White Paper

Data Center & Artificial Intelligence
Agentic AI Infrastructure Sizing

Agentic AI Requires More CPUs

Avoiding GPU Idle Time in Modern Inference Pipelines

Authors
Steve Fowler
Josh Segovia
Lawrence Leung
Laurie Fordham
Stephen Holt

Executive Summary

Inference has replaced training as the dominant AI computing cost, while shifting from GPU heavy single pass inferencing to agentic workflows.[1] These agentic workflows, which function like distributed systems, introduce CPU-intensive stages, such as orchestration, reflection, and planning, which can gate overall throughput and bottleneck GPU utilization.[2] GPUs are expensive in both capital expenditure and operations; any delays caused by a lack of CPU resources can leave these GPUs idle, resulting in a less efficient and more expensive system. This paper examines how increased CPU compute alleviates these bottlenecks, thereby reducing costs by sustaining GPU saturation.[2,3]

It is worth noting that this whitepaper does not suggest replacing the GPU inference function. Instead, it recommends that commercial entities reduce infrastructure overhead and cost by minimizing costly GPU idle time. This can be achieved by expanding pre-work, tool-work, and post-inference work across as many general compute CPUs as possible. After some benchmark testing, we conclude with current and future guidance on the ratios between the AI Accelerator and General Compute CPU.

From Single-Pass Inference to Agentic AI Systems

Traditional single-pass inference follows a straightforward execution model: a user prompt is tokenized, passed through the model once, and the output tokens are generated in a single forward pass.

Agentic AI fundamentally changes inference from a single model call into a coordinated system, much like modern microservices architecture[4] rather than a monolithic application. The orchestration layer is responsible for planning tasks, managing control flow, handling branching and retries, and coordinating interactions among multiple agents and tools, with these functions primarily executed on CPUs. Each agent operates as an independent service invocation and may engage external resources—such as databases, search engines, vector repositories, or APIs, which consequently generate additional CPU, memory, and I/O overhead. For code and reasoning workloads, agentic systems often spin up isolated execution environments, such as sandboxes, interpreters, or test iterations to validate results before proceeding. As these loosely coupled steps execute and repeat across multiturn workflows, CPUs determine overall throughput and directly impact how efficiently GPUs can be utilized.[4] As illustrated in Figure 2, depending on the task the agent is working on, the amount of CPU resources required will fluctuate with the user-requested task.

Leading AI builders (e.g., OpenAI, Anthropic) are increasingly delivering orchestration frameworks, SDKs, and packaged workflows that treat agents as long-running, multistep systems rather than as single model invocations. Platforms now provide explicit agent loops, tool calling interfaces, and observability layers to manage planning, retries, and coordination across tools and services. Several vendors offer agent management platforms that bundle models with orchestration logic, retrieval pipelines, and execution environments, enabling customers to deploy agentic workflows with minimal custom integration code. Collectively, these offerings signal a clear shift across the industry: value is moving up the stack from raw model inference to system-level orchestration, where control flow, tool execution, and workflow management dominate runtime behavior and infrastructure requirements.

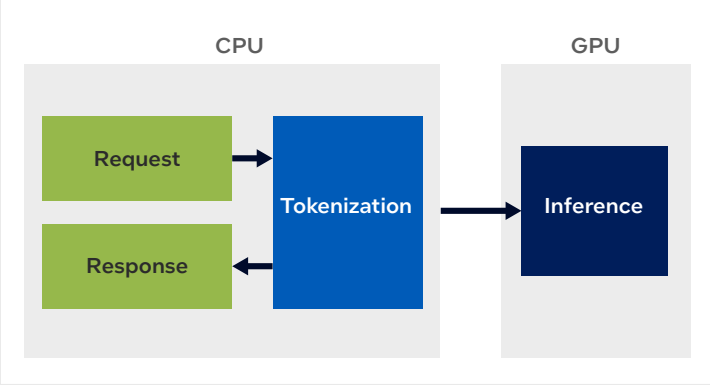

Figure 1: Single-pass inference model.

