OpenVINO Accelerated Agentic RAG: A Comparative Study Across Three Architectures

Planning / Implementation

Retrieval-Augmented Generation has evolved from a static retrieve-then-generate pipeline into more dynamic agentic systems. Instead of retrieving once and answering once, modern systems may decide whether more retrieval is necessary, choose among multiple tools, validate evidence quality, and revise intermediate answers before returning a final response. This shift changes how deployment efficiency should be evaluated. In agentic workflows, runtime depends not only on decoder throughput, but also on orchestration overhead, repeated model calls, retrieval frequency, and control-flow stability.

This paper investigates whether Intel CPU is an acceptable and practical target for such workflows. The key question is not whether CPU matches specialized accelerators on peak token throughput, but whether CPU-based serving can provide competitive end-to-end system performance for realistic agentic pipelines. To study this, the paper compares baseline CPU inference with OpenVINO-optimized Qwen3-8B models across three architectures: judge-based iterative retrieval, ReAct-style agentic RAG, and router-plus-judge multi-tool orchestration.

Experimental Scope

All comparisons were conducted under matched workflow conditions within each architecture, while varying the serving stack and model precision:

- **Agentic RAG Architectures:**
  - Architecture 1 uses judge-based iterative retrieval and provides the richest judged answer-quality metrics, including truthfulness, faithfulness, answer similarity, groundedness, and retrieval-quality judgments.
  - Architecture 2 uses a ReAct-style workflow and emphasizes latency, tool correctness, step count, and pass@k rather than groundedness-style quality judgments.
  - Architecture 3 evaluates multi-tool routing quality and latency across retrieve, web_search, and fred_releases style tool usage.
- **Model selection**
  - **Judge model**
    - Qwen3-14B FP16: used to evaluate retrieval quality and answer quality.
    - OpenVINO Qwen3-14B FP16: the OpenVINO-optimized version of the same judge model, used under the OpenVINO serving setup.
  - **Agent model**
    - Qwen3-8B FP16: baseline agent model for reasoning, tool use, and answer generation.
    - OpenVINO Qwen3-8B FP16: OpenVINO-optimized FP16 agent model.
    - OpenVINO Qwen3-8B INT8: quantized agent model for improved inference efficiency.
    - OpenVINO Qwen3-8B INT4: more aggressively quantized agent model for maximum latency reduction.
- **Dataset**
  - Synthetic data generated by ChatGPT for agentic RAG benchmarking
  - Data sources include:
    - content derived from 5 AI papers for retrieval evaluation
    - Tavily web search outputs
    - Federal macroeconomic calendar data
- **Question set**: **20 total questions**
  - 10 retrieval questions: designed to test document retrieval and answer generation over the AI-paper corpus.
  - 5 Tavily/web search questions: designed to test external web-search tool selection and usage.
  - 5 Federal calendar questions: designed to test the agent's ability to select and use the macroeconomic release tool.
  - Each question was executed **5 times**.
- **Tools**
  - **Retrieval tool**: Used for document-grounded question answering and evidence retrieval.
  - **Web search tool**: Used for questions requiring external or open-web information beyond the retrieval corpus.
  - **Fred_release tool**: Used for questions about economic release schedules and related structured calendar data.
- **Evaluation metrics**
  - **Answer faithfulness:** Measures whether the final answer is supported by the retrieved evidence or tool output.
  - **Answer truthfulness:** Measures whether the answer is factually correct with respect to the reference or source evidence.
  - **Pass@k**: Measures whether at least one of the runs or candidate outputs within the evaluation setting is correct.

- **Tool Correct Rate:** Measures how often the agent selects the correct tool for a given question, such as retrieval, web search, or fred_release.
- **find_missing Steps**: Counts how many times the workflow triggers an additional retrieval-repair step to look for missing evidence before answering.
- **Avg Rounds**: Measures the average number of retrieval/decision cycles used per question, indicating how often the agent needs extra iterations before reaching a final answer.
- **Avg Latency / Question (s)**: Measures the average end-to-end response time per question, including reasoning, tool calls, retrieval, validation, and answer generation.

