WHITEPAPER

SUPERMICRO X14 HGX B200 GPU SERVERS WITH INTEL® XEON® 6 PROCESSORS DELIVER UP TO 1.44X THE CONCURRENT-USER CAPACITY

Intel® Advanced Matrix Extensions (Intel® AMX) Enhance vLLM Inference Through CPU-GPU Co-Serving: A Multi-Agent Architecture Leveraging Supermicro HGX B200 System

Supermicro GPU Server SYS-822GS-NBRT

TABLE OF CONTENTS

Executive Summary

Modern large language model (LLM) serving systems often run most inference workloads on GPUs, while the host CPUs in GPU systems often remain idle after scheduling, tokenization, and data movement are complete. This paper introduces a heterogeneous architecture that co-runs vLLMs on both CPUs and GPUs to improve overall system efficiency through multi-agent or vLLM semantic routing across two inference endpoints: a CPU pathway and a GPU pathway. Instead of reserving every request for the largest GPU-hosted model, the architecture assigns smaller CPU-hosted vLLMs to research, extraction, validation, and short-response tasks when large-model capacity is unnecessary. This enables the GPU-hosted 405B model to focus on long-form generation and complex synthesis, while Intel Xeon 6 processors with Intel® AMX handle CPU-suitable work

© 2026 Copyright Super Micro Computer, Inc. All rights reserved

in parallel. The result is better use of installed compute resources, higher supported user concurrency, and a stronger cost-per-concurrent-user story for mixed LLM/SLM production deployments.

The architecture consists of three coordinated agents: CPU-based Researcher and Reviewer agents responsible for context preparation, rapid feedback, and validation, alongside a GPU-based Writer agent dedicated to high-quality response generation. This division of work reflects a practical application pattern: use smaller, lower-cost CPU inference for fact finding, summarization, critique, and guardrail-style review, and reserve premium GPU capacity for the parts of the workflow that benefit most from the larger 405B model. The closed-loop design is intended to keep CPU and GPU resources engaged simultaneously, reducing idle host capacity and increasing the total number of users served per system.

For marketing and customer-facing positioning, the value is not only raw speed. The same GPU server can support a tiered inference design in which GPU capacity is reserved for high-value generation and CPU capacity absorbs short, repetitive, or validation-heavy tasks. That can reduce the need to scale GPU capacity for each incremental user request and improve the effective cost per concurrent user when the workload mix includes CPU-suitable tasks.

CPU-GPU Co-Serving Increases Concurrency on Supermicro Systems with Intel Xeon 6 and AMX

This implementation extends vLLM with a heterogeneous architecture that splits inference across two coordinated endpoints: a CPU pathway running a Llama-3.1-8B model for lightweight reasoning and compute-intensive tasks such as research extraction, summarization, critique, and response validation, and a GPU pathway running a Llama-3.1-405B model for compute-intensive generation, including long-form structured outputs, deep reasoning, and multi-step synthesis.

The system leverages vLLM's multiprocessing executor, NUMA-aware CPU binding, and OpenMP tuning, with Intel Xeon 6 AMX acceleration optimizing BF16 inference and KV cache management on the CPU side. At the same time, NVIDIA Blackwell-class GPUs handle complex prompts using the 405B model. The two endpoints from each CPU and GPU are deployed as independent vLLM servers. The reference GitHub implementation maps Researcher and Reviewer agents to the CPU 8B endpoint and the Writer agent to the GPU 405B endpoint. The CPU service uses BF16 inference with Intel Xeon 6 AMX, while the GPU service handles high-throughput decoding for the 405B model on Supermicro HGX B200 System. The CPU and GPU endpoints are deployed as independent vLLM servers and orchestrated by the multi-agent application. The supporting CPU-binding workflow generates a Docker Compose override file based on the Intel Xeon 6 CPU-binding table. GPU services run on NUMA-local CPUs, PCT cores can be assigned to latency-sensitive GPU threads, and remaining CPUs can be assigned to CPU-only inference. The same workflow supports deploy and benchmark modes for both the GPU and CPU services.