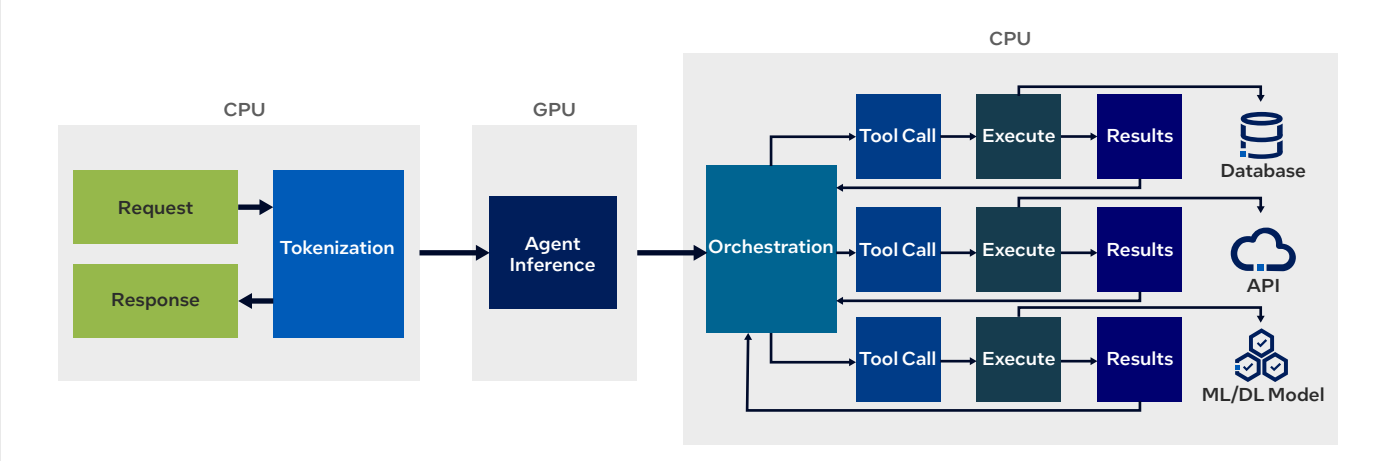

Figure 2: Example agentic workflow.

Test Results

General Agentic AI Workflows

To measure the CPU demand generated by AI-powered agentic workflows, we built an anomaly-detection pipeline based on a real financial services workflow involving SEC regulatory filings. The pipeline works like a junior compliance analyst reviewing filings. First, the CPU does the legwork of loading the data, establishing each company's typical filing pattern, flagging companies that deviate from it, downloading the actual filing documents, and condensing each multi-page document into a short summary. The agent then calls a tool that calculates a score and compares it to a pre-determined threshold. To mimic workflows in which datasets are augmented with additional material, we include a web search enrichment step. Here, the AI model decides when to fill in missing context, such as recent news about the company or regulatory action. The test case concludes by leveraging the LLM call to generate an analysis for each company above the calculated threshold. This experiment was performed on an Intel® Xeon® Platinum 8568Y+ dual-socket system, an Intel® Gaudi® 3 card, and serving Llama 3.1 8B Instruct on a single accelerator card. The results in Figure 4 show that the CPU consumes the majority of the total pipeline, with the enrichment step taking three times as long as LLM inference.