Because the metric sets are not identical across architectures, the cross-architecture discussion focuses on five common themes: latency, workflow efficiency, retrieval or step behavior, task or routing correctness, and the tradeoff between speed and answer-control quality.

Architecture 1: Judge-Based Iterative Retrieval

Architecture 1 uses an LLM judge to decide whether the currently retrieved context is sufficient or whether another retrieval pass is needed before answering. This setting is the clearest testbed for both latency and answer-quality tradeoffs because it records per-question timing, number of rounds, extra retrieval loops, and judged answer metrics.

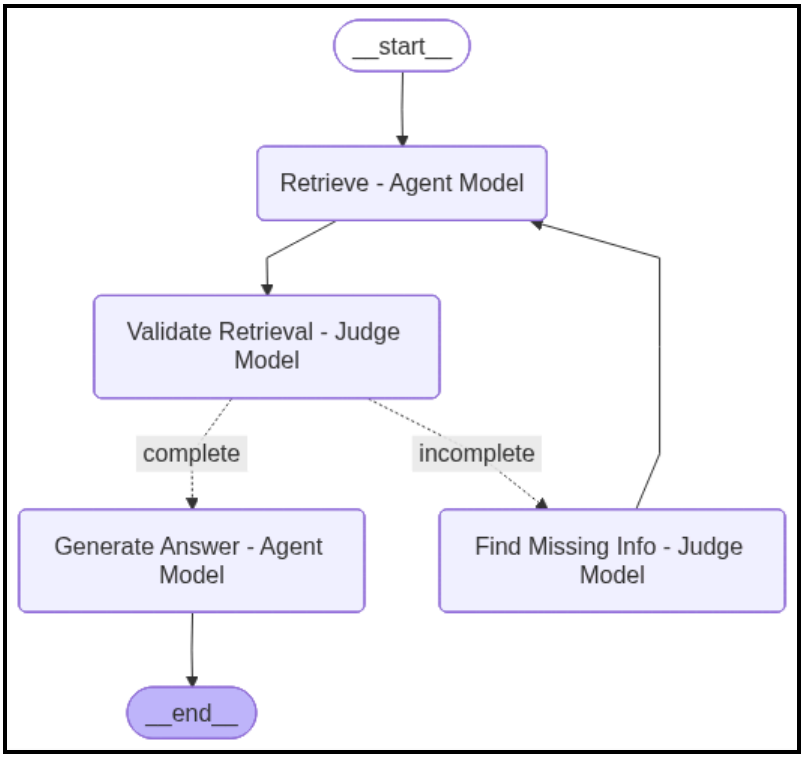

Figure 1. Architecture 1

The following table summarizes the performance of the judge-based iterative retrieval architecture across the baseline and OpenVINO Qwen3-8B variants, highlighting the tradeoffs between latency, retrieval efficiency, and answer quality.

Table 1. Architecture 1 summary

Model	Avg Latency / Question	Avg Rounds	find_missing Steps	Truthfulness	Faithfulness
Baseline FP16	134.3 seconds	1.5	5	3.7	4.2
OpenVINO FP16	70.7 seconds	1.4	4	3.5	4.3
OpenVINO INT8	55.0 seconds	1.2	2	3.3	3.7
OpenVINO INT4	38.1 seconds	1.2	2	3.5	4.1

OpenVINO substantially reduced latency in this architecture, with lower precision providing further gains: INT4 cut average latency by 71.6%, from 134.3 s to 38.1 s per question, while both INT8 and INT4 reduced find_missing steps and two-round questions from 5 to 2. In quality, OpenVINO FP16 offered the best balance, achieving the highest groundedness (0.853) and faithfulness (4.3) with much lower latency than the baseline, whereas INT4 delivered the best efficiency with only modest quality tradeoffs and INT8 was a weaker middle-ground option.

The central conclusion from Architecture 1 is that OpenVINO improves not only model runtime, but also

workflow efficiency. The optimized models require fewer recovery loops, which is especially important in agentic pipelines where repeated retrieval compounds latency quickly.

Architecture 2: ReAct-Style Agentic RAG

Architecture 2 follows a ReAct-style workflow in which the model alternates between thought, action, and observation. The primary metrics here are run-level latency, tool correctness, step count, and pass@k. Detailed step logs were also reviewed to understand output verbosity and reasoning style.