Performance results demonstrate the advantages of Intel Xeon 6 processors with Intel® AMX in a heterogeneous setup that leverages both CPU and GPU compute on the same AI-accelerated system. The benchmark report uses the official vLLM Docker images and the vLLM benchmark suite for both CPU and GPU testing. It follows the vLLM manual benchmark flow, including adaptive concurrency search to find the maximum concurrency closest to service-level thresholds. The CPU endpoint runs Llama-3.1-8B-Instruct with 128 input and 128 output tokens for lightweight reasoning, orchestration, and validation. The Supermicro HGX B200 GPU endpoint runs Llama-3.1-405B-Instruct with 2048 input and 2048 output tokens for large-scale generation and synthesis. In the cited Intel test, CPU and GPU vLLM containers were measured while both sides of the system were kept busy with user requests, reflecting the intended full-system usage model. The GPU-only B200 workload used 12 CPU cores and supported 127 concurrent users. Reusing the remaining Intel Xeon 6 cores for the CPU 8B endpoint added 56 concurrent CPU users, bringing the total to 183 concurrent users, or up to 1.44X more concurrent users than the GPU-only

© 2026 Copyright Super Micro Computer, Inc. All rights reserved

baseline. The benchmark is an LLM/API endpoint-level serving benchmark, not an end-to-end application benchmark of the full multi-agent user flow. The benchmark also showed that proper CPU binding matters: when the CPU model was allowed to contend across all CPUs without binding, CPU-side supported users dropped from 56 to 51, a reduction of approximately 9%.

This parallel execution enables efficient workload distribution across the pipeline and maximizes utilization of both compute domains. With Intel® AMX acceleration on Xeon 6, the system achieves up to 1.44X higher user concurrency by running the workload on both CPU and GPU simultaneously, demonstrating improved throughput and serving efficiency. The evaluation is conducted on a Supermicro SYS-822GS-NBRT HGX platform equipped with 2× Intel Xeon 6 6776P processors (AMX-enabled) and 8× NVIDIA HGX B200 GPUs, highlighting the effectiveness of CPU–GPU co-serving for scalable vLLM inference. For reproduction, use the deployment scripts from the Intel AI TCE vLLM CPU binding branch and follow Benchmark Mode. The vLLM manual flow documents that CPU tests use the CPU container image and the ON_CPU environment variable, while adaptive concurrency can be enabled with ENABLE_ADAPTIVE_CONCURRENCY=1. The benchmark flow generates benchmark_results.md and benchmark_results.json, which are used to compare results across concurrency, TTFT, and TPOT criteria.

In a multi-agent use case, the Researcher agent initiates execution first, the Writer agent begins as soon as partial context or notes become available, and the Reviewer agent starts once a partial draft is produced. This overlapping, staged execution enables pipelined processing across agents, reducing overall end-to-end latency compared to strictly sequential execution.

Without pipelining, the execution is strictly sequential, so end-to-end latency accumulates across stages:

Total latency = Research + Write + Review

With the multi-agent pipelined design, CPU and GPU stages run concurrently with overlapping execution:

Total latency = MAX (Research, Write, Review)

This enables a pipelined execution model where CPU agents continuously prepare, retrieve, and validate context while GPU agents focus on generation, allowing overlapping execution across stages rather than sequential processing; consequently, end-to-end latency shifts from an additive pipeline to a parallelized bound, improving hardware utilization, reducing GPU idle time, stabilizing tail latency, and increasing overall concurrency in production-scale vLLM deployments.

Block Diagram:

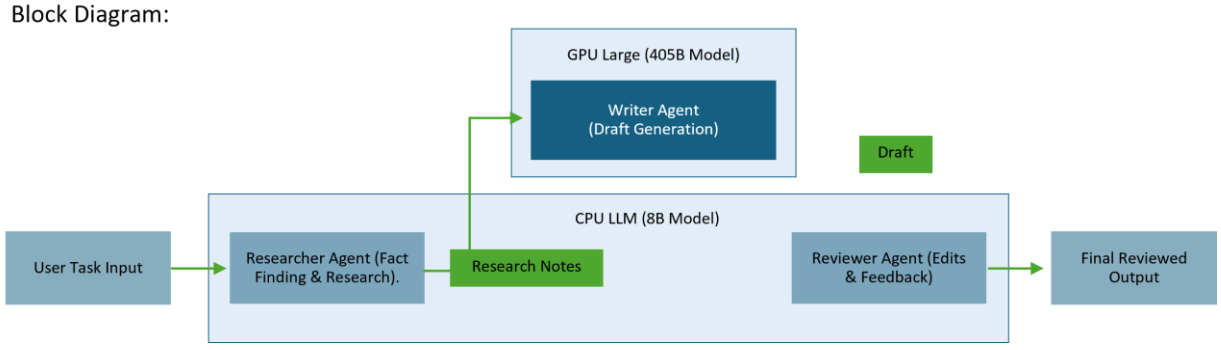

Figure 1 - Block Diagram of the Hardware Design

© 2026 Copyright Super Micro Computer, Inc. All rights reserved

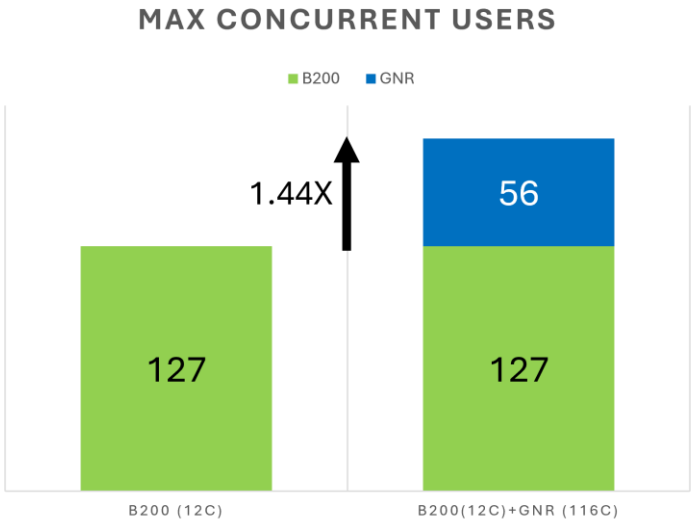

Figure 2 - Performance Comparison of User Concurrency: GPU-Only vs. CPU-GPU Co-Serving[1]

What is Intel® Advanced Matrix Extensions (Intel® AMX)

Intel® AMX is one of two Intel® AI Engines integrated into Intel® Xeon® 6 processors with P-cores, which can help you make the most of your CPU to power AI training and inferencing workloads at scale for benefits including improved efficiency, reduced inferencing, training, and deployment costs, and lower total cost of ownership (TCO). As a built-in accelerator on each CPU core, near system memory, Intel® AMX is often easier to use than discrete accelerators, leading to faster time to value.

How Intel® AMX Works

Intel® AMX is a dedicated hardware block found on the Intel® Xeon® Scalable processor core that helps optimize and accelerate deep learning training and inferencing workloads that rely on matrix math.

Intel® AMX enables AI workloads to run on the CPU rather than offloading them to a discrete accelerator, delivering a significant performance boost. Its architecture supports BF16 (training/inference) and int8 (inference) data types and includes two main components:

- Tiles: These consist of eight two-dimensional registers, each 1 kilobyte in size, that store large chunks of data.
- Tile Matrix Multiplication (TMUL): TMUL is an accelerator engine attached to the tiles that performs matrix-multiply computations for AI.

Together, these components enable Intel® AMX to store more data in each core and compute larger matrices in a single operation. Additionally, Intel® AMX is architected to be fully extensible and scalable.

Supermicro System Used for Testing

The Supermicro SYS-822GS-NBRT is an 8U GPU system designed to integrate eight Supermicro HGX B200 GPUs with dual Intel Xeon 6 processors into a single platform. In this test, CPU binding was used to allocate the CPU resources needed by the GPU-hosted 405B service while preserving remaining cores for CPU-hosted inference. The reported software stack used official vLLM

© 2026 Copyright Super Micro Computer, Inc. All rights reserved

Docker images, the vLLM benchmark suite, vLLM 0.17.0 public release, CPU-binding scripts from the Intel AI TCE vLLM CPU_binding branch, and the public vLLM CPU release container.

System	SYS-822GS-NBRT
CPUs	Dual Intel Xeon 6776P processors (64 cores, 350W TDP)
Memory	2048GB (32x64GB DDR5 6400MT/s [5200MT/s])
GPUs	NVIDIA HGX B200 8-GPU with 5th Generation NVLink® 1.8TB/s, 2.3TB of HBM3e GPU memory per system

Summary

On a Supermicro SYS-822GS-NBRT HGX platform equipped with 2x Intel Xeon 6 6776P processors and 8x Supermicro HGX B200 GPUs, the cited Intel test reports up to 1.44X the supported user concurrency for a CPU-GPU configuration versus a GPU-only baseline. The result comes from a simple but important deployment insight: the 405B GPU service required only a small portion of the host CPUs, while the remaining Intel Xeon 6 cores could run an 8B CPU model for useful agent work. With NUMA-aware CPU binding and optional Priority Core Turbo guidance, the platform can protect the latency-sensitive GPU service while using idle CPU capacity for CPU inference. This pattern is best suited to applications that need both small and large language models, including multi-agent pipelines, LLM/SLM co-serving, and semantic-routing architectures that send suitable requests to smaller models. For organizations deploying mixed LLM and SLM applications, this creates an attractive path to serve more users from the same server and improve cost efficiency without adding another GPU for every lightweight or validation-oriented task.

