate economically productive deployment. The relevant evidence would be flat accepted-outcome volume or low renewal of autonomous workflows despite improving technical performance.

### The reliability path can point in either direction

Better agents may solve a fixed task with fewer calls and retries, reducing work per success. They may also unlock a more difficult class of tasks that requires more resources. An aggregate average can hide those opposing changes. A longitudinal study should compare fixed-task cohorts *and* the changing mix of tasks delegated, rather than interpret a rise in average tokens as declining efficiency.

| Evidence that strengthens expansion                                          | Evidence that weakens expansion                                            |
|------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| Accepted outcome volume grows faster than workload-matched efficiency.       | Outcome volume saturates while fixed-task resource use falls.              |
| More difficult tasks become worth delegating at stable acceptance standards. | Added autonomy mainly produces rejected or low-value work.                 |
| State and traffic rise after sharing, compaction, and tiering are applied.   | Tiny checkpoints and high locality keep memory and transfer growth modest. |

*Exhibit 8.* Directional tests, not predetermined numerical thresholds.

The thesis is strongest when better efficiency broadens economically useful delegation. It is weakest when it merely serves a fixed, limited workload more cheaply.

### 13. Measure outcomes, then forecast resources

*The next useful disclosure is not another isolated token total. It is a joint record of what agents attempted, what they completed, and what completion required.*

A production study should define its cohort, acceptance criteria, and resource boundary. A monitoring run that correctly determines “no action needed” may be a valid outcome; an artifact that fails verification may not. Count completed objectives, not visible output.

| Measurement                        | Why it matters for demand                                                                                |
|------------------------------------|----------------------------------------------------------------------------------------------------------|
| Activations and accepted outcomes  | Separates genuine expansion from retries, abandoned jobs, and nominal agent installations.               |
| Work per accepted outcome          | Includes failed attempts, discarded branches, and disclosed reasoning work; anchors the intensity trend. |
| Depth, context, and model route    | Reveals whether harder tasks, context replay, or model selection explains resource growth.               |
| Elapsed and active service time    | Distinguishes task persistence from occupied accelerator capacity.                                       |
| Reuse intervals and physical bytes | Connects resumed work to a defensible retention policy in each tier.                                     |
| Traffic and restore latency        | Tests whether cache movement beats recomputation in the deployed system.                                 |
| Concurrency and deadlines          | Converts average service demand into peak and reserved capacity.                                         |
| Cost, energy, and acceptance       | Separates commercial price, physical efficiency, and delivered value.                                    |

*Exhibit 9.* Proposed measurement protocol. Report distributions and joint relationships, not just independent averages.

#### Use two complementary comparisons

A **fixed-task panel** tests whether the same work becomes cheaper and faster at stable quality. A **deployment panel** tests whether users delegate more work, harder work, or work with tighter deadlines. Together, they distinguish execution efficiency from the demand mix. Report median and tail results, preserving model and harness versions.

#### Conclusion

Persistent agents create a credible path to higher compute demand by expanding the scope and frequency of delegated work. Inference becomes a means to investigate, act, verify, and resume—not just to produce an immediate response.

Caching, better models, and scheduling can make fixed tasks substantially cheaper. Aggregate demand grows when additional accepted work and broader task scope exceed those savings. Its physical expression is a workload-dependent mix of accelerator service, memory capacity, and transfer traffic, shaped by deadlines and utilization.

The enduring infrastructure thesis is an expansion from human-requested answers to economically valuable delegated work. Forecast the work, measure the efficiency, and derive the hardware requirement—not the other way around.

# Technical appendix and source notes

*The main text uses simple examples to expose mechanisms. These notes define their boundaries and the conditions required to apply them.*

## A. A compact test for whether retained state is worthwhile

Let  $p$  be the probability of another use inside a retention window, and  $q$  the probability that the state remains usable conditional on that use. Let  $C_{\text{recompute}}$  and  $C_{\text{restore}}$  be the costs of rebuilding or restoring it. A simplified single-reuse policy retains state only when:

$$pq (C_{\text{recompute}} - C_{\text{restore}}) > C_{\text{write}} + C_{\text{hold}}.$$

All costs must use the same unit, such as dollars or a declared resource opportunity cost. Holding HBM includes the value of displaced active work. The policy must also satisfy a separate latency constraint. Multiple reuses require summing expected benefits over time; eviction, invalidation, and correlated arrivals can change the result. This is an analytical rule, not a measured universal cache lifetime.

## B. Numerical conventions

The replay example counts all previous generated tokens in later prompts, but does not treat them as fresh prefill under perfect KV reuse. Its 90% case applies to *eligible repeated history*, not to all submitted tokens. It assumes complete compatible prefix reuse, no eviction beyond the stated fraction, and no compression.

Memory calculations use decimal GB ( $10^9$  bytes) and binary GiB ( $2^{30}$  bytes). The 32.768 GB example is raw payload for an illustrative grouped-query, full-attention configuration. It excludes model weights, execution workspace, metadata, allocator overhead, and replicas. Bandwidth calculations are ideal serialization floors, not measured end-to-end restore times.

The scenario ratios are dimensionless sensitivity inputs. They have no assigned forecast dates or likelihoods. The utilization example holds the reference hardware constant; it must not also receive a hardware-efficiency adjustment already included elsewhere. The concurrency example assumes a stable workload with finite mean duration and service demand.