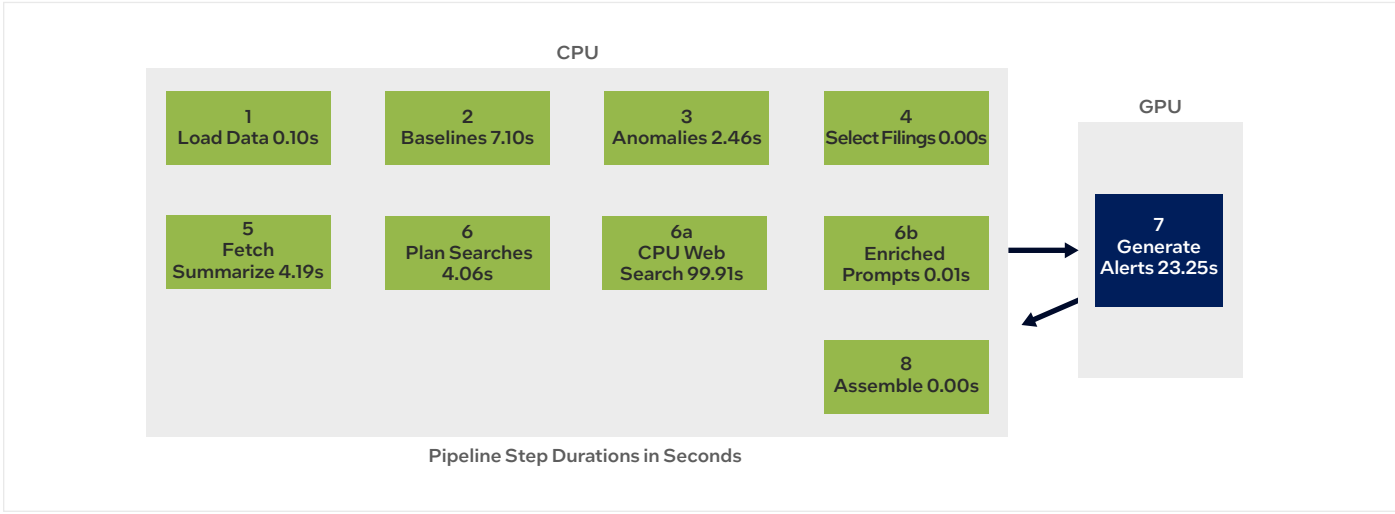

Figure 3: Example of agentic AI workflow steps.

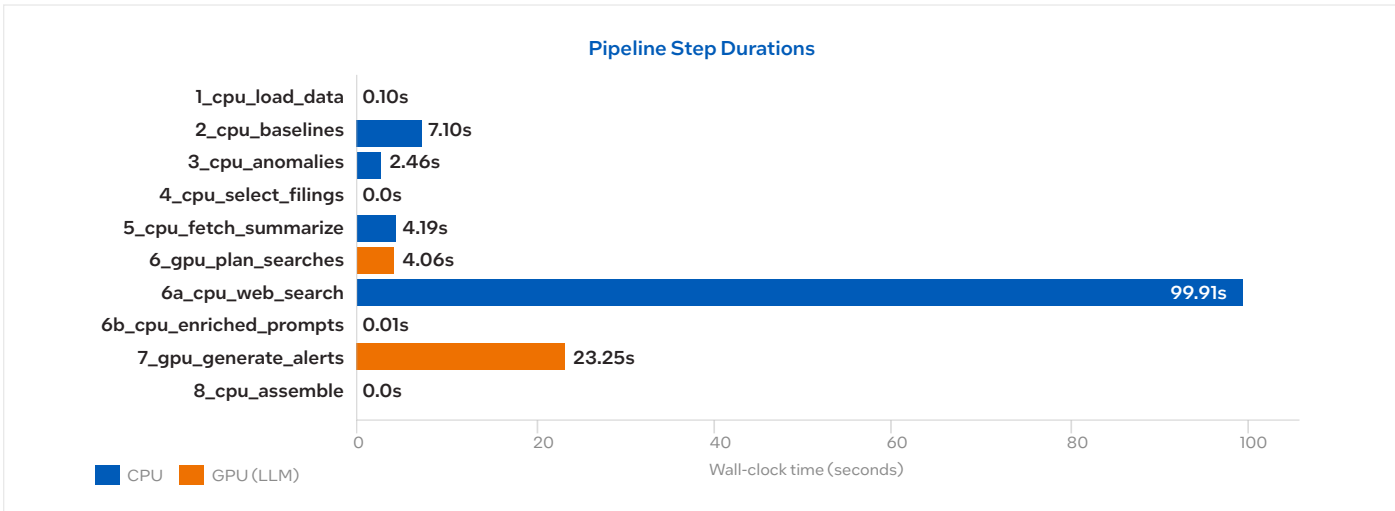

Figure 4: Example of generalized agentic AI workflow.

