RESEARCH-ARTICLE

AskHPC: A ChatBot for High Performance Computing User Support

AKHILESH BONDAPALLI, Clemson University, Clemson, SC, United States

HUIHUO ZHENG, Argonne National Laboratory, Lemont, IL, United States

OLUWASEUN T. AJAYI, Illinois Institute of Technology, Chicago, IL, United States

MURAT KECELI, Argonne National Laboratory, Lemont, IL, United States

HARITHA SIDDABATHUNI SOM, Argonne National Laboratory, Lemont, IL, United States

TAYLOR CHILDERS, Argonne National Laboratory, Lemont, IL, United States

View all

Open Access Support provided by:

Clemson University

Argonne National Laboratory

Illinois Institute of Technology

Published: 15 November 2025

SC Workshops '25: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis
November 16 - 21, 2025
MO, St Louis, USA

Conference Sponsors:
SIGHPC

https://doi.org/10.1145/3731599.3767433
ISBN: 9798400718717

AskHPC: A ChatBot for High Performance Computing User Support

Akhilesh Bondapalli
Clemson University
Clemson, USA
abondap@clemson.edu

Huihuo Zheng
Argonne National Laboratory (ANL)
Lemont, USA
huihuo.zheng@anl.gov

Oluwaseun T. Ajayi
Illinois Institute of Technology
Chicago, USA
oajayi6@hawk.illinoistech.edu

Murat Keceli
Argonne National Laboratory (ANL)
Lemont, USA
kecelim@anl.gov

Haritha Siddabathuni Som
Argonne National Laboratory (ANL)
Lemont, USA
haritha@anl.gov

Taylor Childers
Argonne National Laboratory (ANL)
Lemont, USA
jchilders@anl.gov

Lisa Childers
Argonne National Laboratory (ANL)
Lemont, USA
childers@anl.gov

Yasaman Ghadar
Argonne National Laboratory (ANL)
Lemont, USA
ghadar@anl.gov

Michael Papka
Argonne National Laboratory (ANL)
Lemont, USA
papka@anl.gov

Venkatram Vishwanath
Argonne National Laboratory (ANL)
Lemont, USA
venkat@anl.gov

Rong Ge
Clemson University
Clemson, USA
rge@clemson.edu

Abstract

High-Performance Computing (HPC) systems in the exascale era are increasingly heterogeneous, requiring users to navigate diverse tools, configurations, and best practices. However, essential information is often scattered across fragmented, multimodal documentation, making it difficult and time-consuming to locate. To address this, we present **AskHPC**, an intelligent question-answering ChatBot that delivers accurate, timely, and accessible information through a unified conversational interface. Built on a curated knowledge base integrating user guides, scheduler manuals, and programming documentation, **AskHPC** leverages Large Language Models (LLMs) within a Retrieval-Augmented Generation (RAG) framework. It employs two key techniques to improve HPC query responses: a modality-aware document parsing pipeline that preserves multimodal structure, and a dual-context strategy combining retrieved content (e.g., complete code blocks) with LLM-generated semantics. Evaluation, including a real-world user study, shows **AskHPC** outperforms direct LLM queries and vanilla RAG systems, enhancing user support and accelerating HPC software development.

CCS Concepts

• **Computing methodologies** → **Natural language processing**; *Discourse, dialogue and pragmatics*; • **Information systems** → **Information retrieval**; *Retrieval models and ranking*; • **Software and its engineering** → *Software libraries and repositories*; • **Human-centered computing** → User studies.

Keywords

Large Language Models, Retrieval-Augmented Generation, HPC User Support, ChatBot, Exascale Computing, Multimodal RAG

ACM Reference Format:
Akhilesh Bondapalli, Huihuo Zheng, Oluwaseun T. Ajayi, Murat Keceli, Haritha Siddabathuni Som, Taylor Childers, Lisa Childers, Yasaman Ghadar, Michael Papka, Venkatram Vishwanath, and Rong Ge. 2025. AskHPC: A ChatBot for High Performance Computing User Support. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3731599.3767433

SC Workshops '25, St Louis, MO, USA

ACM ISBN 979-8-4007-1871-7/25/11
https://doi.org/10.1145/3731599.3767433

1 Introduction

High-Performance Computing (HPC) facilities [9, 32, 45] provide critical supercomputing resources [4, 8, 31, 41] that drive scientific discovery and engineering innovation. However, these systems are increasingly heterogeneous and complex as we enter into an era of exascale computing and artificial intelligence (AI), featuring a wide range of architectures, including traditional CPUs, various GPUs and emerging AI accelerators. Effectively utilizing these resources is far from straightforward. Users must follow detailed documentation to develop, optimize, and execute their applications. Due to the complexity and diversity of these systems, documentation is vast and fragmented, spanning facility user guides, job scheduler user docs, compiler manuals, code examples, configuration tables, and best practices. These materials are often scattered across

facility websites, Git repositories, PDFs, and Wiki forums, making it difficult for users to locate relevant information efficiently. This poses a significant barrier to usability, often requiring deep system knowledge and leading to frustration, errors, and reduced productivity.

Since the emergence of ChatGPT [50], Large Language Model (LLM) based conversational AI models have become increasingly popular across diverse domains such as education [60], healthcare [12], and customer service [19], owing to their intuitive interfaces and ability to assist users in navigating complex information. These LLMs have demonstrated the ability to provide advanced, context-aware assistance across a wide range of knowledge domains. They have significant potential to reshape how users interact with complex systems and documentation - by lowering the barrier to accessing specialized knowledge through natural language interaction, thus reducing the cognitive load typically required to search for and interpret technical content.

While general-purpose models offer a unified interface, their direct application to the high-stakes HPC domain is fraught with challenges. These models often suffer from outdated knowledge bases [18, 20], lack the specificity needed to handle the intricacies of specialized HPC systems [28], and are prone to factual inaccuracies or "hallucinations" [11, 18]. In the HPC domain, where accuracy and reliability are key, such errors can have severe consequences — resulting in misconfigurations, failed computations, wasted computational resources, or even violations of facility policies. The following zero-shot query highlights the potential risks of relying on general-purpose conversational models such as ChatGPT for HPC-specific tasks:

Query: Please provide a submission script on ALCF Aurora to run a python job on 1024 nodes.

Answer from GPT-4 Turbo:

```
#!/bin/bash
#COBALT -n 1024                    # Number of nodes
#COBALT -t 30                      # Walltime in minutes
#COBALT -q default                 # Queue name
#COBALT -A your_project_account    # Project allocation
#COBALT -O my_python_job           # Output file prefix

# Load necessary modules
module load conda/2023-10
conda activate myenv

# Optional: Load Intel oneAPI, if needed
# module load intel-oneapi-mpi

# Path to your Python script
SCRIPT=your_script.py

# Launch the job using mpirun
mpirun -np 1024 python $SCRIPT
```

The response from GPT-4 Turbo [47] suffers incorrectness in the following ways: (1) It incorrectly uses Cobalt job submission, which was used by previous systems at Argonne Leadership Computing Facility [9], whereas Aurora [8] is using PBSPro instead; (2) The default conda module should be `frameworks` instead of `conda/2023-10`; (3) The MPI command incorrectly specifies 1024 processes instead of 1024 nodes. The key reason for this failure is that GPT-4 Turbo was trained with a knowledge cutoff in April 2024. While newer models such as GPT-4o and o1 can give slightly better answers (for instance, by identifying the correct PBS scheduler), they are still unable to give completely accurate and satisfactory responses.

To address these critical gaps, we present an intelligent Question Answering (QA) ChatBot system, **AskHPC**, tailored for HPC. **AskHPC** leverages a novel Retrieval-Augmented Generation (RAG) framework [34] to provide users with reliable, context-aware answers through a single interface. By grounding LLM responses in a curated knowledge base built from verified, facility-specific, and up-to-date documentation, our approach directly mitigates the risks of hallucination and outdated information inherent in standard LLMs [18, 20]. For the question asked previously, **AskHPC** provides a fully accurate job submission script, including relevant references:

```
#!/bin/bash -l
#PBS -l select=1024          # selecting 1024 Nodes
#PBS -l place=scatter
#PBS ...[Other PBS settings are omitted]

NNODES=`wc -l < $PBS_NODEFILE`
NRANKS_PER_NODE=12
let NRANKS=${NNODES}*${NRANKS_PER_NODE}

module use /soft/modulefiles
module load frameworks
# ...[oneCCL and Slingshot environment settings are omitted]
mpiexec -np ${NRANKS} -ppn ${NRANKS_PER_NODE} \
--cpu-bind ${CPU_BIND} \
python path/to/application.py
---
References:
- [1] https://github.com/argonne-lcf/user-guides/blob/main/docs/
      aurora/data-science/frameworks/tensorflow.md
```

To provide the LLM with the rich, high-fidelity context needed for accurate and comprehensive answers, **AskHPC** enhances vanilla RAG using two techniques. First, it employs a modality-aware parsing pipeline to preserve the structure of multimodal content (text, code, tables, images) during knowledge base creation. Second, it applies a dual-context strategy that combines high-fidelity original content (e.g., actual code blocks or table data) with LLM-generated semantics at runtime for information retrieval and response synthesis.

