

# Building Blocks of RAG with Intel






Dive into Retrieval Augmented Generation (RAG), the innovative approach that defines how organizations harness the value of their data with large language models (LLMs). Explore some of the Intel hardware and software building blocks that optimize RAG applications, enabling contextual, real-time responses while simplifying deployment and enabling scale.

▶ Tailoring GenAI for your Application . . . . .	2
▶ What is Retrieval Augmented Generation (RAG)? . . . . .	3
▶ Standard RAG Solution Architecture . . . . .	4
▶ Technologies for RAG . . . . .	5
▶ Accelerating RAG in Production. . . . .	6
▶ Opportunities for RAG in Enterprise . . . . .	9
▶ Take the Next Step. . . . .	9

# Tailoring GenAI for Your Application

The public debut of ChatGPT has changed the AI landscape. Enterprises are rushing to take advantage of this new technology to give them a competitive edge with new products, improved productivity and more cost-efficient operations.

Generative AI (GenAI) models, like Grok-1 (300B+ parameters) and GPT-4 (trillions+), are trained on massive amounts of data from the internet and other text sources. These 3rd party large language models are good for general-purpose use cases. However, most use cases for enterprises will require AI models to be trained and/or augmented with your data so the results can be more relevant to your business. Here are some examples of how generative AI can be applied within various industries.

 Consumer Goods and Retails	 Healthcare & Medicine	 Manufacturing	 Media & Entertainment	 Financial Services
<ul style="list-style-type: none"> <li>Virtual fitting rooms</li> <li>Delivery and installation</li> <li>In-store product-finding assistance</li> <li>Demand prediction and inventory planning</li> <li>Novel product designs</li> </ul>	<ul style="list-style-type: none"> <li>Assist busy front-line staff</li> <li>Transcribe and summarize medical notes</li> <li>Chatbots and answer medical questions</li> <li>Predictive analytics to inform diagnosis and treatments</li> </ul>	<ul style="list-style-type: none"> <li>Expert copilot for technicians</li> <li>Conversational interactions with machines</li> <li>Prescriptive and proactive field service</li> <li>Natural language troubleshooting</li> <li>Warranty status and documentation</li> <li>Understanding process bottlenecks, devising recovery strategies</li> </ul>	<ul style="list-style-type: none"> <li>Intelligent search, tailored content discovery</li> <li>Headline and copy development</li> <li>Real-time feedback on content quality</li> <li>Personalized playlists, news digests, recommendations</li> <li>Interactive storytelling via viewer choices</li> <li>Targeted offers, subscription plans</li> </ul>	<ul style="list-style-type: none"> <li>Uncovering trading signals, alerting traders to vulnerable positions</li> <li>Accelerating underwriting decisions</li> <li>Optimizing and rebuilding legacy system</li> <li>Reverse-engineering banking and insurance models</li> <li>Monitoring for potential financial crimes and fraud</li> <li>Automating data gathering for regulatory compliance</li> <li>Extracting insights from corporate disclosures</li> </ul>

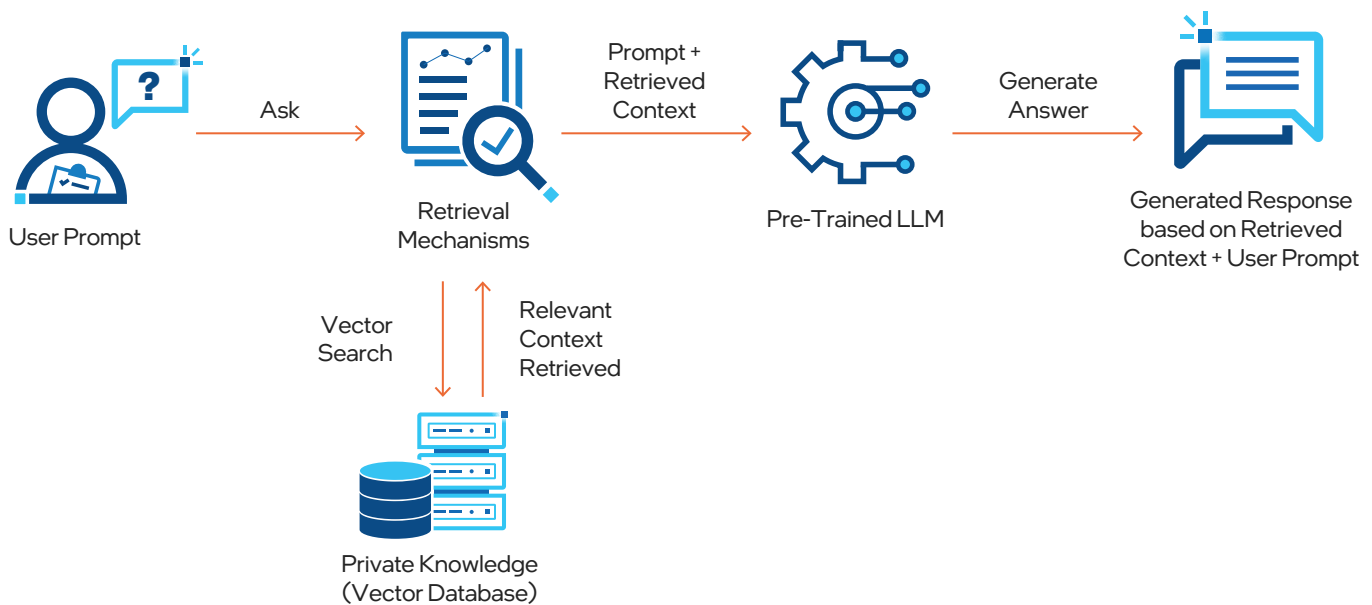
Source: Compiled by MIT Technology Review Insights, based on data from "Retail in the Age of Generative AI,"<sup>9</sup> "The Great Unlock: Large Language Models in Manufacturing,"<sup>10</sup> "Generative AI Is Everything Everywhere, All at Once," and "Large Language Models in Media & Entertainment,"<sup>12</sup> Databricks, April-June 2023.

