

Deploying small language models on Red Hat OpenShift with vLLM

HPE ProLiant Compute DL380 Gen12 with Intel® Xeon® 6 processors





Lowering the cost for enterprise AI inference

Enterprises are rapidly adopting generative AI to power conversational assistants, document summarization, and internal knowledge services—yet many struggle to operationalize these workloads at scale. While GPUs remain essential for highly parallel AI tasks, CPUs can efficiently support many workloads—such as traditional machine learning, data preprocessing, feature engineering, and inference for lighter models—providing a more cost-efficient and scalable compute balance.

To solve these challenges, this solution guide presents an enterprise-ready approach to small language model (SLM) inference using **HPE ProLiant Compute DL380 Gen12** servers powered by **Intel® Xeon® 6 processors**, deployed on **Red Hat® OpenShift** with **virtual large language model (vLLM)**. Together, HPE ProLiant Compute DL380 Gen12 servers and Intel Xeon 6 processors deliver the optimized CPU performance, memory bandwidth, and cache architecture needed to scale AI workloads efficiently across enterprise environments. By leveraging CPU-based inference optimized through advanced memory management, Non-uniform memory access (NUMA)-aware scheduling, and efficient token handling, enterprises can deploy AI services that meet necessary latency and throughput requirements.

To validate this architecture, the solution overview evaluates both bare-metal and containerized OpenShift deployments, analyzing key performance indicators such as time per output token (TPOT), throughput, and concurrent user capacity across real-world AI workloads. single-node OpenShift deployment running vLLM on Intel Xeon 6 processors consistently meet a 100 ms TPOT service-level objective and scale up to 100 concurrent users. Improving scalability and efficiency compared to previous-generation Xeon platforms. This validated architecture enables enterprises to accelerate AI adoption using familiar infrastructure, standardized Kubernetes operations, and a lower total cost of ownership.

Industry verticals and use cases

Table 1. SLM use cases across industry verticals

Industry vertical	Use case 1	Use case 2	Use case 3	Use case 4	Use case 5
Financial services	Customer service virtual assistants	Document and report summarization	Internal knowledge assistants for advisors	Fraud investigation case summarization	Regulatory and compliance Q&A
Healthcare & life sciences	Clinical documentation summarization	Medical knowledge assistants	Administrative workflow automation	Research and clinical trial summarization	Internal IT and HR assistants
Manufacturing & industrial	Maintenance and troubleshooting assistants	Engineering document summarization	Supply chain intelligence assistants	Factory operations knowledge bots	IT and OT support chatbots
Retail & e-commerce	Customer support chatbots	Product catalog and content summarization	Employee knowledge assistants	Policy and training assistants	Conversational access to sales and inventory data
Public sector & education	Citizen service assistants	Policy and regulation summarization	Internal workforce assistants	Education and research assistants	Institutional knowledge management



Solutions architecture

Figure 1 illustrates a layered, CPU optimized AI inference architecture built for enterprise deployment. At the software layer, vLLM, PyTorch, Hugging Face, and oneDNN provide efficient SLM providing optimized memory and token handling. This stack runs on Red Hat OpenShift and Red Hat Enterprise Linux® (RHEL), delivering Kubernetes based orchestration, scalability, and manageability. This jointly optimized software layer builds on Intel® acceleration within oneDNN and CPU-tuned frameworks, enabling HPE ProLiant DL380 Gen12 systems to deliver higher, more consistent inference throughput for production-ready SLM deployments. The infrastructure layer is powered by HPE ProLiant Compute DL380 Gen12 servers with Intel Xeon 6 processors, enabling predictable, cost-efficient AI inference on standard enterprise platforms.

