Optimizing Quantum Computing Simulations: The Role of Swap Memory and Advanced Storage Technologies using QuEST

Ishikka Ladia

Master of Science

School of Informatics

University of Edinburgh

2024

Abstract

This dissertation explores the use of swap memory, particularly SSDs and Compute Express Link (CXL) technology, to enhance quantum computing simulations within the QuEST framework. By benchmarking different configurations, the research reveals that while SSDs extend memory, they increase execution time due to latency. In contrast, CXL significantly improves performance by providing faster access to memory. The findings contribute to optimizing quantum simulations on classical hardware, addressing challenges in scalability and efficiency. Future research should explore alternative storage technologies and advanced memory management strategies to further enhance quantum simulation capabilities.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ishikka Ladia*)

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Antonio Barbalace and Ali Rezeai for their unwavering guidance, insightful feedback, and constant support throughout the course of this research. Your expertise and encouragement have been invaluable.

I am also deeply thankful to the Systems Nuts research team, whose collaboration, resources, and guidance have made this journey both productive and enjoyable. Your contributions have been essential to the success of this thesis.

Thank you all for your support and belief in this work.

Table of Contents

Chapter 1

Introduction

Quantum comoputing is an emerging field which has the potential to solve complex problems that classical computing systems find impossible to solve. On the contrary, the implementation of the quantum circuit simulations on classical computing systems is still limited by both hardware and software due to the exponential increase in the amount of hardware that will be required with the addition of each qubit [1, 2]. The aim of this dissertation is to address this challenge by investigating the potential benefits of using swap memory for quantum computing simulations.

This research is primarily directed towards examining the degree to which swap memory, particularly concerning Solid State Drives (SSDs) or with Compute Express Link (CXL), could be beneficial in increasing the efficiency and availability of quantum simulations. This study aims to achieve this by performing a systematic benchmarking of simulations with different configurations on the swap memory with the aim of establishing the tradeoff between adding more memory and performance of the simulation. The analysis will be centered on various performance metrics such as execution time, CPU power utilization, and general performance to determine the best approach of running quantum simulations on standard computers with limited physical memory.

1.1 Motivation

Quantum computing holds the promise to enable solutions to complex problems and hence revolutionize various fields. A significant challenge however remains that the simulation of the quantum circuits on any classical hardware also remains one of the key problems in this area. Such a simulation is often needed to test and design quantum algorithms which should be executed on the real quantum computers. Growing the size

and number of qubits results in increased required memory and computational power in a two-fold nature, hence making it harder and harder to achieve affordable, large scale, and accurate simulations [3].

The disadvantages of engineering components become clearer while working with large quantum systems where the state vector requires extensive memory which grows exponentially as number of qubits grow. Most of such systems are likely to be greater than the virtual available physical random access memory (RAM) in most systems, prompting the need to incorporate swap memory to increase the capabilities of the hardware.

Swap memory works by using disk storage, like SSDs, to expand the working memory for larger workloads. On the other hand this has an implication on performance since it is faster to load data from RAM than from a disk. Improvement of performance with the help of CXL technology may allow the utilization of larger volumes of memory with access speed comparable to RAM and thus eliminate the need to use slower SSD drives as swap memory [4].

However, the use of swap memory in quantum simulations has not been adequately addressed. This explains the motivation of this dissertation which aims to research about the optimal use of swap memory in quantum simulations.

The findings in this paper suggest that SSDs, though improve the memory that can be used, for the simulations, an increase in the time taken to execute the operations is experienced since they are slower when it comes to access time as compared to RAM. Rather, the addition of CXL as a swap memory option, positively changes the performance of the system, through shorter execution time and better utilization of the CPU.

The following is the outline the structure of the dissertation, organized to deliver comprehensive understanding of the factors affect the application of swap memory in quantum computing simulations:

- Chapter 2: Background: This chapter goes ahead to provide the background of the Quantum Exact Simulation Toolkit (QuEST), swap memory, and benchmarking and what is known. It touches on the scope of quantum simulations on classical hardware and the prospect of overcoming this limitation by leveraging storage devices such as SSD and CXL. The chapter also presents a review of the available literature including the SnuQS framework which is among those scarce literature examining the swap memory in quantum simulations.

- Chapter 3: Methodology: This chapter describes the work undertaken and the design and implementation of the experiments. It outlines the configurations performed, the parameters tested, and the complications faced during the simulations.

- Chapter 4. Analysis and Evaluation. This chapter presents the outcomes of the experiments and provides critical analysis of the data collected. It assesses the efficiency of several swap memory configurations, explains what this entails for the future of quantum simulations as well.

- Chapter 5. Conclusion. In the last section of the dissertation, the most important outcomes of the research performed are highlighted, the scope of limitations of the particular study are outlined, and recommendations are made on what future research can be conducted.

Chapter 2

Background

2.1 QuEST Framework

The Quantum Exact Simulation Toolkit (QuEST) is an open-source framework developed for the classical simulation of quantum circuits. With the advancements made in quantum computing, the problems considered within the computational complexity of the quantum circuits steadily deepen, making it increasingly essential to have scalable and effective tools for quantum simulation. QuEST responds to this need by presenting a high-performance simulator that is effective in multi-core and distributive computing settings [5].

QuEST is intended for performing quantum computations on a classical computer through the emulation of qubits and quantum gates. The framework is implemented in C language and is optimized for performance by taking advantage of efficient linear algebra libraries and parallel computing to cope with the exponentially growing state vector of quantum states. One of these is the ability of QuEST to scale across multiple nodes in high-performance computing (HPC) making it more powerful to perform simulations of larger quantum systems than would be feasible on one machine.

The toolkit encompasses support for both single-node (multi-core) and multi-node (distributed memory) simulations, which is important for managing the intensive memory demands of quantum simulations. QuEST also provides both single and double-precision floating-point arithmetic making it flexible to balance between computational performance and accuracy [6].

Key Features of QuEST:

- **Scalability:** QuEST is quite flexible and well-prepared for scaling up on both

single-node and multi-node platforms. For single-node systems, it is implemented with the support of parallel processing and can employ multi-core CPUs. For larger simulations, QuEST can distribute the computing process across several nodes in a cluster, using Message Passing Interface (MPI) to exchange information between the nodes. This is essential where there is a need to carry out simulations of very large quantum circuits that one would otherwise find hard to operate on a conventional desktop computer.

