Exploring Hybrid Classical-Quantum Compute Systems through Simulation

Muhammad Nufail Farooqi
Leibniz Supercomputing Centre of the
Bavarian Academy of Sciences and Humanities
Garching near Munich, Germany
Email: farooqi@lrz.de

Martin Ruefenacht
Leibniz Supercomputing Centre of the
Bavarian Academy of Sciences and Humanities
Garching near Munich, Germany
Email: ruefenacht@lrz.de

***Abstract*—The computational requirements of applications running in supercomputing centres are growing rapidly and diversifying over time. To meet these needs, developers are constantly developing new processors, domain-specific accelerators, and technologies. Quantum computing is one notable technology that has the potential to speed up certain computations and could be used to enhance High Performance Computing (HPC). However, integrating a quantum computer (QC) into HPC systems is a complex and challenging task, both at the physical and software level. Although various types of accelerators have been integrated into HPC in the past, the integration of QC into HPC presents a unique challenge as it operates on a new computing paradigm and is a scarce resource that needs to be shared among multiple jobs. Furthermore, HPCQC integration is a multidisciplinary field and integration can mean different things to different stakeholders.**

Our approach to HPCQC integration is from the perspective of an HPC centre where QC is integrated into the HPC environment as an accelerator. We identify parameters that are important for the integration, including network latency, HPCQC hybrid application characteristics and fine-grained scheduling. In addition, we develop a custom simulation model that can comprehensively simulate a hybrid classical-quantum compute system at selected level of detail. This simulation model is utilised to simulate the performance of a hybrid application by adjusting its features on a hybrid system with varying parameters. We discover that fine-grained scheduling is critical to performance, and network latency between QC and HPC has a significant impact on hybrid system performance.

***Index Terms*—HPC, quantum computing, hybrid system, workload simulation & modeling, scheduling**

I. Introduction

Rapid advancements and discoveries in science and technology over the last four decades have been made possible by high computational capabilities. This has not only led to breakthroughs in traditional sciences, but has also revolutionised fields such as artificial intelligence and data science. This applicability is driving increased demand for computing, leading not only to growth but also to diversification of use in both academia and industry.

High performance computing technologies have continuously evolved to meet the growing computing demands. The compute power in HPC is harnessed through various architectural advances; These architectures include, but are not limited to, multicore, manycore, accelerators, and a hybrid of these. There are also emerging technologies in development such as neuromorphic computing and quantum computing that are also paving their way to become a part of HPC.

Quantum computing is showing promise for exponential speed-up for certain problems [1]–[5]. Its development has made significant progress recently and is slowly making its way out of physics laboratories and into production environments. Quantum computing, although powerful, has limitations and cannot be used as a stand-alone computer to solve problems of applicable complexity and scale. Its high computational power, combined with the fact that it requires classical computing, makes it a good candidate to use as an accelerator within the HPC ecosystem.

Quantum computers represent a revolutionary shift in computing, with a new paradigm that poses unique challenges for integration into high-performance computing environments. These challenges are diverse and span both physical and software integration. For example, quantum computers operate in cryogenic environments, which requires specialised infrastructure and poses significant challenges for maintenance and repair. In addition, frequent calibration is required to ensure the accuracy and stability of quantum computing hardware. Dynamic circuit compilation and optimisation is also required to ensure efficient and effective use of quantum resources.

In addition, quantum computers are likely to be a limited resource in the near future, meaning that they will need to be shared among many computing jobs. This presents a significant challenge for job scheduling and resource allocation that can effectively utilise quantum computing resources while minimising wait times and maximising efficiency.

Finally, the characteristics of hybrid HPCQC (High-Performance Computing and Quantum Computing) applications remain largely unexplored and need to be carefully analysed to fully understand their potential and limitations. An example of classical-quantum hybrid algorithms are variational quantum algorithms, which need to continuously and synchronously exchange information between the classical and quantum information domains, introducing constraints and overheads that need to be addressed.

