AI for Quantum at the Quantum Science Center: Towards Building a Connective Layer Across the Quantum Computing Stack

Tirthankar Ghosal*, Sharmin Afrose*, Ethan Seefried*, Nick Stapleton†, Nirajan Banjade‡, Romit Ghosh§, Sangram Deshpande¶, Vicente Leyton-Ortega*, Travis Humble*
*Oak Ridge National Laboratory, Oak Ridge, TN, USA
†University of Tennessee, Knoxville, TN, USA
‡Texas State University, San Marcos, TX, USA
§Rutgers University, New Brunswick, NJ, USA
¶North Carolina State University, Raleigh, NC, USA

***Abstract*—The quantum computing stack spans the literature, software frameworks, error-correction toolchains, hardware, and high-performance computing (HPC) systems, each with its own tools that rarely share context. We believe that artificial intelligence can link these layers. This summary describes work by the Agentic Software Team at the Quantum Science Center (QSC) on curated knowledge retrieval, benchmarks for AI-generated quantum code, agents for quantum programming and error correction, and HPC analytics that we are extending toward quantum–HPC (QHPC) co-execution.**

***Index Terms*—quantum computing, large language models, retrieval-augmented generation, multi-agent systems, benchmarking, HPC**

I. Introduction

A researcher planning an error-correction experiment may read papers in one tool, write Qiskit or Stim code in another, check device calibration on a vendor dashboard, and run classical post-processing on an HPC cluster. Context can be lost at each handoff. Agents built on large language models (LLMs) could carry it across, provided they draw on curated sources, are tested against shared benchmarks, and can call the tools a researcher would use. Our systems address these conditions (Fig. 1); they began as separate projects, and connecting them is ongoing work.

II. Knowledge Infrastructure

Quantum RAG applies retrieval-augmented generation (RAG) [1] to the quantum computing literature. Its ingestion pipeline extracts text, captions figures, and builds a vector index, and the model or provider at each stage can be swapped independently.

This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for the United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan.

Retrieval over raw text chunks can miss the definitions and relationships that experts take for granted. The **Quantum Computing LLMWiki** supplies them. An LLM writes wiki pages from the source PDFs and distills them into short concept entries we call nuggets. Nuggets, pages, and source chunks share one index and are linked in a graph, so a query that matches a nugget expands to the passages behind it, with provenance intact.

III. Evaluation and Benchmarking

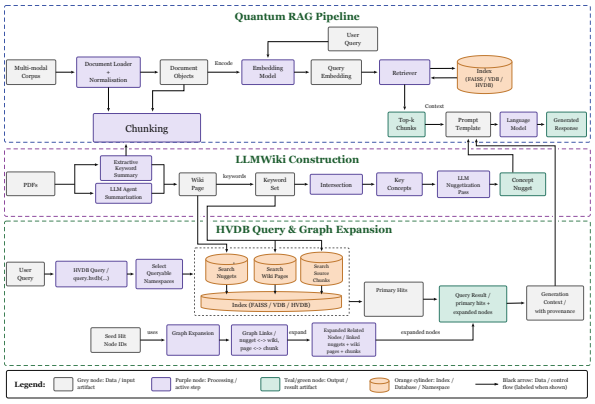
Claims that LLMs can write quantum code are hard to compare without a common test. The **Quantum Computing Leaderboard** evaluates frontier and open-weight models on quantum code generation, using datasets that include Qiskit HumanEval [2]. Automated pipelines check correctness and execution, and per-problem results let us track models over time.

Also, We are preparing open benchmarks that score a complete AI-assisted workflow on optimization, simulation, and graph problems. Entries are judged on solution quality, quantum resource use, reproducibility, and how transparently AI was used.

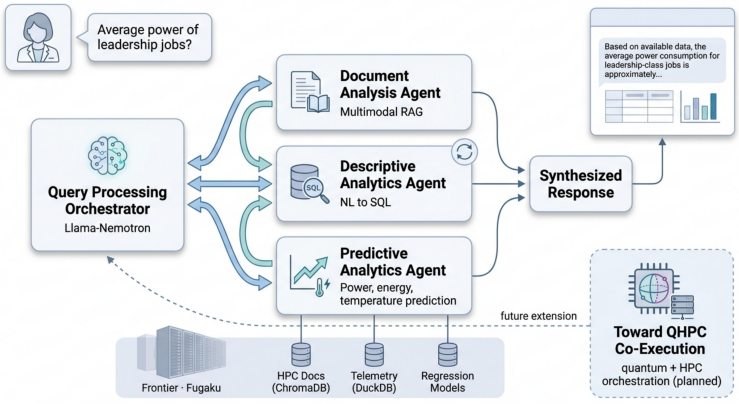
IV. Domain Reasoning and Quantum Systems

OrcheQ is a multi-agent system that, to our knowledge, is the first to cover quantum code generation, circuit optimization, and error correction end to end. An orchestration agent classifies each natural-language request and routes it to a specialist crew for code generation (LLM with RAG), gate optimization (Qiskit, TKET, Cirq), or QEC analysis (Stim [3], PyMatching [4]). A verification agent then applies a four-layer protocol that includes critique by a second LLM, and verified code runs through an MCP server on one of ten backends. Failures at either stage return to the orchestrator. The goal is to let scientists without deep quantum programming experience produce working circuits.