To validate the effectiveness of **AskHPC**, we employ a comprehensive, two-part evaluation strategy. First, to assess the system's practical impact on its intended audience, we conduct a human-centric user study with HPC practitioners. This study measures the tangible benefits of **AskHPC** in terms of correctness and relevance compared to direct LLM querying. Second, we conduct a large-scale automated assessment using over 8,700 synthetically generated question-answer pairs to measure the system's factual correctness and the quality of its retrieval mechanism.

Our key contributions are:

- A domain-specific, RAG-powered **QA ChatBot for HPC user support**, backed by a curated knowledge base built from diverse user and developer documentation sources.
- A novel **LLM-assisted multimodal parsing pipeline** that structures heterogeneous technical content including text, code, tables, and images for effective retrieval.
- A **dual-context strategy** that integrates original content (e.g., code blocks, table data) with LLM-generated semantic context, yielding higher-fidelity and more accurate response synthesis.

- A rigorous **two-pronged evaluation** integrating large-scale automated testing with a real-world user study, demonstrating substantial accuracy gains and practical benefits for HPC users.

2 Background and Motivation

Current user support at HPC facilities, such as those at the Argonne Leadership Computing Facility (ALCF) [9], Oak Ridge Leadership Computing Facility (OLCF) [45], and Livermore Computing (LC) platforms [32], is robust, yet increasingly challenged by the complexity of emerging architectures and workloads. These facilities provide extensive support services, including onboarding, code porting, scaling assistance, and performance optimization. User support is typically delivered through a combination of user documentation [10, 33, 46], helpdesk systems [7, 33, 44], ticketing portals, direct consulting, and targeted training events such as workshops and hackathons [5, 26, 43]. Each facility maintains a team of experts including computational scientists. Additionally, computer scientists serve as catalysts and performance engineers [6, 36, 42]. These experts partner with users to address application-specific challenges, ensure readiness for new systems, and improve scientific productivity on the supercomputers.

User documentation is a critical pillar of HPC user support [10, 33, 46], serving as the first point of reference for users navigating complex systems and software stacks. Well-structured, accurate, and up-to-date documentation empowers users to solve problems independently, accelerates onboarding, and reduces the demand on support staff. As HPC systems become more heterogeneous and software environments grow in complexity, comprehensive documentation becomes essential for scaling support efforts and ensuring that users can fully leverage the capabilities of cutting-edge infrastructure. The documentation includes detailed information on the software and hardware description of the system, account application, software compiling and building, job submission, performance optimization, debugging, etc.

However, the increasing complexity of the documentation poses significant barriers for users. Users have to manually integrate guidance from fragmented documentation repositories—spanning user guides, technical manuals, software documentation, and documentation on facility-specific policies located in disparate sources, such as Wikis, GitHub repositories, Slack channels, and internal knowledge bases. This often forces even experienced users to undertake a trial-and-error process, let alone newcomers who may not even know what information to look for. The expanding diversity of workflows, from traditional simulation to data analytics and machine learning, adds further pressure on support teams to deliver timely, accurate, and context-specific help. To bridge these gaps, there is an increasing need for leadership computing facilities to explore AI-driven support systems that can deliver intelligent, multimodal assistance tailored to user needs.

The primary objective of this work is to develop an intelligent QA service tailored to overcome the aforementioned challenges by leveraging state-of-the-art LLM techniques to enhance the HPC user support system. The key idea is to utilize conversational interfaces for efficient interactive and intuitive information retrieval, moving beyond traditional keyword searches and link navigation. For example, the user can ask a question like, *"How do I run large-scale PyTorch jobs on Aurora? What module should I use, and what are the best settings?"* The ChatBot shall provide an answer, including information on what software framework to use, key software library settings for oneCCL [1], and even provide Intel-specific instructions for PyTorch.

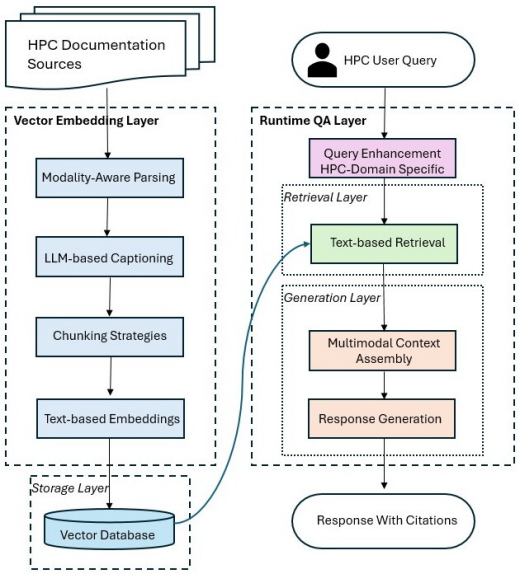
Deploying general-purpose conversational AI models directly for HPC support leads to insufficient, inaccurate and misleading responses. These models often possess outdated knowledge [18, 20], lack the necessary depth and specificity for specialized HPC systems [28], and are prone to factual inaccuracies or "hallucinations" [11, 18]. This is generally unacceptable in the HPC domain, which requires high accuracy. The example shown in Section 1 clearly demonstrates the inadequacy of general-purpose conversational AI models. There is a critical need for a QA service that provides reliable, context-aware, and factually accurate answers grounded in verified, facility-specific documentation.

Fine-tuning and continual pretraining are widely used strategies to adapt LLMs for specialized domains. Fine-tuning involves retraining a pretrained model on labeled, domain-specific datasets to specialize its outputs for downstream tasks [55, 62]. Continual pretraining, in contrast, incrementally updates a pretrained model using unlabeled, domain-relevant text corpora to enhance its general language understanding within the target domain [21, 25]. While both approaches can reduce hallucinations and improve domain alignment, they are not without limitations. In particular, continual updates can lead to catastrophic forgetting, where previously learned general knowledge is overwritten by new domain-specific data [15, 22]. Additionally, these methods are computationally expensive, require frequent retraining, and depend on the availability of curated HPC-specific corpora, which are often fragmented or not publicly accessible. Furthermore, they provide limited adaptability to rapidly changing HPC environments and offer little transparency into how the models use or represent domain knowledge [51, 58].

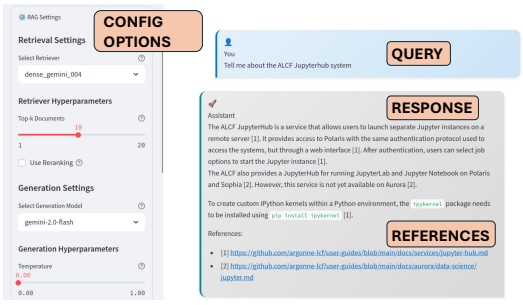
RAG is an alternative approach proposed to mitigate inherent limitations such as knowledge cutoffs and hallucination [20, 34]. It combines the generative power of LLMs with the factual grounding provided by external knowledge bases. In response to a user query, instead of relying solely on the information encoded within the LLM's parameters during training, a RAG system first retrieves relevant information snippets from a specified knowledge corpus [18]. This retrieval step typically involves encoding the query and document chunks into vector embeddings and performing similarity search [14] within an indexed database. The retrieved context is then appended to the original query and fed into the LLM, which generates an answer conditioned on both the query and the provided evidence, thereby producing more factual, specific, and verifiable responses [20].

We adopt the RAG approach to develop the **AskHPC** chatbot. This allows us to specifically select verified, accurate, and up-to-date documents to build the knowledge base. The knowledge base can also be updated independently, enabling rapid incorporation of new documentation and maintaining timeliness in a fast-evolving HPC landscape. What is more, by exposing the original sources used to generate each response, RAG provides critical explainability and traceability, allowing for the validation of technical guidance.

3 System Design and Implementation



(a) Architecture



(b) User Interface

Figure 1: The architecture and user interface of AskHPC: (a) High-level architecture includes the Vector Embedding Layer for knowledge base construction and the Runtime QA Layer for online query processing; (b) User Interface includes Query, with the Response and References generated. A configuration panel on the left is provided to allow advanced users to adjust retrieval and generator parameter settings.

We design the system with the following high-level goals:

(1) **Timeliness**: Ensure the knowledge base remains current through version control that tracks documentation changes and supports dynamic updates.
(2) **Reliability**: Generate answers exclusively from validated documentation, with built-in mechanisms for assessing reliability, ensuring transparency, and referencing sources. The system should also identify and flag knowledge gaps.
(3) **Conformability**: Support flexible configuration of documentation sources, LLMs for embedding and generation, and system hyperparameters to meet diverse user needs.

