Multi-turn Agents require Max Sustained Thread Performance at Rack Scale

Why core count and single-threaded performance don't answer the question, and what Intel Xeon does about it

Mrittika Ganguli, Arjun Kripanidhi, Rajat Agarwal
Intel Data Center Group · August 2026

1. Overview

NVIDIA and AMD have each published a CPU thesis for the agentic era, but neither tells the complete story.

NVIDIA argues [1][4] that the agent-loop is sequential, so per-core speed sets how fast it advances; Vera's Olympus cores are built for maximum single-threaded performance. **AMD** argues nearly the opposite [2]: under a fixed 100 kW rack budget what matters is rack throughput, node count multiplied by node performance. It projects geometric-mean rack performance of 1.0 for Vera, 1.46 for Xeon 6980P, 2.37 for EPYC Turin and 3.30 for its 256-core Venice part.

Read closely, each concedes the other's premise. NVIDIA acknowledges that data centers need high core counts before arguing core count alone is insufficient, so even the strongest advocate of single-thread speed does not claim it is the whole answer. AMD does the mirror image, closing a density paper with a section claiming per-core leadership and softening its own headline to say higher core density *could suggest* support for more concurrent agent workflows. Both hedge because neither axis alone determines the outcome.

Our position is that both are optimizing the input, while customers pay for the output. Cores, clocks and nodes per kilowatt describe **theoretical capacity;** the work a rack could do in principle. What a customer buys is **delivered capacity**: the number of concurrent agent workflows a rack sustains while still meeting the latency target it promises users, known in industry terms as the service-level objective, or SLO. Between the two sits a conversion step neither paper models, covering placement under concurrency, resource balance, offload, and the movement and preservation of agent state.

NVIDIA's own Vera disclosure makes the point for us, because it reports delivered outcomes rather than benchmark scores: When a platform sustains concurrent agents, the actionable question is which term produced it — faster serialized execution, better resource balance under load, fewer stalls in the orchestration path, or state that stayed resident instead of being recomputed. Each answer implies a different purchase, a different fleet design, and a different software investment. Those are exactly the metrics the industry should report, and they are outcomes of the conversion step rather than of any single microarchitectural feature. No one publishes how the result breaks down. This paper supplies that breakdown, and Section 6 states what Intel will publish in the same terms.

2. What each argument gets right, and where it runs out

2.1 NVIDIA: a faster core cannot remove the waiting it does not control

The agent-loop is real, many of its steps do run in sequence, and a faster core does finish a tool call sooner. NVIDIA's sharpest point is economic: in an AI factory the GPU is the revenue asset, so CPU time that leaves a GPU waiting is revenue lost. We agree. Our disagreement is specific: the measurements show that those GPU seconds are lost somewhere other than where a faster core would help.

A production node does not run one agent loop. It runs dozens to hundreds at once, and the published measurements show what happens as that count rises. An instrumented study of agentic inference [3] measured how much the total time for each request was spent on the **host CPU** rather than the GPU. For a single request, the answer was only **0.4 to 0.6% of end-to-end latency**, at every context length from 1K to 100K tokens. There

is almost no host CPU time in a single loop for a faster core to remove. At 32 concurrent sessions the host CPU share rose to **11 to 15%**, and roughly **94% of it was scheduling and queue wait** rather than computation: deciding how to batch requests, looking up cached prompt prefixes, allocating memory blocks, and holding requests in queues until resources free up. Beyond 32 sessions, the host CPU share may continue to rise disproportionately as scheduling and queue-wait contention grows.