Higher user concurrency demonstrates that CPU–GPU co-serving and pipelined execution can substantially enhance the efficiency, scalability, and resource utilization of end-to-end vLLM serving within a single AI-accelerated system. Overall, the results validate heterogeneous co-serving as a practical, scalable strategy for optimizing production vLLM workloads and showcase the performance advantages enabled by Intel® AMX acceleration.

For More Information:

Supermicro SYS-822GS-NBRT: https://www.supermicro.com/en/products/system/gpu/8u/sys-822gs-nbrt
Intel® Advanced Matrix Extensions (Intel® AMX) Details:
https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/what-is-intel-amx.html

Intel AI TCE CPU Binding Deployment and Benchmark Mode: https://github.com/intel-ai-tce/vllm/tree/cpu_binding/benchmarks/cpu_binding
Intel AI TCE Multi-Agent Two-Endpoint Demo: https://github.com/intel-ai-tce/vllm/tree/cpu_binding_demo/benchmarks/cpu_binding/multi_agent_two_endpoints
vLLM Semantic Router Use Case Example: https://vllm-semantic-router.com/
vLLM Benchmark Manual: https://docs.vllm.ai/en/latest/benchmarking/dashboard/#manually-trigger-the-benchmark

© 2026 Copyright Super Micro Computer, Inc. All rights reserved

SUPERMICRO

As a global leader in high performance, high efficiency server technology and innovation, we develop and provide end-to-end green computing solutions to the data center, cloud computing, enterprise IT, big data, HPC, and embedded markets. Our Building Block Solutions® approach allows us to provide a broad range of SKUs, and enables us to build and deliver application-optimized solutions based upon your requirements. Visit www.supermicro.com

INTEL

Intel (Nasdaq: INTC) is an industry leader, creating world-changing technology that enables global progress and enriches lives. Inspired by Moore's Law, we continuously work to advance the design and manufacturing of semiconductors to help address our customers' greatest challenges. By embedding intelligence in the cloud, network, edge and every kind of computing device, we unleash the potential of data to transform business and society for the better. To learn more about Intel's innovations, visit

Visit www.intel.com

[1] Results may vary based on system configuration, workload characteristics, hardware environment, service level objectives, and other operational factors. Performance improvements are dependent on proper setup and optimization. Configuration: 1-node, Supermicro X14DBG-LC+, 2x Intel(R) Xeon(R) 6776P, 64 cores per socket, 350W TDP, HT On, Turbo On, Total Memory 2048GB (32x64GB DDR5 6400MT/s [5200MT/s]), BIOS 1.5, microcode 0x1000405, 3x Unknown NIC, 2x Ethernet Controller X710 for 10GBASE-T, 1x 7T SAMSUNG MZTL67T6HBLC-00AW7, 1x 3.5T SAMSUNG MZTL23T8HCLS-00A07, Ubuntu 22.04.5 LTS, 6.8.0-90-generic. Software: official vLLM Docker images, vLLM benchmark suite, vLLM 0.17.0 public release, CPU-binding scripts from https://github.com/intel-ai-tce/vllm/tree/cpu_binding/benchmarks/cpu_binding, and public.ecr.aws/q9t5s3a7/vllm-cpu-release-repo:v0.17.0. Workload: Llama-3.1-8B-Instruct on Intel Xeon 6 CPU with BF16/AMX BF16, 128 input tokens and 128 output tokens; Llama-3.1-405B-Instruct on Supermicro HGX B200 with BF16, 2048 input tokens and 2048 output tokens. Benchmark method used adaptive concurrency search under SLA thresholds described in the vLLM benchmark manual. CPU and GPU endpoint measurements were taken while both CPU and GPU services were kept busy with user requests. The result is an LLM/API endpoint serving a benchmark, not an end-to-end multi-agent application benchmark. Test by Intel as of March 21, 2026.

© 2026 Copyright Super Micro Computer, Inc. All rights reserved