DEPARTMENT: EXPERT OPINION

Quantum Computers for High-Performance Computing

Travis S. Humble, Alexander McCaskey, Dmitry I. Lyakh, and Meenambika Gowrishankar, *Quantum Computing Institute, Oak Ridge National Laboratory, Oak Ridge, TN, 37830, USA*

Albert Frisch, *Alpine Quantum Technologies, Innsbruck, 6020, Austria*

Thomas Monz, *Alpine Quantum Technologies, Innsbruck, 6020, Austria and also Institut für Experimentalphysik, Universität Innsbruck, Innsbruck, 6020, Austria*

Quantum computing systems are developing rapidly as powerful solvers for a variety of real-world calculations. Traditionally, many of these same applications are solved using conventional high-performance computing (HPC) systems, which have progressed sharply through decades of hardware and software improvements. Here, we present a perspective on the motivations and challenges of pairing quantum computing systems with modern HPC infrastructure. We outline considerations and requirements for the use cases, macroarchitecture, microarchitecture, and programming models needed to integrate near-term quantum computers with HPC system, and we conclude with the expectation that such efforts are well within reach of current technology.

High-performance computing (HPC) systems define the pinnacle of modern computing by drawing on massively parallel processing. This leading paradigm for HPC often relies on specialized accelerators and highly tuned networks to optimize data movement and application performance, whereby many computational nodes are connected by high-bandwidth networks to support shared information processing tasks. Existing computational nodes also support highly concurrent execution with multithreaded processing, and technology trends indicate that future node designs will integrate heterogeneous processing paradigms that include conventional central processing units (CPUs), graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and other specialized processors.[1] The components of these future computational nodes must be tightly integrated to balance data movement with processing power and workload in order to optimize overall system performance.

By comparison, quantum computers (QCs) represent a young yet remarkable advance in the science and technology of computation that are often cited as rivals or successors to state-of-the-art conventional high-performance computing (HPC) systems. The source of this proposed advantage of QCs derives from quantum information processing in which information is encoded in the quantum state of physical systems such as atoms, electrons, and photons.[2] These quantum physical systems present the unique features of quantum coherence and quantum entanglement that permit quantum computing to reduce exponentially the computational time and memory needed to solve many problems from chemistry, materials science, finance, and cryptanalysis among other application domains. The advantage afforded to quantum computing is therefore aptly named the "quantum computational advantage," and there is now a fervent effort to realize quantum computing systems that demonstrate this advantage. Notably, recent efforts have focused on besting the world's leading HPC systems to great effect.[3,4,5]

Many of the most promising applications of quantum computing overlap strongly with existing applications of HPC,[6] which begs the question of how QCs may be integrated with modern HPC to accelerate these

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3099140
Date of current version 14 September 2021.

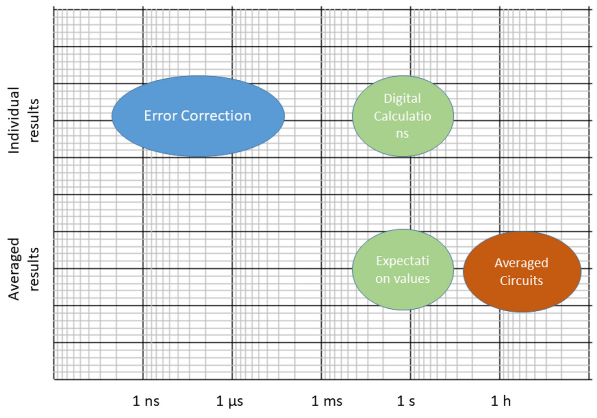

FIGURE 1. Applications of quantum computing categorized by typical execution times. Low-latency, in-sequence processing applications (blue) include quantum error correction and probabilistic state initialization methods that control decisions on nanosecond timescale. Single-circuit applications (green) only return at the end of computation on second timescales. Ensemble-circuit applications (orange), such as application benchmarking or device verification and validation of quantum devices, may take up to hours or more of computational time.

computations. Integrating QC into existing HPC infrastructure prioritizes the use of QCs to address existing computing challenges while leveraging the plethora of existing HPC tools, systems, and workflows for these application contexts. In particular, QCs may be viewed as computational accelerators that integrate into existing HPC systems akin to the current design of leading HPCs using combinations of CPUs and GPUs.

The migration of QCs toward integrated HPC will require many advances in micro- and macro-architecture that address concerns in the difference in infrastructure and performance.[7] For example, quantum processing units (QPUs) represent an early vision of components that may integrate as accelerators for computational nodes. However, existing QC prototypes are based on loosely integrated client–server interactions that lack the sophistication or technological maturity to be used as accelerators.[8] In addition, communication between multiple QPUs lacks the networking protocols needed to support concurrent processing models. This includes both conventional as well as quantum networking. Although existing networking architecture may be leveraged for this purpose, there are outstanding questions about the balance between performance and workload that drive the development of prototype systems.