While you can use your data to fine-tune a model, retraining a model takes additional time and resources. An alternative popular technique, retrieval augmented generation (RAG), creates a domain-specific LLM by augmenting open-source pre-trained models with your proprietary data to develop business-specific results. RAG allows you to keep your data safe and secure without sharing it with third-party large foundation models.

In this introductory guide, we will explain how RAG can be paired with various Intel optimizations and platforms to yield incredible value and performance for production GenAI systems.

# What is retrieval augmented generation (RAG)?

The RAG technique adds dynamic, query-dependent data into the model's prompt stream. Relevant data is retrieved from a custom-built knowledge base stored in a vector database. The prompt and the retrieved context enrich the model's output, delivering more relevant and accurate results. RAG allows you to leverage your data with an LLM while keeping the integrity of your data private, as it's not sent to a third party managing the model. The key components of the RAG workflow can be captured in four simple steps: user query processing, retrieval, context incorporation and output generation. The diagram below illustrates this basic flow.



RAG's utility is not confined to text; it can revolutionize video search and interactive document exploration, even enabling a chatbot to draw on PDF content for answers.

RAG applications are often called "RAG pipelines" due to their consistent data process flow, starting with the user prompt. The prompt is passed through the core component, the retrieval mechanism, which converts it into a vector embedding and uses vector search to find similar content in a pre-constructed vector database (e.g., from PDFs, logs, transcripts).

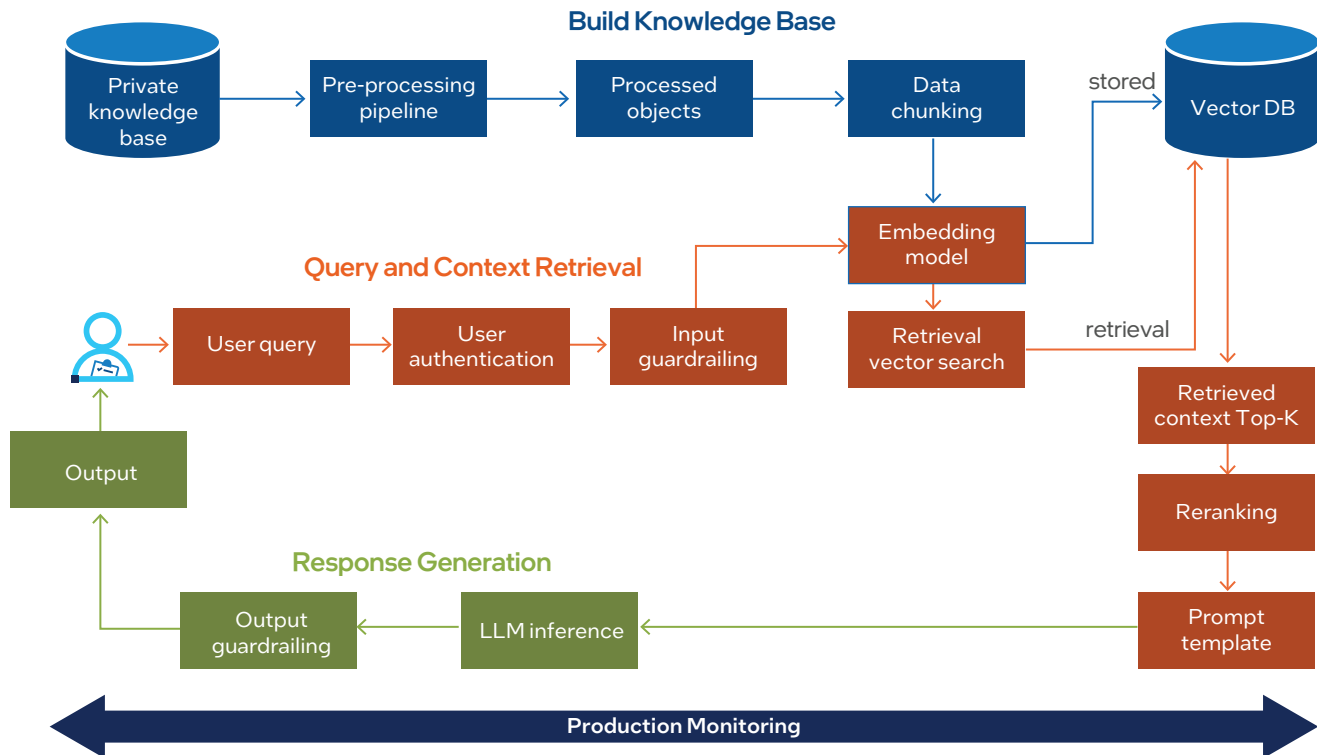
The most relevant data is retrieved, incorporated with the user's prompt, and passed to a model for inference service and final output generation. This context incorporation provides models with additional information unavailable during pre-training, better aligning them with the user's task or domain of interest. Because RAG does not require retraining or fine-tuning the model, it can be an efficient way to add your data to provide context to an LLM.