HPCQC integration involves many stakeholders, and each stakeholder has its own viewpoint on integration, placing emphasis on different things. The majority of HPC centres are expected to adopt quantum technology in the near future, although a small fraction are presently conducting experiments with it. [6]. Therefore, an integration study is necessary to address the concerns of compute centres. As quantum computers are not yet integrated into the HPC environment, we have developed a custom simulation model to explore the parameters important for HPCQC integration on potential hybrid compute system architectures. In particular, we focus on the software aspects that are important for integration from a compute centre's perspective. Highlights of the key contributions for this work are:

- Identify and analyse the challenges ahead for the HPCQC integration.
- Develop a custom simulator that can simulate both the hardware (such as processing units, node layout and network) and software (such as application types and scheduling policies) components of the hybrid system, with a plug-and-play framework to facilitate rapid development and effrotless integration of new components.
- Simulate and analyse a number of design parameters important for the potential HPCQC hybrid compute systems.
- Simulate and analyse HPCQC application characteristics and compare fine grain (task level) scheduling algorithms.

II. HPCQC Integration

HPCQC integration falls into two categories: hardware integration and software integration. The former involves the physical placement of the quantum processors and associated equipment alongside the HPC resources, and the establishment of a communication interface between the two systems. Software integration, on the other hand, involves integrating the software stacks and application paradigms of the two systems, which are quite different due to the fact that classical and quantum systems process information in different ways.

While the list of design parameters for HPCQC integration is extensive, we discuss the following key parameters: One is the impact of latency between the HPC and QC resources, which can have a significant impact on the overall performance of the hybrid compute system. In addition, QC is not a widely available resource and may therefore be used as a shared resource in the HPC environment, requiring careful consideration of issues such as fine-grain scheduling to ensure fair and efficient resource allocation. Finally, the unique characteristics of hybrid applications that combine classical and quantum computing paradigms must also be considered. To effectively implement HPCQC integration, it is essential to carefully evaluate each of these design parameters and develop strategies that can effectively address the challenges they present.

A. QC as Shared Resource (Task Scheduling):

A quantum computer is an expensive resource and is expected to have limited availability in compute centres. Therefore, it won't be feasible to allocate it as a dedicated resource to a job that won't be able to fully utilise the resource, and this will also lead to an excessive increase in waiting time of other jobs requiring quantum resources. An intuitive solution would be to use the quantum computer as a shared resource that can be used by multiple jobs simultaneously. Fine-grain (task-level) scheduling is required to coordinate access to the quantum computer from multiple jobs. We analyse three basic task-level scheduling algorithms and their impact on the overall usage of the hybrid system.

The three scheduling heuristics we use to schedule quantum tasks are First-In-First-Out (FIFO), Priority, and a mixed-mode task scheduling approach. The FIFO method prioritises tasks based on the order in which they are submitted. Tasks that have been in the queue for the longest time are executed first. The priority approach prioritises tasks based on their level of importance. Tasks with the highest priority are given priority over those with a lower priority. If two tasks have the same level of priority, the one that was submitted first is executed first. The priority assigned to a task is determined by the classical resources assigned to the job. The mixed-mode scheduling policy combines elements of both FIFO and Priority scheduling. In this strategy, a set number of tasks are executed using the FIFO method, followed by a set number of tasks executed using priority scheduling. This helps to balance the execution of tasks between those that have been waiting for a long time and those that are more critical or time-sensitive.

B. HPCQC Hybrid Applications:

Jordan [7] has compiled a comprehensive list of quantum algorithms. There are numerous classes of algorithms, but the most interesting class of algorithms for HPC are the variational quantum algorithms, which iteratively perform some classical computations followed by quantum computations. The alternation between classical and quantum computations is synchronous, as both have different compute paradigms that cannot directly interact with each other.

The variational quantum algorithms are promising potential candidates for high-performance computing (HPC) level hybrid applications [8]. While hybrid applications currently exist as algorithms, the development of HPC-level applications has yet to take place. Despite this, it is important to investigate their key characteristics that will be crucial for performance.

