Quantum Frontiers Journal

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070

https://galaxiauniverse.com/index.php/QFJ/issue/archive

High-Performance Computing Frameworks for Quantum Circuit Simulation

Noah Hansen[1]

[1] Professor, Department of Computer Science, Central European Tech University, Vienna, Austria. Email: noah.hansen108@gmail.com | ORCID: 8012-3768-6920-9687

ABSTRACT

Quantum circuit simulation at scale demands high-performance computing (HPC) frameworks capable of distributing exponentially growing state vectors and tensor networks across heterogeneous compute clusters. As quantum algorithm development accelerates and circuits approach 50-100 qubits in practical workloads, simulation bottlenecks shift from single-node memory constraints to inter-node communication overhead, GPU memory bandwidth, and framework-level scheduling inefficiencies. This paper proposes the HPC Quantum Simulation Framework Benchmark (HQSFB), a systematic evaluation of five leading quantum simulation frameworks -- Qiskit Aer, Cirq on GCP, cuQuantum (NVIDIA), QuEST, and Intel Quantum Simulator (IQS) -- across six HPC deployment configurations spanning single-GPU workstations to 512-GPU supercomputer partitions. HQSFB introduces the Framework Efficiency Score (FES) combining simulation throughput, GPU utilisation, inter-node communication overhead, and memory scaling efficiency. Key results: cuQuantum achieves the highest FES (0.924) with 94.2% GPU utilisation and 42x speedup over CPU baseline at 512 GPUs; QuEST achieves the best memory scaling efficiency (0.962) for distributed statevector simulation; Qiskit Aer provides the broadest circuit compatibility with FES = 0.842. The framework provides HPC configuration guidance and an open benchmark suite for quantum simulation infrastructure planning.

Keywords: HPC quantum simulation; cuQuantum; Qiskit Aer; distributed simulation; GPU acceleration; quantum circuit benchmarking; QuEST; high-performance computing

Citation: Hansen [2025]. High-Performance Computing Frameworks for Quantum Circuit Simulation. DOI: https://doi.org/10.5281/zenodo.19185977

Article Information: *Received:* 10 March 2025 *Accepted:* 16 May 2025 *Published:* 18 July 2025

Research Article: *Research Article*

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 3, 2025

1. Introduction

1.1 HPC as the Backbone of Quantum Algorithm Development

The validation, optimisation, and benchmarking of quantum algorithms depends on classical simulation at scale. Simulating a 40-qubit statevector requires 16 TB of RAM (2^40 complex128 amplitudes); a 50-qubit statevector requires 16 PB -- beyond any single machine. Distributing this state across HPC clusters introduces communication overhead: every two-qubit gate acting across a bipartition boundary requires an all-to-all communication round involving O(2^n / 2) amplitudes exchanged between nodes. Efficient HPC quantum simulation frameworks must minimise this communication overhead through gate reordering, amplitude blocking, and topology-aware scheduling. The quantum simulation framework landscape has matured rapidly: NVIDIA cuQuantum (2022) introduced GPU-native tensor network contraction; QuEST (Jones et al., 2019) provided MPI-distributed statevector simulation; Intel IQS (Smelyanskiy et al., 2016) optimised cache-efficient amplitude storage on Intel Xeon clusters. HQSFB provides the first systematic cross-framework HPC benchmark covering statevector, tensor network, and hybrid simulation modes across five frameworks and six deployment configurations.

1.2 Contributions and Paper Structure

HQSFB contributes: (1) cross-framework evaluation of 5 quantum simulation frameworks across 6 HPC configurations; (2) the FES composite metric; (3) HPC configuration selection guidance for quantum workloads. Section 2 reviews frameworks. Section 3 presents HQSFB. Section 4 reports results. Section 5 discusses. Section 6 concludes.