Code Generation Workflows

This additional test aimed to quantify the CPU demand generated by today's typical AI code-generation workflows. To do this, we built a benchmark that mirrors what happens when a developer asks an AI assistant to write and test code. For example, adding a feature to a website or updating an existing user interface. The workflow has two phases: first, the AI model on the GPU or accelerator generates a candidate code solution; second, the CPU executes and verifies the code.

2,140 data science and software engineering problems were drawn from industry standard benchmarks, DS-1000 and BigCodeBench. Each problem includes the description, test cases, any starter code, platform, and difficult metadata. The system and LLM model are the same as in the previous test, with the Intel Xeon 8568Y+ executing each generated code in parallel sandbox environments to verify correctness. Each sandbox is run as a subprocess with limited filesystem access. Actual production systems use containers or VMs, which would add additional CPU overhead.

Code generation (GPU) completed in 62.8 seconds, while sandbox execution on the dual-socket Intel Xeon (CPU) consumed 64.1 seconds, making the CPU the limiting factor even with 80 physical cores and 160 threads fully engaged. Each sandbox averaged 1.8 seconds of execution time, with the longest task taking over 16 seconds, driven by library imports and test suite complexity.

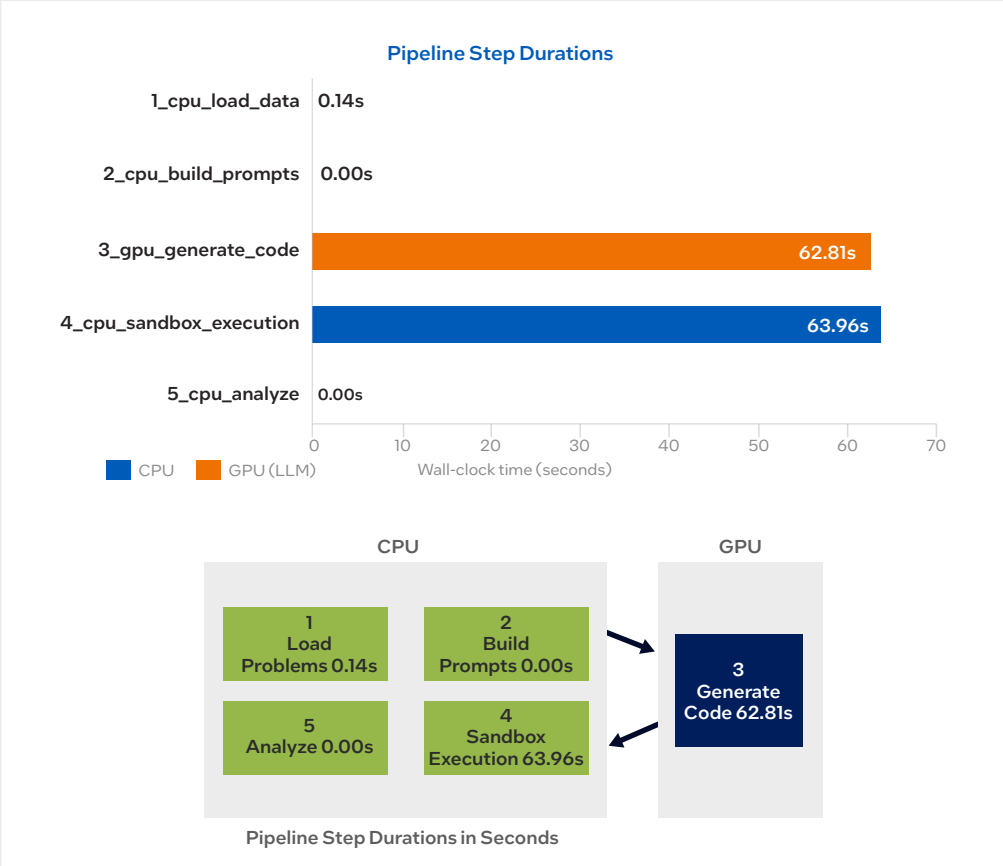

Figure 5: Total time spent generating all 2,140 solutions and high-level workflow.