The significance is in what did not happen. If this work scaled linearly with load, each request would keep paying the same 0.5% and the share would stay flat. Instead, each request consumed over 20 times more host CPU time at 32 sessions than it did running alone, because those scheduling steps serialize against one another and requests wait behind each other. **The study concluded that CPU and GPU co-design is a scheduling problem rather than a compute problem.**

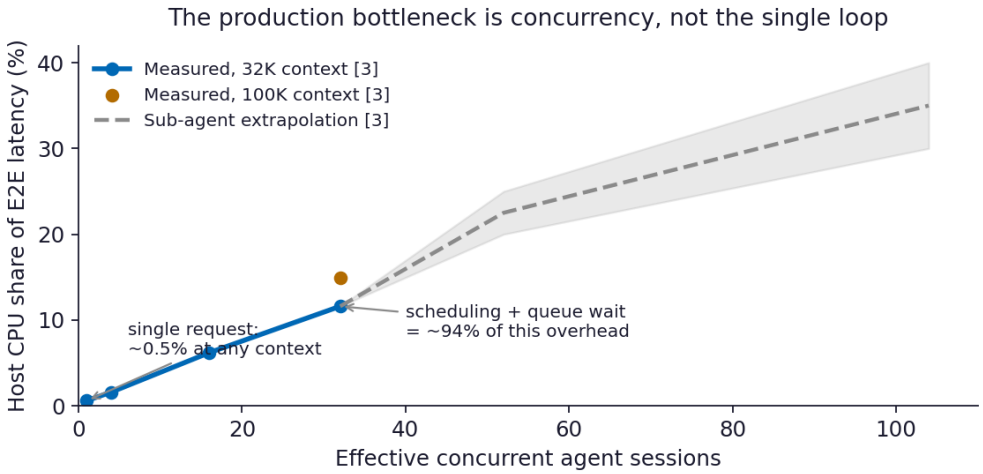


Figure 1. *Host CPU share of end-to-end latency vs. concurrency. Measured points and sub-agent extrapolation from the external study [3] (EXTERNAL). Single-loop CPU time is negligible; contention is not.*

Faster instructions shrink the sequential portion of that work and are welcome there, but they cannot remove queueing, and queueing is the part that grows. Nor does per-core speed address the largest losses. While a tool runs, whether a SQL query, a sandboxed test or a retrieval call, the GPU assigned to that session sits idle no matter how fast the host core is, and if the session's KV cache is evicted while it waits, the GPU pays a multi-second re-prefill on resume. For code-execution agents, the same study estimates the GPU does useful work only **50 to 60% of wall-clock time**. **Keeping agent state resident and placing it well returns more GPU seconds than faster instructions can, because the seconds being lost are spent waiting, not computing.**

2.2 Bandwidth keeps a thread busy; capacity keeps a session alive

Vera leans on very high memory bandwidth per core, a legitimate choice, because a fast core starved of data is not fast in practice. Every CPU shares this constraint: bandwidth is a fixed pool divided across active cores, so it caps how many agents a node can serve at once. Xeon included.

But bandwidth and capacity solve two different problems, and agentic serving needs both:

- **Bandwidth** determines how fast a thread can be fed **while it is running**. It is a throughput question, measured in GB/s.
- **Memory capacity** determines how many agent sessions can stay resident **between their turns**, so their KV cache does not have to be recomputed. It is a residency question, measured in gigabytes of DDR5 and CXL, in last-level cache that holds state, and in a separate I/O cache that prevents DMA traffic from evicting it.

Trading capacity away for bandwidth per core optimizes the first and penalizes the second. It lets an active thread run faster while shrinking the pool that keeps idle sessions warm, so more sessions must be rebuilt on resume. A rebuild costs GPU seconds and a resident session costs none, so the capacity half is where the larger savings sit.

2.3 AMD: the right arithmetic, applied to the wrong workloads

AMD frames the problem correctly. Racks are bought under power constraints, and single-node benchmarks mislead unless multiplied out to a full rack. Node performance times nodes per 100 kW is the right arithmetic for the **first** term of a capacity model, and the paper is candid that it offers directional comparison rather than measured rack benchmarks.