One of the significant factors to consider is the ratio between the classical and quantum compute time within an iteration. As hybrid applications may require communication both before and after the computation of a quantum segment, the number of interactions between these segments will also play a critical role in the performance of the application. These interactions can scale differently depending on the specific application's requirements. For instance, a hybrid application that runs using N processes, where all N processes, a subset of processes M, or a single process out of N processes may carry out quantum tasks, will necessitate varying degrees of communication.

C. Latency between HPC and QC:

Quantum computers are presently only available through cloud services due to the high level of technical expertise and maintenance required to keep the resource functioning. As a result, integrating a quantum computer from the cloud into an HPC system and running tightly integrated classical-quantum

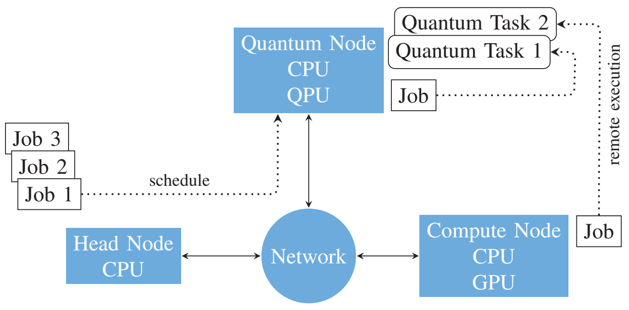

Fig. 1. Diagram of a classical-quantum system architecture model. Any combination of resources and nodes is possible with varying properties such as latency.

hybrid applications on it would considerably impact performance. This is due to the communication latency between the two resources, which adds significant overhead. To measure the impact on performance, there is a need for a thorough investigation of the effect of the latency on HPC resource utilization for hybrid applications.

The information flow from the HPC to QC is a quantum circuit, while qubit readouts are sent back. Therefore, the limited number of qubits currently available means that bandwidth between HPC and QC resources is not an immediate issue when running hybrid HPCQC applications.

III. Simulation Model and Implementation

The hybrid system simulator model is divided into two categories, namely hardware and software components. Hardware components include physical components like processors, node, and network, while software components task, taskgraph, process, job and scheduling. The hybrid system simulator is designed as a plug-and-play model, which means that new components can be added or existing components can be easily modified or replaced. This feature makes the simulator highly adaptable and flexible, allowing developers to create and test new systems with relative ease. Fig. 1 illustrates the core components for an example classical-quantum hybrid system.

In addition to the hardware and software components, an optional component of the hybrid system simulator is the *remote task execution service*. This component is responsible for executing tasks on a remote node if the processing unit required for the task is not available on the current node. This is particularly useful when resources on a compute cluster are shared. In our case, the quantum computer serves as a shared resource and quantum tasks are executed using this service.

The simulator also has the provision to implement and test the impact of new scheduling policies. Scheduling policies can be implemented not only for the jobs submitted to the cluster but also for a task queue of a processing unit. By simulating the impact of different policies, users can optimise the performance and efficiency of their hybrid system.

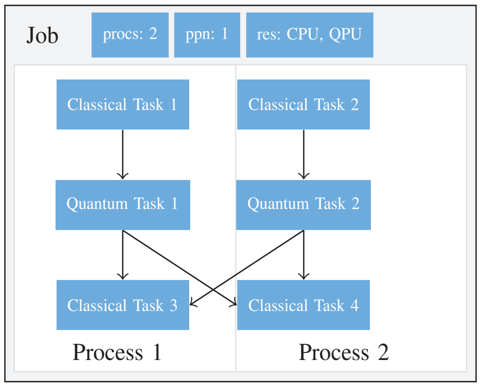

Fig. 2. Example taskgraph showing multiple processes with dependencies on both classical and quantum tasks.

A. Hardware Components

1) Processing Unit: A Processing Unit (PU) represent a computational unit that is capable of carrying out a computational task. The compute model for the PU is implemented inside a method named *compute* of the corresponding PU class. A PU also has an associated task queue where compatible tasks are submitted by the owner node. A *task_handler* processes the task queue, prioritising the tasks in the queue according to a predefined scheduling policy and sending each task to the PU for computation. Once a task has been completed, the *task_handler* receives a signal indicating its completion, allowing it to move on to the next task in the queue.

