#129 ScaleQsim: Highly Scalable Quantum Circuit Simulation Framework for Exascale HPC Systems

Email notification
Select to receive email on updates to reviews and comments.

Shepherd
Sanidhay Bhambay

▼ PC conflicts
Rhonda Righter

Accepted

Final version (4.5MB) Oct 14, 2025, 2:43:12 PM UTC · 38d81420

Submission version

▶ Abstract

Large-scale quantum circuit simulation on high-performance computing (HPC) systems is crucial for developing and verifying quantum algorithms to overcome the limitations of current noisy quantum computers. However, existing simulators face scalability bottlenecks due to memory limits and communication overhead. Many state-of-the-art approaches rely on static circuit partitioning, which makes it difficult to address the exponential growth in problem size as the number of qubits increases. To overcome these challenges, we present ScaleQsim, a highly scalable quantum circuit simulation framework for large-scale HPC systems

[more]

▶ Authors

C. Kim, E. Sohn, S. Kim, A. Sim, K. Wu, H. Tang, Y. Son, S. Kim [details]

Use of generative AI : details

We use LLM to enhance readability by focusing mainly on correcting grammatical errors and typos.

✓ Disclaimer (on the use of generative language models)

Experimental result reproducibility

We will release the code as open-source via GitHub. and also consider Zenodo.

▶ Topics and options

	OveMer	RevExp
Review #129A	3	3
Review #129B	4	4
Review #129C	4	3

5 Comments: S. Bhambay (2), ✪S. Kim, C. Kim (2)

You are an **author** of this submission.

Edit submission Add comment

Reviews and comments in plain text

Review #129A

Overall merit

3. Weak accept

Reviewer expertise

3. I know the material, but am not an expert

Paper summary

This paper proposes an exact state-vector based quantum circuit simulator optimized for use on GPU clusters. The core ideas are: two-phase state-vector partitioning and precomputation of metadata to localize memory access and reduce synchronization overhead. Further, it applies adaptive kernel configuration to adjust execution parameters depending on problem size and available memory. The simulator is open-source and could run exact simulations of quantum circuits of size up to 42 qubits using up to 512 GPUs. The simulator has been shown to outperform leading existing simulators by a factor up to 77.40 across different benchmark circuits.

Reasons to accept

+ Proposes a simple but effective way to keep computation local through inter/intra-partitioning with replicated metadata.
+ Performs exact QFT simulations up to 42 qubits with weak and strong scaling analysis.
+ Outperforms well known QC simulators.
+ The proposed simulator is open-source.

Reasons not to accept

– The authors note that Atlas needs a precompiled plan and argue that including plan time would reduce Atlas's advantage. The authors should quantify this overhead along with any analogous overheads in their own simulator.
– The set of benchmark quantum circuits used for evaluation are limited.

Comments for authors

I think the paper makes a clear and valuable contribution. The authors focus on exact state-vector simulation, where memory and communication overhead are the roadblocks. While I feel like the two-phase partitioning, precomputed metadata (Target Indices/StateMapper) and adaptive kernel configuration are not major departures from existing simulation approaches, but they are smart choices that line up well with the strengths of large GPU clusters.

It would be interesting to investigate whether the state-partitioning and memory optimization techniques proposed here can be extended to simulate noisy quantum circuits of NISQ era (through density matrix partitioning). Also, an ablation study of two-phase partitioning, precomputation of metadata and adaptive kernel would make it clear how much each component contributes to the final speedups.

Review #129B

Overall merit

4. Accept

Reviewer expertise

4. I know a lot about this area

Paper summary

The paper introduces and evaluates ScaleQSim, a classical quantum-circuit simulation framework that deploys several techniques, such as adaptive kernel tuning with precomputed index management, to reduce the overall simulation time. In order to support exact simulation of the output state vector, ScaleQSim employs a two-phase intra-node and inter-node state vector partitioning strategy that evenly distributes the state vector across multiple nodes and GPUs. ScaleQSim is implemented by extending Google's QSim framework with 600 lines of new code in the core CUDA modules. Evaluated on six quantum benchmarks of up to 42 qubits on a GPU cluster of up to 512 GPUs (housed on 128 server nodes), ScaleQSim is demonstrated to achieve up to 77x speedup over the state-of-the-art simulators such as cusvaer, HyQuas, and Atlas.

Reasons to accept

The paper proposes an elegant quantum-circuit simulation solution that beats many of the state-of-the-art simulation frameworks.

The simulator is an exact state vector simulator, that is, no approximation or reduced density compressions are involved in the simulation procedure.

Due to largely localized execution and reduced communication protocols employed by ScaleQSim, the results are surprisingly stable even in shared HPC environments.

Due to the authors' access to supercomputing resources, they are able to show positive results with even 42-qubit quantum circuits running on 512 A100 GPUs.