That first term is where delivered capacity begins, not where it ends. The six modeled workloads, SPECrate integer, server-side Java, NGINX, Redis, Memcached and TPROC-C, are throughput proxie rather than agent workloads. None of them carries a latency target. None reproduces the bursty, stateful, tool-interleaved traffic pattern of agentic serving. And the model itself contains no term for scheduling behavior, memory-bandwidth saturation, accelerator offload or state movement.

Cores turn into agents only if each core can be fed: with bandwidth for the agent's state, with scheduling headroom so requests do not pile up, and with offload so infrastructure work does not eat the core budget. **A core that cannot be fed is theoretical capacity, not delivered capacity.** That applies to AMD's headline number too. The 3.30 figure is its projected rack throughput for the 256-core Venice part relative to a Vera rack scored at 1.0, derived by multiplying benchmark scores by node count under a power cap. It describes what the hardware could do before any conversion loss is applied, which makes it the start of a capacity claim rather than the end of one.

> **The question a buyer should put to every projection, including NVIDIA's, AMD's and ours**
> **At what concurrency? At what latency target? What conversion losses?** A rack number without those three qualifiers describes what the hardware could do if nothing contended, nothing queued, and no session state ever had to move. That is not the workload anyone is buying for.

3. The missing term: conversion

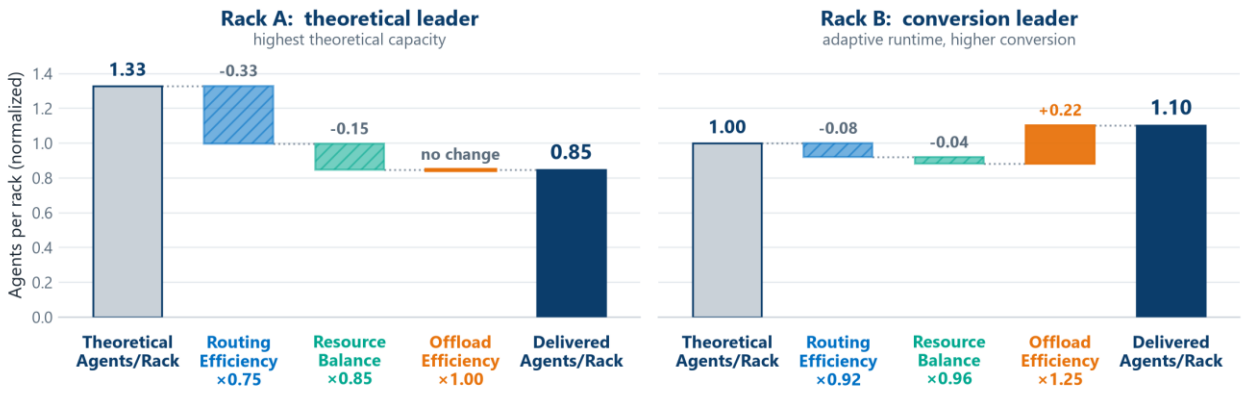
Our capacity model closes the gap between the two narratives with one equation, the single calculation this paper asks the reader to keep:

Delivered Agents/Rack = Theoretical Agents/Rack × Routing Efficiency × Resource Balance × Offload Efficiency

Theoretical Agents/Rack is what both narratives estimate from cores, bandwidth and nodes under power. **Routing Efficiency** measures how well the scheduler turns that capability into placed work when many agents run at once. **Resource Balance** captures what is lost to hot spots such as bandwidth saturation, NUMA imbalance and queue buildup. **Offload Efficiency** credits infrastructure work moved off the cores by accelerators and is the one term that can exceed 1.0. Every platform has these four terms. Almost no one publishes them.

The three conversion terms multiply, and they compound hard enough to reverse a ranking.

A 33% theoretical advantage can still deliver 23% fewer agents



Illustrative values showing the shape of the arithmetic, not a product claim.