2) Node: The simulator consists of two types of nodes: a head node and a compute node. The head node acts as a control center, receiving job submissions from users and managing the scheduling of jobs on the compute node. The compute node is where the actual computation takes place, using the processing units (PUs) available to it.

Jobs are submitted to a head node, which checks the job requirements and maps them to the available PUs on the compute nodes. The head node then schedules the job for processing on the compute node using a scheduling policy. The job scheduling policy we have implemented is the First-Come-First-Serve (FCFS), which schedules jobs in the order in which they are received by the head node. However, this policy can be switched to any other policy depending on the specific requirements of the simulated computing environment. Once a job is scheduled, the compute node processes its associated taskgraph, which represents the computation to be performed by the job.

3) Network: All the nodes in a cluster are connected by a network, which facilitates communication between them to exchange of tasks and messages. Network overhead is accounted for by using a fixed latency model that estimates the time it takes for messages to travel between nodes. At present, the network model is relatively simple, but it is able to transmit jobs, tasks, and communication messages effectively.

However, we plan to implement a more detailed network model in the future. This updated model will dynamically simulate network congestion based on the number of messages being transmitted at any given time, providing for a more accurate representation of network behaviour.

B. Software Components

1) Task: A task can represent a variety of operations, i.e. classical computation, quantum computation or data communication. Each task is associated with an owner node and an owner process, indicating where the task was initiated. If the necessary processing resources are not available on the initiating node, the task can be executed on a remote node. I also has an associated event that is used to signal the completion of the task to its dependent tasks.

2) Taskgraph: A taskgraph is used to model the behaviour of an application. It is represented as a directed acyclic graph, where nodes represent individual tasks, and edges represent dependencies between the tasks. The dependencies ensure that each task is executed in the correct order and that all of its prerequisites are met before it can begin. A number of taskgraphs that are representative of some common application types are pre-built with the simulator. An example taskgraph is shown in Fig. 2.

3) Process: Similar to an operating system process, a process emulates an instance of a taskgraph on an assigned node. It processes the taskgraph and runs the tasks on processing units when they are ready, or offloads them to other nodes if a compatible PU is not available on the current node. Furthermore, each process has its own identification number which is used for communication between processes emulating the same taskgraph.

4) Job: A job encapsulates an application (i.e. a taskgraph) together with its execution requirements, i.e. the number of processes to spawn, the type and number of processing units required. It is analogous to a Slurm [9] job. Once a job has been submitted, its execution requirements are passed on to the job scheduler for reservation and scheduling on the hybrid system.

IV. Results

In Section II, we introduced key aspects of HPCQC integration, and in Section III we introduced the architecture of the simulator. Here we present three use cases to simulate and analyse a number of HPCQC hybrid system design parameters. We also simulate three task-level scheduling algorithms and compare their results. The selection of these use cases and design parameters is based on the discussion of HPCQC integration in [10]. The experimental setup is described below:

The cluster configuration that we simulate is similar to the one shown in Figure 1. It consists of three nodes: a head node, a compute node and quantum node, all connected by an internal network. Jobs are not scheduled directly on the quantum node but are submitted to the head node and are later scheduled on compute or quantum nodes to simulate the scenarios of an on-site quantum computer shared in a cluster, or accessing a quantum computer in the cloud. The compute node, which does not have a quantum processor sends all

TABLE I
Parameters used in simulation setup

Parameter	*Use Case 1*	*Use Case 2*
Cluster network latency	5 ms	
Cluster-QPU network latency	*see x-axis*	10 ms
Control electronics reprogramming time	*see legend*	500 ms
Readout time	0.001 ms	
Classical compute time	1 ms	100 - 0.1 ms
Quantum circuit execution time	0.05 ms	100 - 0.1 ms
# of shots	1000	
# of iterations (coupling frequency)	100	*see x-axis*

...

