- ORIGINAL ARTICLE -

Trends in High Performance Computing and Quantum Computing

Tendencias en Cómputo de Altas Prestaciones y Computación Cuántica

Adrian Pousa[1], Victoria Sanz[1,2], Marcelo Naiouf[1], and Armando De Giusti[1,3]

[1] *III-LIDI, School of Computer Sciences, National University of La Plata, Argentina*
{apousa,vsanz,mnaiouf,degiusti}@lidi.info.unlp.edu.ar
[2] *CIC, Buenos Aires, Argentina*
[3] *CONICET, Argentina*

Abstract

High Performance Computing (HPC) applies different techniques to complex or large-volume applications, relying on both parallel software and hardware, to reduce their execution time compared to running them on a simple computer. On the other hand, Quantum Computing (QC) emerges as a new paradigm that leverages the properties of Quantum Mechanics for computation. QC has an inherently parallel nature and it is expected to solve some problems faster than classical computing. This paper carries out a bibliographic review to examine the point of view of different authors regarding the relationship between HPC and QC. The objective is to determine the trend of this relationship: Will QC replace classical HPC computing? or Will they complement each other? Also, if they were complementary tools, the aim is to answer: How could they be integrated? How will users access these resources?.

Keywords: High Performance Computing, HPC-QC Integration, Hybrid HPC-QC Architecture, Quantum Computing

Resumen

La Computación de Alto Rendimiento (HPC) aplica diferentes técnicas a aplicaciones complejas o de gran volumen, apoyándose tanto en software como en hardware paralelos, para reducir su tiempo de ejecución en comparación con su ejecución en una computadora simple. Por otro lado, la Computación Cuántica (QC) surge como un nuevo paradigma que aprovecha las propiedades de la Mecánica Cuántica para la computación. La QC tiene una naturaleza inherentemente paralela y se espera que resuelva algunos problemas más rápido que la computación clásica. En este artículo se realiza una revisión bibliográfica para examinar el punto de vista de diferentes autores respecto a la relación entre HPC y QC. El objetivo es determinar la tendencia de esta relación: ¿QC reemplazará a la computación HPC clásica? o ¿Se complementarán entre sí? Además, si fueran herramientas complementarias, se busca responder: ¿Cómo podrían integrarse? ¿Cómo accederán los usuarios a estos recursos?.

Palabras claves: Arquitectura HPC-QC Híbrida, Computación cuántica, Cómputo de altas prestaciones, Integración HPC-QC

1 Introduction

High Performance Computing (HPC) is the practice of *aggregating* computational resources in order to improve the performance in solving large-volume problems of science, engineering or business. For that, different programming techniques are applied to create parallel algorithms that will be executed on parallel architectures, with the aim of reducing the execution time of the application. In HPC, the cluster architecture is widely used, that is, computers connected by high-speed networks. Each computer (node) in the cluster can have multiple processing units (CPUs or cores) and eventually multiple graphics cards (GPUs), which are used as coprocessors for general-purpose computing. [1, 2, 3]

Quantum Computing (QC) began in the 1980s, when Paul Benioff, Richard Feynman and David Deutsch proposed to leverage the laws of Quantum Mechanics (QM) for computation. Their major proposals are the viability of quantum computers (i.e., computers that operate under the laws of QM) and the possibility of using such systems to tackle problems that are difficult to solve on classical computers (for example, simulation of quantum systems). [4]

In order to compare the computational power of quantum and classical computers, John Preskill popularized the term *quantum supremacy*, which refers to the potential ability of quantum machines to solve problems that classical machines cannot deal with sufficient precision [5]. However, in practice, the term *quantum advantage* is used, which refers to the significant improvement in execution time achieved by a quantum algorithm compared to the best classical algorithm to solve a problem. In terms of computational

complexity this means providing a superpolynomial speedup over the best possible classical algorithm [6]. An example of this is the quantum algorithm proposed by Shor in 1995 to factorize a large integer [7].

In QC, the unit of information is the quantum bit (qubit). Unlike a classical bit, which can only take the value 0 or 1, a qubit can take the value 0, 1, or both at the same time. That is, the qubit can be in a *superposition* of two quantum states. This quantum superposition enables large-scale parallel processing. Intuitively, those computations that must be applied to both states (0 and 1) can be done in a single operation using a qubit on a quantum computer, in contrast to a classical computer that needs to perform two operations separately (one per state). [4, 8, 9]