The next section will explore the RAG solution architecture and stack.

# Standard RAG Solution Architecture

The following RAG solution architecture provides an overview of the building blocks of a standard RAG implementation. Core components of the flow include **1** building the knowledge base, **2** query and context retrieval, **3** response generation and **4** production monitoring across applications.

## RAG LLM Architecture



Let's expand on some of these components:

### 1 Build the knowledge base:

- **Data Collection:** Assemble a private knowledge base from text-based sources such as transcripts, PDFs and digitized documents.
- **Data Processing Pipeline:** Utilize a RAG-specific pipeline for extracting text, formatting content for processing, and chunking data into manageable sizes.
- **Vectorization:** Process chunks through an embedding model to convert text into vectors, optionally including metadata for richer context.
- **Vector Database Storage:** Store vectorized data in a scalable vector database, ready for efficient retrieval.

### 2 Query and context retrieval:

- **Query Submission:** Users or a subsystem submit queries through a chat-like interface or API calls, authenticated by a secure service.
- **Query Processing:** Implement input safeguards for security and compliance, followed by query vectorization.
- **Vector Search and Re-ranking:** Conduct an initial vector search to retrieve relevant vectors, followed by a re-ranking process to refine results using a more complex model.

### 3 Response generation:

- **LLM Inference and Response Generation:** Combine top context with the user query, process through a pre-trained or fine-tuned LLM and post-process for quality and safety.
- **Response Delivery:** Return the final response to the user or subsystem through the interface, ensuring a coherent and contextually accurate answer.

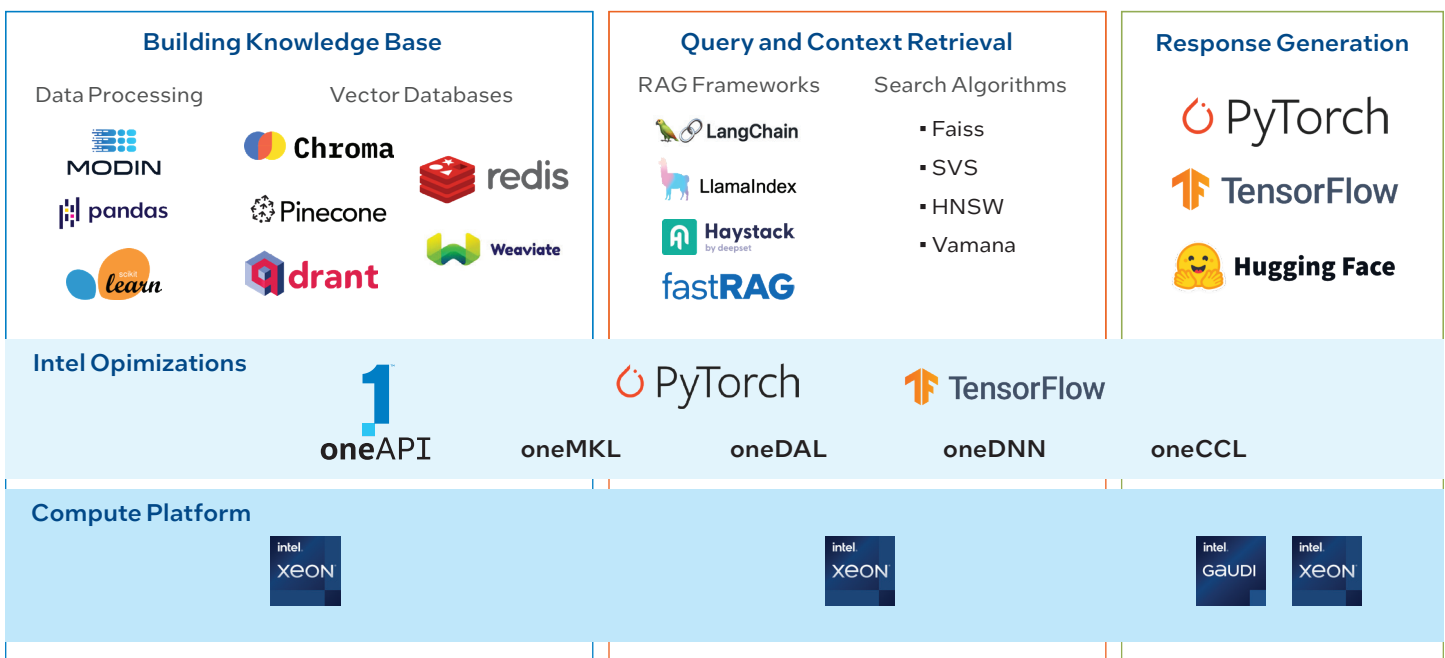
### 4 Production monitoring:

- **Retrieval Performance:** Monitor latency and accuracy of the retrieval process, keeping records for auditing purposes.
- **Re-ranking Efficiency:** Track re-ranking performance, ensuring contextual relevance and speed.
- **Inference Service Quality:** Observe latency and quality of LLM inference, maintaining logs for auditing and improvement.
- **Guardrail Effectiveness:** Monitor guardrails for input and output processing, ensuring compliance and content safety.

# Technologies for RAG

Developing RAG applications usually starts with integrated RAG frameworks, such as Haystack, LlamaIndex, LangChain and Intel Lab's fastRAG. These frameworks streamline development by offering optimizations and integrating essential AI toolchains.

Let's consider the RAG toolchain in the three familiar components: knowledge base construction, query and context retrieval and response generation. Often, RAG frameworks provide APIs that encompass the entire toolchain. Choosing between using these abstractions and leveraging independent components is a thoughtful engineering decision that should be considered carefully.



Intel optimizations bridge the gap between the toolchain and hardware, enhancing performance across the chain while ensuring compatibility and improved functionality on Intel® Xeon® CPUs and Intel® Gaudi® accelerators. These optimizations are integrated into stock frameworks or distributed as add-on extensions, with the goal to decrease the need for extensive low-level programming. This abstraction enables developers to focus on building RAG applications efficiently and effectively, leveraging enhanced performance and tailored solutions for their specific use cases.

Let's explore the various components of the toolchain in more detail.

## Building the knowledge base + context retrieval:

- **Integrated Frameworks:** Haystack and LangChain are notable RAG frameworks that offer high-level abstractions for vector databases and search algorithms, enabling developers to manage complex processes within Python-based environments.
- **Vector Database Technologies:** Pinecone, Redis and Chroma are some key vector database solutions that support popular search algorithms. Scalable Vector Search (SVS) from Intel Labs is another promising addition, expected to integrate with major vector databases by early 2024.
- **Embedding and Model Accessibility:** Embedding models, which are often integrated via Hugging Face APIs, can be seamlessly incorporated into RAG frameworks, making it easier to include advanced natural language processing (NLP) models.

## Response generation:

- **Low-Level Optimizations:** oneAPI performance libraries optimize popular AI frameworks like PT, TF, and ONNX so you can use your familiar open source tools knowing they are optimized for Intel® hardware.
- **Advanced Inference Optimization:** Extensions such as Intel® Extension for PyTorch add advanced quantized inference techniques, boosting performance for large language models.

As you can see, RAG involves several interconnected components and managing them on a single platform, like Intel Xeon CPUs, streamlines configuration, deployment and maintenance. For larger LLMs or high-throughput LLM inference, integrating Gaudi accelerators becomes an optimal solution for fulfilling application needs.

The following section dives into the complexities of RAG in production, addressing various considerations and technologies that help teams achieve successful deployment.

# Accelerating RAG in Production

Many components of the RAG pipeline are computationally intense, while at the same time, end users require low-latency responses. Additionally, because RAG is often used for confidential data, the entire pipeline must be secure. Intel technologies can power the RAG pipeline, contributing to secure performance across compute platforms and helping enable the full power of generative AI tailored to specific domains and industries.

## Computational demand

Generally, LLM inference is the most computationally-intensive phase of a RAG pipeline, particularly in a live production environment. However, creating the initial knowledge base — processing data and generating embeddings — can be equally demanding, depending on the data's complexity and volume. Intel's advancements in general compute technology, AI accelerators and confidential computing provide the essential building blocks for addressing the compute challenges of the entire RAG pipeline while ensuring data privacy and security.

Like most software applications, RAG benefits from a scalable infrastructure tailored to meet end-users' transactional demands. As transaction demand increases, developers may experience increased latency due to the load on compute infrastructure, which becomes saturated by vector database queries and inference calculations. For this reason, it's crucial to have access to readily available compute resources to scale up systems to quickly handle increased demand. Equally important is the need to implement critical optimizations to boost the performance of key components such as embedding generation, vector search and inference.