Figure 2. *Theoretical to delivered, factor by factor. A 33% theoretical advantage finishes 23% behind once the conversion terms are applied (illustrative).*

The technical reason these factors exist is the **latency-knee**. As active agents increase, a node moves from compute-bound to bandwidth-bound to queueing-dominated, and p99 latency stays flat right up until it doesn't. **Delivered capacity is the point where the latency curve crosses the target, not the top of a throughput bar.** Headroom-aware placement, offload and state tiering all work by pushing that knee further right: same latency target, more agents.

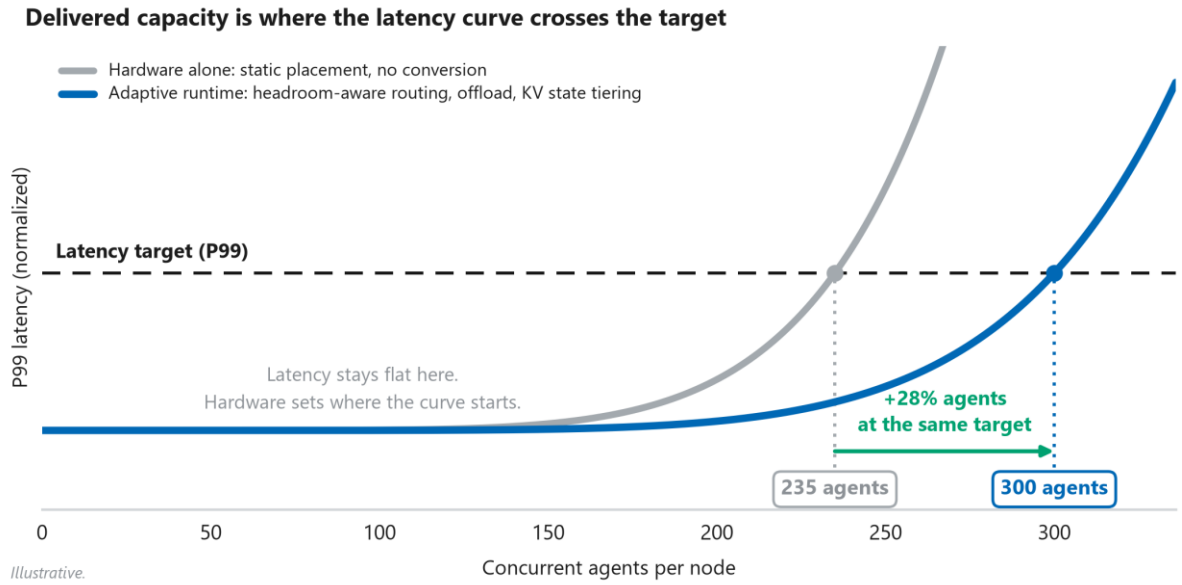


Figure 3. *Capacity is the point where the latency curve crosses the target. Headroom-aware routing, offload and state tiering moves the knee right (illustrative).*

4. The Intel position: adaptive infrastructure

Max Sustained Thread Performance at Scale describes a platform that converts the highest fraction of its theoretical capacity into delivered agents. Six commitments carry it.

- **Bandwidth per active agent, not per spec sheet.** Agentic AI at rack scale is a state-movement problem covering prompt tokens, KV cache, retrieval results and tool outputs. The useful question is not who has more GB/s, but how many concurrent agent states can be moved and updated before the latency-knee.
- **Memory expansion with Intel Flat Memory Mode.** Intel Xeon 6 presents CXL-attached memory and native DDR5 as a single flat address space managed by a hardware memory-side cache. It is transparent to the operating system, needs no tiering software, and moves data at cache-line rather than page granularity, which matters because a single page routinely holds both hot and cold agent state. Pairing CXL cards carrying lower-cost DDR4 with native DDR5 delivered 96% of all-DDR5 performance on a large in-memory OLAP database at roughly 25% lower memory TCO [6]. That is more KV parking capacity per dollar, the resource that decides how many sessions stay warm instead of being recomputed.
- **Session- and state-aware routing.** The router samples live telemetry on every node: CPU, memory, I/O and power headroom, accelerator occupancy and cache state, and where each session's context and KV state reside. Placement follows state, so an agent lands where it can be fed and where its session is already warm. Headroom, not peak speed, is what turns a microarchitectural advantage into delivered capacity.
- **Accelerators as capacity multipliers.** QAT, DSA and IAA hand infrastructure cycles back to the scheduler, including TLS, compression, data movement and analytics primitives. That raises agent density at the same latency target and makes KV compression in host DRAM feasible at line rate.
- **Heterogeneity is an asset.** Granite Rapids, Diamond Rapids and Clearwater Forest are distinct resource profiles covering per-core performance, memory capability and core density, unified under one routing layer that matches agent classes to the right platform.