Beyond computing, it is expected that in the near future *quantum networks*, which use the laws of QM to exchange quantum information between connected devices, will be developed. These networks rely on *teleportation*, a process that allows information to be transmitted from one place to another, even at very large distances. Teleportation is based on *entanglement*, a phenomenon by which two entangled particles behave identically regardless of the distance between them. It is said that both particles share the same state and any change in the properties of one instantly alters those of the other. Through superposition and entanglement of particles, a highly secure and efficient connection can be established, capable of overcoming the limitations of classical networks.[10]

Given the parallel nature of quantum computers and their potential ability to solve problems faster than classical computers, it is inevitable to establish a relationship between HPC and QC. This paper carries out a bibliographic review to examine the point of view of the different authors regarding the relationship between HPC and QC. Specifically, we selected several articles, from 2008 to the present, that introduce different proposals on the use of QC in HPC and we analyzed them seeking to answer questions such as: Will QC replace classical HPC? Can QC be integrated into existing HPC systems? How could such integration be carried out if possible? How will users access these resources?

The rest of the paper is organized as follows. Sections 2 and 3 summarize the fundamentals of HPC and QC respectively. Section 4 describes some works that relate HPC and QC. Section 5 analyzes those proposals to clarify the current state of HPC and QC. Finally, Section 6 presents the main conclusions and future research.

2 Fundamentals of HPC

The idea of *aggregating* computational resources in order to improve the performance in solving large-volume problems dates back to 1971 [11]. However, commodity clusters became popular in the early 1990s, thanks to a drop in computing/communication hardware prices, the extended use of the TCP/IP protocol and the emergence of development tools such as PVM [12] (later replaced by MPI [13]). This section describes the evolution of HPC systems, from their beginnings to the present.

2.1 Clusters

An HPC cluster connects several computers (nodes) through a local network (Figure 1a). The interconnection network may be Ethernet or a high-speed network such as Infiniband or Myrinet. To leverage the power of a cluster, applications must use the distributed-memory programming model. In this model, several processes are launched (one per cluster node) to solve the problem, which communicate among them by exchanging messages. The most used message passing library is MPI.

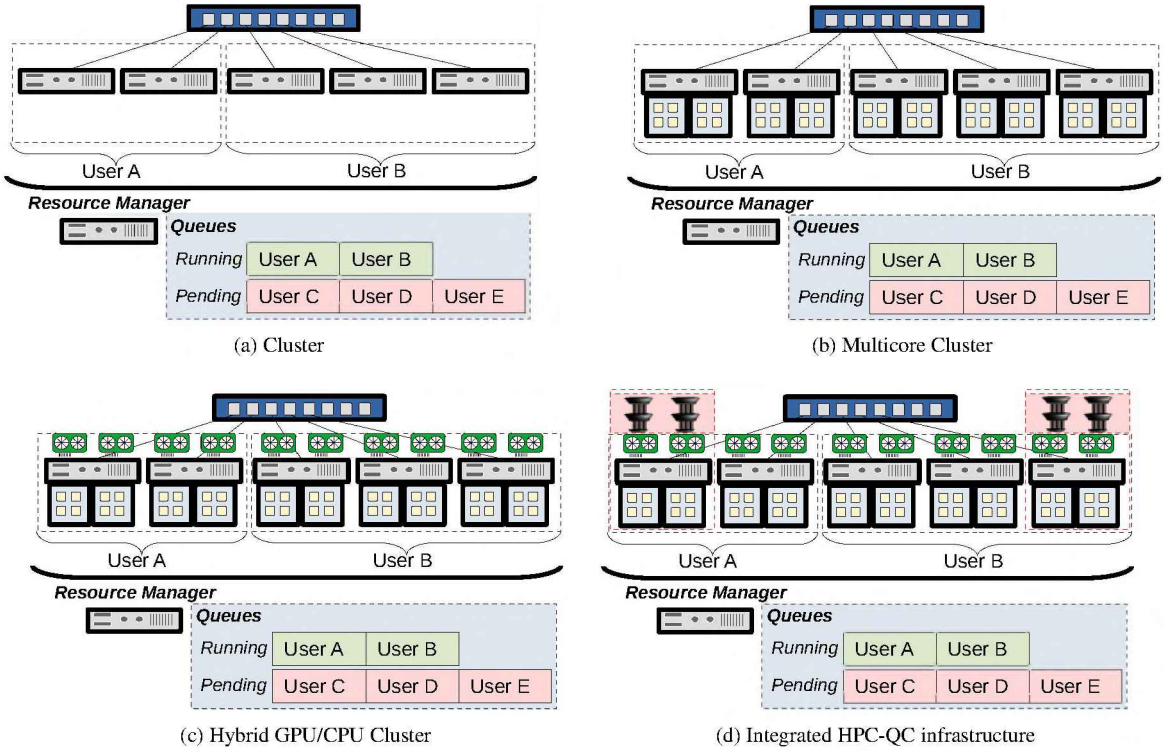
A cluster is often shared by multiple users. To run an application, each user may employ all the nodes or a subset of them, but in an exclusive way. Thus, the application will execute without being interfered by other applications. When a user employs a subset of nodes, the remaining ones can be assigned to other users.