- **High Performance:** The optimization of the performance speed of QuEST is enhanced by applying higher-order libraries of linear algebra for example the use of Intel's Math Kernel Library (MKL) and the OpenBLAS library. These libraries help speed up the matrix multiplications that form the backbone of almost all quantum simulations. QuEST also includes optimizations for low-level architecture such as reducing memory access latencies and maximizing CPU usage.

- **Extensibility:** QuEST has been built such that flexibility is incorporated and therefore users can create their own quantum gates, noise, and collapse models. This enables one to optimize the simulations to the specific requirements of the study where for instance the focus is on how the coherence of the quantum states would influence the quantum algorithms or evaluating the new methods of quantum error correction.

- **User-Friendly Interface:** Besides being aimed at high performance, QuEST has an easy-to-use API that allows users to use a collection of building blocks to create quantum circuits. This makes it usable by those who may be not proficient in parallel computing or high-performance computing (HPC).

- **Support for Noise Models:** Besides the ideal case of simulating ideal quantum circuits, QuEST also has support for noise models which can account for the effects of decoherence and gate errors. This functionality becomes necessary in examining how quantum algorithms withstand stress in practical conditions where qubits are susceptible to different forms of noise and errors.

The reason for choosing QuEST as the simulation platform for this research was based on its performance, scalability, and flexibility. Keeping in mind the objective of quantifying the performance of quantum computing simulations and investigating the application of swap memory, QuEST offers the required level for carrying out

these experiments in a habitable manner. In particular, its capability of handling the exponential expansion of quantum state vectors even in a distributed environment suggests its potential to model advanced quantum circuits that are high in processing power.

Overall, QuEST is one of the state-of-the-art and most complex quantum computer emulators. QuEST's capacity to span multi-core and distributed platforms as well as its user-friendly interface have opened a great opportunity for the objectives of this research. The following sections will delve into how QuEST was used to simulate experiments, particularly for the integration of swap memory and the benchmarking of quantum circuits.

2.2 Swap memory

Swap space or virtual memory is employed in operating systems (OS) to create the illusion of larger physical memory than what is physically present by temporarily moving data that is not being used from the RAM to a certain section of the hard drive, most often referred to as the swap file. The OS will transfer inactive data to swap space if the current physical memory (RAM) is full. This action helps to free RAM to perform more urgent tasks. This allows a system to 'virtualize' its memory and support larger loads than if only physical memory were installed [7, 8]. However, a performance trade-off is often made because of the delay caused by the slower access speeds of disk storage compared to RAM. A simulation that relies heavily on swap memory can experience a substantial slowdown, often referred to as 'thrashing', caused by frequent swapping of data between RAM and disk [9]. This can make simulations impractically slow and may even cause them to fail if the system becomes overwhelmed by the constant swapping.

In relation to quantum computing simulations, the need for swap memory originates in because of the size of the quantum state vectors that have to be stored. The size of the state vector grows exponentially as the number of qubits increase, a quantum system with 'n' qubits has the corresponding state vector of size '2^n' [10, 11, 12]. This quickly goes beyond what physical memory manufacturers have on most systems. For example, a 30 qubit system needs 8GB of memory to store only the state vector, while a 40 qubit system needs 8TB of memory, which is not possible with conventional RAM but is possible with swap memory. Making use of swap memory allows researchers to probe the behavior of quantum systems that are much larger, as well as test more sophisticated

algorithms and probe the boundary of classical simulation of quantum processes.

Quantum simulations, especially those with a large number of qubits, can be quite stressful on the system's memory. While swap memory can extend classical computers' inherent limitations in regards to physical memory, it comes at the cost of increased latency. Hence, the need to understand the trade-offs involved in using the swap memory in important for optimizing the overall performance of the simulations.

2.2.1 SSD as swap memory

In recent times, it has become easy to notice that solid-state drives (SSDs) have taken over a large portion of what was previously reserved for the swap file in hard disk drives (HDDs) because SSDs have the relatively higher data handling efficiency. SSDs contain flash memory for data storage which helps in rapid read and write operations. This implies that they can be used as a supplementary form of RAM, particularly for computational tasks that involve acess to swapped-out data [13, 14].

The benefits of using SSDs as swap memory include:

- Faster Access Times: SSDs are quicker at data retrieval as compared to HDDs, reducing the performance hit when accessing swapped data.
- Reduced Latency: the delay between data requests and delivery is minimized when using SSDs, which is important to maintain the efiicieny of the simulation.
- Enhanced Durability: Modern SSDs are more reliable in demanding computational environments because of their improved durability and lifespan.

However, even with those benefits, the performance of SSD's remains very inadequate when compared to RAM. For that reason, although SSD's may serve to broaden the available memory for quantum simulations, they cannot ultimately compensate for the swiftness of RAM and an appropriate balance must be struck in performance trade offs [15, 16].

2.2.2 CXL as swap memory

Compute Express Link (CXL) is another new technology that is aimed at enabling the memory extension and management processes in high-performance computing environments. CXL is a high speed interconnect that enables the processors to connect

directly to memory and storage devices with low latency. It aims to allow better sharing of resources among CPUs, memory, and accelerators like GPUs or FPGAs [17, 18].

CXL offers a significant advancement in how swap memory can be used in quantum computing simulations. It facilitates the dynamic allocation of memory and allows it to be shared between different computing elements. This provides near-RAM speed access to larger pools of memory [19].

The benefits of using CXL as swap memory include:

- Low Latency: CXL significantly reduces the latency associated with accessing swap memory and is almost as fast as accessing RAM. This is critical to maintain performance for computations that require rapid data access such as quantum simulations.
- Memory Pooling: CXL allows memory from multiple devices to be pooled into a larger unified memory pool which can then be used in a more efficient manner during the course of simulations.
- Dynamic Memory Allocation: CXL provides the ability to dynamically allocate memory resources, which in gives more flexibility to the simulations in terms of memory upscaling or downscaling.

Thus, the balance between large memory size for computations and fast access times can be resolved through CXL technology in quantum computing simulations. CXL can enhance the memory space physically available in the system without compromising the high fast speed necessary for carrying out complex quantum simulations.

