End-to-End Latency Benchmarking for Three Multimodal RAG Architectures

Planning / Implementation

Retrieval-Augmented Generation (RAG) is commonly described as combining parametric generation with non-parametric retrieval over a vector index. This Part 3 in our series of papers on RAG is a latency-first study of three multimodal RAG system designs.

Other papers in this series:

- Part 1: Standard Retrieval Augmented Generation on Intel: From Search to Answers
- Part 2: Accelerating Multimodal RAG with Intel Xeon and OpenVINO: When Vision Meets Language

These multimodal RAG system designs differ in three ways:

- Whether images are summarized into text
- Whether raw images are used at answer time
- Whether retrieval uses text embeddings or multimodal embeddings

We define an end-to-end (E2E) latency methodology that separates index-time latency (document ingestion + indexing) from query-time latency (retrieve + generate). Before the E2E comparison, we quantify two key contributors to latency variance:

- Multimodal embedding dimension vs embedding throughput
- Multimodal generation responsiveness vs image resolution and visual tokenization, including a controlled complexity study using synthetic patterns.

Benchmark areas we will cover:

- Embedding-size sensitivity: how vector dimension (and model choice) changes retrieval and overall latency.
- Latency breakdowns: ingest → summarize (if any) → embed → index → retrieve → generate.
- E2E latency under three architecture Architectures.

Experimental Design

This section outlines the evaluation framework used to compare the three multimodal RAG architectures under consistent latency and quality criteria. The design ensures controlled measurement of retrieval effectiveness, generation quality, and system-level performance trade-offs across all configurations.

Metrics

We measure index-time and query-time separately, plus streaming responsiveness:

Index-time latency:

- PDF parsing / content extraction time
- Image summarization time (Architectures 1–2 only)
- Embedding generation time (text and/or image)
- Vector store creation + indexing time
- Total indexing time

Query-time latency:

- Retrieval latency (ANN search)

Streaming breakdown:

- TTFT (time-to-first-token): responsiveness
- TPOT (time per output token): throughput proxy

Embedding-Size Benchmark Matrix

For each Architecture, we benchmark on different size of embeddings. The clean way to do this is to hold everything constant and vary only the embedding model dimension.

Multimodal embedding sizes: Small/Medium/Large depending on the chosen multimodal embedder (often 512-1024 + effective dims in CLIP-like families).

- 3 Multimodal embedding model sizes (CLIP-style multimodal representations):
  - Small: 512-d
  - Medium: 768-d
  - Large: 1024-d
- 8 square resolutions: 224, 336, 384, 448, 512, 672, 896, 1024
  - Each resolution repeated 10 times → 80 samples/model
- 4 image complexity patterns: Gradient, Solid, Checkerboard, Noise

Architectures Under Test

To systematically evaluate design trade-offs in multimodal RAG, we implemented three representative architectural patterns that differ in retrieval strategy, modality handling, and generation workflow. The following describes the framework of three architectures.

- Architecture 1: “Summarize → Text Retrieval → Text-only Answer”
  1. Use a multimodal LLM (VLM) to produce text summaries from images.
  2. Embed all content as text (image summaries + tables + text).
  3. Pass retrieved text to a text LLM for answer generation.
- Architecture 2: “Summarize → Text Retrieval + Image Refs → Multimodal Answer”
  1. Use a VLM to create text summaries from images.
  2. Embed summaries and retrieve them, preserving references to original images.
  3. Pass raw image(s) + retrieved text chunk(s) to a VLM for answer generation.
- Architecture 3: “Multimodal Embeddings → Multimodal Retrieval → Multimodal Answer”
  1. Use a multimodal embedding model (e.g., CLIP-style) to embed images directly.
  2. Retrieve relevant images (and associated text chunks) via similarity search over multimodal vectors.
  3. Pass the retrieved raw image(s) and text chunk(s) to a VLM for answer generation.

Vector Database and search infrastructure

Chroma is a common similarity search library used for fast ANN over dense vectors. It can be used for Semantic Search, RAG, Recommendation System and Image and Multimedia Retrieval.