Resource Managers, such as OpenPBS [14], Torque [15] and Slurm [16], are used to manage cluster resources. Users ask the manager for the resources needed to run their applications. If the request can be satisfied, the manager allocates the demanded nodes and launches the execution of the application. On the contrary, it queues the request and serves it later, when resources become available.

2.2 Multicore and Multicore Clusters

In the past, the evolution of processors was based on increasing clock frequency, which automatically improved the performance of applications. However, this strategy also raises the temperature of the processor and leads to a higher energy consumption to dissipate the heat. Hence, this trend came to an end in the early 2000s, with the emergence of multicore processors. Multicores include several independent processing units (cores) and improve the performance of applications designed to exploit task-level parallelism.

Multicore cluster systems (Figure 1b) benefit from having a higher number of processing units. To fully exploit multicore machines, applications should use the shared-memory programming model. In this model, several threads are created (one per core) that communicate with each other through shared memory. The most used tools for developing shared-memory applications are Pthreads [17] and OpenMP [18]. However, as an alternative, the application can use the distributed-memory programming model (i.e. running one process per core and communicating them through



(a) Cluster

(b) Multicore Cluster

(c) Hybrid GPU/CPU Cluster

(d) Integrated HPC-QC infrastructure

Figure 1: HPC systems

messages). Applications that run on a multicore cluster can use both programming models (hybrid model) and combine different tools (i.e. MPI and OpenMP or MPI and Pthreads).

As mentioned above, cluster nodes are not shared among users. Users ask the *Resource Manager* for exclusive nodes. Therefore, if a user employs only a subset of cores from the requested node, the remaining cores cannot be employed by other users.

2.3 Hybrid GPU/CPU Clusters

Graphics cards (GPUs) were designed to accelerate graphics processing, freeing the CPU from that task. In the beginning, they were composed of non-programmable processors, dedicated to specific tasks (implemented by hardware). Nowadays, GPUs include programmable processors, allowing solving general-purpose problems, which gives place to the concept of GPGPU (General-Purpose Computing on GPUs).

GPUs have demonstrated to be more efficient than multicore processors and clusters for data-intensive computing applications, which exhibit a high degree of parallelism (SIMD) and work on regular data. However, GPUs did not replace those architectures, but rather were included as coprocessors inside each cluster node (Figure 1c).

Applications that run on hybrid GPU/CPU clusters can use only the GPUs or use GPUs and CPU cores in a collaboratively way. Some articles [19, 20] show the advantage of this integration.

Regarding resource allocation, when a user asks for GPUs, the *Resource Manager* serves the request by assigning nodes that include them. If the user employs only one type of resource from each node (GPUs or CPU cores), the remaining resources cannot be employed by other users.

3 Fundamentals of QC

The qubit is the unit of quantum information. A qubit can be in one of the basic states (0 or 1) or in the superposition state (0 and 1). A qubit in a superposition state has two associated probabilities, which represent the probability of being 0 or 1 respectively. It should be taken into account that when "observing" or "measuring" a qubit in a superposition state, its state collapses towards the most probable basic state (either 0 or 1). For this reason, several runs (or *shots*) must be performed to obtain meaningful results when using a quantum algorithm. [4, 8, 9]

The superposition principle endows quantum computers with an inherent parallelism. In a quantum system of N-qubits, 2^N states can be simultaneously superposed, so that the effect of applying an operation to N-qubits is equivalent to applying it to all states in parallel.

The first quantum computers appeared in the 1990s and were limited to 2 or 3 qubits. Currently there are individual quantum machines with tens of qubits [21, 22, 23, 24]. These computers operate on qubits

using *quantum gates*, which are analogous to the logical gates used by classical computers and allow to change the state of the qubit [25]. Quantum machines must be in a controlled environment to avoid or mitigate *decoherence*, which arises from the interactions of a qubit with the environment and leads to errors in quantum information. For this reason, they require extremely low temperatures, close to absolute zero (-273 C), and need to be isolated from external signals (radio, light, electromagnetic). This explains why quantum machines are difficult to commercialize to the point that today they are provided by a few institutions and accessed through Cloud services [26].

In the same decade, the first quantum algorithms (QAs) were proposed [7, 27, 28]. A QA applies operations to the qubits, with the objective of increasing the probability of the desired states and decreasing that of the undesired ones, and ends with the measurement, which collapses the state of the system to a basic state with a certain probability, producing a classical output (bits).