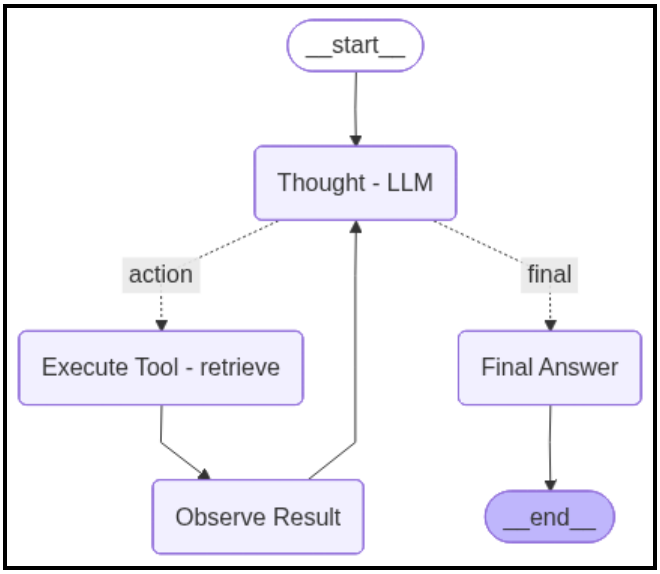

Figure 2. Architecture 2

The following table summarizes the performance of the ReAct-style agentic RAG architecture across the baseline and OpenVINO Qwen3-8B variants, focusing on latency, step efficiency, and tool-use correctness.

Table 2. Architecture 2 summary

Model	Avg Latency / Question (s)	Avg Intermediate Tokens / Run	Pass@k	Avg Steps	Avg Tool Calls	Truthfulness	Faithfulness
Baseline FP16	123.4 seconds	1142.8	1	2.12	1.12	3.99	3.89
OpenVINO FP16	98.1 seconds	1134.3	1	2.09	1.09	4.06	3.92
OpenVINO INT8	67.4 seconds	1124.5	1	2.07	1.07	4.04	3.79
OpenVINO INT4	50.9 seconds	1106.8	1	2.04	1.03	4.14	3.91

The latency trend is clear again: INT4 > INT8 > FP16 > baseline. OpenVINO FP16 reduces average latency by 20.5%, OpenVINO INT8 by 45.4%, and OpenVINO INT4 by 58.8% relative to the baseline.

For ReAct Agent system, Intermediate token consumption changed only slightly, which suggests that OpenVINO mainly improves generation efficiency rather than shortening the reasoning process itself; INT4 therefore combines the lowest latency, the fewest steps, and the lowest intermediate token usage, while FP16 remains slightly stronger on faithfulness. Intermediate token consumption changed only slightly, which suggests that OpenVINO mainly improves generation efficiency rather than shortening the reasoning process itself; INT4 therefore combines the lowest latency, the fewest steps, and the lowest intermediate

token usage, while FP16 remains slightly stronger on faithfulness.

Architecture 3: Router + Judge Multi-Tool Orchestration

The updated Architecture 3 experiments are the strongest evidence for a mature multi-tool agent pipeline. Unlike the earlier ReAct-style outputs, these runs show that the agent is correctly using multiple tools rather than collapsing almost entirely to retrieval.