Results

This section presents the benchmark results in a bottom-up way. We first isolate two components that frequently dominate multimodal RAG latency:

- Multimodal embedding speed as a function of embedding dimension and input resolution
- Multimodal generation latency as a function of image resolution (visual tokens) and image complexity

We then connect these component measurements to the end-to-end indexing and query latency observed across the three RAG architectures.

Embedding Results

To isolate the cost of multimodal retrieval in Architecture 3, we benchmark CLIP-style image embedding across three model sizes (512, 768, and 1024 dimensions). We report per-image embedding latency and throughput over multiple input resolutions to quantify how embedding dimension and image size affect ingestion and query-time embedding overhead.

Table 1. Describe the Mean embedding time per image and Throughput

Model size	Dimension	Mean time (s)	p95 (s)	Throughput (imgs/s)
0.15B	512	0.0385 sec	0.0509 sec	~26.0 images/sec
0.4B	768	0.4101 sec	0.4294 sec	~2.44 images/sec
1B	1024	0.5947 sec	0.6628 sec	~1.68 images/sec

Key takeaway: Moving from 512 → 768 → 1024 dimensions in these runs produces a large step-up in embedding time (around 10× from 512 to 768, then around 1.45× from 768 to 1024).

Embedding Speed vs Resolution Input

Comparing mean time at the smallest vs largest resolution (224^2 vs 1024^2):

- 512-d: ~0.0302s → ~0.0508s (~1.68×)
- 768-d: ~0.3936s → ~0.4240s (~1.08×)
- 1024-d: ~0.6079s → ~0.6187s (~1.02×, effectively flat within noise)

In this setup, we found that resolution has **minor impact** on medium/large embedding models and **moderate impact** on small embedding models. Dimension remains the dominant factor.

Multimodal Generation Speed vs Pixel Size and Pattern Complexity

This section characterizes the latency behavior when a VLM must process raw images (relevant to Architecture 2 and Architecture 3 answer generation).

Resolution scaling on Generation Latency

To understand the query-time cost of using raw images (Architectures 2 and 3), we measure how the vision-language model's latency scales with input resolution. We use a simple gradient image to keep content constant while increasing pixel count, and report both time-to-first-token (TTFT) and time per output token (TPOT). This isolates the image encoding and prefill overhead from the downstream text decoding rate.

Table 2. Multimodal Generation Speed on Different Resolutions

Resolution	Pixels	Visual tokens	Avg TTFT (s)	Avg TPOT (s/token)
224^2	50,176	63	0.1935	0.0590
512^2	262,144	323	0.4511	0.0606
896^2	802,816	1023	1.1691	0.0599
1024^2	1,048,576	1368	1.8004	0.0595

Key takeaways: TTFT is almost linear in visual tokens. Fitting a simple linear model to 8 our resolution points shows TTFT is well-approximated by:

$\text{TTFT} \approx 0.102 + 0.00116 \times (\text{visual\_tokens})$

For this VLM configuration, the “cost to start responding” is driven far more by how many visual tokens the encoder produces than by the downstream text decoding rate (TPOT).

Pattern complexity comparison at fixed resolution

We tested solid, gradient, checkerboard, and noise at the same resolution. The visual token count is identical across patterns (so the VLM’s image-encoding workload is basically the same), and the measured latency differences are tiny.

With 512×512 images

- TTFT range across patterns: 0.4442s → 0.4528s ($\Delta \approx 0.009$s, ~1.9% of mean)
- TPOT range across patterns: 0.05865 → 0.05992 s/token ($\Delta \approx 0.0013$, ~2.1% of mean)

With 1024×1024 images

- TTFT range across patterns: 1.7800s → 1.7948s ($\Delta \approx 0.015$s, ~0.83% of mean)
- TPOT range across patterns: 0.05970 → 0.06051 s/token ($\Delta \approx 0.0008$, ~1.35% of mean)

Key takeaways: Across synthetic complexity patterns (solid/gradient/checkerboard/noise), TTFT and TPOT vary by less than ~2% at fixed resolution, while increasing resolution from 512^2 to 1024^2 increases TTFT by ~4× with negligible TPOT change. This indicates image resolution (not pixel-level complexity) is the primary determinant of multimodal generation responsiveness in this configuration.