Here we examine how much of the underlying infrastructure for HPC will benefit future QC systems including data management, process scheduling, control, networking, and user management. We discuss the possibilities and requirements for combining QCs with HPC, based on an assumption of close physical proximity between the QC and HPC systems. We consider the possibility for integrating QCs both through cloud-based access and tight integration. We account for acceptable latency in different use-cases and address technical approaches for how these requirements can be met.

APPLICATIONS OF QUANTUM COMPUTING

There is a wide range of promising application use cases for QC, and many of these overlap with the existing uses of HPC. The utilizations include the simulation of high-dimensional physical models, the design, verification, and validation of complex logical systems, and inference, pattern matching, and search over large datasets. Underlying these use cases are algorithms that take advantage of the logic from both conventional computing and quantum computing, and we defer those details to other reviews.[6] Instead, we highlight several issues that arise when considering the micro- and macro-architectures of these hybrid computing models, including characteristics of the quantum devices that support these calculations and how those devices may interface with conventional HPC nodes.

We will coarsely classify applications of quantum computing in terms of the expected clock-speed of the QPU, the acceptable communication and control latency within the accelerated node, and the amount of data typically transmitted. We will term these use cases as 1) in-sequence processing, 2) single-circuit applications, and 3) ensemble circuit applications. A graphical representation of this classification is shown in Figure 1, and we include a detailed description below.

In-Sequence Processing

This grouping represents applications, which demonstrate a need for low-latency communication and control decisions for successful execution of a quantum program. The selection includes applications of quantum computing that require individual outcomes calculated by the QPU to be transmitted and processed during the remaining run time of the quantum computation. Two prominent examples are the operation of quantum error correction (QEC) and, more generally, the conditional preparation of quantum states based on intermediate measurements. For some existing applications of QEC, it is necessary to process intermediate information, i.e., syndrome measurements to identify the error syndrome that arises during an encoded computation. The resulting syndrome analysis then leads to new control signals that modifies, i.e., "corrects," the remaining quantum operations. In a similar way, the quantum register within the QPU may be reinitialized during run time as a result of a previous measurement outcome, e.g., for efficient quantum state preparation. Conditional state preparation processes issue new instructions and pulses to modify the execution sequence. A similar paradigm also applies to measurement-based quantum computing, which is of particular interest in blind quantum computing for future high-security applications.

For example, consider a hypothetical QPU with a few 1,000 encoded qubits that support error correction. Each of these encoded qubits shall consist of 100 physical qubits, where 25 ancilla measurements each characterize the errors in a single encoded qubit. The control and preparation of these encoded qubits must occur faster than the typical decoherence timescale that characterizes the QPU technology. The target feedback time may be 10 ns for superconducting technologies and 10 μs for trapped-ion technologies. The QPU must evaluate and transfer the peak rates of 10 Tbps and 10 Gbps, respectively, with latencies of a few nanoseconds or microseconds.

These requirements may be met, in principle, using existing technologies. For example, the communication buses that connect existing GPUs with each other and with CPU main memory can support up to about 50 Gbps per lane. Similarly, state-of-the-art Ethernet transport may meet the requirements cited for trapped ions. While these examples may meet the data transfer and latency requirements, existing technologies may not be directly suitable for quantum computing environments. In particular, the QPUs may be installed in cryogenic environments with demanding heating restrictions to prevent additional noise. Indeed, these examples suggest that the in-sequence processing requirements will only be realistic for on-premise installation of a QC within an HPC node.

Single-Circuit Applications

This grouping represents the many current use-cases for quantum computing that are based on single-circuit evaluations. This group includes domain applications, e.g., analog quantum simulations, variational methods for chemistry, materials science, and high-energy physics simulations, optimization and combinatorial problems, as well as many others. The notable features of these single-circuit applications is that the underlying quantum program is static during execution, does not change its state based on intermediate measurement outcomes, and completes circuit execution before returning results. In general, such programs may be executed repeatedly to generate a distribution of measurements and a resulting statistical characterization, but such choices do not change the requirements placed on the control of the QPU.

In addition, many single-circuit application may iteratively modify the circuit or parameters. Key examples include variational and adaptive methods. Individual results are only dependent on a single execution instance and independent execution of these multiple circuits instances is possible, i.e., an example of hybrid parallelism. Consequently, the single-circuit application use case is well aligned with higher latency communication between the host node managing task delegation and the QPU performing that task.