This paper focuses on the use of swap memory, specifically in context of utilizing SSDs and CXL technology, to perform large-scale quantum simulations even with insufficient physical RAM. However, as the complexity of quantum computing simulations increases, the need for efficient memory management becomes of greater importance. The work presented here provides evidence on how well SSDs and CXL can be utilized as swap memory in the simulation studies conducted using the QuEST framework. This includes benchmarking the performance impacts, assessing the balance between speed and memory capacity, and understanding how these technologies could make it possible to perform bigger quantum simulations. The results of these experiments aim to contribute to a broader understanding of how to optimize quantum simulations on classical hardware, particularly in environments with limited computational resources.

There has been limited research focused on the use of swap memory in quantum simulations, with the SnuQS framework being one of the few notable studies in this area. 'SnuQS: Scaling Quantum Circuit Simulation using Storage Devices' by S. Kim, J. Lee, and J. Kim [20] aligns with the objectives of this dissertation. This paper proposes the SnuQS framework that seeks to solve the problem of insufficient physical memory in classical simulations by deploying SSDs as extensions of the RAM.

SnuQS's main innovation is in its implementation of sophisticated tiered memory management which aims to optimize data transfer to and from the RAM and the SSD. This is made possible by the inclusion of storage devices which make it possible to model larger quantum circuits than what would otherwise be feasible, due to limitations in RAM.

This paper aims to add to the concepts SnuQS introduces by studying swap memory more comprehensively. SnuQS concentrates only on SSDs, whereas this research examines different storage options and benchmarks their effect on the performance of the simulations.

2.3 Benchmarking

Benchmarking is essential to evaluate the performance and efficiency of quantum computing simulations. It is important to assess the effectiveness of frameworks like QuEST to emulate quantum systems, manage computational resources, and scale with increasing problem sizes.

Quantum computations are very complex and a simple increase in the number of qubits results in an increase in the computational resources needed exponentially. This calls for intensive benchmarking to establish performance levels of the different approaches to simulation. Benchmarking includes testing the simulators with various parameters such as the number of qubits, depth of the circuit, noise amount, and available memory. This process helps identify their strengths, weaknesses, and potential areas for optimizations, thereby building the understanding of the trade-offs between speed, accuracy, scalability, and resource consumption [6].

Benchmarking is important to:

- Analyze: Ascertain the efficacy of the resources, such as any of the processors including the CPU or GPU, or even memory components, being used by a quantum simulator. This evaluation includes measuring execution time consumption,

memory utilization, and determining how well a quantum circuit simulation can scale with the increase in the quantum circuit size.

- Compare Accuracy: Evaluate the accuracy of the simulation through a comparison between their outcome and the predetermined analytical methods or alternative simulators, which is highly necessary when assessing the approximations as well as noise models.

- Identify Bottlenecks: Find out the shortfalls of the simulation, which may be in poor utilization of memory, large use of swap memory, or ineffective mapping of the workload.

- Guide Optimization: Improve the processes of simulation by inferring information from these indexes, which could be in the form of enhancing the algorithms, reengineering the memory management systems, or considering hardware optimization.

With the progress of quantum computing, benchmarking of this nature and its reproducibility becomes more important to provide a means of comparing simulators and evaluating their suitability for real-world applications.

Numerous platforms have been implemented that allow for reproducible benchmarking of quantum computational simulations. Among them, MQTbench [21] and Yao benchmark [22] stand out for the advantages they bring to the field. Additionally, their user-friendliness and easy availability make them accessible tools for research, facilitating a consistent and reliable benchmarking process.

MQTbench: MQTbench is a set of performance benchmarks proposed by the Munich Quantum Toolkit (MQT)team, which provides a shared bench for different types of quantum computation resources. MQTbench takes into consideration the performance of more than just quantitative metrics of the quantum simulation and the precision within which computing is performed while emphasizing the potential for qubit numbers to scale up. It facilitates the assessment of the performance of a framework within large-scale quantum simulations task evaluation which is crucial especially when looking at the performance of the framework concerning the memory limitations [21].

Nonetheless, MQTbench is not capable of performing any benchmarking that employs single-qubit gates such as Pauli-X and Hadamard which this research specifically intends to use. Benchmarking with single-qubit gates is important as these are the basic

components of a quantum circuit and their qualities have a significant effect on the success and efficiency of quantum computations. The focus on these fundamental gates helps to understand the essence of the quantum simulation and allows for more accurate tuning and improvement of the system.

Yao benchmark: Yao is a versatile open-source framework, written in Julia, suitable for composing and evaluating quantum algorithms. It comprises components for the building of quantum circuits, quantum dynamics simulations, and benchmarks, as well as implementing quantum benchmarks. Yao makes it possible to construct a benchmark for any simulator or quantum hardware and test their performance relative to the classical simulator or quantum hardware. It is also possible to extend the benchmarks to accommodate any research needs [22].

Unlike MQTbench, Yao facilitates benchmarking with single-qubit gates, making it an ideal tool for analyzing fundamental quantum operations. Additionally, Yao integrates Google Benchmark, which adds numerous features such as detailed performance analysis, customizable benchmarking configurations, and support for complex benchmarking scenarios. Yao's modularity and flexibility are particularly advantageous for the objectives of this paper, specifically in analyzing the performance of QuEST-based simulations using benchmarks generated through Yao.

Some other benchmarking frameworks that were explored included QASMBench [23], QUARK [24], PAS [25], HpQC [26] etc.

In essence, benchmarking is crucial to this research and provides a structured approach to analyzing the QuEST framework's performance in simulating quantum circuits. Through detailed benchmarks, this research aims to:

- Quantify Performance: Establish metrics for quantum simulations using execution time, scalability, and memory usage.
- Evaluate Swap Memory Impact: Focus on the influence of swap memory on the performance, comparing scenarios with and without it to understand trade-offs.
- Compare storage technologies: Compare the effect of using SSD versus CXL as swap memory by examining the differences in performance and efficiency between these technologies.

Chapter 3

Methodology

In this research, the simulations conducted are mainly based on the Pauli X gate since this is the core quantum computation gate, equivalent to the classical NOT gate, that serves the function of flipping the state of the qubit. The choice of the Pauli X gate as the benchmark quantum operation was driven by the primary objective of the experiments to evaluate the efficacy of the memory management techniques on simulation performance without the complication of multi-qubit operations or more complex algorithms.

The goal was to monitor the effect of swap memory on the simulation time and overall efficiiency, especially when the physical memory could not handle the entire quantum state vector. The design of the experiments was meant to analyze the performance benefit or disadvantage of using SSD and CXL for swap memory in different swappiness configurations.