## Data privacy and security

- **Secure AI Processing:** Intel® [Software Guard Extensions](#) (Intel® SGX) and Intel® [Trust Domain Extensions](#) (Intel® TDX) boost data security via confidential computing and data encryption in CPU memory during processing. These technologies are crucial for handling sensitive information, contributing to the creation of secure RAG applications with encrypted data throughout the pipeline's various parts. This is an essential feature for RAG applications that require secure processing of sensitive data during vector embedding generation, retrieval, or inference.
- **Implement Proper Guardrailings:** In RAG applications, guardrailings involves implementing measures to manage the behavior of the LLM within the RAG system. This includes monitoring the model's responses, helping to adhere to guidelines and best practices, and controlling its output to decrease the risks of toxicity, unfair bias, and privacy breaches. Guardrailings in RAG applications helps maintain the trust and responsible usage of the LLM while ensuring it aligns with the system's overall goals and requirements.

## Open-source optimizations

### Embedding optimizations

- **Quantized Embedding Models:** Intel Xeon processors can take advantage of quantized embedding models to optimize the generation of vector embeddings from documents. A great example is [bge-small-en-v1.5-rag-int8-static](#), a version of BAAI/BGE-small-en-v1.5 quantized with Intel® [Neural Compressor](#) and compatible with [Optimum-Intel](#). Retrieval and re-ranking performance tasks using the quantized model on the Massive Text Embedding Benchmark (MTEB) reveals a < 2% difference between the floating-point (FP32) and the quantized INT8 version, on MTEB performance metric, while enhancing throughput (see footnote 1,3).

In a recent study with Hugging Face, we evaluated throughput for peak encoding performance in terms of documents per second. Overall, for all model sizes, the quantized model shows up to 4x improvement compared to the baseline bfloat16 (BF16) model in various batch sizes. Read more here: <https://huggingface.co/blog/intel-fast-embedding>

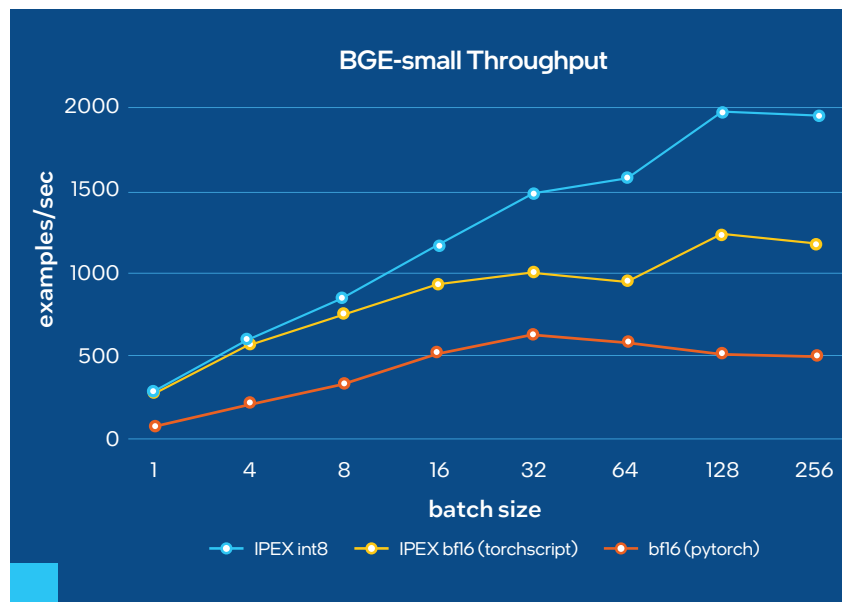
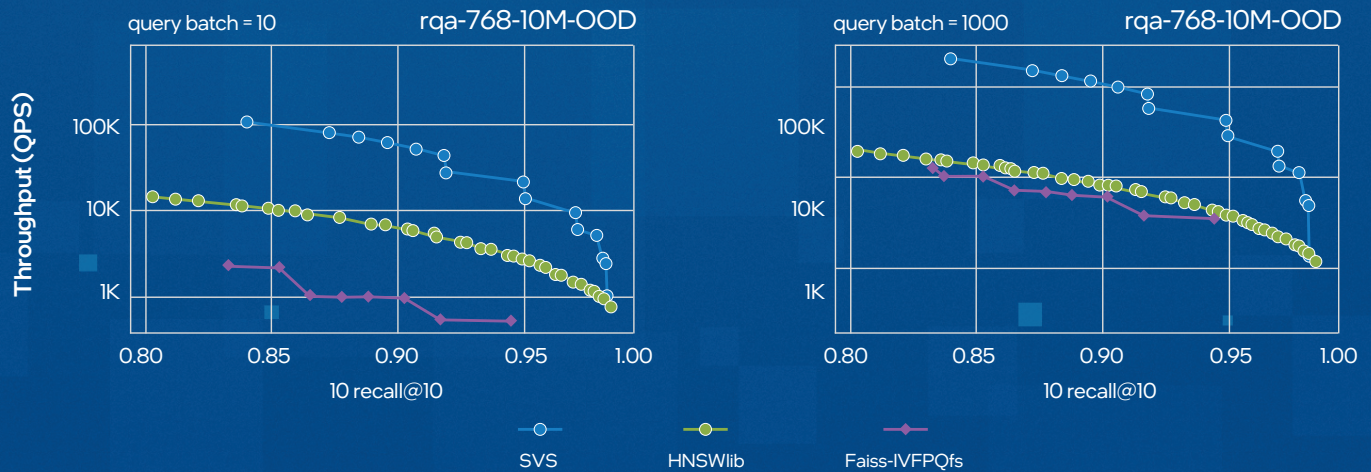