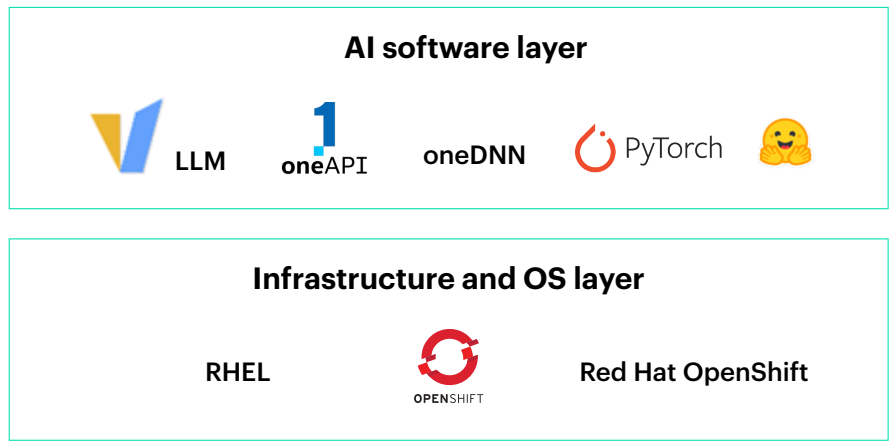


Figure 1. Architecture overview for SLM on DL380 Gen12 servers

This architecture seamlessly integrates advanced CPU-optimized inference with the robust container orchestration capabilities of Red Hat OpenShift. By combining HPE ProLiant Compute DL380 Gen12 servers and Intel Xeon 6 processors, the solution offers high availability and simplified management for enterprise AI workloads. The use of vLLM enables efficient deployment and scaling of SLMs within Kubernetes environments, supporting dynamic resource allocation and rapid response to changing AI demands. The end-to-end CPU-optimized AI inference software stack is enabled with open-source tools/frameworks—vLLM, PyTorch, oneDNN, Huggingface.

Table 2 shows sizing guidance for small, medium, and large deployments' conversational chatbot customers.

Table 2. Sizing guidance

	Small	Medium	Large
Number of concurrent users (256 input / 1024 output token)	100	400	800
Number of concurrent users (1024 input / 2048 output token)	60	240	480
Number of concurrent users (1024 / 1024 token count)	64	256	512
HW configuration	1 x HPE ProLiant Compute DL380 Gen12 with 2x Intel Xeon 6767P Processors	4 x HPE ProLiant Compute DL380 Gen12 with 2x 6767P Processors	8 x HPE ProLiant Compute DL380 Gen12 with 2x 6767P Processors
Total processor count	2	8	16

Server configuration

Table 3. Hardware configuration

System	HPE ProLiant Compute DL380 Gen12
Model name	HPE ProLiant Compute DL380 Gen12 2U rack
Processor	Intel Xeon 6767P (64 cores per socket), 336 MiB
Sockets	2
Memory	1024 GB (16 x 64 GB DDR5 6400 MT/s)
NIC	1x Ethernet Controller 1 Gb MT2892 family
Disk	1x 1 Tb HPE NS204i-u Gen11 NVMe Hot Plug Boot Optimized Storage Device
Operating system	RHEL CoreOS 9.6.20250729-1 (Plow)
TDP	350W

Software configuration

Table 4. Software configuration

Workload	vLLM serving benchmark
Application	Meta-LLAMA 3.1 8B-Instruct
Tools/compilers	GCC 12.3.0
Middleware, framework, runtimes	Python 3.10.12, Torch 2.6.0+cpu, intel_extension_for_pytorch-2.6.0+cpu, intel-openmp 2024.2.1, vLLM v0.10.2
Num prompts	Sweep—Between 1 to 128
Input tokens	256/1024
Output tokens	256/1024/2048
Libraries	
Orchestration	Red Hat OpenShift 4.19.7
Containers and virtualization	Podman 5.6.0 crio version 1.33.4-4.rhaos4.20.git08e7df2.el9

Red Hat instructions and best-known methods guide

The [Red Hat instruction guide](#) provides concise, step-by-step best practices for optimizing OpenShift deployments, including feature gate configuration, pod resource tuning, and NUMA-aware scheduling. It enables enterprises to boost scalability, security, and performance for CPU-based AI inference with vLLM and serves as a practical reference for IT administrators and engineers.



Use case #1: Conversational chatbot performance with Llama 3.1-8B T

This use case evaluated the performance of the Meta-Llama 3.1-8B-Instruct conversational model on Intel Xeon-based platforms running Red Hat OpenShift, with the goal of boosting concurrent user support while meeting a strict 100 ms TPOT SLA. As shown in Figure 2, one HPE ProLiant Compute DL380 Gen12 with 2x Intel Xeon 6 processors supports up to 120 concurrent users at 256 input/output token lengths. Testing used vLLM's benchmark_serving.py with fixed prompt sizes (256, 1024 tokens) and incremental output targets (256 to 2048 tokens), enabling tensor parallelism, cache prefixing, chunked prefill, and the v1 architecture. The deployment was optimized for sub-NUMA clustering to efficiently scale across CPU sockets, delivering high inference throughput on HPE solution architecture.