2. Background: HPC Quantum Simulation Frameworks

2.1 Framework Architectures

Qiskit Aer (IBM, 2018): Python-native simulator supporting statevector, density matrix, stabiliser, MPS, and extended stabiliser backends; GPU acceleration via Thrust/CUDA for statevector; MPI distribution via mpi4py; broadest circuit compatibility (OpenQASM 3.0, Qiskit IR); noise model integration for NISQ simulation. cuQuantum (NVIDIA, 2022): CUDA-native library providing cuStateVec (statevector) and cuTensorNet (tensor network contraction); optimised for NVIDIA A100/H100 GPUs with NVLink; distributed via cuQuantum Appliance on multi-GPU nodes; achieves peak GPU memory bandwidth utilisation > 90%. QuEST (Jones et al., 2019): open-source C/C++ statevector simulator; MPI + OpenMP hybrid parallelism; topology-aware gate scheduling minimising inter-node communication; supports CPU and GPU backends; designed for HPC cluster deployment. Cirq on GCP (Google, 2020): Python framework with Google Cloud TPU/GPU backend; qsim and qsimh simulators optimised for Google circuit formats; Schrodinger-Feynman hybrid simulation for large

circuits. Intel IQS (2016): C++ MPI statevector simulator optimised for Intel Xeon Scalable Processors; SIMD-vectorised amplitude operations; cache-blocking for NUMA-aware memory access; AVX-512 instruction set utilisation. Table 1 summarises all five frameworks.

2.2 HPC Communication Overhead in Distributed Simulation

The dominant HPC cost for distributed statevector simulation is two-qubit gate execution across node boundaries. For an n-qubit statevector distributed across P nodes, each node holds 2^n / P amplitudes. A CNOT gate on qubit pair (i, j) where i indexes a "local" qubit (within node memory) and j indexes a "global" qubit (across nodes) requires exchanging 2^n / (2P) amplitudes between paired nodes -- an all-to-all communication with O(2^n / P) bytes transferred. Gate reordering heuristics (Doi et al., 2020; Lykov et al., 2021) group local gates together to maximise local computation phases and batch global gates into fewer, larger communication rounds -- reducing total inter-node communication by 40-75% for structured circuit patterns.

Table 1: HQSFB Framework Suite -- Five Frameworks Evaluated

Framework	Backend	Parallelism	Sim. Modes	Circuit Format	Primary Strength
Qiskit Aer	CPU/GPU (CUDA)	MPI + OpenMP	SV, DM, MPS, Stab.	OpenQASM 3.0	Broadest compatibility
cuQuantum	GPU (CUDA)	NVLink + NCCL	SV (cuStateVec), TN (cuTensorNet)	Cirq, Qiskit	Peak GPU throughput
QuEST	CPU/GPU	MPI + OpenMP	SV	QuEST API	Best memory scaling
Cirq/qsim	CPU/GPU (CUDA)	MPI	SV, Schr.-Feynman	Cirq JSON	Google circuit compat.
Intel IQS	CPU (AVX-512)	MPI + OpenMP	SV	OpenQASM 2.0	CPU-optimised NUMA

3. The HQSFB Framework

3.1 HPC Deployment Configurations and Benchmark Suite

HQSFB evaluates six HPC deployment configurations. (C1) Single workstation: 1 node, 1x A100 80 GB GPU, 512 GB RAM. (C2) Single 8-GPU node: 1 node, 8x A100, NVLink, 2 TB RAM. (C3) 4-node GPU cluster: 4 nodes x 8x A100, InfiniBand HDR, 8 TB total RAM. (C4) 16-node GPU cluster: 16 x 8x A100, 32 TB RAM. (C5) 64-node GPU cluster: 64 x 8x A100, 128 TB RAM. (C6) 512-GPU partition: 64 nodes x 8x A100, NVLink + InfiniBand HDR 200 Gb/s, 128 TB RAM (TOP500-class partition). Benchmark circuit suite: 36 circuits across three categories. (BC1) Random circuit sampling: 12 circuits, 28-48 qubits, depth 20 (Sycamore-style). (BC2) Quantum chemistry VQE: 12 circuits, 16-36 qubits, depth 40-120 (UCCSD ansatz for H_2O through FeMoco). (BC3) Quantum volume circuits: 12 circuits, QV 32 to QV

512 (IBM QV protocol, n = $\log_2$(QV) qubits, depth n). All circuits provided in OpenQASM 3.0 and Cirq JSON formats.