The basis for all such simulations is embedded in the fact that with the increasing number of qubits, the quantum state space grows exponentially. For an n qubit system, the state vector requires 2n complex amplitudes and thus high memory as n gets larger. Considering this exponential growth, it becomes impractical to physically keep all the stored information in the RAM state vector beyond a particular number of qubits, thus it necessitates the inclusion of swap memory for preventing the limitation of the system capacity.

In order to tackle the memory problem, this work integrates external memory of SSD and CXL to quantum computing simulations of the QuEST quasi-classical framework. Compared to traditional hard disks, SSDs (Solid State Drives) have a large large storage capacity and relatively fast speed as compared to traditional hard drives. On the other hand CXL (Compute Express Link) is recently developed technology, fast and offers high memory capacity.

The simulations seek to investigate the following two main frameworks on swap use over CXL and SSD:

1. SSD-Only Configuration: Here the swap space is entirely used by a SSD and the aim is to analyse the quantum simulations while relying on SSD as the only mode to execute overload memory beyond physical RAM.

2. SSD and CXL Configuration: In this setup swap memory is allocated to both SSD and CXL serving as a swap, but with CXL having the higher priority. This configuration aims to measure the increase in performance due to the joint utilization of SSD and a faster memory technology.

The simulations were carried out using the Yao benchmark with QuEST , which stands out because it is designed with distributed memory systems in mind [5], an important feature when modeling large quantum systems that exceed the RAM of a machine. The simulations were run on a high performance computing (HPC) cluster with a total of 500 GB of physical RAM, 3.6 TB of Solid State Device (SSD) and 128 GB of Compute Express Link (CXL) swap memory.

The deterministic requirements are the memory requirements which can be expressed as follows:

$$\text{Memory (in bytes)} = 16 \times 2^n$$

where n is the number of qubits. The multiplication by 16 is due to the need to store complex numbers which themselves comprise a real and an imaginary part, each part stored as a double precision (8 byte) floating point number respectively. For example:

$$\text{35 qubits require } 16 \times 2^{35} \approx 512 \text{ GB}$$

$$\text{36 qubits require } 16 \times 2^{36} \approx 1024 \text{ GB or 1 TB}$$

When considered all these requirements it is obvious that simulating 35 qubits does require more than what the 500 GB of physical RAM available can support and simulation of 36 qubits even surpass the combined RAM and CXL capacity and thus uses the SSD as a swap. Therefore the swap memory configuration was determinant in simulating quantum circuits that were beyond the physical RAM, that is, qubits 35 and 36.

SSD-Only Experiments:

- This set of experiments was conducted using only the SSD as swap memory. The umber of qubits ranged from 4 to 36 whereby the high qubits counts of 35 and 36 required that additional swap memory.

- The swappiness parameter was modified in steps of 10, with regard to RAM and swap space allocation. To ensure consistency, each setting was tested multiple times.

- Performance metrics including execution and CPU time were recorded and analyzed on completion of one of the tasks. Execution time determines the total simulation time, inlcuding I/O operations while CPU time indicates the amount of time the central processing unit worked on matters of simulation only, without any idle periods.

CXL And SSD Experiments:

- In the case where both the CXL memory and the SSD were used for swap memory allocation, experiments were carried out following the same steps as those for SSD-only tests with varied swappiness parameter and over a qubit range of 4 to 36.

- The objective was to compare the performance of the CXL and SSD configuration against the SSD-only setup, focusing on how the prioritized use of CXL influenced overall simulation efficiency.

- The aim was to analyze the difference made my using the CXL and SSD configuration as compared to only using SSD, mainly to note the effect of priortising CXL on rate of efficiency of simulation as a whole.

A configuration utilizing solely CXL was not tested out since it was clear that the available 128 GB of CXL would not be enough to simulate 36 qubits as only 35 qubits could be managed with just RAM and CXL. In this regard, there was also no prioritization of experiments with SSD on top of CXL as there was no notable sufficiency expected on the performance one would gain by utilizing an SSD only setup as opposed to an SSD on top of prioritization setup for just the 35 and 36 qubit simulations.

The swappiness allocation parameter was modified via system configuration control files with an aim to vary the proportion of physical memory and virtual memory used. In both configurations described, the swappiness parameter which is an OS parameter used for controlling the system's tendency to use swap is adjusted from 10 to 100 in intervals of 10. This alteration makes it possible to test how aggressive or conservative memory swapping will impact the performance of the simulations, particularly when

the system is nearing its memory limits. A low-value swappiness (e. g. 10 or 20) is useful when data is desired to be kept in RAM, but cases where large swap is on fast storage devices a larger value (e. g 60 or 70) can be benefitial. The default swappiness is set to 60, indicating the system to use swap when RAM usage is around 60

Key Challenges Faced:

- During the course of this research, one of the key difficulties that was faced was the performance drop during the simulation of 35 and 36 qubit. As predicted, they resorted to the use of swap memory and this negatively affected their performance in terms of speed and even time taken to complete certain tasks. For example, performing a simulation of 35 qubits using only an SSD took 25 minutes while performing a simulation of 36 qubits with the same rig took 2.5 hours to complete. However, on a CXL and SSD setup, execution time was appreciably less, 35 qubits was done within 15 minutes and 36 was done under 2 hours.

- Another difficulty arose when it was necessary to handle the data locality of the swap memory. The quantum state vector which is a growing function of the number of qubits is more dynamically changing and thus in need of frequent updates throughout the simulation. During this time the system encountered a problem dubbed ”thrashing” where the CPU has to swap more RAM and SSD data than actually executes machine instructions, as parts of the state vector that are being worked upon are stored on the slower swap memory.

- Additionally, challenged were encountered to ensure reproducibility and consistency of results since incorporating swap memory introduces variability. Although multiple repetitions of the simulations helped solve this issue, the considerable length of running a simulation posed a challenge on its own.

Some alternative approaches that could be considered include:

- Using Larger Capacity SSDs or NVMe Drives: One possible method could be replacing the standard SSDs with NVMe drives as they can lower the swap memory latency. However, this approach was avoided and the research concentrated on investigating the present hardware configuration [27].

- Adjusting Memory Management Algorithms: Consideration was given to the possibility of formulating user-specific memory management algorithms within the operating environment of the computing device. However, this was considered

to be beyond the objectives of the present study that was geared towards meshing into the existing tools and settings based in typical HPC setups.