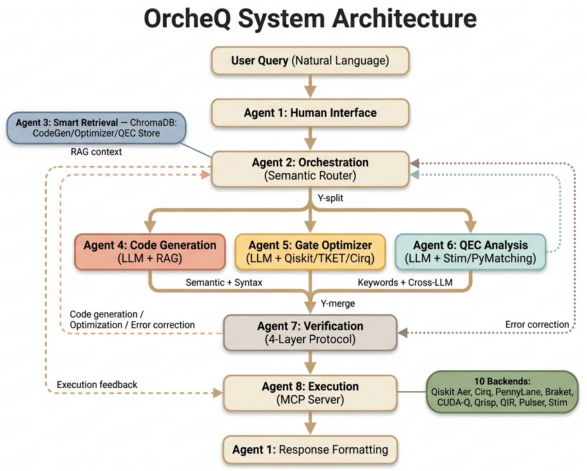
ChatQEC is a RAG assistant for quantum error correction. It retrieves from a curated corpus of roughly 120 sources, a



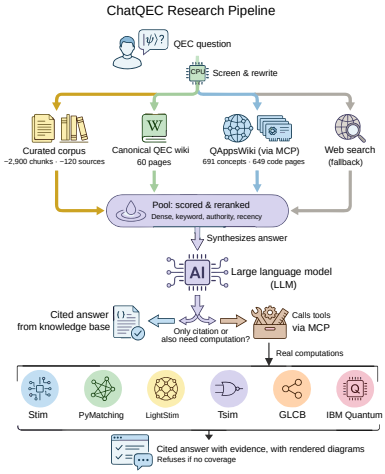
(a) Quantum RAG and LLMWiki



(b) EPIC



(c) OrcheQ



(d) ChatQEC

Fig. 1. System architectures: (a) Quantum RAG with LLMWiki, (b) EPIC, (c) OrcheQ, and (d) ChatQEC.

60-page QEC wiki, and a concept wiki, falls back to web search, and reranks candidates by similarity, keyword match, authority, and recency. When a question needs computation, the LLM calls tools such as Stim and PyMatching and can use live calibration data to place a circuit on specific IBM or IQM qubits. If no source covers the question, ChatQEC says so instead of guessing.

V. HPC Orchestration and QHPC Co-Execution

EPIC [5] replaces batch-oriented HPC analytics pipelines with natural-language queries, such as the average power draw of leadership-class jobs. A Llama-Nemotron orchestrator sends each query to one or more of three fine-tuned agents: a document agent using multimodal RAG over HPC documentation, an NL-to-SQL agent over telemetry, and a predictive agent that estimates power, energy, and temperature. Their outputs are merged into one answer. On SQL generation, the fine-tuned agent scored 87.4% against 69.3% for GPT-4o at 19× lower cost, and the same architecture ran on data from Frontier and Fugaku without modification. We are extending the orchestrator to schedule quantum and classical resources together for QHPC co-execution.

VI. Conclusion

No single system yet spans the path from a question to an executed, verified result, but the pieces fit. LLMWiki and Quantum RAG can supply context to OrcheQ and ChatQEC, the benchmarks can show whether those agents actually improve, and EPIC's orchestrator could dispatch their jobs to quantum and HPC resources. Building these connections is our next step.

Acknowledgment

This material is based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Quantum Science Center.

References

[1] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020, pp. 9459–9474.
[2] S. Vishwakarma *et al.*, "Qiskit HumanEval: An evaluation benchmark for quantum code generative models," in *Proc. IEEE Int. Conf. Quantum Computing and Engineering (QCE)*, 2024.
[3] C. Gidney, "Stim: A fast stabilizer circuit simulator," *Quantum*, vol. 5, p. 497, 2021.

[4] O. Higgott and C. Gidney, "Sparse Blossom: Correcting a million errors per core second with minimum-weight matching," *Quantum*, vol. 9, p. 1600, 2025.
[5] A. M. Karimi, W. Shin, J. Hines, T. Ghosal, N. S. Sattar, and F. Wang, "EPIC: Accelerating iterative analytics in HPC operations with orchestrated multi-agent systems," in *Proc. ISC High Performance*, Hamburg, Germany, Jun. 2026, pp. 1–12, doi: 10.23919/ISC.2026.11520478.