The overall system architecture is depicted in Figure 1, including an offline **Vector Embedding Layer** for constructing the knowledge base, and an online **Runtime QA Layer** for processing user queries to generate grounded answers. Central to our design is a novel approach to handling the diverse data modalities (text, code, tables, images) inherent in technical documents, ensuring that the full richness of the source material informs the final response.

3.1 Documents Selection

Addressing the breadth and depth of questions highlighted across different categories necessitates grounding the QA service in a comprehensive, accurate, and up-to-date knowledge base. We preselect a relatively comprehensive list of documentation sources provided in Table 1 for building our knowledge base. It includes user guides from three major US Leadership computing facilities [10, 33, 46], documentation for two popular schedulers in PBSPro [3] and Slurm [57], and documentation of four dominant programming models - CUDA [40], HIP [2], SYCL [56] and Intel oneAPI [27]. We integrate all these diverse documents into one unified knowledge base to support queries that may need information retrieved from multiple sources.

Table 1: Data Sources for the Knowledge Base

Data	Type / format	#Char
ALCF User Guide [10]	Markdown	1,419,414
OLCF User Guide [46]	RST	1,292,381
LLNL User Guide [33]	Web	1,502,278
PBS Documentation [3]	PDF	514,412
Slurm Documentation [57]	Web	3,929,498
CUDA C Programming Guide [40]	Web	1,282,897
SYCL Documentation [56]	PDF	838,869
AMD HIP Documentation [2]	Web	3,824,417
Intel OneAPI Documentation [27]	Web	1,221,829

We design our framework to support continual expansion of the knowledge base, enabling seamless integration of new documentation and updates of existing ones to ensure that users always have access to the most current and comprehensive information. The document sources are defined in a YAML configuration file, and the user can easily extend the list to include new documents. We also have a built-in version control system to keep track of updates to the existing documents. The system will regularly ingest and index updated content.

3.2 Vector Database Construction

The Vector Embedding Layer, illustrated on the left side of Figure 1, transforms the documentation sources listed in Table 1 into an indexed, searchable representation suitable for retrieval. This multi-stage process begins with **Modality-Aware Parsing**, where carefully designed custom parsers convert documents (HTML, PDF, RST, etc.) to Markdown and dissect them into discrete structural elements, explicitly tagging each element with its modality (text, code blocks, tables, images) while preserving crucial metadata like source origin and section context.

A key innovation follows: **LLM-based Semantic Captioning**. This is applied specifically to non-text elements like code snippets, data tables, or architectural diagrams, which convey vital information but are not easily retrievable in their raw form. We employ Gemini 2.0 Flash LLM to generate concise, descriptive text captions for each element. The generated captions act as semantic proxies, providing rich metadata such as the purpose of a code block, the

content of a table, or the description of an image. This text-centric indexing approach avoids the complexities of purely multimodal embedding spaces while still enabling effective retrieval based on semantic similarity. For example, below is a code block from the LLNL Documentation [33] and the generated semantic caption, which is much more descriptive than the original text surrounding the code block.

```
SHELL = /bin/sh

.SUFFIXES: .cc .o

CXX = /usr/tce/packages/rocmcc/rocmcc-6.1.2-magic/bin/amdclang++
CPPFLAGS = -D__HIP_PLATFORM_AMD__ -I/opt/rocm-6.1.2/include
CXXFLAGS = -O3 --cuda-gpu-arch=gfx942 -std=c++11 --rocm-path=/opt/
      rocm-6.1.2 -x hip
HIPFLAGS = -mllvm -amdgpu-early-inline-all=true -mllvm -amdgpu-
      function-calls=false -fhip-new-launch-api --driver-mode=g++
LDFLAGS =
LDLIBS =

all: rush_larsen_gpu_hip

rush_larsen_gpu_hip: rush_larsen_gpu_hip.cc
        $(CXX) $(CPPFLAGS) $(CXXFLAGS) $(HIPFLAGS) $+ $(LDFLAGS) $
             (LDLIBS) -o $@

check:
        ./rush_larsen_gpu_hip 10 .01

clean:
        rm rush_larsen_gpu_hip
```

The following code block is a Makefile designed to compile a C++ file (`rush_larsen_gpu_hip.cc`) for AMD GPUs using the Clang compiler with the ROCm framework. It defines compiler flags (`CPPFLAGS`, `CXXFLAGS`, `HIPFLAGS`), linker flags (`LDFLAGS`, `LDLIBS`), and the compilation process. The Makefile creates an executable named `rush_larsen_gpu_hip`, includes a `'check'` target to run the executable with specified arguments, and a `'clean'` target to remove the executable. It specifies the ROCm path and uses specific flags for HIP (Heterogeneous-compute Interface for Portability) compilation.

The textual content—comprising original text elements and the generated captions—is then processed using configurable **Chunking Strategies**. This involves segmenting the text into manageable units, balancing semantic coherence with retrieval granularity [14]. The text chunks and captions are then converted into embedding vectors through **Text-based Embeddings** using sentence-transformer models. These vectors are then ingested into a Milvus vector database [59] in the **Storage Layer**. We selected Milvus [59] for the vector database due to its proven efficiency and scalability in large-scale vector similarity search [64]. This ensures our system can readily accommodate future growth in documentation corpora and increasing query loads.

The entire pipeline produces a set of Markdown files totaling 16.7 MB in the intermediate representation, and a vector database ranging from 60 to 80 MB for a chunk size of 1024 characters.

3.3 Query Processing and Answer Generation

In the **Runtime QA Layer**, the incoming user's query first undergoes a **Query Enhancement**, in which we utilize LLMs to refine, expand, or decompose the user's query into potentially multiple, more precise search terms to improve the likelihood of retrieving relevant information. This is demonstrated in the example shown below:

Query: What are the architecture differences between Aurora and Frontier?

Enhanced Queries

1. Aurora supercomputer architecture details.
2. Frontier supercomputer architecture details.
3. Compare Aurora supercomputer architecture to Frontier supercomputer architecture.

The enhanced query is passed to the **Retrieval Layer** to find the most semantically similar text/caption chunks from the indexed vector database. We support dense retrieval with a variety of indexing algorithms, including IVF_FLAT [59] for approximate nearest neighbor search, and FLAT for an exact search.

The retrieved text/caption chunks proceed to the **Generation Layer**, which incorporates the second phase of our core novelty: **Multimodal Context Assembly**. Instead of merely passing the retrieved text and captions to the generator, the system retrieves the *original, complete non-text element* (e.g., the full code block with syntax highlighting information, the structured table data, or the image reference/data). This assembled context, comprising relevant text passages *and* high-fidelity non-textual elements, provides the generator LLM with significantly richer and more accurate information than either text-only descriptions or uncaptioned elements. With the assembled context, **Response Generation** then synthesizes a coherent, informative answer with citation references linking back to the original documentation sources for transparency, enabling users to verify the information and explore the context further.

The architecture is highly configurable, supporting diverse LLMs (open-source via local inference using frameworks like vLLM [30], or proprietary models via APIs) for captioning, query enhancement and generation, and embedding models, allowing adaptation to various deployment constraints and performance needs.

3.4 User Interface

The **AskHPC** user interface shown in Figure 1(b) utilizes a main chat panel for queries and responses. The panel renders text, code, tables and images in Markdown format. Inline citations (e.g., "[1]", "[2]") link each generated statement to the URLs of the original source documents, specified in a "References" section for verification. A sidebar provides advanced users control over retrieval (embedding model, top-K) and generation (LLM, temperature) settings, highlighting the system's modular and configurable RAG architecture.

Overall, we would like to highlight that our system design uniquely integrates three LLM-driven novelties that do not exist in common RAG systems. Firstly, **semantic metadata enrichment** uses LLMs to generate descriptive text captions for non-text elements (code, tables, images), creating rich, searchable proxies. Secondly, **user query in-context enhancement** applies LLMs to refine and expand user queries, boosting retrieval relevance. Thirdly, our approach to **multi-modal answer synthesis** is distinct: instead of relying solely on text or captions, we provide the generator LLM with the retrieved text (and captions) *alongside* the original, complete non-text elements, facilitating superior context understanding.

4 Experiment Design

4.1 Real-World User Evaluation

We have initiated a systematic user evaluation study during the deployment of AskHPC at a US DOE Leadership Computing facility. The study involves 15 HPC experts and users from diverse backgrounds in computer science and various domain sciences. Participants posed realistic questions and rated the answers on a five-point scale based on correctness and relevance.