Parameter	*Use Case 3*	*Use Case 4*
Cluster network latency	5 ms	
Cluster-QPU network latency	*see legend*	0 ms
Control electronics reprogramming time	*see legend*	500 ms
Readout time	0.001 ms	
Classical compute time	1.81 - 0.18 ms	*see legend*
Quantum circuit execution time	0.18 - 1.81 ms	*see legend*
# of shots	1000	100
# of iterations (coupling frequency)	100	*variable*

quantum tasks to the quantum node, which sends the results back after computation. We measure throughput in terms of quantum jobs completed per unit time, as this is one of the key performance metrics for HPC clusters. The total simulation time is 24 hours and we calculate throughput as the average number of jobs completed per hour. The taskgraph that we simulate is representative of the class of Variational Quantum

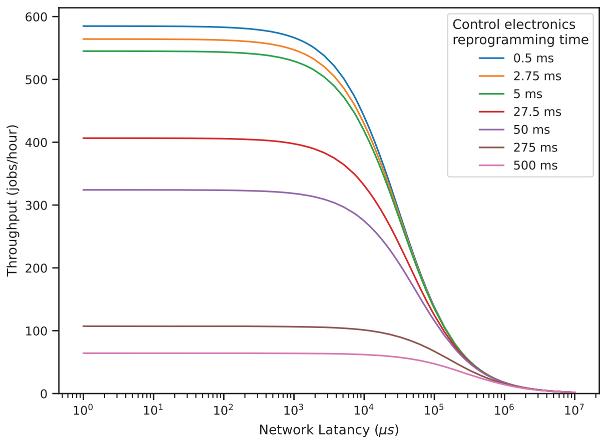

Fig. 3. This plot shows the relationship between network latency, reprogramming time, and the resulting throughput. The control electronics reprogramming time is shown in the legend. We see that both latency and reprogramming time has a significant effect on the overall throughput of the system in different regions of the parameter space.

Algorithms [11]. It consists of two tasks: a classical compute task and a quantum task that depends on the classical compute task. The taskgraph runs iteratively for a given number of steps. The parameter values for the system configuration to be simulated and the use cases are listed in Table I.

A. *Use Case 1: Throughput vs Network Latency*

An ideal scenario would be to equip each compute node with Quantum Processing Unit (QPU). However, currently, this is not feasible due to cost, space, maintenance, and design constraints. Therefore, the effect of communication latency on the performance of the system needs to be analysed.

Fig. 3 shows the throughput for network latency ranging from 1 microsecond to 100 seconds. The latency here corresponds to the latency between the quantum node and a quantum processor. A lower value of latency is analogous to simulating a QPU attached to the node and a higher value to a QPU residing remotely, e.g. in the cloud. Each line corresponds to a different value for control electronics reprogramming time that is the time required to reprogram the control electronics for a QPU when a new circuit needs to be executed.

As apparent from Figure 3, for lower values latency, the throughput does not vary significantly but for higher values i.e. the range for cloud access the throughput can be reduced drastically. The control electronics reprogramming time also plays an important role in increasing the throughput.

B. *Use Case 2: Throughput vs Coupling Frequency*

Besides system configuration, the system performance is also dependent on the application characteristics. One key aspect is the coupling frequency i.e. the number of times communication happens between the classical and quantum compute parts of the application per second. Two common types are loosely coupled and tightly coupled applications. An example of a loosely coupled application is a one-time quantum circuit that does not need communication. Variational quantum algorithms are examples of tightly coupled applications.

Fig. 4 shows throughput as we increase coupling frequency i.e. the number of iterations from 1 to 1000. The throughput decreases as we increase the coupling frequency, because of the increased overhead. Overhead here is a combination of the latency, control electronics reprogramming time and qubit readout time. Thus while loosely coupled applications may not be affected the performance of the system if run on a cloud but running tightly coupled applications can degrade the performance significantly.

C. *Use Case 3: Throughput vs Classical to Quantum Compute Time Ratio*

The computation time spent in the classical part of an application and the quantum part is not usually equal. In this use case, we evaluate the throughput affected by the classical to quantum compute time ratio for different values of network latency and control electronics reprogramming times.