- **Host of the GPU rack.** "GPU utilization is revenue" is our argument too, and the measurements show where that revenue leaks: 11 to 15% of request time on host-side scheduling, and GPUs doing useful work only 50 to 60% of wall-clock for code-execution agents [3]. Xeon recovers those losses through host software and state management, including sub-agent-aware routing and KV parking in host DRAM.

5. The hardware behind the conversion

Routing, scheduling, and resource balancing are vendor-agnostic ideas. The silicon they run on is not. Intel ships two things competing server CPUs do not: on-die fixed-function engines (QAT, DSA, IAA and AMX), and a portfolio of distinct execution profiles under one instruction set, spanning a single-threaded P-core in Diamond Rapids, SMT2 P-cores in Granite Rapids and dense E-cores in Clearwater Forest. That portfolio matters because an agent turn is not a single kind of work.

A single agent turn breaks into five segments by share of wall-clock. The largest, the model-call itself at roughly 40%, runs on the GPU. The remaining four are the host-side phases:

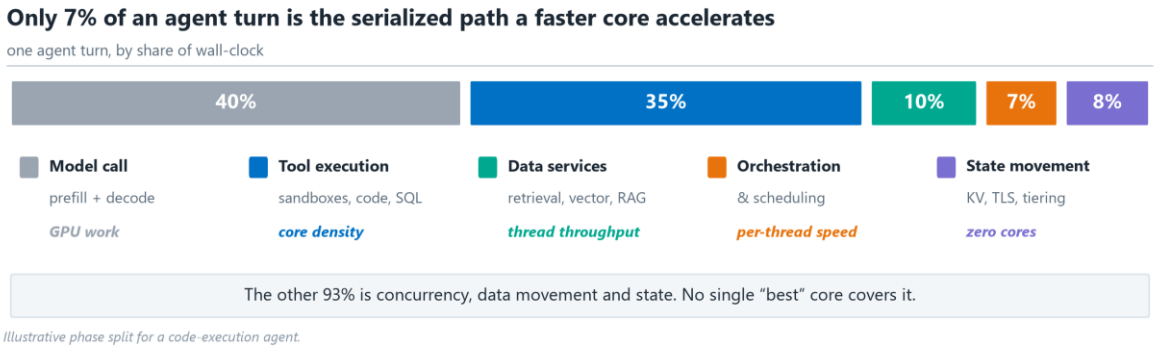


Figure 4. *Anatomy of an agent turn, showing where the wall-clock time actually goes (illustrative).*

Those four host-side phases have four different execution signatures, and the one a faster core improves, the sequential orchestration path, is the smallest of them at roughly 7% of the turn. This is where the two published arguments become easier to place. **NVIDIA's single-thread thesis optimizes the orchestration phase. AMD's core-density thesis optimizes the tool-execution phase.** Each is right about the phase it targets and silent about the other three, which is why the two arguments never meet: they are not describing the same portion of the workload. A CPU built around one phase cannot serve the other three well, so the useful question is not which single core design wins, but whether a platform can cover all four.