4.2 LLM-as-a-Judge Evaluation

To evaluate **AskHPC** more comprehensively, we conducted an automated study to complement the user evaluation. This study assessed retrieval and generation performance by measuring the quality of retrieved context, the relevance and accuracy of generated answers, and the impact of different model and system configurations. As a baseline, we compared against responses from the underlying LLM without the knowledge base, isolating the contribution of the RAG architecture.

4.2.1 Evaluation Dataset: LLM generated synthetic QA Pairs. Following the methodology in [35], we generate a set of HPC domain-specific question-answer (QA) pairs using the long-context `Gemini 2.0 Flash` LLM [23]. The input to the model consists of segmented portions of our entire document corpus, which we used to construct the vector database. Each segment contains up to 12,000 characters. This process yields a dataset of 8,772 synthetic QA pairs. We further use `Gemini 2.0 Flash` to classify these pairs into 25 distinct categories, as shown in Table 3. Each QA pair can belong to more than one category. These synthetic questions are designed to reflect realistic queries from HPC users. The full dataset of 8,772 QA pairs is used for the majority of our evaluation studies. To support more targeted evaluations, we also construct two smaller subsets:

- **250 stratified QA pairs** by selecting 10 pairs from each category. This subset supports the hyperparameter tuning and model scaling experiments.
- **67 multiline-code QA pairs** by selecting queries where the ground truth contains at least one code segment of 20 lines or more. This subset supports the comparative study of AskHPC and vanilla RAG.

4.2.2 Evaluation Methodology: LLM-as-a-Judge. We employ an automated evaluation methodology using LLM-as-a-Judge [63]. This approach leverages the advanced reasoning and language understanding capabilities of modern LLMs to assess the quality of generated answers against the ground truth. While the field is rapidly evolving, LLM-based evaluation is increasingly recognized for correlating better with human judgment compared to traditional lexical overlap metrics (e.g., BLEU, ROGUE) [17, 63], especially for nuanced tasks like evaluating factual correctness and semantic relevance in QA systems. Specifically, in our evaluation, we utilize the `Gemini 2.0 Flash` Model as the evaluator. For consistency, where applicable (e.g., for embedding ground truth/generated answers for similarity checks within metrics), the `Gemini text-embedding-004` model [24] is used.

4.2.3 Evaluation Metrics. We evaluate **AskHPC** using key metrics from the RAGAS framework [17], assessing both the quality of the final generated answer and the effectiveness of the intermediate retrieval step using an LLM judge.

Answer Correctness (AC). This primary metric measures the factual accuracy and semantic relevance of the generated answer compared to the ground truth answer. It combines two components:

- **Factual Correctness (FC):** Assesses the factual alignment between the generated answer and the ground truth. The LLM identifies true positive (TP), false positive (FP), and false negative (FN) statements to compute an F1-score:

$$FC = \frac{TP}{TP + 0.5 \times (FP + FN)} \tag{1}$$

- **Semantic Similarity (SS):** Calculates the cosine similarity between the vector embeddings of the generated answer and the ground truth answer.

The final *Answer Correctness* score is a weighted average:

$$AC = \alpha \times FC + (1 - \alpha) \times SS \tag{2}$$

We set $\alpha = 0.75$, the default value in RAGAS. Higher scores (approaching 1.0) indicate answers that are both factually accurate and semantically aligned with the ground truth.

Context Precision. This metric evaluates the relevance of the retrieved context chunks provided to the generator LLM. Higher scores indicate that the retrieved context is focused and contains less irrelevant information. Context precision is given by:

$$\textit{Context Precision}\ @\text{K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Number of relevant items in top K}} \tag{3}$$

where K is the total number of chunks in the retrieved contexts, $v_k \in \{0, 1\}$ is the relevance indicator for Chunk-k, and precision@k is the precision for Chunk-k defined as:

$$\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k} + \text{false positives@k}} \tag{4}$$

Context Recall. This metric assesses the completeness of the retrieved context concerning the ground truth answer. It measures whether the retrieved chunks contain all the necessary information required to formulate the ground truth answer.

Together, *Answer Correctness* provides an end-to-end measure of quality, while *Context Precision* and *Context Recall* offer diagnostic insights into the retrieval stage [20, 53]. This allows us to pinpoint bottlenecks, compare different retrieval strategies, and understand the relationship between retrieval quality and final answer generation, particularly when tuning hyperparameters like chunking strategies [14].

4.2.4 Models Under Investigation. To understand the impact of model choice on performance, our experiments utilize a range of both open-source and proprietary models for embedding and generation:

- **Embedding Models:** We investigate four `Snowflake Arctic` embed series models (xs/s/m/l) [37], known for their strong performance on retrieval benchmarks, and `text-embedding-004` [24] from Google as a state-of-the-art proprietary baseline.
- **Generator LLMs:** We explore six models in the `Qwen2.5 Instruct` series (0.5B, 1.5B, 3B, 7B, 14B, 32B) [54]. As a state-of-the-art proprietary baseline, we use Google's `Gemini 2.0 Flash` [23].

This selection allows us to compare performance across different model sizes and architectures, assessing the trade-offs between computational cost and QA quality within our RAG framework.

4.2.5 Evaluation Experiments. We perform controlled experiments to isolate the effects of different system configurations:

(1) **Retrieval Hyperparameter Tuning:** Evaluating the impact of chunk size, overlap ratio, and index type (IVF_FLAT vs. FLAT) on *Context Precision*, *Context Recall*, and *Answer Correctness*.
(2) **Model Scaling Effects:** Measuring how embedding and generator model sizes influence QA performance to identify cost-effective configurations.

Using a synthetic 8.7k QA dataset and the optimal configuration identified, we employ an LLM-based evaluator to conduct large-scale, end-to-end systematic assessments of **AskHPC**'s performance

5 Experiment Results

5.1 End-to-End Evaluation Results

5.1.1 Real-World User Evaluation. Fifteen participants took part in the evaluation study. Across 108 evaluated responses, **AskHPC** achieved a mean score of **4.7/5.0**, substantially outperforming direct queries to Gemini 2.0 Flash, which scored **2.4/5.0**. These results from the ongoing user evaluation demonstrate the clear benefits of our RAG-based approach in a real-world HPC setting.

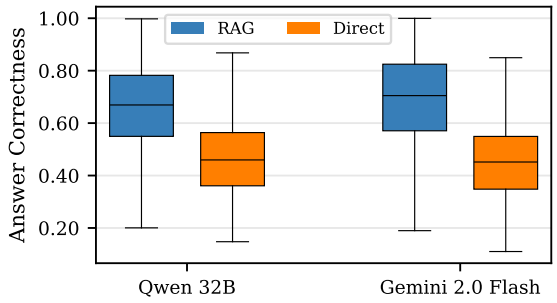


Figure 2: *Answer Correctness* (AC) distribution comparison between RAG and Direct LLM querying on the full 8.7k dataset. Results shown for Gemini (`text-embedding-004` + `Gemini 2.0 Flash`) and Qwen/Snowflake (`snowflake-arctic-embed-l` + `Qwen2.5-32B-Instruct`) configurations using optimal retrieval parameters. RAG demonstrates significantly higher median AC scores and superior overall correctness distributions in both cases.

5.1.2 LLM-as-a-Judge Evaluation. We evaluated our RAG system against direct LLM querying on the full 8.7k QA dataset, using optimal retrieval parameters determined. We measured *Answer Correctness* (AC) for both RAG and direct LLM querying across the dataset. Two configurations, Proprietary (`Gemini`) and Open-Source (`Qwen`), were tested as shown in Table 2. Both configurations use IVF_FLAT indexing for embedding-based retrieval, each paired with its respective optimal chunking parameters identified through hyperparameter search: the former with a chunk size of 4096 characters and 20% overlap, and the latter with a chunk size of 1024 characters and 20% overlap.

Table 2: LLM setup in the two RAG configurations

Conf.	Embedding Model	Generator Model
`Gemini`	`text-embedding-004`	`Gemini 2.0 Flash`
`Qwen`	`snowflake-arctic-embed-l (v2.0)`	`Qwen2.5-32B-Instruct`

Our **AskHPC** system, powered by RAG, demonstrates substantial and consistent improvements in *Answer Correctness* across both proprietary and open-source LLM configurations, as illustrated in Figure 2. Specifically, in the `Gemini` configuration, RAG achieves a median *Answer Correctness* score of 0.70—markedly higher than the 0.45 median from direct querying. The interquartile range (IQR) for RAG spans from 0.57 to 0.82, compared to 0.35 to 0.55 for direct querying, indicating significantly higher-quality responses. Similarly, in the `Qwen` configuration, RAG yields a median *Answer Correctness* of 0.67 (IQR: 0.55–0.78), outperforming the direct query baseline, which has a median of 0.46 (IQR: 0.36–0.56). In both cases, the entire answer correctness distribution under RAG is shifted toward higher values, highlighting its effectiveness in producing more accurate and reliable responses.