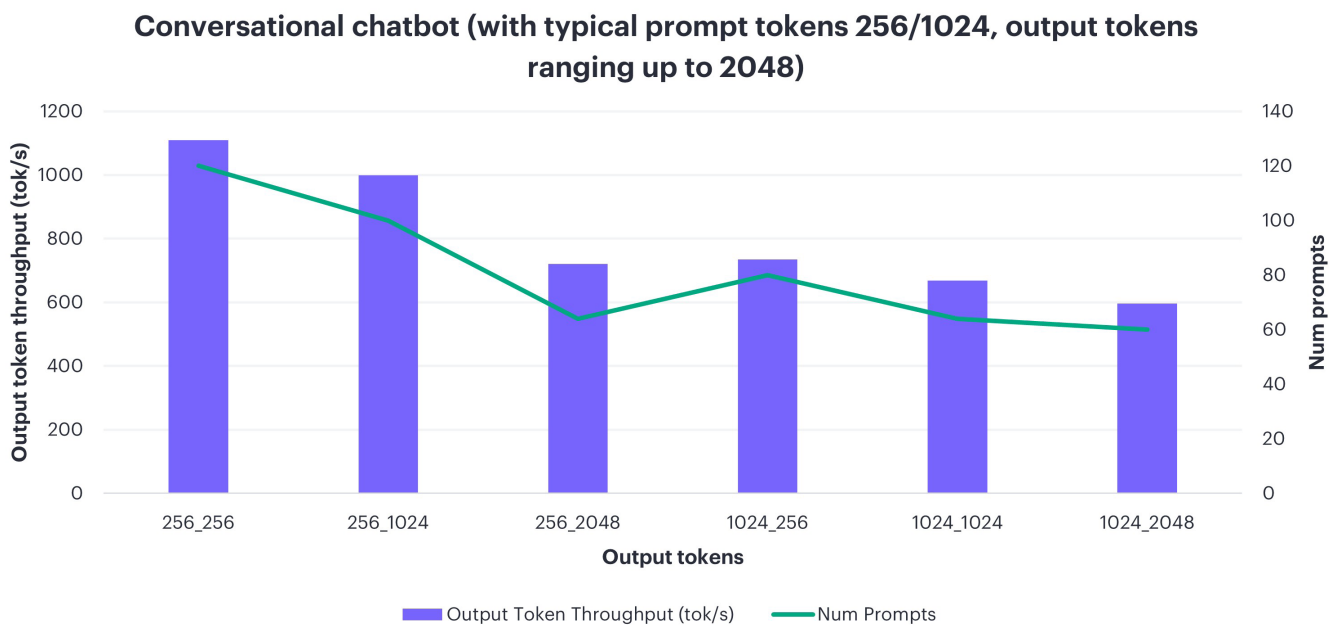


Figure 2. Conversational chatbot performance

The conversational chatbot performance charts demonstrate how the architecture sustains high throughput while maintaining strict latency targets across varying prompt sizes and output lengths. As input and output token counts increase, the results show predictable scaling behavior, balancing output tokens per second with supported concurrent users. This highlights the effectiveness of vLLM optimizations, NUMA aware scheduling, and CPU tuning in delivering consistent inference performance. The charts reinforce that the platform can support real world conversational workloads with stable response times, enabling reliable user experiences while scaling concurrency on standard enterprise CPU infrastructure.

Results summary—Conversational chatbot

The chart (Figure 2) illustrates two key performance metrics: The left y-axis showcases output tokens per second (throughput), and the right y-axis showcases the Maximum concurrent users supported while maintaining the 100 ms TPOT SLA. The x-axis shows the various input/output token combinations used to run the benchmark.

The results demonstrate that Intel Xeon 6 processors, when deployed with an upstream vLLM inference server, consistently meet the stringent 100 ms TPOT SLA. These improvements are driven by higher base CPU frequencies, expanded L3 cache capacity, and increased memory bandwidth, enabling superior throughput, scalability, and efficiency for enterprise-grade SLM inference workloads. This jointly engineered environment incorporates Intel acceleration within oneDNN and CPU-optimized frameworks, empowering HPE ProLiant DL380 Gen12 systems to deliver higher, more consistent inference throughput for production-ready SLM deployments.

Use case #2: Document summarization Performance with Llama 3.1-8B T

This use case evaluates a long-context generative AI workload focused on **document summarization combined with interactive conversational querying** over large inputs while maintaining chat history. The scenario reflects enterprise workflows where users analyze lengthy documents and continue asking contextual questions without losing prior conversation context. Such capabilities are critical for applications like legal contract review, financial report analysis, research summarization, and enterprise knowledge-based exploration. The benchmark measures system behavior under large prompt sizes ranging from **16K to 65K tokens**, representing realistic long-document interactions. The goal is to understand how well an optimized inference deployment can support large-context reasoning while enabling responsive conversational experiences for document-centric AI applications.