The codebase and experimental implementation of ScaleQSim are already open-sourced for the research and development communities.

Reasons not to accept

The paper does not evaluate and validate whether the simulator is outputting correct state vectors or not.

The results related to the temporal ScaleQSim simulator overhead beyond the simulation time are potentially missing.

The work is somewhat limited in the number and types of quantum circuits that are simulated in the evaluation sections.

Comments for authors

Thank you for submitting this work to SIGMETRICS. I really enjoyed reading it, and I had many a moment of “Why didn’t anyone think of this before?” while reading it. Of course, that’s what makes a good technical solution — something that is lightweight, low-overhead, elegant, and intuitively simple. I think ScaleQSim is successful at achieving this feat with very positive results. The paper evaluates all expected state-of-the-art simulator frameworks that are based on products from major providers (e.g., IBM and NVIDIA) and shows that ScaleQSim largely outperforms them in most cases. The performance improvement of ScaleQSim even makes it possible to run larger circuits on a given allocation of GPU nodes, which other simulators may not be able to accommodate. ScaleQSim can achieve all of this while ensuring that no approximation or lossy compression techniques are used — maintaining the exactness of the output state vector.

I am also positively surprised by the variability figures provided in Fig. 13 and Sec. 4.7. Such low variability is welcome in a shared HPC setting. I was curious as to the GPU setup that was used for the particular experiments shown here. I would expect that as more GPUs are employed (e.g., increasing from 2 GPUs to your largest experiment of 512 GPUs), the variability (not just raw numbers but also normalized by the mean execution time) would increase. This is not due to the communication and I/O contention from other traffic, but also the contention caused between the threads within the simulation program. Have you noticed this behavior as well? How well is the stability maintained with an increase in GPU count? It is appreciable that you have open-sourced the simulator codebase, allowing researchers and practitioners to run it on their local systems and derive insights about performance gains and variations.

Having said all this, as described below, there are a few aspects that, if addressed, could help elevate this work. Firstly, even though this work performs an exact simulation of the output state vector, it is important to perform a validation evaluation regarding the correctness of the simulator for completeness. One metric that could be employed for this is a pairwise inner product of the output vectors generated by ScaleQSim and an existing state-of-the-art simulator that is known to produce the correct output. Because the simulators, provided by Qiskit and cuQuantum are widely used, their output is broadly regarded as being correct. They would be appropriate baselines for validating the correctness of the ScaleQSim simulator output.

Secondly, in regard to the weak scaling results presented in Fig. 8, it is mentioned that while Atlas performs better than ScaleQSim in some cases, Atlas would have to produce its execution plan prior to execution, which would add to its runtime. This made me think that there must be some amount of classical beyond-simulation temporal overhead associated with ScaleQSim as well, right? The inter- and intra-partitioning, index precomputing, gate aggregation into tasks, state generation, etc., aspects must have their own overheads as well. Is this time included in the presented results for ScaleQsim? Is this part of the “Initialization” chunk in Fig. 12? If not, it would be good to have these results included and compared in the evaluation.

Thirdly, I would also say that the evaluated circuits, while they have a large qubit count, are not very deep (i.e., their gate count is not very large). Something like a Variational Quantum Eigensolver (VQE) circuit with the UCCSD ansatz might provide a high-depth circuit that could potentially help highlight the gains of ScaleQSim even further. On that note, it would be better to evaluate the work using more than six benchmarks to get a more general view of the simulator's performance; although, this is a minor weakness.

Regarding the selection of the benchmarks, it appears to me that all six of the evaluated circuits are continuous circuits from front to back with measurements at the end; that is, none of them have reset operations or conditional gates or mid-circuit measurements. How would the simulator deal with cases like these? For instance, conditional if-statement-based gates might impact the gate aggregation into tasks and how these tasks can be executed, serializing some of them to resolve prior dependencies for the condition-based branching.

Lastly, there is a minor typo in the legend of Fig. 12 on page 19. One of the legend handle labels says “Calulate Target indices”. It should be “Calculate Target Indices”.

Review #129C

Overall merit

4. Accept

Reviewer expertise

3. I know the material, but am not an expert

Paper summary

This work introduces a new, scalable, and distributed method, "ScaleQsim", for performing quantum simulations on high-performance computing systems. ScaleQsim utilises full-scale vector simulation to execute a dynamic plan during runtime. ScaleQsim has three major components: (i) two-phase partitioning of the full state vector across nodes and then GPUs, (ii) precomputation and broadcast of per-gate target indices in a statespace structure so that each GPU can operate independently, and (iii) adaptive kernel parameter tuning to match workload and memory. Simulations show up to 42 qubits on 512 GPUs and speedups up to 77.4× over state-of-the-art baselines (Qsim, Atlas, HyQuas).