Figure 1: Throughput for BGE small source: <https://huggingface.co/blog/intel-fast-embedding>

## Vector search optimizations

- **CPU-Optimized Workloads:** Vector search operations are highly optimized on Intel Xeon processors, particularly with the introduction of Intel® Advanced Vector Extensions 512 (Intel® AVX-512) in 3rd generation processors or later. Intel AVX-512 leverages the fused multiply-add (FMA) instruction, which combines multiplication and addition in a single operation, enhancing inner product calculations — a fundamental operation in vector search. This capability significantly improves throughput and performance by reducing the number of instructions needed for computation.
- **Scalable Vector Search (SVS):** Scalable Vector Search (SVS) technology delivers fast vector search capabilities, optimizing retrieval times and improving overall system performance. SVS optimizes graph-based similarity search using locally-adaptive vector quantization (LVQ), which minimizes memory bandwidth requirements while maintaining accuracy. The result is significantly reduced distance calculation latency and higher performance in throughput and memory requirements, as demonstrated in the figure below.



**Figure 2.** Query per Second (Throughput) performance of SVS compared to other well adopted implementations, HNSWlib and FAISS's. The figure shows the QPS vs recall curves for the rqa-768-10M-OOD dataset (10M 768-dimensional embeddings generated with the dense passage retriever model RocketQA [QDLL21] with out-of-distribution queries). (Footnote 2,3) source: <https://intellabs.github.io/ScalableVectorSearch/benchs/static/latest.html>

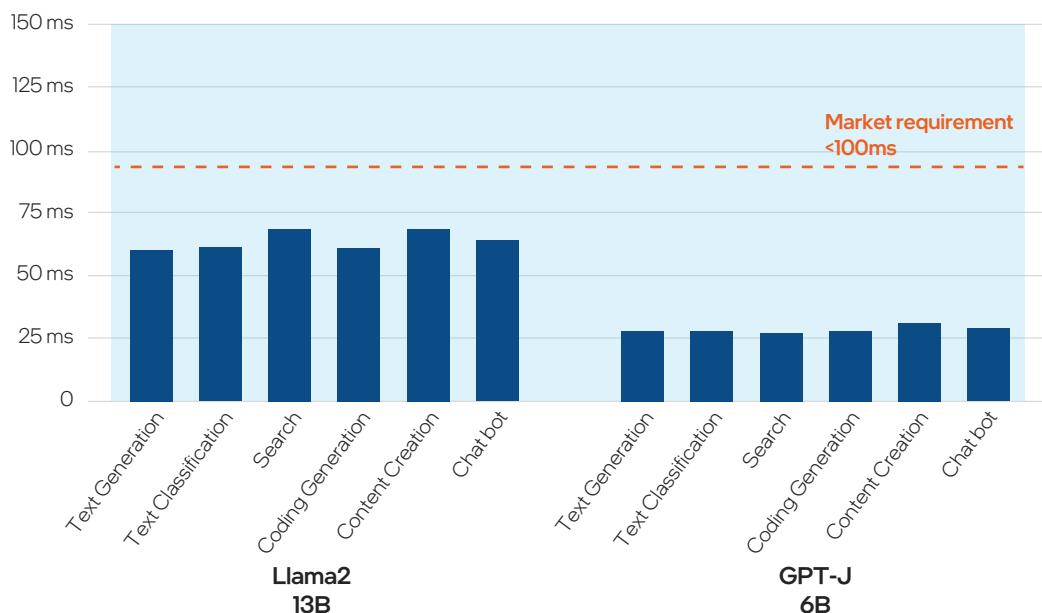
### Inference Optimizations

RAG primarily involves inference operations, which Intel Xeon processors support through advanced model compression techniques. These techniques enable operations at reduced precisions (BF16 and INT8) without significant performance loss. For larger models and high throughput requirements, Intel Gaudi accelerators offer excellent price/performance benefits and can replace CPUs and other accelerators for RAG inference. In this section, we'll outline various inference-specific optimizations and opportunities.

- **Intel Advanced Matrix Extensions (Intel AMX):** The 4th and 5th Gen Intel Xeon Scalable processors incorporate Intel AMX, enabling more efficient matrix operations and improved memory management.