4 HPC-QC: a bibliographical review

In [29] the authors address the issue of large-scale quantum information processing and propose using a High Performance Quantum Computer as a generic resource for multiple-user quantum information processing. Specifically, they suggest to use a cluster that connects quantum computers (nodes) through a network with 3D topology. This topology allows to partition the cluster into isolated regions, which can be assigned to different users for simultaneous use. In this way, conflicts that arise with traditional topologies, where the quantum network is shared among users, are avoided. Those conflicts appear because in that shared context, a user could transport information from other users, leading to errors.

A similar idea was presented in [30], which proposes: connecting quantum nodes through a quantum interconnection, taking advantage of quantum teleportation, and extending MPI to support communication in this type of architectures.

Ian Foster, a leading researcher in the area of HPC, published a short article [31], in collaboration with other authors, which relates HPC, Artificial Intelligence/Machine Learning (AI/ML), QC and communications. The authors mention that the infrastructure needed to support HPC and AI/ML shares many features. For example: GPUs originally used in HPC for general-purpose computing have become essential for AI/ML and, although QC has not reached sufficient maturity, it is expected to add a complementary computing capability to other systems. They also claim that quantum accelerators will not be suitable for all workloads. They conclude that understanding how these three technologies might complement each other and share infrastructure requires early thinking and

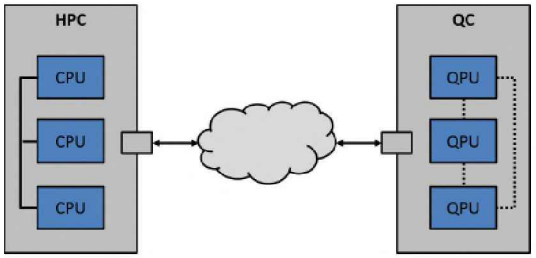


Figure 2: Loose integration between an HPC system and a QC system

planning. In addition, they highlight the current difficulty in accessing this type of infrastructure. They recommend that such infrastructure should be a combination of co-located resources at computing centers, coordinated access to specific testbeds and to commercial Cloud resources. Only in this way, the needs of the most demanding, interdisciplinary research will be met.

Other authors [32] proposed two ways of integrating Quantum Processing Units (QPUs) into HPC systems: loose and tight. The loose integration (Figure 2) is a client-server model, where the HPC system interacts with an isolated QC server via network; the QC server may be on a dedicated network or may be part of a larger computational network, and it may host multiple QPUs that interact with each other through a quantum interconnect. The tight integration (Figure 3) moves the QPUs closer to the CPUs, creating a single, tightly connected system; in this case, three alternatives are proposed: a) allowing multiple CPUs to interact with a single QPU, b) associating each CPU with a dedicated QPU, c) connecting the QPUs through a quantum network and allowing them to be accessed by multiple CPUs through interfaces. Also, the authors consider that new metrics should be developed to evaluate performance. In HPC, FLOPS (*Floating Point Operations per Second*) is a key performance metric, but it is meaningless for quantum systems, whose computational model is not based on performing floating point operations but rather on manipulating states. They propose to measure the performance of a QPU by evaluating the quantum gates applied and the amount of *shots*, since quantum algorithms are usually probabilistic.

More recently, the same authors [33] proposed integrating QPUs into HPC systems and use them as specialized accelerators, in a manner similar to what happens with GPUs. This involves a careful selection of the workloads that will be assigned to these devices. They predict that future HPC designs will be heterogeneous, with multiple accelerators coexisting within a node. However, some technological barriers challenge this integration, for example: the insulation and cooling required by QPUs, incompatible with the usual noise and vibrations of a server room, and the lack of software to support the interaction of these hardware systems. Their contribution is a description of a QPU

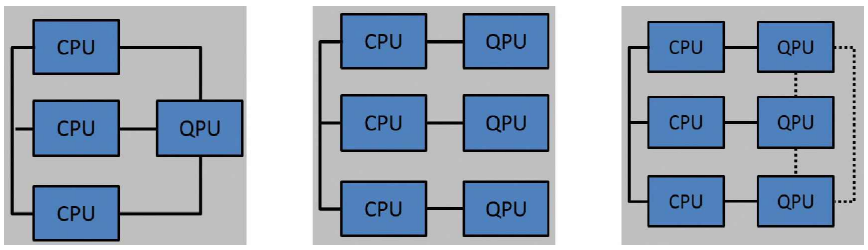


Figure 3: Tight integration of CPUs and QPUs: three alternatives

accelerator design and a language hierarchy that would make the integration possible, as explained next.