Reasons to accept

1) Authors come up with an efficient way of using a full state vector simulation for a quantum circuit, which gives accurate results for the output of the circuit. This approach is crucial for validating quantum algorithms, verifying hardware behaviour, and benchmarking performance.

2) The way ScaleQsim do the partitioning of the sequence of gate operations required to execute a quantum circuit first across nodes and then within each node across GPUs avoids static, precompiled circuit plans used by proprietary works. This partitioning improved the adaptability and scalability of running large quantum circuits with a larger number of qubits.

3) predistribution of target indices, which will provide the information on which qubits will be affected by the application of the gate. The target indices are stored in the state space metadata and broadcast to all nodes before execution. The predistribution of metadata to all nodes significantly reduces the communication overhead between nodes. Moreover, the overhead index generation is small, with a duration of approximately 0.58 seconds and a size of 54.81 KB at 36 qubits.

4) The numerical results comparing ScaleQsim with other state-of-the-art techniques for finite quantum algorithms are strong. Clearly, highlighting the advantages of using ScaleQsim compared to others in terms of scalability and simulation time for large quantum circuits.

5) Authors introduced an Adaptive kernel parameter adjustment algorithm that dynamically adjusts the number of CUDA blocks based on the gate size and the portion of the state vector assigned to each GPU. This adaptive scheme is crucial for managing memory and GPU mismatching.

Reasons not to accept

1) The ScaleQsim approach relies on the global metadata broadcast across nodes with GPUs. As the system scales with more nodes, the communication overhead of distributing metadata across nodes will increase. In this work, the simulations are presented with 40 qubit state information distributed across 128 GPUs. However, for a fault-tolerant task, we need qubits of the order of 100s or even thousands, which will increase the number of GPUs and the communication overhead required to broadcast metadata.

2) In quantum algorithms like QFT, the Atlas gives a lower simulation time as compared to the ScaleQsim (see Fig. 8 (a)) for 36 qubits. Even for ghz, the simulation time for both Atlas and ScaleQsim is comparable. It is not clear whether ScaleQsim saves simulation time for a larger class of quantum algorithms. I think it is better to check the applicability of ScaleQsim for more quantum algorithms, such as QEC encoding circuits.

3) The benchmarking algorithms considered in this work are small.

4) Authors have given the link to the GitHub repo, which is not anonymous.

Comments for authors

1) It's better to have a separate section, probably after the ScaleQsim design section, to clearly outline the significant differences that ScaleQsim has from other approaches like Cusvaer, HyQuas, and Atlas.

2) All reported results focus on performance like simulation time, speedup, scaling, time breakdown, and time variability across Figures 1 and 7–13; there are no results on output state fidelity, nor any gate-wise error accumulation analysis.

3) In addition, it is assumed that the full state-vector simulation is exact, but it does not validate whether there are errors in the state vector itself.

4) Overall, the paper is very well written, and the ScaleQsim architecture is clearly explained with all components.

@A1 Sanidhay Bhambay (Shepherd) · Sep 26

Congratulations on the acceptance of your paper. The reviews are overall positive, but it would be good to incorporate some of the suggestions listed below.

1) It is assumed that the full state-vector simulation is exact, but it does not validate whether there are errors in the state vector itself—some comments or discussion on this.

2) All six benchmark circuits are continuous circuits from front to back with measurements at the end; that is, none of them have reset operations or conditional gates or mid-circuit measurements. How would the simulator deal with cases like these? For instance, conditional if-statement-based gates might impact the gate aggregation into tasks and how these tasks can be executed, serialising some of them to resolve prior dependencies for the condition-based branching. (comment on this)

3) In Fig. 8, you note that Atlas would pay plan-generation time; ScaleQsim also has classical setup overheads (partitioning, index precompute, task aggregation, state init). Are these costs included in your reported times, for example, under “Initialisation” in Fig. 12? If not, please report end-to-end runtimes, including them for a fair comparison.

4) Also address the minor comments from each reviewer.

@A2 Sunggon Kim (Author) · Oct 3

Dear Sanidhay Bhambay,

We sincerely thank the reviewers and the shepherd for their thoughtful and valuable feedback on our submission. Their valuable comments have provided us with important insights into both the strengths and areas for improvement in our work. Based on these suggestions, we have carefully prepared the following revision plan.

The following is the revision plan.

1. State-Vector Fidelity Verification : We will measure between the final state vectors produced by the original Qsim and ScaleQsim to verify numerical accuracy and include the results in the Evaluation section.

2. Time Breakdown Refinement (Figure 12) : We believe that a more detailed breakdown can better reveal potential sources of runtime overhead. Although GPU memory initialization, allocation, and target index precomputation are already included, we will further decompose the “Others” category to separately report the time spent on state initialization (partitioning), task aggregation, and so on.