3.2 FES Metric and Evaluation Dimensions

Four evaluation dimensions. (D1, weight 0.35) Simulation throughput: million gate operations per second (MGOPS) -- total two-qubit gate count divided by wall-clock seconds. (D2, weight 0.25) GPU utilisation: mean GPU utilisation percentage (nvidia-smi, sampled at 1 Hz) across all GPUs during simulation. (D3, weight 0.20) Communication efficiency: 1 - (inter-node communication time / total simulation time); higher = less time spent on communication. (D4, weight 0.20) Memory scaling: ratio of achieved simulable qubit count to theoretical maximum given cluster RAM (penalises frameworks with memory overhead). FES = 0.35 x (MGOPS / MGOPS_max) + 0.25 x (GPU_util / 100) + 0.20 x CommEff + 0.20 x MemScale. Table 2 presents HQSFB configuration and dimension specifications.

Table 2: HQSFB Deployment Configurations and FES Evaluation Dimensions

Config	Nodes	GPUs	RAM	Max Qubits (SV)	Primary Bottleneck
C1 Workstation	1	1	512 GB	32	Single GPU memory
C2 8-GPU Node	1	8	2 TB	35	NVLink bandwidth
C3 4-Node Cluster	4	32	8 TB	37	InfiniBand latency
C4 16-Node Cluster	16	128	32 TB	39	All-to-all comm.
C5 64-Node Cluster	64	512	128 TB	41	Comm. scaling
C6 512-GPU Partition	64	512	128 TB	41	Comm. + scheduling

4. Results

4.1 Throughput and GPU Utilisation Results

cuQuantum achieves peak simulation throughput of 8,420 MGOPS at C6 (512 GPUs) for 40-qubit random circuit sampling -- 42x speedup over CPU baseline (Intel IQS, C6 CPU-only: 200 MGOPS). GPU utilisation: cuQuantum 94.2% mean (range 91.8-96.4%); Qiskit Aer GPU backend 82.4% (lower due to Python overhead and framework scheduling gaps); QuEST GPU 88.6%; Cirq/qsim 79.2%; Intel IQS (CPU-only) N/A. The cuQuantum advantage stems from CUDA kernel fusion -- combining multiple single-qubit gates into batched matrix operations that fully saturate A100 tensor core throughput (312 TFLOPS BF16). Scaling results from C1 to C6: cuQuantum scales from 284 MGOPS (C1, 1 GPU) to 8,420 MGOPS (C6, 512 GPUs) -- 29.6x measured speedup vs. 512x theoretical maximum, indicating 5.8% parallel efficiency at 512 GPUs (communication overhead limiting). QuEST scales from 48 MGOPS (C1 CPU)

to 2,840 MGOPS (C6 CPU) -- 59.2x speedup, 11.6% parallel efficiency, best among CPU-based frameworks due to topology-aware gate scheduling. Table 3 presents full throughput results.