- Conducting such experiments, this research adds to understanding how quantum computing simulations will perform using swap memory. The results of these experiments will be further discussed in the next section.

Chapter 4

Results and Evaluation

A series of simulations has been performed using the QuEST framework to understand the effects of using SSD and CXL as a swap memory on the large-scale quantum simulations focussing on those with a higher qubit count. The results are provided for the two cases (‘SSD only’ and ‘CXL with SSD’) in the form of several plots containing performance indices such as execution time, CPU time, and performance ratio.

4.1 SSD-only Analysis

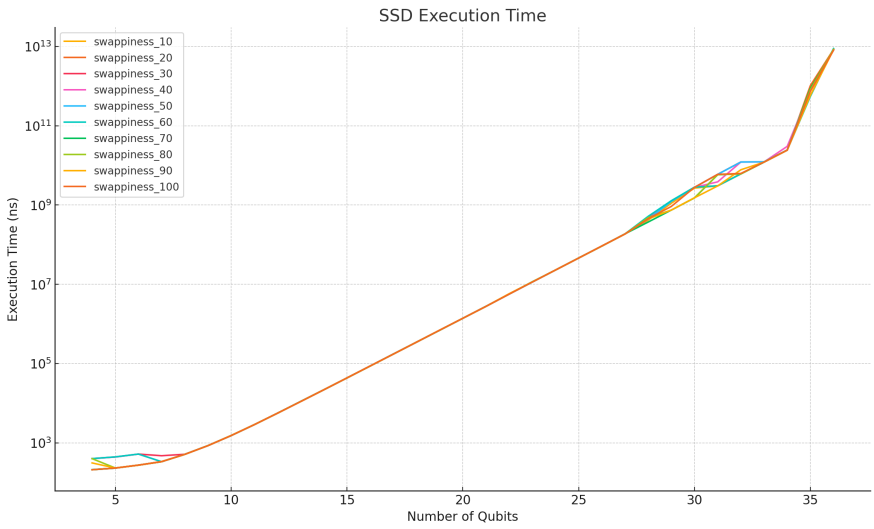

Figure 4.1: Execution and CPU times for SSD-only Analysis for 4-36 qubits

The analysis starts with the SSD-only configuration where execution and CPU times are computed at different qubit numbers ranging from 4 to 36. The results, as shown by figure 4.1, reveal the fact that execution time increases sharply with an increase in

the number of qubits particularly when the system has to use swap memory owing to low physical RAM. The access time delays incurred by the SSD greatly detracted from the performance of the simulation and led to longer execution times as the number of qubits increased.

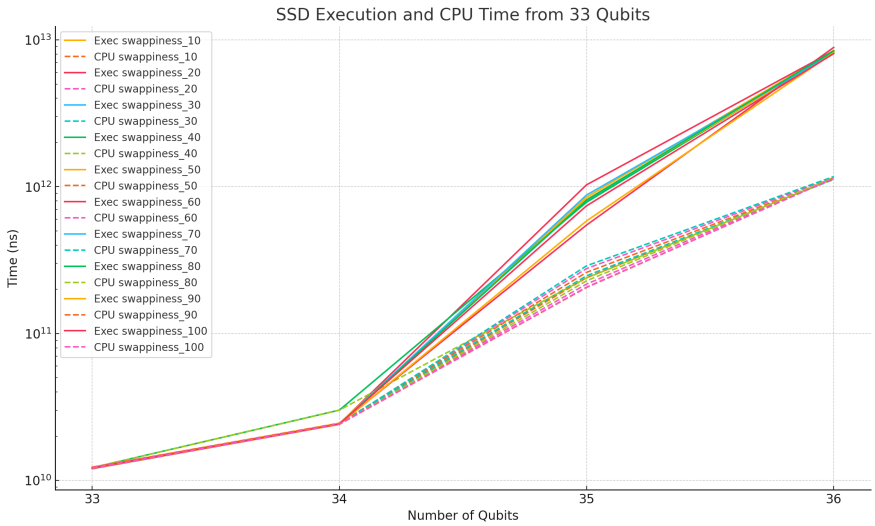

Figure 4.2: Execution and CPU times for SSD-only Analysis for 33-36 qubits

The corresponding CPU time analysis reveals that the SSD configuration demands more processing power as the qubit count grows. However, figure 4.2 highlights that the overall time taken is significantly more than the time the CPU spent executing the instructions when swap memory is used (number of qubits >= 35). This additional time is a result of the fact that usage of SSD space as swap memory is rather not efficient since more memory resources are used and less computations are done. This suggests that although SSD can volume up the memory available for quantum simulations, it comes at the price of increased latency. As figure 4.1 depicts, when the number of qubits gets larger the execution times become extremely undesirable, hence revealing the disbenefits of relying solely on an SSD in memory intensive quantum simulations.

The results also show that varying the 'swappiness' parameter did not make a significant impact on the performance. This is because the simulations reach a state where swap memory is continuously used and the system's overall performance is effected by the speed of swap memory instead of any set OS policy that dictates when to swap data.

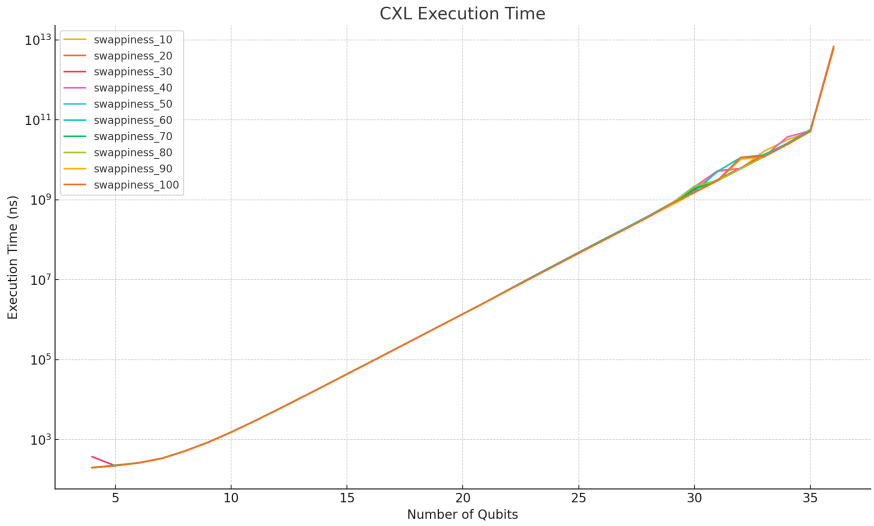

Figure 4.3: Execution and CPU times for CXL and SSD Analysis for 4-36 qubits

4.2 CXL and SSD Analysis

When CXL is added to the swap memory configuration with SSD, results indicate that execution time and CPU time improve significantly. Particularly, figures 4.6 and 4.7 depicts that when using swap memory (number of qubits $>=35$) computation time of SSD and CXL indicates that the CXL configuration is the better performer when compared with SSD-only.