Figure 3 illustrates the computational impact of **long context prefill processing** on inference performance. With a **16K-token input context**, the system maintains high output token throughput while scaling to a larger number of concurrent prompts, benefiting from efficient batching and parallel token generation during the decode phase. When the input context expands to **65K tokens**, throughput decreases and supported concurrency drops due to the significantly higher **prefill compute and memory bandwidth requirements**, as the model must process a much larger attention window before generation begins. These results highlight the fundamental trade-off in long-context LLM workloads: **larger prompt contexts increase reasoning capacity but introduce higher latency and reduce concurrent serving efficiency.**

Document summarization with query, interactive chat with history of larger input tokens of up to 16,384/65,536

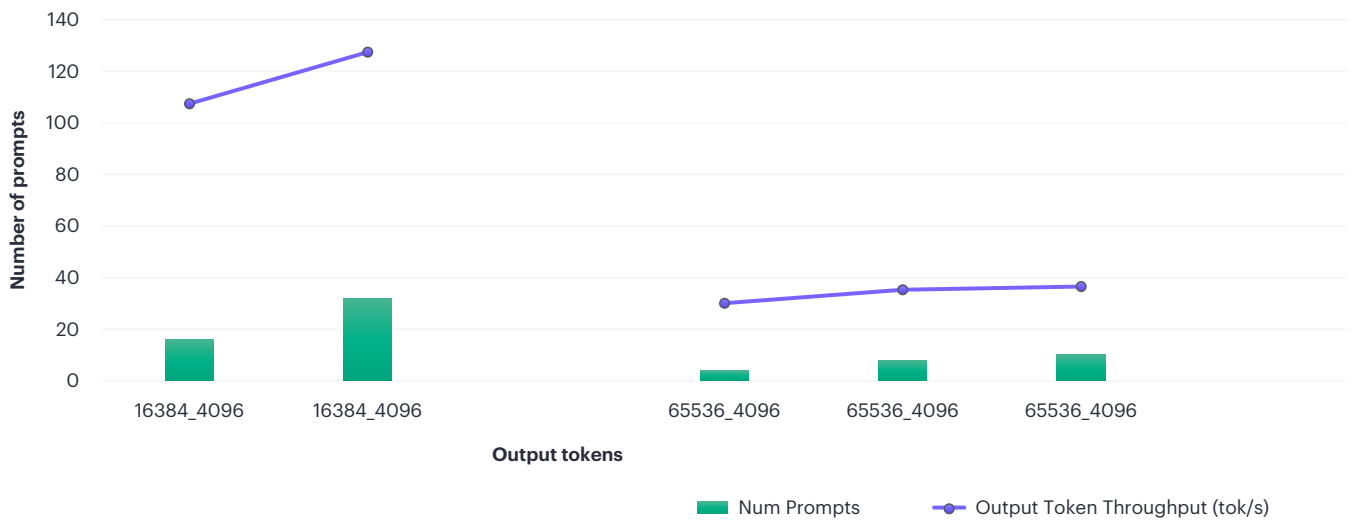


Figure 3. Document summarization

Results summary—Document summarization

The document summarization performance charts highlight the platform's ability to process large input documents while maintaining efficient output generation. The results show how throughput scales as input token sizes grow to tens of thousands, demonstrating stable tokens per second performance even with extended context lengths. At the same time, the charts illustrate controlled concurrency, emphasizing optimization for latency sensitive summarization tasks rather than maximum user count. This behavior reflects effective use of vLLM features such as chunked prefill, prefix caching, and NUMA aware tuning, enabling reliable, production ready document summarization on CPU based enterprise infrastructure.



Lowering the barrier to scalable enterprise AI

This solution overview demonstrates that vLLM-optimized AI inference on Intel Xeon 6 processors running Red Hat OpenShift provides a scalable and cost-effective solution for Enterprises. By enabling high-throughput, low-latency SLM inference on standard CPU infrastructure, this validated solution allows enterprises to deploy AI workloads using familiar platforms and operational models. The result is a practical, production-ready approach that simplifies deployment, reduces infrastructure complexity, and enables organizations to scale AI services with confidence, performance consistency, and operational efficiency.

Key takeaways

- CPU based SLM inference is a viable, cost-efficient alternative to GPUs for many enterprise workloads.
- Enterprise AI can be scaled pragmatically without over reliance on GPUs.
- AI inference can align with existing enterprise platforms and operating models.
- Predictable performance enables AI to move from pilots to production.
- Validated reference architectures reduce risk and accelerate decision-making.

Visit [HPE.com](https://www.hpe.com)

Learn more at

[HPE ProLiant Compute](#)

[Intel Xeon 6](#)

[Chat now](#)

© Copyright 2026 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Intel Xeon is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries. All third-party marks are property of their respective owners.

a00157626ENW

HEWLETT PACKARD ENTERPRISE

[hpe.com](https://www.hpe.com)

HPE

intel