End to End Latency Summary

We tested the End-to-End comparison on a popular paper, Attention is All You Need, across 3 architectures.

- Vision Language Model: OpenVINO/Qwen2-VL-7B-Instruct-fp16-ov
- Large Language Model: meta-llama/Llama-3.1-8B-Instruct converted to OpenVINO IR
- Multimodal Embedding Model : openai/clip-vit-base-patch32
- Text Embedding Model: sentence-transformers/all-MiniLM-L6-v2
- Vector Database: Redis

Indexing Total Time

This subsection reports the one-time indexing cost to prepare the corpus for retrieval. We decompose total indexing time into document loading, image-to-text summarization (Architectures 1-2 only), and ingest/index construction. Index Total Time (s) = Document Loading + Image-to-Text Summarization + Ingest.

Table 3. Indexing among three architecture (one time cost for corpus)

Architecture	Document Loading (s)	Image→Text Summarization (s)	Ingestion Row per second	Index Total (s)
architecture_1	37.479	127.218	254	165.024
architecture_2	38.144	127.890	352	166.831
architecture_3	36.924	0.000	67.7	37.776

Key Takeaway: architecture 3 indexes **~4.4× faster** than architecture 1 and 2 because it eliminates **image summarization**, which dominates index time for architecture 1 and 2.

Generation Phase Time

Next, we analyze the query-time path by separating retrieval latency from generation latency. We report retrieval time, time-to-first-token (TTFT), and time-per-output-token (TPOT) to capture both responsiveness and decoding speed.

Estimated Generation (s) = TTFT + TPOT × Answer Length

Table 4. Retrieval + Generation metrics (per query averages)

Architecture	Retrieval (s)	TTFT (s)	TPOT (s/token)
architecture_1	0.02	0.548	0.054
architecture_2	0.02	0.997	0.067
architecture_3	0.13	1.056	0.067

Key Takeaway: architecture 2 has the **fastest retrieval**, while architecture 1 has the **best TTFT/TPOT**.

Retrieval and Generation Quality

To assess the qualitative performance of the three multimodal RAG architectures, we conducted an automated evaluation using a preloaded VLM Qwen2-VL-7B as the judge model. The evaluation was performed with over 50 representative questions, resulting in averaged scores across 50 total evaluations per architecture. The judge model evaluated the retrieval metrics: Precision@5, MRR, nDCG@5, and responses metrics: **answer relevance, groundedness, and context coverage**, providing a structured comparison of reasoning quality and evidence alignment across architectures.