Figure 4a describes their QPU accelerator design. The QPU is composed of: a quantum control unit (QCU), quantum execution units (QEUs), and a quantum register. The QCU represents the interface between the QPU and the external system, and its role is to analyze incoming instructions and issue operands to the QEUs. Each QEU implements a subset of quantum gates operations, and different QEUs can operate in parallel. The quantum register stores quantum information (n-qubits); its measurement causes the collapse of some elements to a basic state, this is collected by the appropriate QEU and converted to a digital signal, and finally this information is transmitted to the QCU and returned to the host or collected for further processing.

Figure 4b) describes the language hierarchy to allow the host program to interact with the QPU. The highest-level language must have keywords, data types and functions that enable quantum operations; it may be an existing programming language augmented with specialized libraries or a dedicated quantum programming language [34] [35]. The program may be compiled or interpreted, and may access/control accelerators or other devices and system resources. The role of the host is to issue instructions to the QPU. The instruction set of the QPU (ISA) is very important as it abstracts its capabilities. Then, the stream of instructions and data sent to the QPU must be translated into operation codes (opcodes) that will trigger specific QEUs. Finally, the specification of how QEUs implement opcodes (gate and register usage) represents the lowest-level language (not visible to the user). Figure 5 summarizes the steps involved in the interaction.

Another more recent article [36] presents a conceptual middleware that facilitates quantum-classical integration, by providing unified resource access and management. This middleware is composed of four layers: workflow, workload, task, and resource. The workflow layer receives high-level task descriptions (classical, quantum, or composite), their dependencies, and the data they use, resolves dependencies and prepares the execution. The workload layer organizes the execution of the tasks issued by the previous layer;

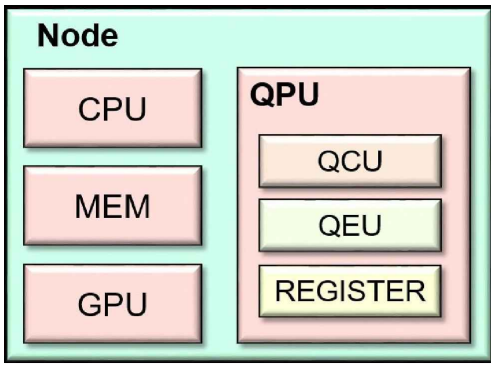


Figure 4: (a) HPC node with a QPU accelerator. (b) Language hierarchy used to program and control a QPU.

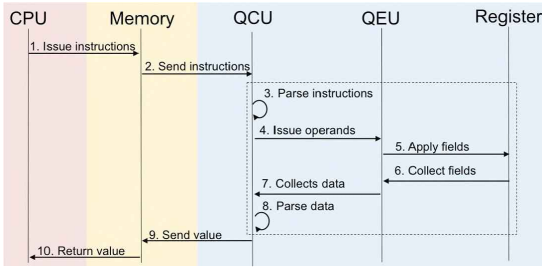


Figure 5: CPU-QPU interaction sequence diagram

depending on the tasks type (classical, quantum or composite), the workload manager selects the appropriate system resource for each task and assigns tasks to the resources. The task layer comprises several task managers, one per system resource; each one manages

the allocation, acquisition, scheduling and monitoring of local resources to ensure the tasks are executed successfully. The resource layer represents the classical (CPUs, GPUs, FPGAs, etc.) and quantum resources (QPUs) where tasks are executed.

In [37] other authors claim that quantum-classical integration is needed in order to demonstrate practical quantum advantage. They show, without much detail, a hybrid memory-centric architecture (Figure 7), integrated in an HPC data center, accessible through the Cloud and based on container technology. They implement the proposed architecture by combining real high-performance classical infrastructure (CPUs, GPUs, etc.), simulated QPUs and advanced machine learning tools, with the aim of experimenting with hybrid quantum-classical algorithms. They say that quantum circuits are hard to simulate classically, since the time and memory required scale exponentially with the number of qubits. However, they highlight that some classes of algorithms scale more favourably, although still exponentially, for specific sets of quantum circuits (e.g. tensor networks). They run optimization, machine learning and simulation algorithms on the presented hybrid architecture and demonstrate that the hybrid quantum-classical approach significantly outperforms the classical approach, for these algorithms.

Another recent review article [38] states that advancements in QC have made clear that it will not replace conventional HPC, but can rather be integrated into current heterogeneous HPC infrastructures, as an additional accelerator, thus allowing the optimal use of both paradigms. The authors study several quantum programming tools (QPTs) and investigate whether they can be integrated with existing HPC tools. They conclude that existing QPTs only partially cover the requirements needed for full HPC-QC integration and that future work on QPTs should consider integration as a crucial aspect and provide a simple way to use them in an HPC context.