Fig. 5 shows throughput for ratios between classical and quantum compute times ranging from 10:1 to 1:10. The quantum compute time refers to the time it takes to execute the circuit once. Throughput does not change significantly even in the case of higher overhead for applications with a higher quantum compute time as the overhead cost is small compared to the quantum compute time. However, throughput becomes sensitive to the overhead (latency and reprogramming time) for applications with lower quantum compute time.

D. *Use Case 4: Task-Level Scheduling Polices*

In this use case, we simulate a cluster of 1024 classical nodes with 48 cores each. QPUs are directly attached to quantum nodes and are shared among all jobs.

The workload is generated using the logs of HPC only workload from a compute centre. We simulate variational algorithm with different classical and quantum compute time ratios. Total computation time of a job is taken from the log

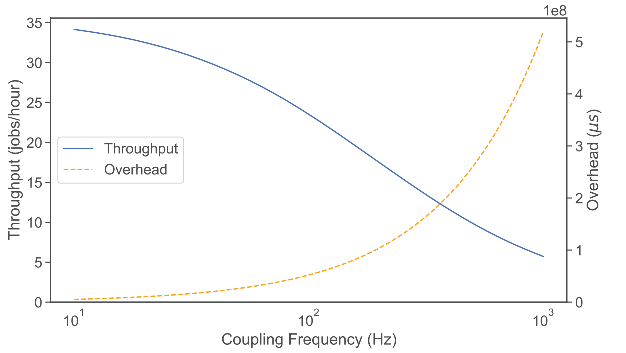

Fig. 4. This plot shows that as the coupling frequency increases the overall throughput suffers and as the coupling frequency increases the overhead also increases. Overhead is the combination of latency, control electronics reprogramming time and qubit readout time.

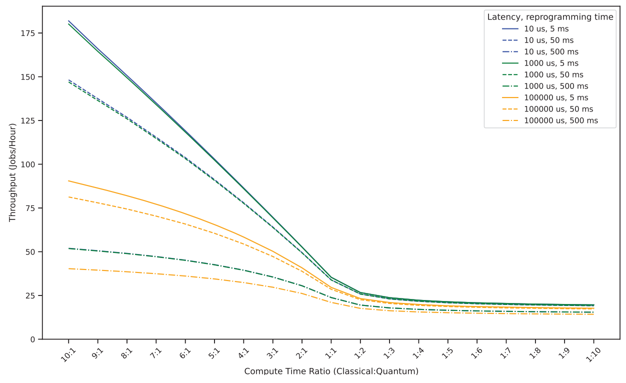

Fig. 5. This plot shows the throughput as a function of the computational ratio as classical:quantum. Latency and reprogramming times are shown in the legend. As seen, with a higher quantum to classical work ratio the through plateaus whereas if the quantum component is not oversubscribed then latency and reprogramming time are dominant.

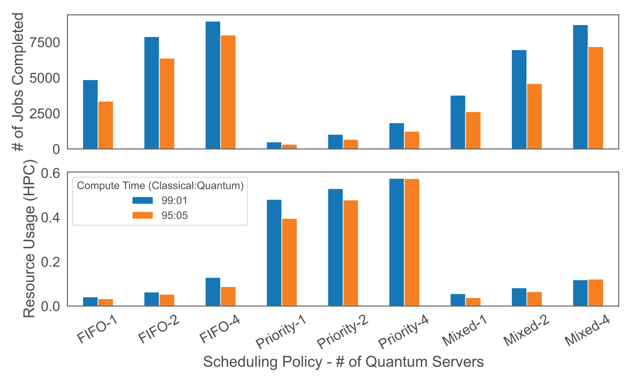

Fig. 6. This plot shows cluster's throughput (top) in terms of the total number of jobs completed and the average of classical resources usage efficiency(bottom) for the scheduling policies with different number of quantum servers and different workloads (classical-quantum compute time ratios). Both FIFO and Mixed completes high number of jobs while the Priority scheduling policy has the highest resource usage as the policy prioritises the quantum tasks that belongs to a job with most allocated HPC resources.

and distributed among classical (95-99%) and quantum (05-01%) compute times. We simulate one month of execution where job arrival times are taken from the log and are scheduled using the Slurm's backfill scheduler.