Execution times are shorter because CXL is less latency sensitive and has faster access speeds, even as the qubit count rises. Additionally, CXL+SSD configuration has CPU times that are mostly constant over the cycle and are lower than when only SSD is in use. This stability suggests that CXL benefits both the execution and CPU time, hence, enabling the system to concentrate on the computation part rather than on memory swap and management.

Percentage of improvement in execution time due to CXL is shown in figure 4.5. This is calculated by averaging on the execution times for different swappiness values and alling the following formula

$$\frac{\text{CXL time} - \text{SSD time}}{\text{SSD time}} \times 100$$

The data collected for the 35 qubit simulation, which is when the CXL is used, shows a significant increase in performance (reduction in execution time). The data for the 36 qubit simulation, which is when SSD is used alongside CXL, shows a dip in performance and steep increase in execution time (figure 4.4). This highlights the

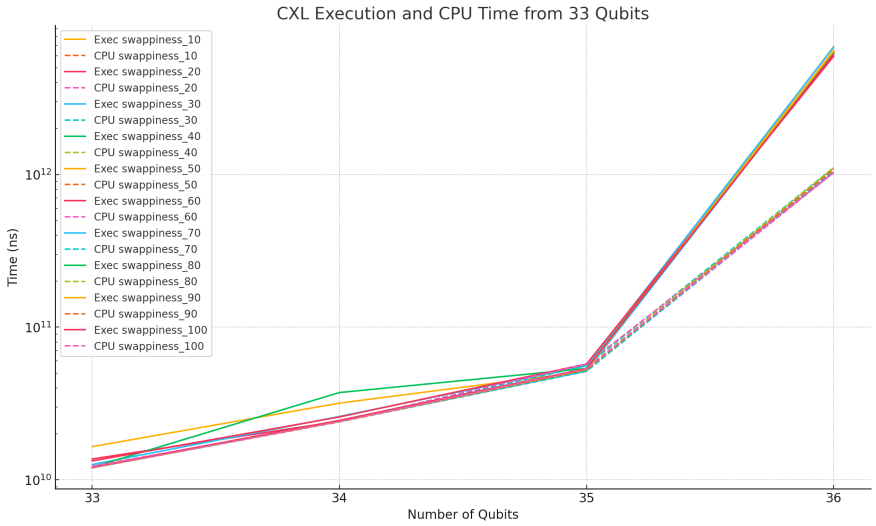

Figure 4.4: Execution and CPU times for CXL and SSD Analysis for 33-36 qubits

considerable advantage which CXL offers over its competitors, making the technology a suitable choice for large scale quantum simulation.

4.3 Evaluation

The results from this study clearly demonstrate that indeed SSD can improve the available memory for quantum simulation but the performance levels get compromised. When the number of qubits is high, the weaknesses of disks that depend on SSD only configurations start to show themselves, and the execution times grow significantly. The opposite is true when CXL is combined with SSD as it brings about a lot of improvements in performance. With reduced execution and CPU times and improved efficiency gains, CXL is a good fit for situations where high efficiency performance ratios are required but physical RAM is not up to the demand.

Key findings from this evaluation include:

- Scalability: CXL manages the increase of memory related to the increase in the number of qubits more effectively than the SSD. This scalability is important to enable quantum simulations to cross the barrier of classical hardware.

- Efficiency: CXL's reduced execution and CPU times lead to more efficient simulations, shortening the simulation time and providing more trustworthy results. This efficiency is of great use in high-performance computing applications which are time and resource intensive.

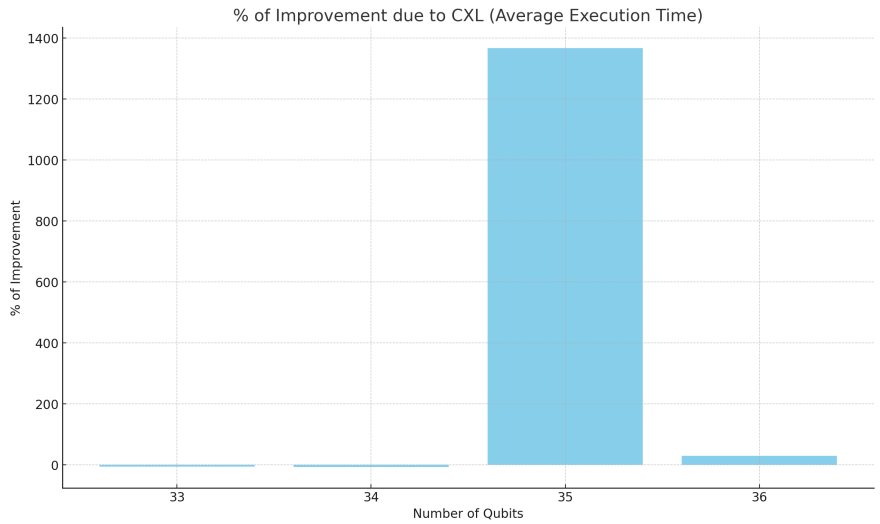

Figure 4.5: Percentage of improvement in execution time due to CXL (33-36 qubits)

- Practical Application: Factors such as the percentage of improvement due to CXL are very significant in that they are a demonstration of the possible use of CXL in quantum simulations on large scales. The incorporated architecture will allow users and researchers to run more advanced simulations without the drastic slowing down of the system typical of an SSD only system configuration.