- **Open Source SOTA Inference Optimization Tools:** Intel contributes to and extends popular deep learning frameworks like PyTorch, TensorFlow, Hugging Face, DeepSpeed, etc. Of interest for the RAG workflow are the opportunities to optimize LLMs by implementing model compression techniques like quantization. [Intel® Extension for PyTorch](#) currently provides a variety of state-of-the-art (SOTA) LLM quantization recipes such as SmoothQuant, weight-only quantization, and mixed precision (FP32/BF16). The figure below showcases the inference latency performance of an INT8-quantized Llama 2 model running on a single-socket 4th Gen Intel Xeon platform.

**5th Gen Xeon best market requirements on LLM latencies**  
Single node 2S 5th Gen Xeon 8592+ (64C) Large Language Model Next Token Latency



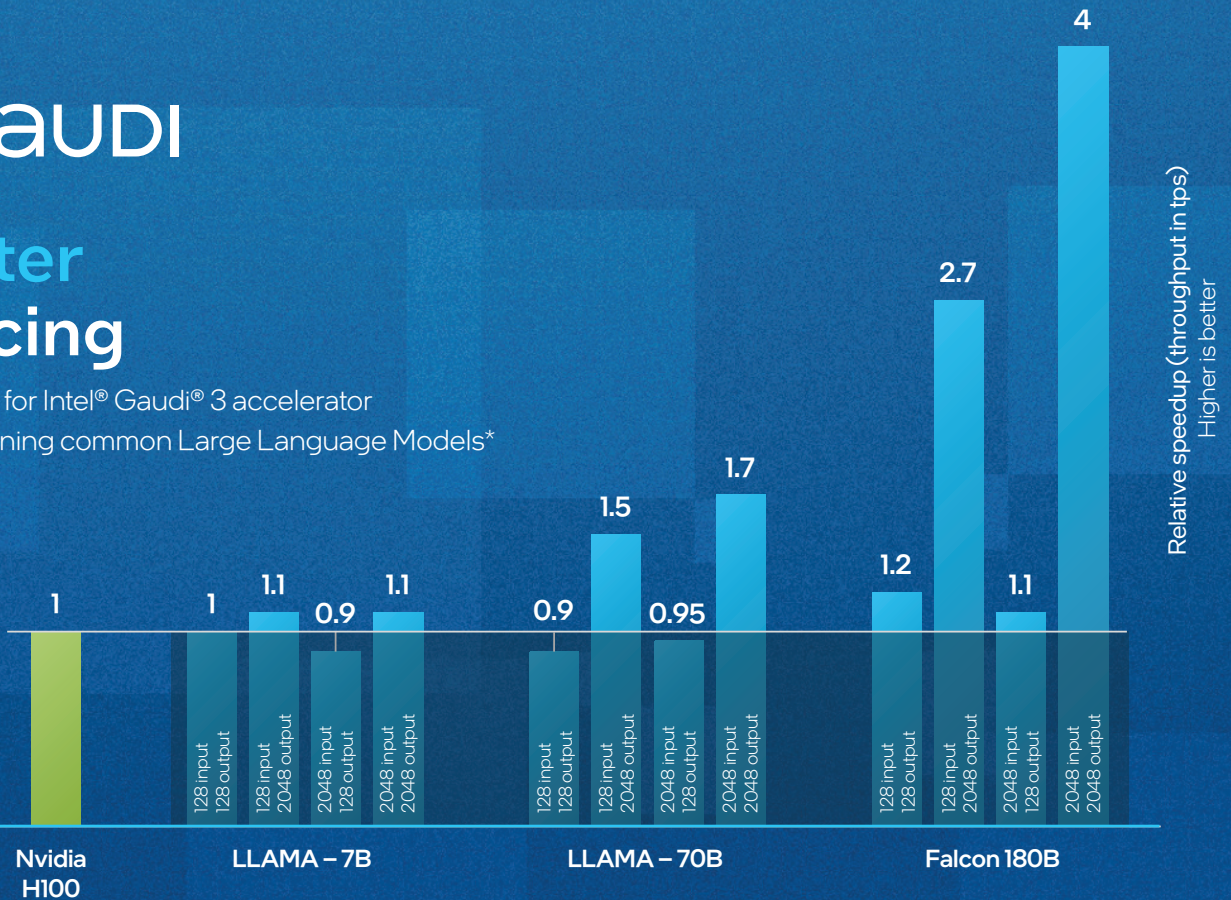
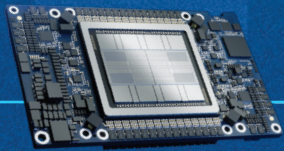
See backup configuration for workload and configurations. Results may vary.

**Figure 3.** Llama 2 13B and GPT-J 6B performance on 5th Gen Intel® Xeon® Scalable processors<sup>3</sup>

# intel GAUDI

## 1.5x faster inferencing

Average projection for Intel® Gaudi® 3 accelerator vs. Nvidia H100, running common Large Language Models\*



\*Source: NV H100 comparison based on <https://nvidia.github.io/TensorRT-LLM/performance.html#h100-gpus-fp8>, Mar 28th, 2024. Reported numbers are per GPU. Vs Intel® Gaudi® 3 projections for LLAMA2-7B, LLAMA2-70B & Falcon 180B projections. Results may vary.