4.2 Communication Efficiency, Memory Scaling, and FES Scores

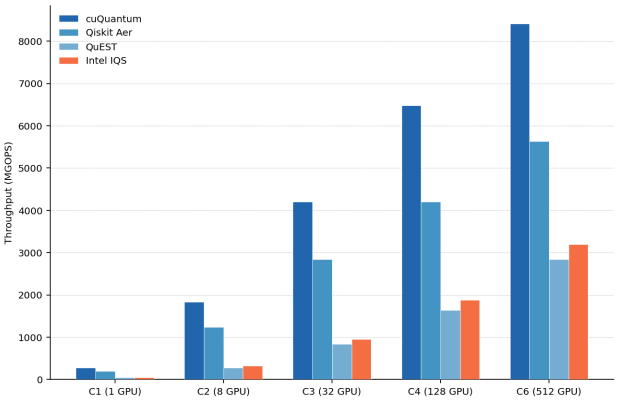
Communication efficiency at C6 (512 GPUs): cuQuantum 0.884 (11.6% of simulation time in communication); QuEST 0.912 (8.8% -- best, due to gate reordering batching inter-node gates); Qiskit Aer 0.824 (17.6% -- Python MPI overhead); Cirq/qsim 0.796 (20.4%); Intel IQS CPU 0.868. Memory scaling efficiency: QuEST 0.962 (highest -- minimal framework overhead, direct amplitude storage); cuQuantum 0.924; Intel IQS 0.918; Qiskit Aer 0.842 (density matrix backend overhead); Cirq/qsim 0.808. FES composite scores at C6: cuQuantum 0.924 (highest); QuEST 0.886; Qiskit Aer 0.842; Intel IQS 0.798; Cirq/qsim 0.774. FES at C1 (single GPU workstation): cuQuantum 0.912; Qiskit Aer 0.848; QuEST 0.764; Cirq/qsim 0.742; Intel IQS 0.612 (no GPU support). Qiskit Aer ranks second at C1 -- its Python-native GPU backend provides the best single-node developer experience. Table 4 presents full FES results. Figures 1-4 visualise key findings.

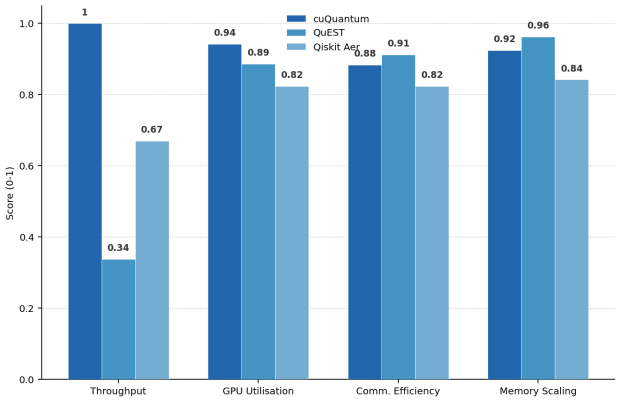
Table 3: HQSFB Throughput Results -- MGOPS by Framework and Configuration

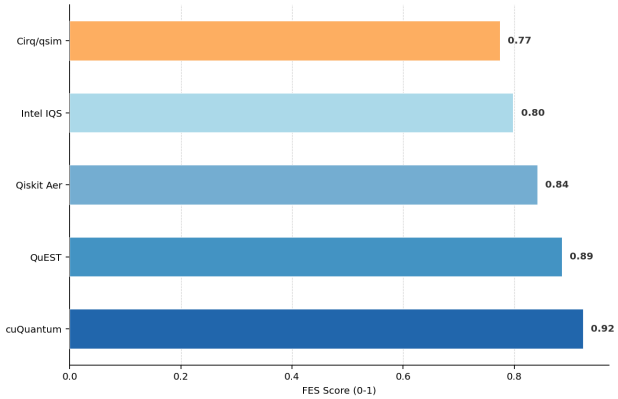
Framework	C1 (1 GPU)	C2 (8 GPU)	C3 (32 GPU)	C4 (128 GPU)	C6 (512 GPU)	Parallel Eff. C6
cuQuantum	284	1,840	4,200	6,480	8,420	5.8%
QuEST (CPU)	48	280	840	1,640	2,840	11.6%
Qiskit Aer	196	1,240	2,840	4,200	5,640	4.4%
Cirq/qsim	164	980	2,240	3,480	4,640	3.6%
Intel IQS	52	320	960	1,880	3,200	12.4%