## C. Evidence and denominator discipline

The Copilot abstract’s aggregate token count differs from the prompt/completion totals in Table 3; this paper uses neither aggregate. Its telemetry also excludes reasoning tokens. Table-specific session and user-turn definitions are preserved rather than equated with successful tasks. [1]

A benchmark pass rate is evidence about its tested acceptance conditions, not an audited business success rate. Provider token ratios, system throughput improvements, and agency energy projections retain their own baselines and scopes. None is used as a stand-alone global demand multiplier.

**Terms.** *Prefill*: processing input context. *Decode*: generating new tokens. *KV cache*: stored attention keys and values. *HBM*: high-bandwidth accelerator memory. *Goodput*: completed service meeting its quality or latency target. *Accepted outcome*: a result satisfying the declared objective and verification standard.

# References

*Primary research and first-party documentation. Titles are clickable. Versions and evidence classes are retained where material to the analysis.*

- [1] Liu, B., Qiu, H., Goiri, Í., Fonseca, R., Bianchini, R., and Choukse, E. (2026). *Agentic Coding in the Wild: Characterizing GitHub Copilot at Production Scale*.

arXiv:2608.00101v1. Microsoft / University of Illinois production telemetry. See Tables 3–4 and the session hierarchy.

- [2] Anthropic Engineering (2025). *How we built our multi-agent research system*.

June 13. First-party engineering observations; token comparisons are internal and workload-specific.

- [3] vLLM and Mooncake engineering teams (2026). *Serving Agentic Workloads at Scale with vLLM x Mooncake*.

May 6. First-party analysis of Codex / SWE-bench Pro traces and serving experiments.

- [4] Williams, J., Farris, D., Farris, J., and Jiao, J. (2026). *SWE-Serve: Benchmarking Agentic Engineering for Production Inference Serving*.

arXiv:2609.26777v1. Controlled engineering benchmark; 53 tasks and 11 models.

- [5] Bai, L., et al. (2026). *How Do AI Agents Spend Your Money? Analyzing and Predicting Token Consumption in Agentic Coding Tasks*.

arXiv:2604.22750v2. Measurement study of agentic coding trajectories and token variability.

- [6] Chang, C., et al. (2026). *From LLM Inference to Agentic Workloads: Characterization and Implications for Serving Systems*.

arXiv:2608.15127v1. AgentSysBench; controlled deployments and production traces.

- [7] OpenAI (accessed September 30, 2026). *Background mode*.

Developer documentation. Capability evidence for asynchronous model execution.

- [8] OpenAI (accessed September 30, 2026). *Scheduled tasks in ChatGPT*.

Product documentation. Capability evidence for scheduled and supported event-triggered work.

- [9] Kwa, T., et al. (2025; revised 2026). *Measuring AI Ability to Complete Long Software Tasks*.

arXiv:2503.14499v4; NeurIPS 2025. Human-time-based capability horizon, not agent runtime.

- [10] OpenAI (2024). *Learning to reason with LLMs*.

September 12. First-party o1 evaluation of training-time and test-time computation.

- [11] OpenAI (accessed September 30, 2026). *Reasoning models*.

Developer documentation. Reasoning-token accounting and usage behavior.

## References, continued

- [12] Kwon, W., et al. (2023). *Efficient Memory Management for Large Language Model Serving with PagedAttention*.  
SOSP 2023; arXiv:2309.06180. KV allocation, sharing, and fragmentation.
- [13] Ainslie, J., et al. (2023). *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*.  
EMNLP 2023; arXiv:2305.13245. Grouped-query attention and key/value head structure.
- [14] DeepSeek-AI (2024; revised 2025). *DeepSeek-V3 Technical Report*.  
arXiv:2412.19437. Architecture report, including multi-head latent attention.
- [15] Liu, Z., et al. (2024). *KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache*.  
ICML 2024; arXiv:2402.02750. Controlled evaluation of KV-cache quantization.
- [16] Qin, R., et al. (2024; revised 2025). *Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving*.  
FAST 2025; arXiv:2407.00079. KV-centric serving with pooled memory and storage.
- [17] Zhong, Y., et al. (2024). *DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving*.  
OSDI 2024; arXiv:2401.09670. Workload- and latency-aware serving-system evaluation.
- [18] Kang, H., et al. (2026). *ThunderAgent: A Simple, Fast and Program-Aware Agentic Inference System*.  
arXiv:2602.13692v3. Controlled evaluation of program-aware model and tool scheduling.
- [19] Xiao, Y.-A., Gao, P., Peng, C., and Xiong, Y. (2025; revised 2026). *Reducing Cost of LLM Agents with Trajectory Reduction*.  
FSE 2026; arXiv:2509.23586v2. AgentDiet; token and cost reduction experiments.
- [20] Lindenbauer, T., Slinko, I., Felder, L., Bogomolov, E., and Zharov, Y. (2025). *The Complexity Trap: Simple Observation Masking Is as Efficient as LLM Summarization for Agent Context Management*.  
NeurIPS 2025 Deep Learning for Code Workshop; arXiv:2508.21433v3. Controlled context-management study.
- [21] Cottier, B., Snodin, B., Owen, D., and Adamczewski, T. (2025). *LLM inference prices have fallen rapidly but unequally across tasks*.  
Epoch AI, March 12. Original price analysis at benchmark-performance thresholds; excludes reasoning models.
- [22] International Energy Agency (2026). *Key Questions on Energy and AI*.  
April 16. Agency analysis and projections; executive summary used for the 2025–2030 electricity outlook.