AskHPC demonstrates a consistent advantage over direct LLM query in all 25 distinct categories, as illustrated in Table 3. The mean AC score from RAG is consistently above 0.64; whereas the score from direct LLM query is mostly lower than 0.46. In all the categories, RAG shows better results for more than 80% of the questions.

AskHPC provides the most substantial relative improvements in categories demanding precise, facility-specific factual information. Topics such as "Account Creation", "System Access and Authentication", "File Transfer and Storage", and "User Policies and Procedures" show the largest gains. Conversely, the relative advantage of RAG, while still significant, is less pronounced for categories involving broader and more general computational concepts, such as "Code Optimization and Performance Tuning" and "Parallel Programming and Debugging". In these areas, the base LLM's parametric knowledge, drawn from a general training corpus, might already contain partial information needed for the response, thus narrowing the performance gap relative to RAG. These findings again highlight RAG's strength in grounding answers with documented specifics that direct LLMs often lack or hallucinate.

Overall, this category-level analysis confirms that RAG significantly enhances *Answer Correctness* across the full spectrum of HPC user queries. The benefit is most striking for fact-based, system-specific questions common in operational tasks. Still, the improvement remains considerable, even for more complex but general topics that require more reasoning.

5.2 Performance on Multiline Code Queries

To specifically evaluate **AskHPC**'s ability to handle complex, code-centric queries, we analyzed its performance on a challenging subset of our dataset. This subset consists of 67 queries for which the ground-truth answer is a multiline code snippet of 20 lines or more. This scenario is critical for HPC user support, where providing correct and complete code examples is often necessary.

Table 3: Performance comparison of RAG vs Direct LLM querying across different HPC categories. Columns show QA pair count, AC score for RAG and Direct, and the win rate defined as the fraction of queries where RAG AC > Direct AC. `Gemini Flash 2.0` is used as the Generator LLM in both cases.

Category	#QA	RAG	Direct	Win Rate
Software Usage	4584	0.69	0.46	0.84
Resource Allocation and Management	2445	0.69	0.45	0.84
Parallel Programming and Debugging	2270	0.67	0.48	0.80
Specialized Hardware Utilization	1755	0.68	0.45	0.84
Code Optimization and Perf Tuning	1381	0.64	0.46	0.80
Software Environment Configuration	1001	0.71	0.42	0.88
Compilation and Linking	984	0.68	0.45	0.84
System Architecture and Configuration	917	0.70	0.43	0.85
Job Scheduling and Queuing	912	0.68	0.44	0.83
Job Submission	819	0.69	0.44	0.87
Troubleshooting Common Issues	510	0.71	0.43	0.87
File Transfer and Storage	468	0.72	0.39	0.89
System Access and Authentication	430	0.72	0.40	0.89
Job Scripting	423	0.67	0.43	0.85
User Policies and Procedures	325	0.73	0.41	0.89
System Monitoring and Logging	295	0.67	0.42	0.81
Software Installation	266	0.72	0.42	0.89
Best Practices	217	0.66	0.43	0.84
Data Management and Archiving	205	0.71	0.41	0.86
Job Optimization	195	0.68	0.43	0.82
Data Analysis and Visualization	139	0.70	0.42	0.84
Account Management	132	0.71	0.41	0.88
Advanced Usage	93	0.68	0.45	0.86
Group Account Management	51	0.66	0.44	0.84
Account Creation	38	0.72	0.36	0.95

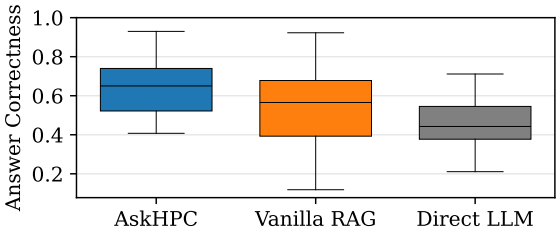


Figure 3: Answer Correctness (AC) for multiline code queries. Results shown for Gemini (`text-embedding-004` + `Gemini 2.0 Flash`)

As shown in Figure 3, **AskHPC** outperforms both vanilla RAG and direct LLM querying for this task. It achieves a mean *Answer Correctness* of 0.64, compared to 0.52 for vanilla RAG and 0.45 for direct querying. The box plot shows higher median performance and an upward shift in the entire interquartile range, indicating consistently better accuracy. **AskHPC** also improves the minimum correctness, reducing the occurrence of low-quality responses seen in vanilla RAG on complex queries.

This improvement stems from **AskHPC** 's core innovations. Modality-aware parsing preserves code block structure, semantic captioning generates rich descriptions, and dual-context assembly provides the LLM with complete, high-quality code snippets. Together, these features enable more accurate and complete answers for complex queries requiring multiline code, where vanilla RAG systems struggle.

5.3 Ablation and Component Configuration Analysis

Optimizing individual components is critical to maximizing RAG performance. We first perform a systematic hyperparameter search on the retrieval subsystem, evaluating the effects of chunking strategies and vector index types using a 250-question subset of the QA dataset. We then analyze how varying the sizes of the embedding and generator LLMs influences end-to-end QA accuracy, enabling us to identify cost–performance trade-offs for practical deployment.

5.3.1 Chunking Strategy. We assessed the impact of chunk size and overlap fraction on key metrics (*Answer Correctness*, *Context Precision*, *Context Recall*), evaluating two distinct model configurations shown in Table 2. Detailed results are presented in Figure 4.

The `Gemini` configuration is characterized by large context windows in both the generator LLM (approx. 4 million characters) and the embedding model (`text-embedding-004`, approx. 8k characters). We observed consistent improvements in *Answer Correctness* as chunk size increased up to 4096. *Context Precision* also increased with larger chunk sizes in this setup, indicating effective processing of larger retrieved contexts.

In contrast, the `Qwen` configuration exhibits markedly different behavior. *Answer Correctness* peaks at a chunk size of 1024 and declines thereafter. *Context Precision* continues to increase up to a chunk size of 2048. The decline in *Answer Correctness* is, therefore, primarily due to the limited context length of the `Qwen` generator LLM (approx. 128k characters), making it less capable of handling large chunk sizes without loss in answer quality. *Context Precision* decreases for chunks larger than 2048, particularly when overlap exceeded 0.2. This decline is linked to the embedding model used (`snowflake-arctic-embed-l` v2.0), which has a maximum context window of 512 tokens (approx. 2048 characters). Chunk sizes exceeding this limit likely lead to degraded embedding quality, negatively impacting precision.

We found that introducing a chunk overlap of 0.2 generally enhanced performance, particularly for *Context Precision*. This suggests that overlap helps maintain semantic continuity across chunk boundaries, mitigating potential information loss. However, the optimal overlap ratio may depend on the specific structure and content of the source documents.

These contrasting results highlight that the optimal chunking strategy is highly dependent on the specific embedding and generator models used, particularly their context-handling capabilities, as well as the semantic structure and content of the documents.

5.3.2 Approximate vs. Exact Search Algorithms. Vector databases commonly use Approximate Nearest Neighbor Search (ANNS) [39] to achieve scalability, often trading off some accuracy compared to Exact Nearest Neighbor Search (ENNS). To assess the validity of using ANNS in our setting, we compared the performance of the Milvus IVF_FLAT index (ANNS) with the FLAT index (ENNS), using the configurations shown in Table 2 and zero chunk overlap to isolate the effects of indexing strategy.

We found that the differences in *Answer Correctness*, *Context Precision* and *Context Recall* between IVF_FLAT and FLAT were minimal for all chunk sizes we tested (results omitted). We thus

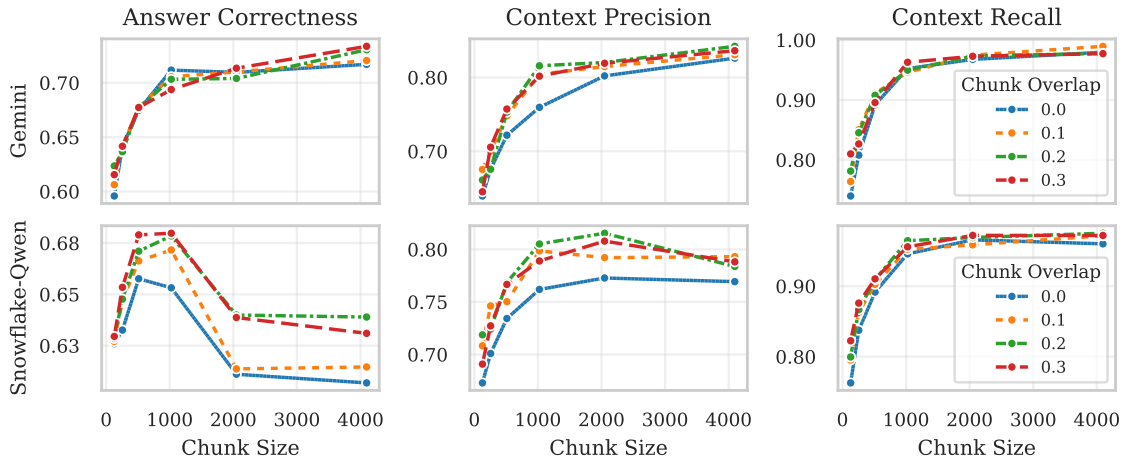


Figure 4: The impact of the chunk size and overlap ratio on the performance of RAG with `Gemini` (upper panel) and `Snowflake-Qwen` (lower panel) Generator LLMs.

choose IVF_FLAT over FLAT for computational efficiency considerations.