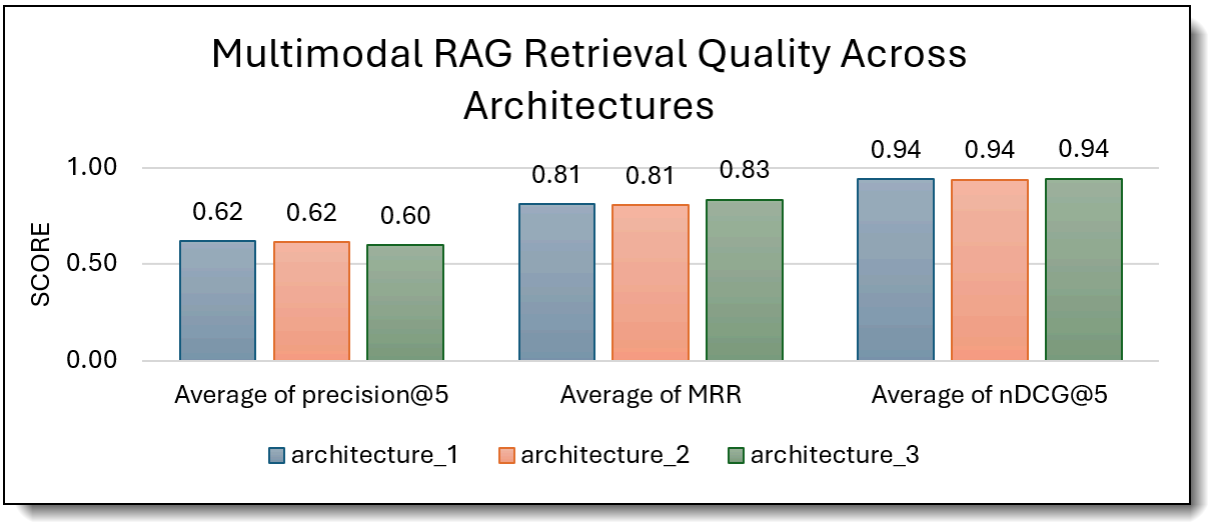

Figure 1. Multimodal RAG Retrieval Quality Chart

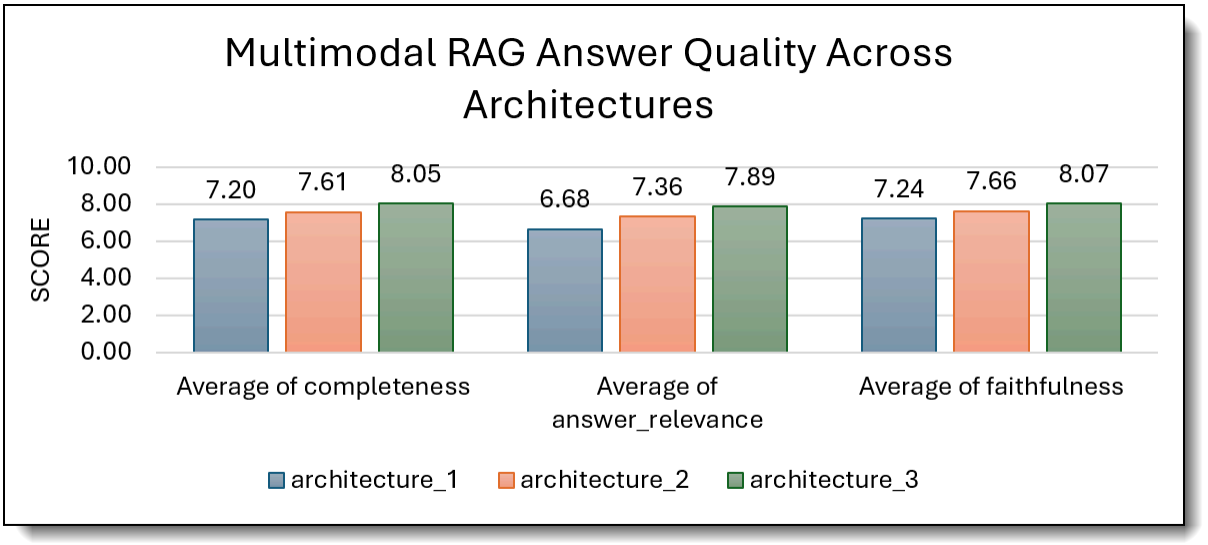

Figure 2. Multimodal RAG Answer Quality Chart

Key Takeaways

- **Architecture 3 delivers the strongest answer quality**, indicating that multimodal embedding–based retrieval can improve completeness and faithfulness when paired with multimodal generation.
- **Architecture 2 provides the best retrieval precision**, suggesting that text-mediated retrieval improves top-k relevance.
- **Architecture 1 is the most stable baseline**, but does not outperform the hybrid or fully multimodal designs.

Overall, hybrid or multimodal retrieval strategies outperform text-only pipelines in both ranking effectiveness and answer quality.

Conclusion and Implementation Recommendation

This section summarizes the practical tradeoffs observed in the benchmarks and translates them into implementation guidance. We focus on how indexing cost, retrieval latency, and multimodal generation overhead interact to determine overall end-to-end latency and user experience.

Architecture Recap and Conclusion

We recap the three architectures and highlight where each one fits best. The goal is not to declare a single universal winner, but to make the latency and quality trade-offs explicit so teams can choose the right design for their workload and constraints.