We compare three scheduling policies described in Section II; FIFO, Priority and Mixed on cluster configurations with 1,2 and 4 quantum servers. We also simulate two classical:quantum computation time ratios (99:01 and 95:05) of the HPCQC hybrid applications in the workload. We look into two performance matrices i.e.throughput and HPC resources usage to compare the performance of the task-level scheduling policies. Throughput is represented in terms of the total number of jobs that are completed during the entire simulation period. Thus, we keep track of the idle and busy times for the HPC resources and compute the average resource usage efficiency for all the completed jobs.

Figure 6 shows the throughput and resource usage. FIFO and Mixed scheduling perform better than priority-based scheduling by completing more jobs. The Priority scheduling completes fewer jobs as it priorities the quantum resource for bigger size jobs and allow them to finish first while small jobs keep waiting for the quantum resource. The Priority scheduling significantly increases the HPC resource usage as it reduces the waiting time for big jobs by allowing them to use the quantum resource out-of-order. However, the downside with the Priority scheduling is that small jobs suffers from starvation for the quantum resource thus results in low number of completed jobs. The Mixed scheduling reduces the effects of starvation by alternating between FIFO and Priority polices resulting in completing more jobs and completing higher total core-hours jobs. There is no clear winning scheduling policy that perform better on all performance metrics. The selection of a scheduling policy would therefore be based on a centre's goals and priorities that they want to achieve.

V. Related Work

The integration of quantum computing (QC) into high-performance computing is a relatively new area of research, and progress in this field is still in its early stages. While there have been several discussions and proposals have been put forward regarding the integration of quantum computing into HPC systems, concrete steps have yet to be taken. Some of the integration discussions are as follows.

In their paper, Martin et al. [10] examine various hybrid application workflows and analyse existing hybrid architectures, such as quantum computing in the cloud and its future integration on-premises or on-node. The authors also propose the concept of onloading to increase resource utilisation, where HPC resources can be utilised for tasks related to QC.

Humble's paper [12] highlights technical challenges for HPCQC integration but despite these challenges, Humble sees significant potential for progress. The paper concludes that further research and development is needed to fully realise the full potential of HPCQC.

Bartsch et al. [8] identify variational quantum algorithms as a suitable fit for hybrid HPCQC applications in the NISQ era, and predict that QC will serve as an accelerator for HPC in the future. The authors emphasise the importance of developing an efficient and standardised quantum software stack to facilitate the integration of HPC and QC.

Simulators have been used to simulate specific operations of a compute clusters. For example, [13] and [14] simulates job scheduling, task mapping and application simulation on target platforms. They focus on the interconnect models and simulate the message passing communication behaviour of application. Similarly, [15] and [16] do scheduling simulation using Slurm simulator. They model system workload and generate synthetic workload but no hardware and application simulation.

VI. Conclusion

In summary, we have discussed several key aspects of the integrating quantum computers into the supercomputing environment. In addition, we introduced a classical-quantum compute system simulation model, which can simulate these systems and generate performance metrics relevant to supercomputing centres. We simulated and analysed a simple mid-size classical-quantum compute system with various integration and system design parameters such as latency, control electronics reprogramming time, coupling frequency, and classical-to-quantum compute time ratio. We also simulated a real workload for HPCQC hybrid applications and analysed the performance of three task-level scheduling policies.

HPCQC hybrid applications comprise two compute paradigms with a sequential workflow that require rigorous synchronization during execution. To achieve high performance for these tightly coupled applications, an on-premises quantum computer would be necessary as latency plays a critical role. Moreover, state-of-the-art task-level scheduling algorithms would be required to share quantum computers among jobs and optimise system-wide resource utilisation.