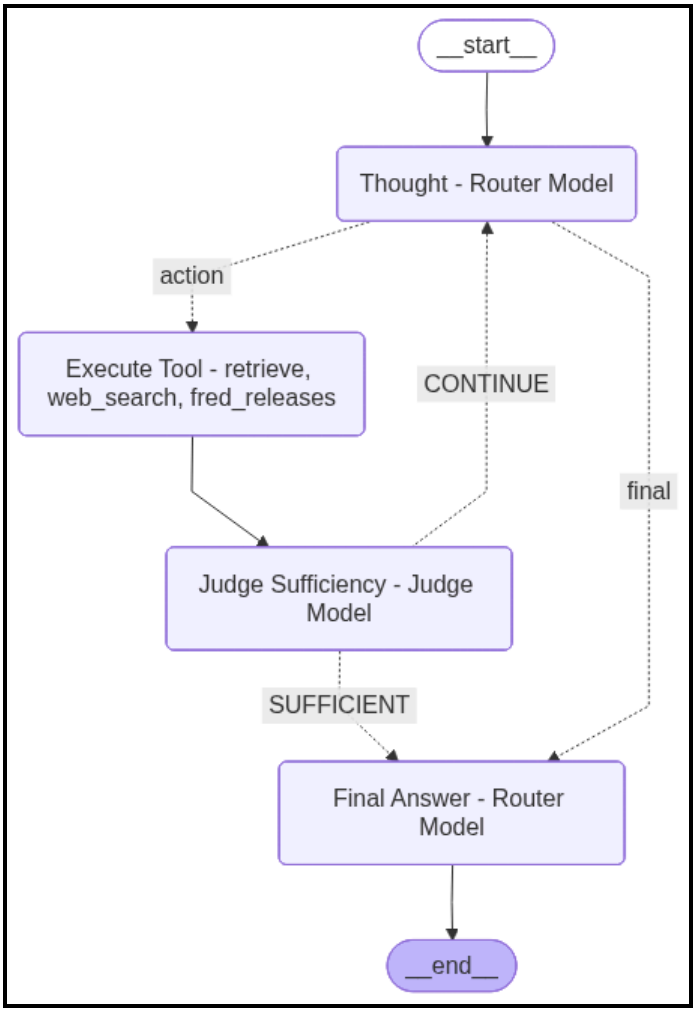

Figure 3. Architecture 3

The following table summarizes the performance of the router-plus-judge multi-tool architecture across the baseline and OpenVINO Qwen3-8B variants, highlighting the tradeoff between latency, tool-routing accuracy, and judged answer quality.

Table 3. Architecture 3 summary

Model	Avg Latency / Question (s)	Tool Correct Rate	Pass@k	Retrieve Correct	FRED Correct	Web Search Correct	Truthfulness	Faithfulness
Baseline FP16	91.4 seconds	0.95	1.00	0.90	1.00	1.00	4.64	4.43
OpenVINO FP16	72.0 seconds	0.98	1.00	0.96	1.00	1.00	4.72	4.58

Model	Avg Latency / Question (s)	Tool Correct Rate	Pass@k	Retrieve Correct	FRED Correct	Web Search Correct	Truthfulness	Faithfulness
OpenVINO INT8	49.8 seconds	0.93	0.95	0.88	1.00	0.96	4.37	4.23
OpenVINO INT4	40.0 seconds	0.95	1.00	0.94	1.00	0.92	4.60	4.39

All OpenVINO variants improved latency in Architecture 3, with INT4 delivering the fastest performance and FP16 achieving the strongest overall routing and judged quality. Overall, the results show a clear tradeoff between maximum efficiency and best control quality, with OpenVINO INT4 favoring speed and OpenVINO FP16 favoring quality.

Cross-Architecture Discussion

This section examines how OpenVINO optimizations impact performance, efficiency, and quality across different agent architectures.

- OpenVINO consistently improves CPU inference efficiency
- INT4 is the strongest efficiency setting
- FP16 is the safest quality-preserving OpenVINO option
- Architecture matters as much as precision

OpenVINO consistently improves CPU inference efficiency

Across all three architectures, OpenVINO reduces end-to-end latency. This is the most stable result in the study. In Architecture 1, INT4 reduces latency by 71.6%; in Architecture 2, INT4 reduces latency by 58.8%; and in Architecture 3, INT4 reduces latency by 56.3%. The effect is not isolated to one workflow. It appears in iterative retrieval, ReAct, and multi-tool routing.

Although OpenVINO consistently improved latency, its broader effect depended on the agent design.