Based on the preceding studies, we adopt a chunk size of 1024 characters for `Qwen` models and 4096 characters for `Gemini` models, each with a 0.2 overlap, and utilize the IVF_FLAT ANNS indexing algorithm. These hyperparameters were used consistently in all other stages of our evaluation.

5.3.3 Model Size Study. This study investigates how the size of the embedding and generator models affects the RAG system's *Answer Correctness*, using the optimal retrieval parameters identified for the `Qwen` configuration (chunk size 1024, overlap 0.2, IVF_FLAT index). We evaluated combinations of `Snowflake Arctic` embedding models (`xs`, `s`, `m`, `l`) and `Qwen2.5-Instruct` generator LLMs (0.5B to 32B parameters). Our results show that different embedding models yield similar levels of answer correctness, but the choice of generator model has a significant impact on the final accuracy.

Minimal Impact of Embedding Model Size: We observed that increasing the size of the `Snowflake` Arctic embedding model (from xs to l) yielded only negligible improvements in the final *Answer Correctness* for any given `Qwen` generator LLM as is shown in Table 4. The marginal gains in retrieval metrics (*Context Precision* and Recall) with larger embedding models do not translate into noticeably better synthesized answers. Therefore, the smallest `xs` embedding model (22M parameters) is a highly efficient choice with sufficient retrieval quality.

Noticeable Impact of Generator Model Size: In stark contrast to the embedding models, we found that increasing the size of the `Qwen2.5` generator LLM from 0.5B to 32B results in a consistent noticeable improvement in *Answer Correctness* for all embedding models. Larger generator LLMs appear substantially better equipped to understand the nuances of the query, process the retrieved multimodal context effectively, and synthesize more accurate and complete answers, likely due to their increased reasoning and language processing capacity. The highest *Answer Correctness* (0.68 on the sampled dataset) was achieved with the `Qwen2.5` 32B generator.

The RAG system consistently outperforms direct querying of LLMs across all model size settings. As shown in Figure 5, the RAG approach achieves *Answer Correctness* scores that are 0.1–0.2 points higher than those from direct queries using the corresponding `Qwen2.5` models. Most notably, the RAG system powered by the smallest `Qwen2.5` generator model (0.5B parameters) attains an *Answer Correctness* of approximately 0.46—surpassing the performance of the state-of-the-art Gemini 2.0 Flash model when queried directly (*Answer Correctness*: 0.43). This result underscores the critical role of retrieval augmentation in QA systems, demonstrating that even compact, efficient models can outperform much larger foundation models when paired with effective retrieval.

Table 4: Mean Retrieval Metrics vs. embedding model size on the Sampled QA Dataset

Embedding Model	Precision	Recall
`snowflake-arctic-embed-xs`	0.76	0.94
`snowflake-arctic-embed-s`	0.77	0.93
`snowflake-arctic-embed-m`	0.80	0.96
`snowflake-arctic-embed-l`	0.80	0.97

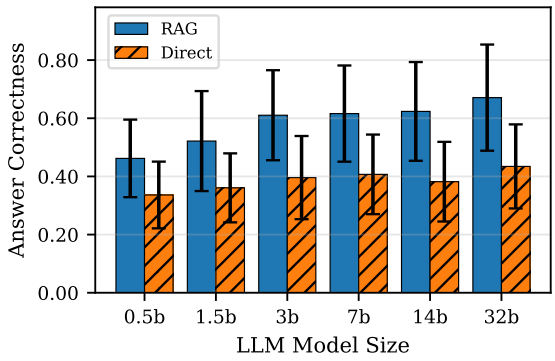


Figure 5: *Answer Correctness* mean and standard deviation comparison: RAG (using `snowflake-arctic-embed-xs`) vs. Direct LLM querying across `Qwen2.5` generator sizes.

6 Discussion

Our extensive evaluation clearly demonstrates that the proposed RAG framework substantially improves the reliability and accuracy of question-answering services for HPC users, outperforming direct queries to state-of-the-art LLMs. These results highlight the essential role of grounding model responses in a carefully curated and structured knowledge base derived from HPC documentation. In the following section, we provide a detailed analysis of the strengths and limitations of **AskHPC**, offering insights into its current capabilities and areas for future enhancement.

6.1 Overall Reliability of AskHPC

The quality of the answers generated is directly related to the *Answer Correctness* (AC) score. Examples of answers with different AC are listed in Table 5. We manually reviewed the quality of a subset of answers and summarized our findings as follows:

- **AC ≈ 0.65 – Correct and Trustworthy**: Generally accurate and aligned with the query, though not always exhaustive or free of minor extraneous content.
- **AC ≈ 0.45 – Partially Correct**: Contains some relevant information, but key details are missing or mixed with irrelevant content, limiting reliability.
- **AC ≈ 0.25 – Insufficient or Irrelevant**: Offers little value, typically off-topic, incomplete, or a refusal to answer due to lack of information.

For direct LLM queries without RAG, the interpretation of AC scores is similar with respect to partial or general correctness. However, very low scores (e.g., < 0.25) often indicate severe hallucinations or fundamentally incorrect information rather than refusal or vagueness based on retrieved context.

These benchmarks illustrate how AC scores correlate with the practical utility and trustworthiness of the generated answers, confirming that scores around 0.65 for RAG represent generally reliable and useful responses, while scores below 0.25 indicate significant issues, manifesting differently in RAG (refusal/irrelevance) versus direct LLM (hallucination). For completeness, scores greater than 0.8 indicate near-perfect alignment with the ground truth.

6.2 Analysis of Underlying Retrieval and Generation Dynamics in AskHPC

The strength of our RAG system stems from its ability to retrieve relevant context from verified documentation, enabling the generator LLM to synthesize accurate answers. Analysis of internal metrics (Figure 6) reveals that the retrieval component consistently achieves high *Context Recall*, indicating that the necessary information is typically found within the knowledge base. However, occasional suboptimal performance, reflected in lower *Answer Correctness* scores, often correlates with lower *Context Precision*. This suggests that the primary challenge is not failing to find relevant information, but rather including excessively noisy or irrelevant context that hinders the generator's ability to pinpoint the precise answer.

A case-by-case scrutinization of the failing cases (AC < 0.25), amounting to between 2-3% of all 8772 cases, confirmed three main causes of failing response: (1) Low precision and low recall due to rare information in the document; (2) High recall but low precision due to excessive noise in the retrieved contexts; and (3) High recall and high precision but still poor response due to scattered answers in multiple chunks, resulting in the Generator LLM failing to summarize the answer. We also found that most of the failing cases for `Gemini` were due to retrieval failure, whereas we saw a greater proportion of generator failures for `Qwen`.

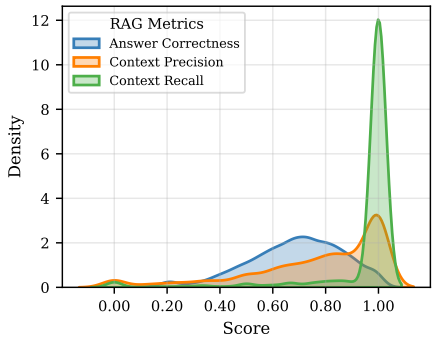


Figure 6: RAG metrics KDE distributions on the full dataset (Gemini). High *Context Recall* with slightly lower *Context Precision* suggests noisy context (low precision), and not missing facts (high recall) is the main factor behind lower Answer Correctness scores.

Addressing these limitations, particularly improving *Context Precision*, is key to further enhancing reliability. Strategies include exploring advanced retrieval techniques like hybrid search, employing re-ranking models to filter retrieved context, and refining chunking strategies or embedding models for better technical nuance capture.