The following table summarizes the structural differences among the three multimodal RAG architectures, highlighting variations in image processing, embedding strategy, retrieval type, and generation workflow.

Table 5. Multimodal RAG Architecture summary table

Aspect	**Architecture 1**	**Architecture 2**	**Architecture 3**
Image processing	VLM summarization	VLM summarization	CLIP embedding
Indexing speed	Slow	Slow	Fast
Image embedding	Text (summary)	Text (summary)	Visual (CLIP)
Text embedding	sentence-transformers	sentence-transformers	NA
Answer generation	LLM (text only)	VLM (multimodal)	VLM (multimodal)
Raw image in answer	No	Yes	Yes
Retrieval type	Semantic (summary)	Semantic (summary)	Visual similarity
Models needed	VLM + LLM	VLM	VLM + CLIP

The following compares the three architectures across key performance dimensions, including indexing efficiency, response latency (TTFT), answer quality, and retrieval behavior.

Table 6. Multimodal RAG performance among architectures

Metric	**Arch 1**	**Arch 2**	**Arch 3**
Indexing time	Slow	Slow	Fast (10-100x)
Answer TTFT	Fast	Slow	Slow
Answer quality	Good	Good	Better
Retrieval accuracy	Semantic	Semantic	Visual

The following summary highlights the fundamental trade-offs between indexing cost, retrieval strategy, and answer-time performance across the three multimodal RAG architectures.

Index-Time vs Query-Time Dominance:

- **Architectures 1–2**: dominated by **summarization latency** (VLM compute). Unless summarization is cached/reused across many queries, this pushes total time up.
- **Architecture 3**: dominated by extraction + embedding + index build; avoids summarization entirely.

Retrieval Quality vs Latency:

- **Architecture 1** can be fast in streaming responsiveness and cheaper at answer time, but risks losing fine-grained visual details because images are reduced to summaries.
- **Architecture 2** preserves visual fidelity at answer time while keeping retrieval in a text space; it is a strong balanced design for charts/figures.
- **Architecture 3** bets on multimodal embedding quality to retrieve the right images without summarization; this can be excellent for “find the figure/screenshot” tasks but can degrade if the embedding model struggles with domain visuals.

Implementation Recommendation

Based on the measurements above, we provide a simple decision guide that maps common enterprise use cases (fast document onboarding, interactive Q&A, visually grounded answers) to the architecture that best matches the latency and quality requirements.

Table 7. Business recommendation when implementing Multimodal RAG

Business model / product goal	**Best architecture**	**Estimated Ingestion Time /MB file**	**Estimated Generated Tokens**	**Generative Time (s)**	**Number of Concurrent User (SLA-bound)***
Interactive chat copilot and cost sensitive (text-first UX)	Arch 1	~78.21 s/MB	250 tokens	~36	~44 users (LLM)
Balanced enterprise multimodal assistant	Arch 2	~79.07 s/MB	300 tokens	~43	~10 users (VLM)
Premium multimodal analyst copilot (accuracy-first)	Arch 3	~17.90 s/MB	250 tokens	~37.5	~10 users (VLM)
Figure/image-centric search	Arch 3	~17.90 s/MB	150 tokens	~24.5	~10 users (VLM)

* Concurrent users measured under SLA constraints: TTFT ≤ 10s and TPOT ≤ 130ms on Intel Xeon 6787P with OpenVINO.

The results demonstrate that architectural choice directly influences scalability, latency, and deployment cost:

- Architecture 1 maximizes concurrency and is best suited for high-scale, text-dominant enterprise deployments.
- Architecture 2 offers a balanced approach, combining improved retrieval precision with strong multimodal reasoning for general enterprise assistants.
- Architecture 3 delivers the highest answer completeness and faithfulness, making it ideal for premium, reasoning-intensive multimodal use cases, albeit with lower concurrency.

These findings provide clear guidance for aligning multimodal RAG system design with product goals, infrastructure budgets, and SLA commitments.