5 Discussion

Some authors seem to think about the replacement of HPC systems with their quantum equivalent [29, 30], by proposing the use of quantum clusters without integrating them with classical HPC architectures. On the other hand, the remaining articles make it clear that QC will not replace traditional HPC, but rather will be complementary tools.

In [32] the authors propose hybrid quantum-classical architectures, which may even integrate multiple interconnected QPUs. However, current research efforts are focus on the manufacture of a single, more efficient quantum machine and it seems difficult to have, in the short term, interconnected quantum machines.

Foster [31] suggests that QC will not be suitable for all workloads, and thus will complement classical HPC architectures when necessary. Something similar happened when GPUs began to be used for general-purpose computing. These boards did not replace clusters or multicore processors, but rather were integrated as accelerators for specific tasks. Some articles [19, 20] show the advantages of CPU-GPU collaborative computing. Britt [32] raises the possibility of using quantum machines as coprocessors, as happens with GPUs. Perelshtein [37] demonstrates the practical advantage of this approach by running specific algorithms on a hybrid architecture, which combines high-performance classical infrastructure (CPUs, GPUs) and simulated QPUs. Likewise, Britt [32] claims that new metrics, based on quantum gates and the number of shots, should be considered.

With regards to HPC-QC integration, the articles [32, 33, 36, 38] include proposals about how to integrate these architectures, the use of middlewares, the management of quantum-classical resources, and the integration of quantum programming tools with existing HPC tools. Today, quantum computers require a controlled environment and are not integrated with classical HPC systems. For this reason, they can only be accessed individually through Cloud services.

HPC-QC integration involves several challenges. Britt [33] explains some of them: the incompatibility between quantum and classical computing technologies, and the effort to manage the interaction between both systems and errors in quantum information.

In conclusion, HPC-QC integration should be considered as a crucial aspect, HPC-QC infrastructure should be easily accessed by users and easy-to-program. [31, 38]

6 Conclusions and Future Work.

From the bibliographic review carried out in this work, we can conclude that classical HPC systems and quantum machines should be used in a complementary manner. In particular, quantum devices should act as coprocessors, as happens today with GPUs. QC will be used for solving specific problems, for which it is possible to obtain a significant improvement in performance, with respect to classical HPC solutions. Today, quantum machines are accessed through Cloud services and are not integrated with classical HPC systems. There are several challenges related to the incompatibility between classical-quantum technologies and the fault-tolerance ability for long-running applications. Achieving a successful HPC-QC integration requires vendors to provide easy access to the infrastructure and user-friendly programming tools.

As future work, we plan to develop quantum algorithms and compare their performance with that of their classical counterparts. Although currently available quantum computers have few qubits, they are useful to determine what types of workloads are more benefited by QC. In the near future, the number of