6.3 Limitations and Future Vision

Our future work focuses on addressing the previously identified limitations and expanding the service's utility:

- **Retrieval Enhancement:** We are actively investigating advanced retrieval strategies, including hybrid search (combining dense and sparse embeddings like BM25), cross-encoder re-ranking for improved context precision, and leveraging knowledge graphs extracted from documentation for more structured querying.
- **Community-Driven Improvement:** We plan to release the framework as open-source, incorporating mechanisms for logging failed queries and collecting user feedback on missing information or desired documentation sources. This collaborative approach aims to continually refine and expand the knowledge base and system capabilities, evolving the service into an indispensable resource for the HPC community.
- **Real-Time System Status Data Integration:** We are developing mechanisms to connect the QA service with live system monitoring tools from the facilities, enabling queries about dynamic information like job queues, system load, and user allocations.

7 Related Work

Various HPC-related ChatBots have been developed using different methods. For example, HPC-GPT fine-tunes a LLaMA model on HPC domain data to answer cluster-specific queries and even

Table 5: Examples of responses from AskHPC with different *Answer Correctness* (AC) scores

Query : What environment variables are highly recommended to set when using multiple nodes for PyTorch on Frontier? **Ground Truth** : MASTER_ADDR (head node IP via hostname -i for node communication) and NCCL_SOCKET_IFNAME (set to hsn0 for optimal NCCL inter-node communication). Crucial for distributed training performance. **RAG Answer** : MASTER_ADDR (host IP via hostname -i for process discovery) and NCCL_SOCKET_IFNAME (set to hsn0 for NCCL interface on high-speed network) [3]. Also recommended: MASTER_PORT=3442 for process sync [1]. **AC Score** : 0.6393 **Interpretation** : Accurately identifies the two key variables (MASTER_ADDR, NCCL_SOCKET_IFNAME) and their importance. Includes MASTER_PORT, which is relevant but not part of the "crucial" set in Ground Truth.	**Query** : Explain the purpose of the WithDeleted flag in sacctmgr and describe how resources are handled when deleted. **Ground Truth** : WithDeleted flag shows previously deleted data. Resources deleted <24h after creation are removed; older ones are marked deleted but kept. Flag shows these marked items. **RAG Answer** : The WithDeleted flag displays info on deleted data [1, 2, 3]. Handling depends: Assoc./Users/WCKeys/Resources deleted <24h removed; else marked deleted [1, 2, 3]. **AC Score** : 0.4999 **Interpretation** : Correctly identifies the flag's purpose and the general mechanism (quick deletion vs. marking). However, it adds unnecessary granularity by resource type, thereby adding excessive detail.	**Query** : Describe the purpose of the triad function in the SYCL code, and explain the operations it performs on the allocated memory. **Ground Truth** : The triad function performs C[idx] = A[idx] + B[idx] * scalar using USM memory allocated via malloc_shared, executed in parallel across sub-devices via SYCL queues, measures time, checks results, and frees memory. **RAG Answer** : I couldn't find the information in the documents provided. For more details, please consult the following resources. (Lists retrieved references) **AC Score** : 0.1890 **Interpretation** : The RAG system explicitly refuses to answer due to perceived lack of context, even if information might be present in retrieved documents. This refusal provides no value and is characteristic of scores below 0.25.

detect parallel programming issues [16]. Similarly, chatHPC, an end-to-end pipeline aligns LLMs with HPC tasks to provide an interactive assistant for system questions and job script generation [61]. Other efforts have blended AI with existing HPC interfaces: a conversational interface called "hey-CAI" helps run and profile HPC applications via natural dialogue [29]. These projects highlight a clear trend toward specialized QA solutions in HPC, combining domain-specific data and interfaces to help users navigate supercomputing resources and troubleshoot issues more effectively. However, they focus more on using conversational interfaces to help specific HPC tasks such as software development and workflow management, while ours targets general HPC user support. In addition, they typically use continual pretraining and fine-tuning rather than RAG.

Recent efforts have explored the use of LLMs with RAG architectures to improve HPC user support. Biggs et al. develop an HPC assistant based on a RAG pipeline constructed from documentation specific to the Idaho National Laboratory's HPC environment [13]. Similarly, HyCE, an extension of the RAG paradigm, incorporates dynamic, user-specific information retrieved from live HPC systems, such as current job queue status or available computational resources, to generate more context-aware responses [38]. Both systems share a similar motivation to **AskHPC** in addressing the challenge of fragmented and complex HPC documentation through intelligent, LLM-powered interfaces. Our **AskHPC** system distinguishes itself from prior work not only in the scope and specificity of the underlying knowledge bases but also in the design of its RAG infrastructure. While previous implementations rely on standard, "vanilla" RAG pipelines with limited retrieval and reasoning capabilities, **AskHPC** introduces several key innovations. These include multimodal data indexing, modality-aware segmentation of content types (e.g., text, code, tables, and diagrams), and query enhancement techniques that significantly improve both retrieval accuracy and answer generation. Additionally, unlike earlier efforts, **AskHPC** is accompanied by a rigorous and comprehensive evaluation framework. To the best of our knowledge, it is the first systematically validated RAG-based QA system purpose-built for HPC, demonstrating architectural advances alongside measurable gains in response correctness and transparency.

It is worth mentioning that several general-purpose question-answering services, such as ChatGPT's Browse capability through Search [48] and Deep Research [49], Perplexity AI Deep Research [52], and Google Gemini [23], aim to provide users with up-to-date information by accessing publicly available web resources, similarly addressing the information gap and hallucination issue. They differ from our work in the following key aspects. Firstly, their knowledge sources are primarily the open web, which may contain unverified or outdated information. In contrast, our **AskHPC** system is built upon a curated knowledge base of official, well-controlled documentation from HPC facilities and vendors, ensuring guaranteed reliability and accuracy within the HPC domain. Secondly, these general-purpose services inherently cannot access data residing exclusively on private or local networks, whereas **AskHPC** can be configured to conditionally retrieve information from such internal sources including real-time cluster queue status and user account info. The accessibility of these general-purpose tools often comes with licensing costs, which do not align with the open science principles prevalent in many publicly funded HPC facilities like those supported by the U.S. Department of Energy.

8 Conclusion

We present a domain-specific Question Answering ChatBot system **AskHPC** for user support powered by LLM within a RAG framework. Extensive evaluations demonstrate our system significantly outperforms direct querying of state-of-the-art LLMs across diverse HPC topics and model scales. Our specialized, multimodal-aware RAG framework represents a significant advancement in building reliable AI support for HPC.

This research delivers a robust solution for trustworthy and efficient information access to the HPC community, reducing operational risks associated with inaccurate guidance. While acknowledging current limitations, such as reliance on static data and occasional retrieval imprecision, future directions include integrating live system information, enhancing retrieval mechanisms through hybrid search and re-ranking, and fostering open-source development.

Acknowledgments

This work is supported in part by the U.S. National Science Foundation under Grant CCF-2350323.

This work used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. We thank the AI/ML team at Argonne Leadership Computing Facility for helpful discussions.

References

[1] 2025. oneAPI Collective Communications Library (oneCCL). https://github.com/uxlfoundation/oneCCL original-date: 2019-09-09T21:57:46Z.

[2] Advanced Micro Devices. 2025. Programming Model - HIP Programming Guide. ROCm Documentation. https://rocm.docs.amd.com/projects/HIP/en/latest/understand/programming_model.html Accessed on 2025-04-13.

[3] Altair Engineering, Inc. 2021. PBS Professional 2021.1.2 User's Guide. https://2021.help.altair.com/2021.1.2/PBS%20Professional/PBSUserGuide2021.1.2.pdf Accessed on March 22, 2025.

[4] Argonne Leadership Computing Facility. 2024. ALCF AI Testbed. https://www.alcf.anl.gov/alcf-ai-testbed Accessed: 2025-04-14.

[5] Argonne Leadership Computing Facility. 2024. ALCF Events. https://www.alcf.anl.gov/events Accessed: 2025-04-14.

[6] Argonne Leadership Computing Facility. 2024. ALCF People. https://www.alcf.anl.gov/about/people Accessed: 2025-04-14.

[7] Argonne Leadership Computing Facility. 2024. ALCF Support Center. https://www.alcf.anl.gov/support-center Accessed: 2025-04-14.

[8] Argonne Leadership Computing Facility. 2024. Aurora Supercomputer. https://www.alcf.anl.gov/aurora Accessed: 2025-04-14.

[9] Argonne Leadership Computing Facility. 2025. Argonne Leadership Computing Facility (ALCF). https://www.alcf.anl.gov/. Accessed: 2025-04-13.

[10] Argonne Leadership Computing Facility (ALCF). Accessed: 2025-03-22. ALCF Systems User Documentation. https://github.com/argonne-lcf/user-guides. Online; Accessed on March 22, 2025.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. ACM, Virtual Event Canada, 610–623. doi:10.1145/3442188.3445922

[12] Jean-Emmanuel Bibault, Benjamin Chaix, Arthur Guillemassé, Sophie Cousin, Alexandre Escande, Morgane Perrin, Arthur Pienkowski, Guillaume Delamon, Pierre Nectoux, and Benoît Brouard. 2019. A chatbot versus physicians to provide information for patients with breast cancer: blind, randomized controlled noninferiority trial. *Journal of medical Internet research* 21, 11 (2019), e15787.