Figure 4. LLM inference performance on Intel Gaudi 3

### Inference complexity and Intel Gaudi

One of the benefits of RAG is that since you depend less on the LLM's "knowledge" and more on its "language modeling capabilities," you can use models with a much lower parameter count. In many cases, a smaller 7B parameter model with RAG can beat larger models with tens of billions of parameters on domain-specific tasks associated with the RAG model's knowledge base.

Highly specialized tasks may sometimes require larger models and, consequently, specialized accelerators like Intel Gaudi processors. For RAG applications requiring the highest throughput or lowest latency, run LLM inference on the highest-performing AI accelerator available, such as an Intel Gaudi 3 processor.

### Explore Intel Gaudi RAG resources to learn more

- [Multi-Modal RAG Demo at Intel® Vision 2024](#) from Intel Labs Cognitive AI team
- A scalable [Retrieval Augmented Generation \(RAG\) application using Hugging Face tools](#) as a way of deploying optimized applications utilizing the Intel Gaudi 2 accelerator



# Opportunities for RAG in Enterprise

## Retail

Retailers face the challenge of recommending products that match their customers' diverse and changing preferences. Traditional recommendation systems may not effectively account for the latest trends or individual customer feedback, leading to less relevant suggestions.

Implementing a RAG-based recommendation system enables retailers to dynamically incorporate the latest trends and individual customer feedback into personalized product suggestions. This system enriches the shopping experience by offering relevant, timely and personalized product recommendations, driving sales and customer loyalty.

[LEARN MORE >](#)

## Manufacturing

In manufacturing, unexpected downtime due to equipment failure is a significant cost driver. Traditional predictive maintenance models may miss subtle anomalies that precede a failure, especially in complex machinery where historical failure data may be limited or nonexistent.

A RAG-based anomaly detection system for predictive maintenance can analyze vast amounts of operational data in real-time, comparing it against an extensive knowledge base of equipment performance to identify potential failures before they occur. This approach minimizes downtime and maintenance costs while extending equipment life.

[LEARN MORE >](#)

## Financial services

Providing personalized financial advice at scale can be challenging due to the vast amount of ever-changing financial data and regulations. Customers expect quick, relevant and personalized financial advice that traditional chatbots cannot always accurately provide.

A RAG model enhances a financial advice chatbot by dynamically pulling the most current financial data and regulations to generate personalized advice. By leveraging a vast knowledge base, the chatbot can provide clients with tailored investment strategies, real-time market insights and regulatory advice, enhancing customer satisfaction and engagement.

[LEARN MORE >](#)



## Take the next step

When you are ready to kick-start your implementation, Intel provides a suite of resources to help you get started, from hardware access in the Intel® Tiber™ Developer Cloud to ubiquitous compute in major cloud providers like Google Cloud Platform, Amazon Web Services, and Microsoft Azure. For developers seeking code samples, walkthroughs, training and more, please visit [Intel Developer Zone](#).

## Intel® Tiber™ Developer Cloud

Accelerate AI development using Intel®-optimized software on the latest Intel® Xeon® processors and GPU compute.

Accelerate AI development using Intel®-optimized software on the latest Intel® Xeon® processors, Intel® Gaudi® Accelerators, and other Intel platforms.



Access Intel hardware and start building RAG applications on cloud providers like Amazon Web Services, Google Cloud Platform and Microsoft Azure.



## Your Official Source for Developing on Intel® Hardware and Software

Explore Intel's most popular development areas and resources.

[Intel GenAI Development Resources](#)



<sup>1</sup> Performance claims based on 4th gen Intel Xeon 8480+ with 2 sockets, 56 cores per socket. Pytorch model was evaluated with 56 cores on 1 CPU socket. IPEX/Optimum setups were evaluated with ipexrun, 1 CPU socket, and cores ranging from 22-56. TCMalloc was installed and defined as an environment variable in all runs. See [www.intel.com/performanceindex](http://www.intel.com/performanceindex) for details. Results may vary.

<sup>2</sup> Performance claims based on a 2-socket 4th generation Intel® Xeon® Platinum 8480L CPU with 56 cores per socket, equipped with 512GB DDR4 memory per socket @4800MT/s speed, running Ubuntu 22.04.12 For the deep-96-1B, dataset we use a server with the same characteristics except that it is equipped with 1TB DDR4 memory per socket @4400MT/s speed. See [www.intel.com/performanceindex](http://www.intel.com/performanceindex) for details. Results may vary.

<sup>3</sup> Performance varies by use, configuration and other factors. Learn more at [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure. Your costs and results may vary. Intel technologies may require enabled hardware, software or service activation. © Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.  
0424/SN/MESH/PDF 358260-001US