3. Addition of Variational Quantum Eigensolver (VQE) Circuit

: Since our current benchmark circuits have large qubit counts but relatively shallow depth, we will incorporate a VQE circuit to increase circuit diversity and provide additional experimental results in the Evaluation section.

4. Performance Variability with Increasing GPU Count : The current results in Section 4.7 (Performance Variability and Stability) are based on a fixed number of GPUs. To reflect Reviewer 2’s feedback, we will conduct additional experiments to examine whether performance variability remains stable as the number of GPUs increases and include the results in the paper.

5. Fix general typos.

The revision plan I have prepared so far is as follows. For the two items listed below, revisions are difficult to address due to time and development scopes, so I have prepared responses to address them instead.

---

1. Analysis of time overhead caused by Atlas's plan generation

: We recognize that this plan-generation step introduces non-negligible overhead. So, to account for the overhead introduced by Atlas’s plan-generation phase, we directly contacted the authors and received guidance on how to reproduce their setup. The authors confirmed that their experiments were conducted on the Perlmutter, and we also attempted to replicate the same process in a Perlmutter. However, the plan generation step did not function as expected, and we are still encountering significant difficulties in reproducing it successfully.

Although we requested additional details, the information provided was insufficient to fully replicate the procedure. We will continue to follow the authors’ instructions as closely as possible, but if the plan generation cannot be completed, including this step directly in our measurement will not be feasible. Nevertheless, we will explicitly state in the paper that such plan generation is an inherent part of Atlas’s workflow and introduces non-negligible overhead.

2. All six benchmark circuits are continuous circuits from front to back with measurements at the end; that is, none of them have reset operations or conditional gates or mid-circuit measurements. How would the simulator deal with cases like these? For instance, conditional if-statement-based gates might impact the gate aggregation into tasks and how these tasks can be executed, serialising some of them to resolve prior dependencies for the condition-based branching. (comment on this)

: ScaleQsim is built upon Google's Qsim, which is designed as a state-vector simulator optimized for static quantum circuits that execute from start to end with measurements performed only at the final stage. This design enables aggressive gate fusion and task-level parallelism, allowing efficient execution of the entire circuit as a single computational pipeline. Consistent with this design choice, the current version of ScaleQsim also does not fully support dynamic circuits involving mid-circuit measurements, conditional branching, or reset operations. Consequently, all benchmark circuits in ScaleQsim are restricted to continuous circuits with measurements at the end.

Please do not hesitate to share any further thoughts. We truly appreciate your time in shepherding our paper.

@A3 Changjong Kim (Author) · Oct 10

We sincerely thank the reviewers and the shepherd for their thoughtful feedback on our submission. Their valuable comments have provided us with important insights into both the strengths and areas for improvement in our paper.

We inform you that we have incorporated all of the items in the revision plan, including:

1. State-vector fidelity verification between Qsim and ScaleQsim. (Section 4.7)
2. Time Analysis refinement with a more detailed decomposition of the "Others" category. (Section 4.6)
3. Addition of the VQE circuit to diversify the benchmark circuits. (Section 4.1, 4.4)
4. Additional experiments on performance variability as the number of GPUs increases. (Section 4.8)
5. Typo corrections throughout the paper.
6. The data for cusvaer at 16 GPUs was mistakenly plotted using the 8 GPU results. We identified this issue and have corrected it in the camera-ready version. (Section 4.3 - Strong Scalability)

In addition, regarding the analysis of time overhead caused by Atlas's plan generation, we followed the authors' guidance and carefully reproduced the steps described in their GitHub repository. According to the Atlas GitHub, the plan-generation phase typically takes around 17 minutes on Perlmutter. However, despite repeated attempts over two days, we were unable to complete the plan generation.

Although we directly contacted the authors and received additional guidance, the information provided was not sufficient to fully replicate the procedure. Thus, including this step directly in our measurements was not feasible. Instead, we have added a clear explanation of this overhead to the "Initialization" category in our time analysis and reflected its impact. We also explicitly state in the paper that such plan generation is an inherent part of Atlas's workflow and introduces non-negligible overhead.

Note that, except for the acknowledgment section (which is currently being updated), all planned revisions have been fully integrated into the camera-ready version.

Thank you again for your guidance and support throughout this process.

Best regards, Changjong Kim

_Camera_ready___SIGMETRICS_26__ScaleQsim__Highly_Scalable_Quantum_Circuit_Simulation_Framework_for_Exascale_HPC_Systems__KCJ_ (3).pdf (4.5MB)

@A4 Sanidhay Bhambay (Shepherd) · Oct 14

Thanks for revising the manuscript and addressing the comments. It looks good. Please go ahead and submit the camera-ready version.

@A5 Changjong Kim (Author) · Oct 14

Thank you for your assistance. I will do that.

Thanks Changjong Kim