- In **Architecture 1**, OpenVINO improved both runtime and workflow efficiency. The optimized variants not only answered faster, but also required fewer additional retrieval-repair cycles. This is important because iterative retrieval amplifies latency when the system repeatedly invokes the judge and retrieval steps. In this setting, OpenVINO therefore improved the full workflow, not just raw inference.
- In **Architecture 2**, OpenVINO again improved runtime substantially, but quality gains were limited. The judged metrics stayed relatively close across variants, and the step-level traces showed that all models generated similarly long, verbose reasoning outputs. This suggests that the main bottleneck in this ReAct setup was reasoning and tool-selection behavior, not model speed alone.
- In **Architecture 3**, the routing policy was already strong, and the agent used all three tools effectively. As a result, OpenVINO translated more directly into practical system gains: latency dropped sharply while overall routing quality stayed high. This architecture therefore showed the clearest production-style tradeoff between best quality and best efficiency.

INT4 is the strongest efficiency setting

Across all three architectures, OpenVINO INT4 delivered the lowest latency:

- in Architecture 1, it produced the largest reduction in average latency and fewer extra retrieval loops,
- in Architecture 2, it was the fastest despite similar intermediate reasoning-token consumption,
- in Architecture 3, it again had the best runtime while maintaining strong routing quality.

This makes INT4 the strongest choice when the primary objective is low latency, lower serving cost, or higher CPU throughput.

FP16 is the safest quality-preserving OpenVINO option

Although INT4 was fastest, OpenVINO FP16 most consistently delivered the strongest quality profile:

- in **Architecture 1**, it achieved the best groundedness and faithfulness while still substantially reducing latency,
- in **Architecture 2**, it remained competitive on judged metrics even though the architecture itself limited quality differentiation,
- in **Architecture 3**, it achieved the best overall tool-correct rate, retrieval precision and recall, truthfulness, and faithfulness.

This makes FP16 the safest option when the goal is to retain the strongest control quality, answer quality, or routing stability while still benefiting from OpenVINO acceleration.

Architecture matters as much as precision

Taken together, the results support a system-level claim rather than a narrow model-speed claim. Intel CPU paired with OpenVINO can be a practical deployment target for Agentic RAG because the relevant metric is not only token throughput, but also:

- How many reasoning or retrieval rounds are triggered
- How reliably the agent chooses the correct tool
- How much intermediate reasoning is generated
- How well latency improvements preserve answer quality

This is why the benefit of OpenVINO was strongest in architectures where the workflow itself was already well structured. When the control policy was strong, as in Architecture 3, OpenVINO translated cleanly into practical deployment gains. When the control policy was weaker, as in Architecture 2, OpenVINO still improved latency, but it could not by itself correct architectural weaknesses.

Practical Deployment Guidance

The following table presents a cross-architecture comparison of the three agentic RAG designs, summarizing how latency, workflow efficiency, and quality tradeoffs vary across the baseline and OpenVINO model configurations.

Table 4. Practical deployment guidance

Priority	Recommended Variant	Reason
Strongest answer/control quality	OpenVINO FP16	Best faithfulness and groundedness in Architecture 1; Best routing correctness in Architecture 3.
Lowest latency on Intel CPU	OpenVINO INT4	Fastest across all three architectures, with large end-to-end latency reductions.

Conclusion

This study shows that Intel CPU is a credible inference platform for Agentic RAG, especially when paired with OpenVINO. Across three distinct architectures, OpenVINO consistently reduced latency, and lower precision further improved efficiency. The strongest speed result came from OpenVINO INT4, which was the fastest option in every architecture. However, the best overall deployment choice depends on workflow goals. OpenVINO FP16 most consistently preserved answer quality and routing quality, while INT4 maximized throughput and reduced end-to-end delay.

The three architectures also highlight an important systems insight. In agentic RAG, performance is shaped not only by model precision, but also by the structure of the workflow itself. Judge-based iterative retrieval benefits from fewer recovery loops, ReAct remains sensitive to tool-selection policy, and router-plus-judge systems can translate inference optimization directly into strong multi-tool performance when the routing policy is already well designed. Overall, the results support a practical claim: for CPU-based Agentic RAG, OpenVINO makes Intel deployment substantially more competitive, and INT4 or FP16 can be selected depending on whether the priority is efficiency or control quality.

Server configuration

The experiments were conducted on a CPU-based server platform with the hardware and software configuration shown below.

In this paper, we decided to use a server with 5th Gen Intel Xeon processors. This will allow us to do a gen-to-gen comparison in a future paper, where we can examine the performance gains when using a server with Intel Xeon 6 processors.