[13] Brandon S Biggs and Kaylee Dalton. [n. d.]. Supplementing HPC Support with a Science Gateway AI Assistant. ([n. d.]).

[14] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X Retrieval: What Retrieval Granularity Should We Use? doi:10.48550/arXiv.2312.06648 arXiv:2312.06648 [cs].

[15] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.

[16] Xianzhong Ding, Le Chen, Murali Emani, Chunhua Liao, Pei-Hung Lin, Tristan Vanderbruggen, Zhen Xie, Alberto E. Cerpa, and Wan Du. 2023. HPC-GPT: Integrating Large Language Model for High-Performance Computing. In *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. 951–960. doi:10.1145/3624062.3624172 arXiv:2311.12833 [cs].

[17] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 150–158.

[18] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Barcelona Spain, 6491–6501. doi:10.1145/3637528.3671470

[19] Asbjørn Følstad, Cecilie Bertinussen Nordheim, and Cato Alexander Bjørkli. 2018. What makes users trust a chatbot for customer service? An exploratory interview study. In *Internet Science: 5th International Conference, INSCI 2018, St. Petersburg, Russia, October 24–26, 2018, Proceedings 5*. Springer, 194–208.

[20] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. doi:10.48550/arXiv.2312.10997 arXiv:2312.10997 [cs].

[21] Zheng Gong, Kun Zhou, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2022. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5923–5933.

[22] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).

[23] Google. [n. d.]. Gemini 2.0 Flash. Google Cloud Vertex AI Documentation. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash Accessed on April 12, 2025.

[24] Google. 2024. Google text embeddings (text-embedding-004). https://cloud.google.com/vertex-ai/docs/generative-ai/embeddings/get-text-embeddings. Accessed: 2025-04-12.

[25] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964* (2020).

[26] HPC @ LLNL. 2025. LC Training Events. https://hpc.llnl.gov/updates-events/training-events Accessed: 2025-08-11.

[27] Intel Corporation. 2025. Intel oneAPI Programming Guide. https://www.intel.com/content/www/us/en/docs/oneapi/programming-guide/2025-0/overview.html. Version 2025.0; Online; Accessed on March 22, 2025.

[28] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 7969–7992.

[29] Pouya Kousha, Arpan Jain, Ayyappa Kolli, Saisree Miriyala, Prasanna Sainath, Hari Subramoni, Aamir Shafi, and Dhableswar K. Panda. 2022. "Hey CAI" - Conversational AI Enabled User Interface for HPC Tools. In *High Performance Computing*, Ana-Lucia Varbanescu, Abhinav Bhatele, Piotr Luszczek, and Baboulin Marc (Eds.). Springer International Publishing, Cham, 87–108. doi:10.1007/978-3-031-07312-0_5

[30] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 611–626.

[31] Lawrence Livermore National Laboratory. 2024. El Capitan High-Performance Computing. https://www.llnl.gov/news/highlights/el-capitan-high-performance-computing Accessed: 2025-04-14.

[32] Lawrence Livermore National Laboratory. 2025. High Performance Computing at Lawrence Livermore National Laboratory. https://computing.llnl.gov/. Accessed: 2025-04-13.

[33] Lawrence Livermore National Laboratory (LLNL). Accessed: 2025-03-22. LLNL Documentation. https://hpc.llnl.gov/documentation. Online; Accessed on March 22, 2025.

[34] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[35] Fengchen Liu, Jordan Jung, Wei Feinstein, Jeff D'Ambrogia, and Gary Jung. 2024. Aggregated Knowledge Model: Enhancing Domain-Specific QA with Fine-Tuned and Retrieval-Augmented Generation Models. In *Proceedings of the 4th International Conference on AI-ML Systems*. ACM, Baton Rouge Louisiana USA, 1–7. doi:10.1145/3703412.3703434

[36] Livermore Computing. 2025. Livermore Computing: Development Environment Group. https://computing.llnl.gov/livermore-computing/development-environment-group Accessed: 2025-04-14.

[37] Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-Embed: Scalable, Efficient, and Accurate Text Embedding Models. arXiv:2405.05374 [cs.CL] https://arxiv.org/abs/2405.05374

[38] Yusuke Miyashita, Patrick Kin Man Tung, and Johan Barthélemy. 2024. LLM as HPC Expert: Extending RAG Architecture for HPC Data. arXiv:2501.14733 (Dec. 2024). doi:10.48550/arXiv.2501.14733 arXiv:2501.14733 [cs].

[39] Marius Muja and David G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 11 (Nov. 2014), 2227–2240. doi:10.1109/TPAMI.2014.2321376

[40] NVIDIA Corporation. Accessed: 2025-03-22. NVIDIA CUDA C Programming Guide. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. Online; Accessed on March 22, 2025.

[41] Oak Ridge Leadership Computing Facility. 2024. Frontier Supercomputer. https://www.olcf.ornl.gov/frontier/ Accessed: 2025-04-14.

[42] Oak Ridge Leadership Computing Facility. 2024. OLCF Staff Directory. https://www.olcf.ornl.gov/directory/ Accessed: 2025-04-14.

[43] Oak Ridge Leadership Computing Facility. 2024. OLCF Training Archive — OLCF Documentation. https://docs.olcf.ornl.gov/training/training_archive.html

Accessed: 2025-04-14.

[44] Oak Ridge Leadership Computing Facility. 2024. OLCF User Assistance Center. https://docs.olcf.ornl.gov/support/index.html#olcf-user-assistance-center Accessed: 2025-04-14.

[45] Oak Ridge Leadership Computing Facility. 2025. Oak Ridge Leadership Computing Facility (OLCF). https://www.olcf.ornl.gov/. Accessed: 2025-04-13.

[46] Oak Ridge Leadership Computing Facility (OLCF). Accessed: 2025-03-22. Oak Ridge Leadership Computing Facility User Documentation. https://github.com/olcf/olcf-user-docs. Online; Accessed on March 22, 2025.

[47] OpenAI. 2024. GPT-4 Turbo. https://platform.openai.com/docs/models/gpt-4-turbo Accessed: 2025-04-14.

[48] OpenAI. 2024. Introducing ChatGPT Search. https://openai.com/index/introducing-chatgpt-search/ Accessed: 2025-04-14.

[49] OpenAI. 2024. Introducing Deep Research. https://openai.com/index/introducing-deep-research/ Accessed: 2025-04-14.

[50] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[51] Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934* (2023).

[52] Perplexity AI. 2024. Perplexity AI. https://www.perplexity.ai/ Accessed: 2025-04-14.

[53] Fabio Petroni, Federico Siciliano, Fabrizio Silvestri, and Giovanni Trappolini. 2024. IR-RAG @ SIGIR24: Information Retrieval's Role in RAG Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Washington DC USA, 3036–3039. doi:10.1145/3626772.3657984

[54] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 Technical Report. arXiv:2412.15115 [cs.CL] https://arxiv.org/abs/2412.15115

[55] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[56] James Reinders, Ben Ashbaugh, James Brodman, Michael Kinsner, John Pennycook, and Xinmin Tian. 2021. *Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL*. Apress. doi:10.1007/978-1-4842-5574-2

[57] SchedMD. Accessed: 2025-03-22. Slurm Workload Manager Documentation. https://slurm.schedmd.com/. Online; Accessed on March 22, 2025.

[58] Heydar Soudani, Evangelos Kanoulas, and Faegheh Hasibi. 2024. Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. ACM, Tokyo Japan, 12–22. doi:10.1145/3673791.3698415

[59] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data*. ACM, Virtual Event China, 2614–2627. doi:10.1145/3448016.3457550

[60] Rainer Winkler and Matthias Söllner. 2018. Unleashing the potential of chatbots in education: A state-of-the-art analysis. In *Academy of management proceedings*, Vol. 2018. Academy of Management Briarcliff Manor, NY 10510, 15903.

[61] Junqi Yin, Jesse Hines, Emily Herron, Tirthankar Ghosal, Hong Liu, Suzanne Prentice, Vanessa Lama, and Feiyi Wang. 2024. chatHPC: Empowering HPC users with large language models. *The Journal of Supercomputing* 81, 1 (Nov. 2024), 194. doi:10.1007/s11227-024-06637-1

[62] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792* (2023).

[63] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 46595–46623. https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf

[64] Zilliz. 2024. *Milvus Performance Report*. Technical Report. Zilliz. https://zilliz.com/resources/whitepaper/milvus-performance-benchmark Accessed on April 12, 2025.