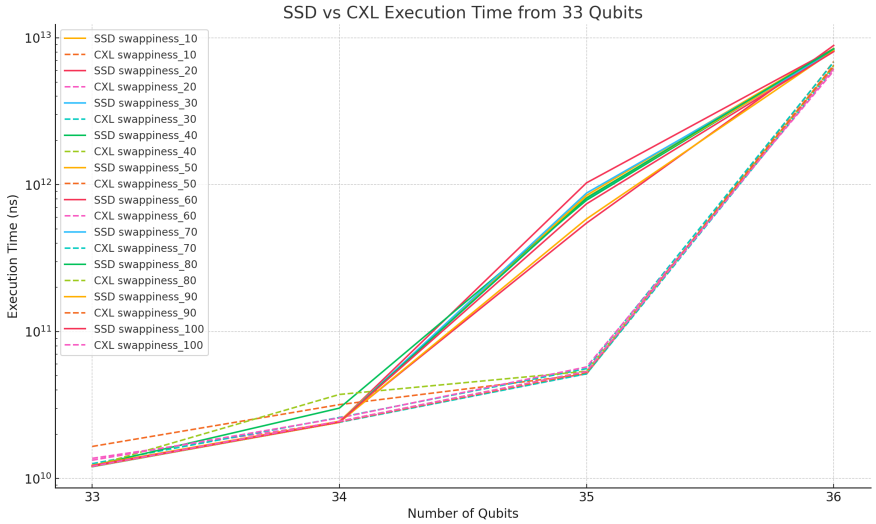

Figure 4.6: Execution time comparison for SSD vs CXL (33-36 qubits)

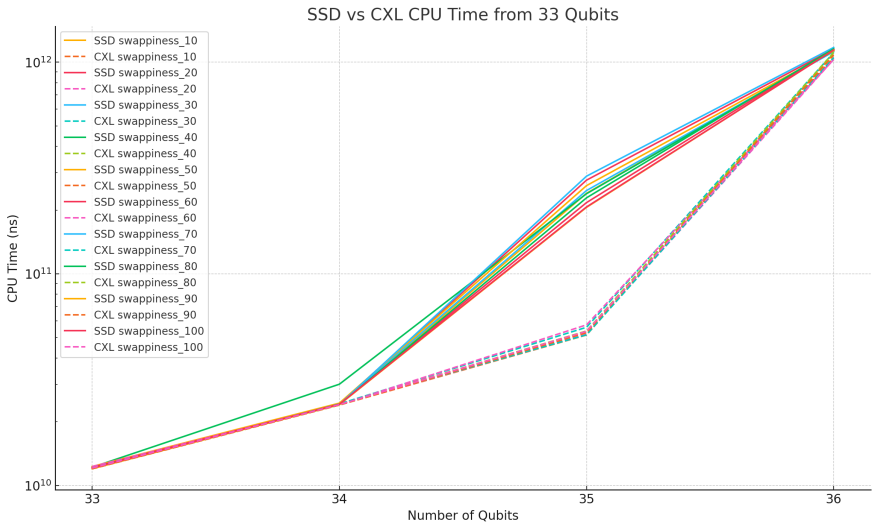

Figure 4.7: CPU time comparison for SSD vs CXL (33-36 qubits)

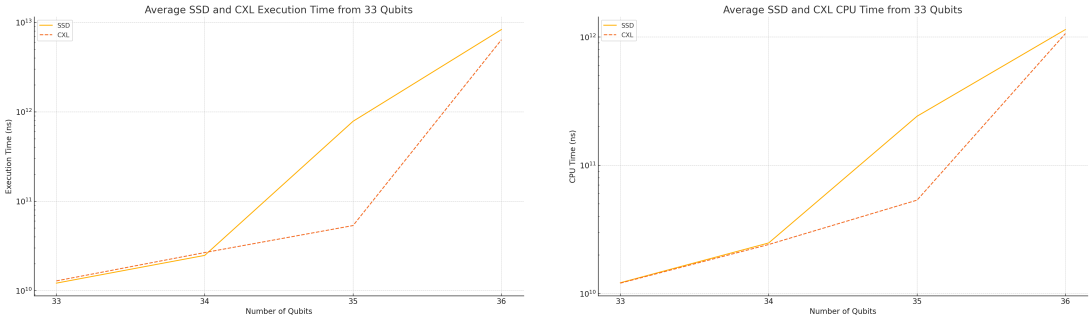

Figure 4.8: Average execution and CPU time comparison for SSD vs CXL

Chapter 5

Conclusions

The goal of this research was to examine the potential benefits of swap memory in the quantum computing simulations performed using the Quantum Exact Simulation Toolkit (QuEST). This essentially looked into the use of Solid State Drives (SSD) and Compute Express Link (CXL) technology in order to increase the memory space available and how this affects the quantum simulations in regard to the increased number of qubits.

There are several significances of this research to the general field of study. First, it was indicated that swapping memory into power SSDs would enhance the amount of memory available for the quantum simulations for larger quantum systems which would require more physical RAM than available in the systems. However, as most of the added memory was put on SSDs, there were problems because access to SSDs was many times slower than to RAM, resulting in much longer execution times and poorer performance of the simulations as the configurations were 35 and later 36 qubits. The CPU was virtually "thrashing" wherein a vast majority of the operations revolved around moving data back and forth between the SSD and the CPU, instead of executing any computations, particularly in the SSD only mode configuration.

On the other hand, the application of CXL technology as a swap memory option reflected a significant enhancement in performance. CXL can provide near-RAM speed while enlargening memory spaces. CXL combined with SSDs performed better when compared to SSDs alone, particularly in qubit rich simulations. This indicates that CXL technology together with SSDs could be a better way of providing more memory in quantum simulations without compromising performance. This in particular is of great relevance regarding the future of simulations of quantum computing on classical systems. They show that whilst SSDs may be deployed to increase the memory, incorporation of CXL technology is essential in enhancing performance in such manner that more

complex and larger quantum simulations may be achieved. These findings enhance understanding of the challenges and implications of using swap memory that is based on diverse storage technologies, and also help in the design of more efficient quantum simulations for environments with memory constraints.

5.1 Future Work

Based upon the results of this thesis, it is possible to state some directions for further investigations. In particular, it would be interesting to look into the utilization of other more sophisticated storage solutions such as NVMe drives that have high access speeds. This would then help in understanding the usability of such swap memory with the NVMe compared to CXL for effective executing quantum simulations.

The future work could also include designing and incorporating into the simulation framework some real time memory management strategies. Such algorithms may be able to suit the evolving nature of quantum state vectors and shield or minimize the effects of “thrashing” while improving the efficiency of the simulation as a whole. Furthermore, the use of machine learning to forecast and control the amount of memory needed at different stages of the simulation, rather than waiting until memory is entirely used, could also be a way of optimizing resources and improving performance.