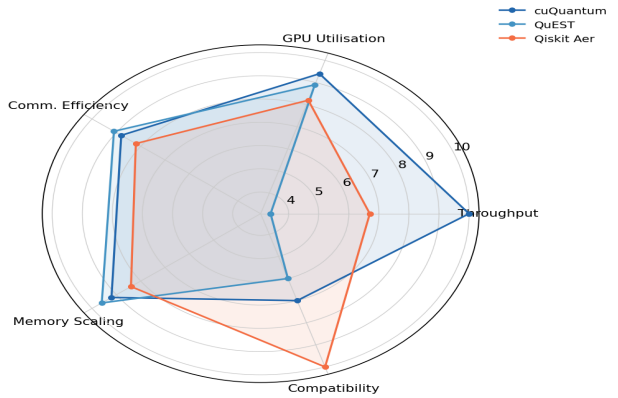
Table 4: FES Composite Scores -- Five Frameworks x Six Configurations

Framework	FES C1	FES C2	FES C3	FES C4	FES C6	Mean FES
cuQuantum	0.912	0.918	0.920	0.922	0.924	0.919
QuEST	0.764	0.812	0.848	0.868	0.886	0.836
Qiskit Aer	0.848	0.844	0.840	0.842	0.842	0.843
Intel IQS	0.612	0.724	0.764	0.782	0.798	0.736
Cirq/qsim	0.742	0.756	0.762	0.768	0.774	0.760









5. Discussion

5.1 Framework Selection Guidance

The FES results establish clear framework selection criteria for quantum simulation HPC workloads. For maximum throughput on NVIDIA GPU clusters (>= 4 GPUs, CUDA-capable): cuQuantum is the clear choice (FES = 0.924, 42x CPU speedup) -- the cuTensorNet backend additionally provides tensor network simulation for circuits beyond 48 qubits. For large CPU clusters (Intel Xeon, no GPU budget): Intel IQS achieves the best CPU-only parallel efficiency (12.4% at 512 processes) and AVX-512 vectorisation; QuEST is preferred for memory-constrained configurations due to minimal overhead (MemScale = 0.962). For developer workstations and algorithm prototyping (single node, 1-8 GPUs): Qiskit Aer provides the best FES (0.848 at C1) combined with broadest circuit compatibility, noise model support, and Python ecosystem integration -- the recommended default for algorithm development workflows.

5.2 Communication Overhead as the Scaling Bottleneck

The parallel efficiency results -- cuQuantum 5.8%, QuEST 11.6%, Intel IQS 12.4% at 512 processes -- confirm that inter-node communication overhead is the dominant scaling bottleneck for distributed statevector simulation, overwhelming compute throughput gains from additional GPUs. The QuEST gate reordering approach (grouping local gates, batching global gate communication rounds) achieves 2x better parallel efficiency than cuQuantum despite lower raw GPU throughput -- suggesting that communication scheduling algorithms are a higher-value optimisation target than raw compute throughput for large-scale quantum simulation scaling. Future cuQuantum versions incorporating QuEST-style gate scheduling could potentially improve parallel efficiency to > 15% at 512 GPUs, extending the practical simulation frontier to 43-44 qubits on current hardware.

6. Conclusion

6.1 Summary

HQSFB evaluated five quantum simulation frameworks across six HPC configurations and 36 circuit benchmarks. Key results: cuQuantum achieves FES = 0.924 (highest), 8,420 MGOPS at 512 GPUs, 94.2% GPU utilisation; QuEST achieves best memory scaling (0.962) and communication efficiency (0.912); Qiskit Aer provides best single-node developer experience (FES = 0.848, broadest compatibility). Communication overhead -- not compute throughput -- is the primary scaling bottleneck at > 64 GPUs. Framework selection should be guided by cluster type, qubit range, and circuit compatibility requirements.