Acknowledgment

Funded by the German Federal Ministry for Education and Research under grants 13N15689 (DAQC), 13N16063 (Q-Exa), 13N16188 (MUNIQC-SC), and 13N16078 (MUNIQC-ATOMS), the German Federal Ministry for Economic Affairs and Climate Action under grant 01MQ22004C (QuaST), and the Bavarian State Ministry of Science and the Arts as part of Munich Quantum Valley (MQV).

References

[1] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6:1–6:20, 2018.

[2] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020.

[3] H. Liu, G. H. Low, D. S. Steiger, T. Häner, M. Reiher, and M. Troyer, "Prospects of quantum computing for molecular sciences," *Materials Theory*, vol. 6, no. 1, p. 11, 2022. [Online]. Available: https://doi.org/10.1186/s41313-021-00039-z

[4] P. S. Emani, J. Warrell, A. Anticevic, S. Bekiranov, M. Gandal, M. J. McConnell, G. Sapiro, A. Aspuru-Guzik, J. T. Baker, M. Bastiani, J. D. Murray, S. N. Sotiropoulos, J. Taylor, G. Senthil, T. Lehner, M. B. Gerstein, and A. W. Harrow, "Quantum computing at the frontiers of biological sciences," *Nature Methods*, vol. 18, no. 7, pp. 701–709, 2021. [Online]. Available: https://doi.org/10.1038/s41592-020-01004-3

[5] A. Ajagekar and F. You, "New frontiers of quantum computing in chemical engineering," *Korean Journal of Chemical Engineering*, vol. 39, no. 4, pp. 811–820, 2022. [Online]. Available: https://doi.org/10.1007/s11814-021-1027-6

[6] IQM and ATOS, "Untangling the hpc innovation dilemma through quantum computing," International Data Corporation (IDC), Nov. 2021. [Online]. Available: https://www.meetiqm.com/uploads/documents/IQM-Atos-State-of-quantum-HPC-research-2021.pdf

[7] S. Jordan, "Quantum algorithm zoo." [Online]. Available: https://quantumalgorithmzoo.org/

[8] V. Bartsch, G. Colin de Verdière, J.-P. Nominé, D. Ottaviani, D. Dragoni, C. Bouazza, F. Magugliani, D. Bowden, C. Allouche, M. Johansson, O. Terzo, A. Scarabosio, G. Vitali, F. Shagieva, and K. Michielsen, "¡ qc — hpc ¿: Quantum for hpc," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5555960

[9] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[10] M. Ruefenacht, B. G. Taketani, P. Lähteenmäki, V. Bergholm, D. Kranzlmüller, L. Schulz, and M. Schulz, "Bringing quantum acceleration to supercomputers," 2022. [Online]. Available: https://meetiqm.com/uploads/documents/IQM_HPC-QC-Integration-Whitepaper.pdf

[11] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, pp. 625–644, sep 2021.

[12] T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowrishankar, A. Frisch, and T. Monz, "Quantum computers for high-performance computing," *IEEE Micro*, vol. 41, no. 5, pp. 15–23, 2021.

[13] M. A. Obaida and J. Liu, "Simulation of hpc job scheduling and large-scale parallel workloads," in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 920–931.

[14] K. Ahmed, M. Obaida, J. Liu, S. Eidenbenz, N. Santhi, and G. Chapuis, "An integrated interconnection network model for large-scale performance prediction," in *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM-PADS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 177–187.

[15] C. Galleguillos Miccono, Z. Kiziltan, A. Netti, and R. Soto, "Accasim: a customizable workload management simulator for job dispatching research in hpc systems," *Cluster Computing*, vol. 23, 03 2020.

[16] G. P. Rodrigo, E. Elmroth, P.-O. Östberg, and L. Ramakrishnan, "Scsf: A scheduling simulation framework," in *Job Scheduling Strategies for Parallel Processing - 21st International Workshop, JSSPP 2017, Orlando, FL, USA, June 2, 2017, Revised Selected Papers*, ser. Lecture Notes in Computer Science, D. Klusácek, W. Cirne, and N. Desai, Eds., vol. 10773. Springer, 2017, pp. 152–173.