Note: The SR680a V3 includes NVIDIA GPUs, however for the purposes of this paper, the GPUs were not used. The paper only examined the effectiveness of CPU-based inference for Agentic RAG workflows, using OpenVINO on Intel CPUs.

Table 5. Hardware configuration

Component	**Details**
System	Lenovo ThinkSystem SR680a V3
CPU	2× Intel Xeon Platinum 8568Y+
CPU topology	48 cores/socket, 2 threads/core, 192 total threads
CPU frequency	Up to 4.0 GHz
Memory	2.0 TiB DDR

Table 6. System software configuration

Software	**Version**
OS	Ubuntu 24.04.3 LTS (Noble Numbat)
Kernel	6.8.0-94-generic
Python	3.12

Table 7. Key Python package versions

Package	**Version**
torch	2.9.1+cu130
transformers	4.57.6
langchain	1.2.10
langgraph	1.0.8
chromadb	1.4.1
sentence-transformers	5.2.2
openvino	2026.1.0
optimum-intel	1.27.0
pandas	2.3.3
opencv-python	4.13.0.92
tavily-python	0.7.23
unstructured	0.21.5
huggingface_hub	0.36.2

Package	Version
onnxruntime	1.23.2

References

See the following documents for more information:

- Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401
- Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. https://arxiv.org/abs/2210.03629
- Asai et al. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. https://arxiv.org/abs/2310.11511
- Yan et al. Corrective Retrieval-Augmented Generation. https://arxiv.org/abs/2401.15884
- Intel Advanced Matrix Extensions overview. https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/advanced-matrix-extensions/overview.html
- OpenVINO documentation: Running inference. https://docs.openvino.ai/2024/openvino-workflow/running-inference.html
- An Engineering Guide to Evaluating AI Agents. https://levelup.gitconnected.com/an-engineers-guide-to-evaluating-ai-agents-62716f8553a8
- Building Agentic Adaptive RAG with LangGraph for Production https://ai.plainenglish.io/building-agentic-rag-with-langgraph-mastering-adaptive-rag-for-production-c2c4578c836a

Model Cards

9. Qwen/Qwen3-8B — baseline 8B agent model.
10. Qwen/Qwen3-14B — baseline 14B judge model.
11. OpenVINO/Qwen3-8B-fp16-ov — OpenVINO FP16 8B agent model.
12. OpenVINO/Qwen3-8B-int8-ov — OpenVINO INT8 8B agent model.
13. OpenVINO/Qwen3-8B-int4-ov — OpenVINO INT4 8B agent model.
14. OpenVINO/Qwen3-14B-fp16-ov — OpenVINO FP16 14B judge model.

Authors

Kelvin He is an AI Data Scientist at Lenovo. He is a seasoned AI and data science professional specializing in building machine learning frameworks and AI-driven solutions. Kelvin is experienced in leading end-to-end model development, with a focus on turning business challenges into data-driven strategies. He is passionate about AI benchmarks, optimization techniques, and LLM applications, enabling businesses to make informed technology decisions.

David Ellison is the Chief Data Scientist for Lenovo ISG. Through Lenovo's US and European AI Discover Centers, he leads a team that uses cutting-edge AI techniques to deliver solutions for external customers while internally supporting the overall AI strategy for the Worldwide Infrastructure Solutions Group. Before joining Lenovo, he ran an international scientific analysis and equipment company and worked as a Data Scientist for the US Postal Service. Previous to that, he received a PhD in Biomedical Engineering from Johns Hopkins University. He has numerous publications in top tier journals including two in the Proceedings of the National Academy of the Sciences.

Related product families

Product families related to this document are the following:

- Artificial Intelligence
- Intel Alliance

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

© Copyright Lenovo 2026. All rights reserved.

This document, LP2427, was created or updated on April 23, 2026.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP2427
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP2427.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®
ThinkSystem®

The following terms are trademarks of other companies:

Intel®, the Intel logo, OpenVINO®, and Xeon® are trademarks of Intel Corporation or its subsidiaries.

NVIDIA® is a trademark of NVIDIA Corporation.

Other company, product, or service names may be trademarks or service marks of others.