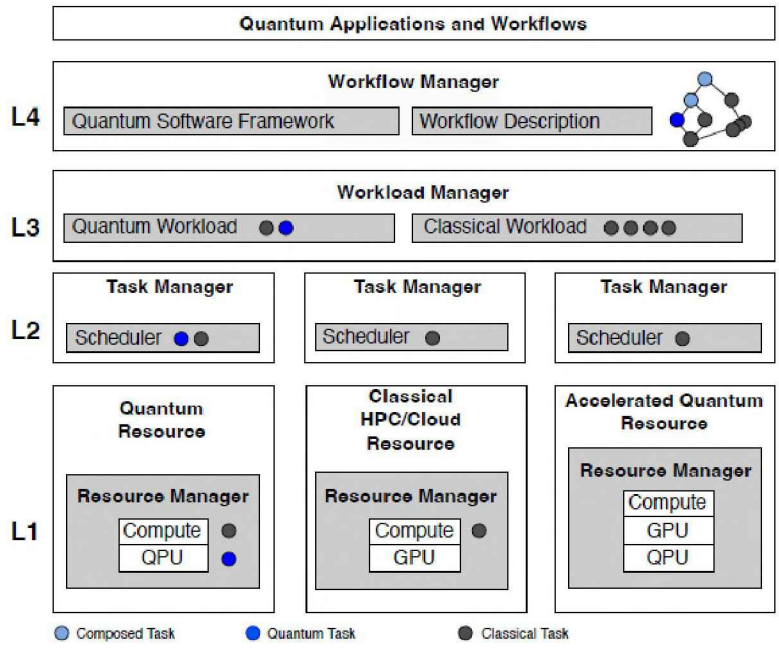


Figure 6: Conceptual middleware to facilitate quantum-classical integration

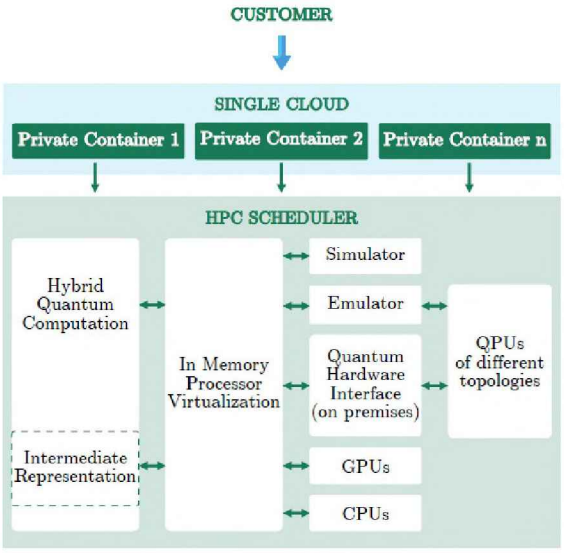


Figure 7: Hybrid classical-quantum architecture accessible through the Cloud

qubits is expected to increase exponentially. With a significant number of qubits, it will be possible to study the scalability of certain algorithms.

Competing interests

The authors have declared that no competing interests exist.

Authors' contribution

The authors confirm contribution to the paper as follows:

(AP, VS): Conceptualization, Methodology, Investigation, Validation, Redaction - original manuscript, Redaction - review and editing

(MN, ADG): Supervision

All the authors have read and approved the final version.

References

[1] A. Grama, A. Gupta, G. Karypis, V. Kumar, "An Introduction to Parallel Computing", 2nd ed., Addison Wesley, 2003.

[2] I. Foster, "Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering", Pearson, 2023.

[3] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, A. White, "Sourcebook of Parallel Computing", Morgan Kaufmann, 2003.

[4] S. Golestan, M. R. Habibi, S. Y. Mousazadeh Mousavi, J. M. Guerrero, J. C. Vasquez, "Quantum computation in power systems: An overview of recent advances", Energy Reports, Volume 9, Pages 584-596, 2023. DOI: https://doi.org/10.1016/j.egyr.2022.11.185

[5] J. Preskill, "Quantum computing and the entanglement frontier", arXiv preprint arXiv:1203.5813v3, 2012. DOI: https://doi.org/10.48550/arXiv.1203.5813

[6] A. Papageorgiou, J. F. Traub, "Measures of quantum computing speedup", arXiv preprint arXiv:1307.7488v1, 2013. DOI: https://doi.org/10.48550/arXiv.1307.7488

[7] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", arXiv:quant-ph/9508027v2, 1995. DOI: https://doi.org/10.48550/arXiv.quant-ph/9508027

[8] C. Hughes, J. Isaacson, A. Perry, R. F. Sun, J. Turner, "What Is a Qubit?. In: Quantum Computing for the Quantum Curious", Springer, Cham, 2021.

[9] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions", Softw: Pract Exper, 52(1): 66-114, 2020. DOI: https://doi.org/10.48550/arXiv.2010.15559

[10] B. Lackey, "Quantum networking: A roadmap to a quantum internet", Microsoft Azure Quantum Blog. Accessed: Feb. 29, 2024. https://cloudblogs.microsoft.com/quantum/2023/11/01/quantum-networking-a-roadmap-to-a-quantum-internet/

[11] G. Hager, G. Wellein, "Introduction to HPC for Scientists and Engineers", CRC Press, 2010.

[12] Parallel Virtual Machine (PVM), https://www.epm.ornl.gov/pvm/pvm_home.html

[13] Message Passing Interface (MPI), https://www.mpi-forum.org/

[14] OpenPBS Resource Manager, https://www.openpbs.org/

[15] Torque Resource Manager, https://adaptivecomputing.com/cherry-services/torque-resource-manager/

[16] Slurm Resource Manager, https://slurm.schedmd.com/

[17] Posix Threads (Pthreads), https://posix.opengroup.org/

[18] Open Multi-Processing (OpenMP), https://www.openmp.org/

[19] V. Sanz, A. Pousa, M. Naiouf, A. De Giusti, "Accelerating Pattern Matching with CPU-GPU Collaborative Computing", Algorithms and Architectures for Parallel Processing, ICA3PP 2018, Lecture Notes in Computer Science, Vol 11334, Pages 310–322, Springer, Cham, 2018. DOI: https://doi.org/10.1007/978-3-030-05051-1_22