Moreover, the scope of the simulations could also be extended to incorporate algebraic or linear algebraic quantum algorithms employing multi-qubit circuits, so that a clearer picture of the performance of the swap memory for various types of computations may emerge. This would then mean that thorough research could eventually lead to proper conclusions concerning the application of different memory management practices in quantum computer simulation regardless of the benchmarking of a considerable part of quantum circuits.

Lastly, there may be the possibility of re-evaluating the use of swap memory in a hybrid quantum-classical computing setting as more advances in quantum computing hardware are made. This and similar future research aims to investigate how computational resources typically hosted in classical simulation frameworks like QuEST could find deployment in quantum hardware so as to ease the computational load on the classical systems and optimize simulation capabilities.

Bibliography

[1] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, 2018.

[2] D. Babukhin, A. Zhukov, and W. Pogosov, "Hybrid digital-analog simulation of many-body dynamics with superconducting qubits," *Physical Review A*, vol. 101, no. 5, p. 052337, 2020.

[3] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "Cutqc: using small quantum computers for large quantum circuit evaluations," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. ACM, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1145/3445814.3446758

[4] S.-P. Yang, M. Kim, S. Nam, J. Park, J. yong Choi, E. H. Nam, E. Lee, S. Lee, and B. S. Kim, "Overcoming the memory wall with CXL-Enabled SSDs," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA: USENIX Association, Jul. 2023, pp. 601–617. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/yang-shao-peng

[5] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, "Quest and high performance simulation of quantum computers," *Scientific reports*, vol. 9, no. 1, p. 10736, 2019.

[6] A. Jamadagni, A. M. Läuchli, and C. Hempel, "Benchmarking quantum computer simulation software packages," *arXiv preprint arXiv:2401.09076*, 2024.

[7] M. Bielski, C. Pinto, D. Raho, and R. Pacalet, "Survey on memory and devices disaggregation solutions for hpc systems," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed*

Computing and Applications for Business Engineering (DCABES). IEEE, 2016, pp. 197–204.

[8] Y. Park and H. Bahn, "Management of virtual memory systems under high performance pcm-based swap devices," in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 2, 2015, pp. 764–772.

[9] P. J. Denning, "Thrashing: Its causes and prevention," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, 1968, pp. 915–922.

[10] P. W. Shor, "Quantum computing," *Documenta Mathematica*, vol. 1, no. 1000, pp. 467–486, 1998.

[11] R. LaRose, "Distributed memory techniques for classical simulation of quantum circuits," *arXiv preprint arXiv:1801.01037*, 2018.

[12] Y. Zhao, Y. Chen, H. Li, Y. Wang, K. Chang, B. Wang, B. Li, and Y. Han, "Full state quantum circuit simulation beyond memory limit," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[13] J. Lee, S. Park, M. Ryu, and S. Kang, "Performance evaluation of the ssd-based swap system for big data processing," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2014, pp. 673–680.

[14] S. Bergman, N. Cassel, M. Bjørling, and M. Silberstein, "Znswap: Un-block your swap," *ACM Transactions on Storage*, vol. 19, no. 2, pp. 1–25, 2023.

[15] G. F. Oliveira, S. Ghose, J. Gómez-Luna, A. Boroumand, A. Savery, S. Rao, S. Qazi, G. Grignou, R. Thakur, E. Shiu *et al.*, "Extending memory capacity in modern consumer systems with emerging non-volatile memory: Experimental analysis and characterization using the intel optane ssd," *IEEE Access*, 2023.

[16] X. Ouyang, N. S. Islam, R. Rajachandrasekar, J. Jose, M. Luo, H. Wang, and D. K. Panda, "Ssd-assisted hybrid memory to accelerate memcached over high performance networks," in *2012 41st International Conference on Parallel Processing*. IEEE, 2012, pp. 470–479.

[17] J. Sim, S. Ahn, T. Ahn, S. Lee, M. Rhee, J. Kim, K. Shin, D. Moon, E. Kim, and K. Park, "Computational cxl-memory solution for accelerating memory-intensive applications," *IEEE Computer Architecture Letters*, vol. 22, no. 1, pp. 5–8, 2022.

[18] Y. Fridman, S. Mutalik Desai, N. Singh, T. Willhalm, and G. Oren, "Cxl memory as persistent memory for disaggregated hpc: A practical approach," in *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 983–994.

[19] D. Gouk, M. Kwon, H. Bae, S. Lee, and M. Jung, "Memory pooling with cxl," *IEEE Micro*, vol. 43, no. 2, pp. 48–57, 2023.

[20] D. Park, H. Kim, J. Kim, T. Kim, and J. Lee, "Snuqs: scaling quantum circuit simulation using storage devices," in *Proceedings of the 36th ACM International Conference on Supercomputing*, 2022, pp. 1–13.

[21] N. Quetschlich, L. Burgholzer, and R. Wille, "Mqt bench: Benchmarking software and design automation tools for quantum computing," *Quantum*, vol. 7, p. 1062, 2023.

[22] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, "Yao. jl: Extensible, efficient framework for quantum algorithm design," *Quantum*, vol. 4, p. 341, 2020.

[23] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, pp. 1–26, 2023.

[24] J. R. Finžgar, P. Ross, L. Hölscher, J. Klepsch, and A. Luckow, "Quark: A framework for quantum computing application benchmarking," in *2022 IEEE international conference on quantum computing and engineering (QCE)*. IEEE, 2022, pp. 226–237.

[25] H. Bian, J. Huang, J. Tang, R. Dong, L. Wu, and X. Wang, "Pas: A new powerful and simple quantum computing simulator," *Software: Practice and Experience*, vol. 53, no. 1, pp. 142–159, 2023.

[26] H. Bian, J. Huang, R. Dong, Y. Guo, and X. Wang, "Hpqc: a new efficient quantum computing simulator," in *Algorithms and Architectures for Parallel Processing: 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2–4, 2020, Proceedings, Part II 20*. Springer, 2020, pp. 111–125.

[27] Q. Xu, H. Siyamwala, M. Ghosh, T. Suri, M. Awasthi, Z. Guz, A. Shayesteh, and V. Balakrishnan, "Performance analysis of nvme ssds and their implication on

real world databases," in *Proceedings of the 8th ACM International Systems and Storage Conference*, 2015, pp. 1–11.