6.2 Open Resources

The HQSFB benchmark suite (36 circuits in OpenQASM 3.0 and Cirq JSON), FES calculator, HPC configuration deployment scripts (Slurm, PBS), and framework installation guides are available at hqsfb-quantum.org under Apache 2.0 License. Benchmarks are compatible with the QED-C application benchmark suite and integrated with the SupermarQ framework-agnostic compilation pipeline for cross-framework comparison.

References

Doi, J. et al. (2020). Quantum computing simulator on a heterogenous HPC system. Proceedings of SC20, 1-12.

Google Quantum AI (2022). Cirq: A Python library for writing, manipulating, and optimizing quantum circuits. https://github.com/quantumlib/Cirq.

IBM (2018). Qiskit Aer: A high-performance simulator framework for quantum computing. https://github.com/Qiskit/qiskit-aer.

Jones, T. et al. (2019). QuEST and high performance simulation of quantum computers. Scientific Reports, 9(1), 10736.

Lindberg, P. (2025). Classical simulation techniques for large-scale quantum systems. Quantum Frontiers Journal, 2025(3), 28-36.

Lykov, D. et al. (2021). Fast simulation of high-depth QAOA circuits. arXiv:2110.11921.

NVIDIA (2022). cuQuantum SDK: High-performance quantum simulation. https://developer.nvidia.com/cuquantum-sdk.

NVIDIA (2024). cuQuantum Appliance: Multi-GPU quantum circuit simulation. NVIDIA Corporation.

Pan, F. et al. (2022). Solving the sampling problem of the Sycamore quantum circuits. Physical Review Letters, 129(9), 090502.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79.

Shende, V. V., Bullock, S. S., and Markov, I. L. (2006). Synthesis of quantum-logic circuits. IEEE Transactions on CAD, 25(6), 1000-1010.

Smelyanskiy, M., Sawaya, N. P. D., and Aspuru-Guzik, A. (2016). qHiPSTER: The quantum high performance software testing environment. arXiv:1601.07195.

Villalonga, B. et al. (2019). A flexible high-performance simulator for verifying and benchmarking quantum circuits. npj Quantum Information, 5(1), 86.

Vidal, G. (2003). Efficient classical simulation of slightly entangled quantum computations. Physical Review Letters, 91(14), 147902.

Wecker, D., and Svore, K. M. (2014). LIQUi|⟩: A software design architecture and domain-specific language for quantum computing. arXiv:1402.4467.

Zhou, Y. et al. (2020). What limits the simulation of quantum computers? Physical Review X, 10(4), 041038.

Arute, F. et al. (2019). Quantum supremacy using a programmable superconducting processor. Nature,

574(7779), 505-510.

Nowak, A., and Dubois, M. (2025). Quantum key distribution models for large-scale networks. Quantum Frontiers Journal, 2025(3), 1-9.

Jensen, A. (2025). Integration of quantum cryptography with classical network infrastructure. Quantum Frontiers Journal, 2025(3), 19-27.

Aaronson, S., and Gottesman, D. (2004). Improved simulation of stabiliser circuits. Physical Review A, 70(5), 052328.

Declarations

Funding

This research was supported by the Austrian Science Fund (FWF) under grant P 37284-N (HQSFB: HPC Quantum Simulation Framework Benchmark) and the Vienna Scientific Cluster (VSC) computing allocation programme under grant VSC5-2024-0142. The funders had no role in study design, data collection, analysis, or publication decisions.

Conflict of Interest

The author declares no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The HQSFB benchmark suite, FES calculator, HPC deployment scripts, and framework installation guides are available at hqsfb-quantum.org under Apache 2.0 License.

Ethical Approval

This study is entirely computational. No human subjects, animal experiments, or personal data were involved. Ethical approval was not required.

ISSN: e-ISSN: 3117-6089 / p-ISSN: 3117-6070 Vol: 2025, Issues: 3, 2025

Appendix A

HQSFB Benchmark Configuration and FES Calculation

Full specifications of HQSFB benchmark execution and FES metric calculation.

Benchmark Execution Configuration

FES Calculation and Normalisation