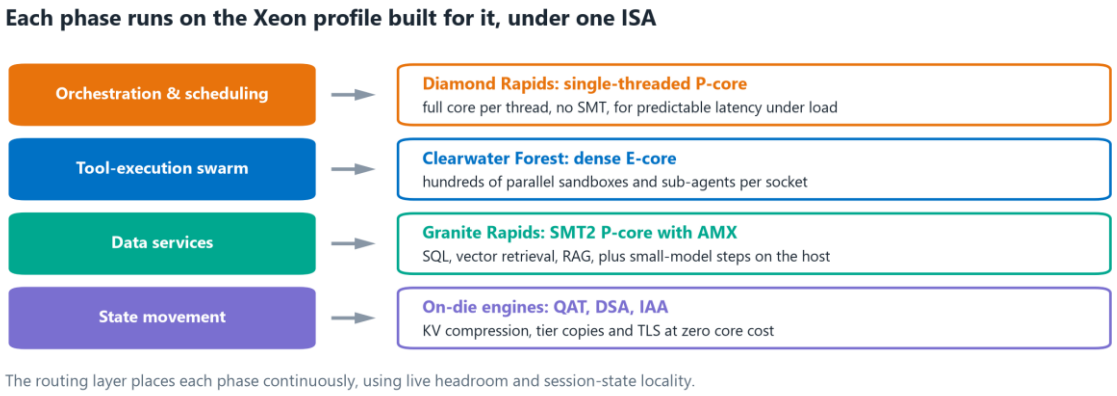


Figure 5. *Phase to profile. Each phase of the agent turn runs on the Xeon execution profile built for it.*

Diamond Rapids removes SMT so every thread owns a full core, which is what predictable per-thread latency under load requires, and it serves the sequential loop steps directly. Running hundreds of concurrent sandboxes is a density problem, and IAA-accelerated snapshotting delivers roughly 1.2x faster completions [5]. Data services belong on SMT2 P-cores, with AMX keeping embeddings, reranking and guardrails on the host instead of hopping off-node. State movement runs on the on-die engines at zero core cost. **A fleet that places each**

phase on the profile built for it, converts more of its theoretical capacity than any uniform fleet built around a single "best" core.

6. Summary

We are asking the industry to be judged on delivered capacity, so we will report Intel results in those terms: **agents per rack** at a stated p99 latency target on real agentic traces; **Routing Efficiency**, actual agents versus theoretical; the **infrastructure offload factor** and agent-density gain from QAT, DSA and IAA at the same target; **host efficiency** on GPU-attached configurations, replicating the published external benchmark matrix on a Xeon host for a like-for-like baseline; and **agents per watt** under the same 100 kW rack constraint the industry has adopted.

Every thread fed. Every session warm. More agents delivered per rack, on Intel Xeon.

References

[1] I. Buck, "AI Innovators Adopt NVIDIA Vera: Why Max Single-Threaded CPU at Scale Matters," NVIDIA Blog, July 7, 2026.

[2] AMD, "Performance Projections Methodology: Computing in the Agentic AI Era," June 2026.

[3] A. Luo, "CPU-GPU Co-Design for Agentic LLM Inference," May 14, 2026. https://andyluo7.github.io/llm/amd/mi300x/vllm/lmcache/performance/2026/05/14/cpu-gpu-codesign-agentic-inference-mi300x

[4] NVIDIA, "Inside NVIDIA Vera CPU: Olympus Cores Built for Maximum Single-Threaded Performance in Agentic AI," NVIDIA Developer Blog, 2026.

[5] D. Lazarev et al., "Sabre: Hardware-Accelerated Snapshot Compression for Serverless MicroVMs," USENIX OSDI '24.

[6] "Exploiting Locality in Flat Memory with CXL for In-Memory DBMSs", SAP & Intel, DaMoN workshop, SIGMOD '25

Figure 1 reproduces measured data from the cited study [3]. Figures 2 through 5 are illustrative: not measured product results.