Conclusion

This architectural shift has direct implications for how inference infrastructure must be sized and evaluated. Just as with microservices-based applications, performance and cost efficiency are no longer determined by a single component, but by how well the entire system is balanced.

In Agentic AI deployments, insufficient CPU capacity can stall orchestration, delay tool execution, and slow verification loops, leaving expensive GPUs underutilized while they wait for the control plane (agent orchestration) to catch up. Optimizing agentic workloads requires increasing the number of CPUs to act as the control and orchestration layer, sustaining the distributed, multi-step nature of modern AI workflows while efficiently feeding accelerators.

Current Recommendation

Based on testing for this paper, using Intel Xeon 6767P CPUs with Nvidia B200 GPUs for both Agentic AI and Code Generation use cases, we found that a CPU-to-GPU ratio of 1:1 to 1.4:1 keeps the GPUs busy. For a detailed breakdown of the changes that affect this ratio, refer to the appendix.

Future Recommendation

As we look forward to future high-performance GPUs, if pairing current-generation CPUs with newer GPUs, the CPU-to-GPU ratio could reach 7:1, depending on the workload and how close future GPU performance comes to published forecasts.

For example, when Agentic AI expands to the physical world, AI errors can put lives at risk. Because of this risk, it requires a lot of general compute CPUs to perform sequential error-correction calculations before any interaction with the real world occurs. If we look at the expanding use of Robotics, we see the continued heavy use of CPUs, with GPUs handling intensive computation and CPUs handling everything else. Running the simulated training environments, coordinating the broader system, and delivering precise, low-risk, real-time commands to every joint and actuator are all tasks that depend on the CPU. Under-spec the number of CPUs, and the entire pipeline suffers, introducing risk and wasting money regardless of GPU capability.

As AI workloads evolve from single-pass inference to complex agentic pipelines, CPUs have emerged as a critical, and often underestimated, component of inference infrastructure. Organizations that fail to provision sufficient CPU capacity risk leaving their expensive GPU accelerators idle should currently target a CPU-to-GPU ratio of 1:1 to 1.4:1 to maximize throughput, minimize waste, and future-proof their AI deployments.

Appendix

Sizing Guidance

In the recommendations section, we provide a CPU to GPU ratio for current (as of March 2026) hardware of 1:1 to 1.4:1. Another way to look at this is the number of CPU Cores per GPU, which would be 86:1 to 120:1. In this section, we want to provide some guidance on how changes to the workload or hardware may change this ratio.

Model Size (Parameters)	Quantization	Change	Example Ratio
8 B	FP4	More CPU needed	1.4:1
70B	FP4	Baseline	1:1
405B	FP4	Less CPU needed	0.8:1

Changing the model size affects the number of tokens per second generated by the GPU, with smaller sizes resulting in the GPUs doing more work in the same amount of time, requiring additional CPUs to keep them busy. Upgrading to a newer CPU generation, which would increase the amount of work the CPU can do in the same amount of time, would reduce the number of CPUs needed to keep the GPUs busy. The overall guidance is that if the GPU can generate more tokens per second, then additional CPUs are needed to keep them busy.

[1] https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-next-big-shifts-in-ai-workloads-and-hyperscaler-strategies

[2] https://openai.com/index/new-tools-for-building-agents/

[3] https://www.anthropic.com/engineering/advanced-tool-use

[4] https://developers.redhat.com/articles/2026/02/11/agentic-ai-design-reliable-workflows-across-hybrid-cloud