[20] V. Sanz, A. Pousa, M. Naiouf, A. De Giusti, "Efficient Pattern Matching on CPU-GPU Heterogeneous Systems", Algorithms and Architectures for Parallel Processing, ICA3PP 2019, Lecture Notes in Computer Science, Vol 11944, Pages 391–403, Springer, Cham, 2020. DOI: https://doi.org/10.1007/978-3-030-05051-1_22

[21] Amazon Braket: Quantum Computers, QuEra, https://aws.amazon.com/es/braket/quantum-computers/quera/

[22] Amazon Braket: Quantum Computers, Rigetti, https://aws.amazon.com/es/braket/quantum-computers/rigetti/

[23] IBM: Quantum Computing Technology, https://www.ibm.com/quantum/technology

[24] Google, Quantum AI.: Quantum Computer Datasheet, https://quantumai.google/hardware/datasheet/weber.pdf

[25] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, H. Weinfurter, "Elementary gates for quantum computation", arXiv:quant-ph/9503016v1, 1995. DOI: https://doi.org/10.48550/arXiv.quant-ph/9503016

[26] Amazon Braket, https://aws.amazon.com/braket/

[27] D. Deutsch, R. Jozsa "Rapid solution of problems by quantum computation", Proceedings of the Royal Society of London A439:553, 1992. DOI: https://doi.org/10.1098/rspa.1992.0167

[28] L. K. Grover, "A fast quantum mechanical algorithm for database search", Proceedings, 28th Annual ACM Symposium on the Theory of Computing, 1996. DOI: https://doi.org/10.48550/arXiv.quant-ph/9605043

[29] S. J. Devitt, W. J. Munro, K. Nemoto, "High Performance Quantum Computing", arXiv preprint arXiv:0810.2444, 2008. DOI: https://doi.org/10.48550/arXiv.0810.2444

[30] T. Haner, D. S. Steiger, T. Hoefler, M. Troyer, "Distributed Quantum Computing with QMPI", Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2021, Article No.: 16, 2021. DOI: https://doi.org/10.48550/arXiv.2105.01109

[31] S. Banerjee, I. Foster, W. Gropp, "Infrastructure for Artificial Intelligence, Quantum and High Performance Computing", arXiv:2012.09303, 2020. DOI: https://doi.org/10.48550/arXiv.2012.09303

[32] K. A. Britt, T. S. Humble, "High-Performance Computing with Quantum Processing Units", ACM Journal on Emerging Technologies in Computing Systems, Volume 13 Issue 3 Article No.: 39 pp 1–13, 2015. DOI: https://doi.org/10.1145/3007651

[33] K. A. Britt, F. A. Mohiyaddin, T. S. Humble, "Quantum Accelerators for High-Performance Computing Systems", arXiv preprint arXiv: 1712.01423, 2017. DOI: https://doi.org/10.48550/arXiv.1712.01423

[34] P. Selinger, "Towards a quantum programming language", Mathematical Structures in Computer Science, vol. 14, pp. 527–586, 8, 2004. DOI: https://doi.org/10.1017/S09601295040042

[35] D. C. Steiger, T. Haner, M. Troyer, "Projectq: An open source software framework for quantum computing", arXiv preprint arXiv:1612.08091, 2016. DOI: https://doi.org/10.48550/arXiv.1612.08091

[36] N. Saurabh, S. Jha, A. Luckow, "A Conceptual Architecture for a Quantum-HPC Middleware", arXiv preprint arXiv:2308.06608v1, 2023. DOI: https://doi.org/10.48550/arXiv.2308.06608

[37] M. Perelshtein, A. Sagingalieva, K. Pinto, V. Shete, A. Pakhomchik, A. Melnikov, F. Neukart, G. Gesek, A. Melnikov, V. Vinokur, "Practical application-specific advantage through hybrid quantum computing", arXiv preprint arXiv:2205.04858, 2022. DOI: https://doi.org/10.48550/arXiv.2205.04858

[38] A. Elsharkawy, M. Xiao-Ting, P. Seitz, Y. Chen, Y. Stade, M. Geiger, Q. Huang, X. Guo, M. A. Ansari, C. B. Mendl, D. Kranzlmüller, M. Schulz, "Integration of Quantum Accelerators with High Performance Computing – A Review of Quantum Programming Tools", arXiv preprint arXiv:2309.06167, 2023. DOI: https://doi.org/10.48550/arXiv.2309.06167

Citation: A. Pousa, V. Sanz, M. Naiouf and A. De Giusti. *Trends in High Performance Computing and Quantum Computing*. Journal of Computer Science & Technology, vol. 24, no. 2, pp. 111–119, 2024.
DOI: 10.24215/16666038.24.e11
Received: April 15, 2024 **Accepted:** May 24, 2024.
Copyright: This article is distributed under the terms of the Creative Commons License CC-BY-NC-SA.