Additional papers on RAG

This paper is Part 3 of 4 of a series of papers on Retrieval Augmented Generation.

The papers in the series are as follows:

- **Part 1**: Standard Retrieval-Augmented Generation: implement and benchmark a text-only RAG pipeline on Intel platforms, evaluate embedding and generation model combinations, and quantify retrieval latency, TTFT, TPOT, and throughput to establish CPU-first performance baselines from search to answer generation. Read the paper here: Standard Retrieval Augmented Generation on Intel: From Search to Answers
- **Part 2**: Multimodal RAG: extend the standard pipeline to support cross-modal retrieval and grounding (e.g., text with images/tables/code), compare hybrid indexes and multimodal encoders/LLMs, and quantify modality-aware latency and throughput alongside retrieval quality. Read the paper here: Accelerating Multimodal RAG with Intel Xeon and OpenVINO: When Vision Meets Language
- **Part 3**: This paper
- **Part 4**: Agentic Long-Context RAG: introduce planning and tool use with validation loops (self-check, re-query, re-rank), add long-context memory, and evaluate end-to-end task success, stability across multi-step trajectories, and cost–latency trade-offs. Coming soon.

System Configuration

The configuration of the server used in our testing is as follows:

- CPU: Intel Xeon 6787P processor, 86 cores / 172 threads @ 3.8 GHz
- GPU: None
- Memory: 16x 64 GB DDR5 @ 6400 GHz
- Cache: 336 MB
- OS: Ubuntu 22.04.5 LTS (Linux kernel 6.8.0-59-generic)
- Python: 3.12.3
- OpenVINO: 2025.4.0-20398
- Transformer: 4.45.0

Resources

This paper is Supplemental to the Part 2 on Retrieval Augmented Generation Series. Please check out detailed throughput and number of concurrent users test in Part 2.

Part 1: Standard Retrieval Augmented Generation on Intel: From Search to Answers
https://lenovopress.lenovo.com/lp2322-standard-retrieval-augmented-generation-on-intel-from-search-to-answers

Part 2: Accelerating Multimodal RAG with Intel Xeon and OpenVINO: When Vision Meets Language
https://lenovopress.lenovo.com/lp2350-accelerating-multimodal-rag-with-intel-xeon-and-openvino

For more information, see these resources:

- Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP.” arXiv:2005.11401
  https://arxiv.org/abs/2005.11401
- MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text
  https://arxiv.org/abs/2210.02928
- OpenVINO Toolkit:
  https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html
- OpenVINO GenAI Model:
  https://openvinotoolkit.github.io/openvino.genai/docs/use-cases/image-processing/
- Huggingface OpenVINO Toolkit:
  https://huggingface.co/OpenVINO/collections
- Intel Advanced Matrix Extensions (Intel AMX):
  https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/what-is-intel-amx.html
- Intel Xeon 6787P (Intel Xeon 6 / Granite Rapids) product page:
  https://www.intel.com/content/www/us/en/products/sku/241844/intel-xeon-6787p-processor-336m-cache-2-00-ghz/specifications.html

Model cards:

- OpenVINO/Qwen2-VL-7B-Instruct-fp16-ov
  https://huggingface.co/OpenVINO/Qwen2-VL-7B-Instruct-fp16-ov
- meta-llama/Llama-3.1-8B-Instruct
  https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
- sentence-transformers/all-MiniLM-L6-v2
  https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
- openai/clip-vit-large-patch14
  https://huggingface.co/openai/clip-vit-large-patch14
- openai/clip-vit-base-patch32
  https://huggingface.co/openai/clip-vit-base-patch32
- laion/CLIP-ViT-H-14-laion2B-s32B-b79K
  https://huggingface.co/laion/CLIP-ViT-H-14-laion2B-s32B-b79K

Author

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.

Related product families

Product families related to this document are the following:

- Artificial Intelligence

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.

This document, LP2371, was created or updated on February 19, 2026.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP2371
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP2371.

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®

The following terms are trademarks of other companies:

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

Linux® is the trademark of Linus Torvalds in the U.S. and other countries.

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