As a specific example, current cloud-based models of quantum computing rely on a host computer to distribute self-contained quantum programs to dedicated but remote QC hardware. These programs are relatively small in size as measured by bits and correspondingly generate a modest amount of information, which is due to almost entirely the limited size and coherence lifetimes of existing QC hardware. Additional metadata may expand the data collected. Execution of the distributed programs is often moderated by a server task running on conventional computing hardware, in which a queue supports multiprogram and multiuser operations. We expect these runtime models to become more sophisticated through tighter integration of the conventional control system with the QC hardware components.

The resulting latency of the server, which includes queuing, context switching between job executions, and the necessary circuit compilation process, may be on the order of seconds and sometimes longer depending on overall demand. However, this latency does not influence the quantum computation itself. Instead, the requirement for the server to transmit and receive several kilobytes of information over the course of several seconds is easily met by existing Ethernet technologies.

Ensemble-Circuit Applications

The final grouping of applications that we consider represent programs that require averaging of results over an ensemble of different circuit instance. Unlike single-circuit applications, this grouping represents those methods that require results collected from several different circuits in order to perform a joint evaluation. These include methods for characterization, verification, validation, and debugging of QPUs, which may be used to certify computational performance in the process of

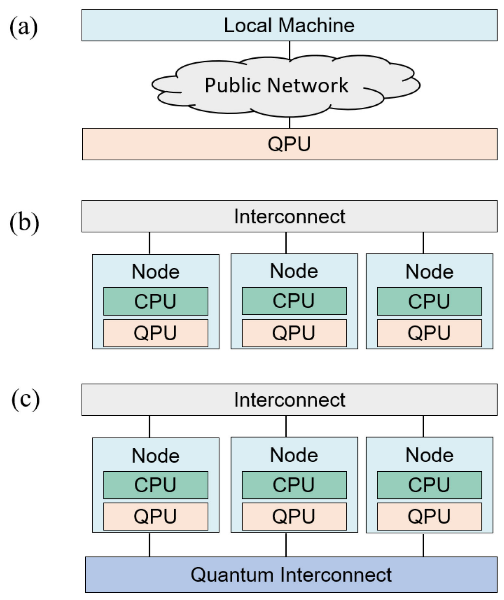

FIGURE 2. Three macroarchitectures for integrating quantum computing with conventional computing. (a) A local machine remotely accesses a QPU through public cloud network. (b) A network of quantum-accelerated nodes communicate through a common interconnect. (c) A network of quantum-accelerated nodes communicate through both conventional and quantum networks.

meeting a known standard. For example, several recent experimental demonstrations of quantum computational advantage have relied on ensemble circuit evaluations for comparison with results from conventional HPC simulations of those same circuits.

Ensemble-circuit applications lend themselves to highly parallel execution as the circuit instances can be issued independently. However, aggregation of the results from all circuit instances is necessary for postprocessing to complete the calculation, e.g., as in benchmark or characterization results. Managing these parallel tasks with existing multithreading and interprocess methods offer a convenient means of implementing traditional fork-join programming models. These parallelized approaches offer opportunities to manage the near-concurrent, high-latency sampling of many different circuits.

The data requirements for ensemble-circuit applications may range from a linear scaling of the requirements for single-circuit applications to more complex scaling based on result-driven feedback. An example of the latter is reflected in iterative characterization methods, such as adaptive process tomography, or hierarchical testing methods for complex quantum programs. In addition, postprocessing of these resulting datasets may require extensive computational resources.

MACROARCHITECTURE DESIGN

We now examine the possible macroarchitectures integrating QCs with conventional computing systems and, especially, HPC systems. In Figure 2(a), we present the current example of hybrid computing in which a remotely accessible QPU is dedicated to executing standalone quantum programs. The local machine system coordinates interactions with the QPU through a public network but details of the conventional computing system are indistinct for the purposes of these interactions. This macroarchitecture is representative of current interactions with commercial QPUs, in which the connection is mediated by wide-area networking such as the Internet. This type of integration is due to the ease with which the devices are physically connected as well as the simplicity with which programming and execution are implemented. While this macroarchitecture can meet the high latency requirements of single-circuit and ensemble-circuit applications, it may not be optimal due to limitation on the number of processes that can be executed. In addition, Figure 2(a) offers poor support for in-sequence applications due to the high-latency connection.

A leading example of a more performant architecture is shown in Figure 2(b), where a network of conventional computing nodes individually interface with a QPU. This design permits coordination of processing and data across low-latency connections. This architecture also supports the paradigm of parallel processing in which the decomposition of both single-circuit and ensemble-circuit workloads into concurrent tasks may accelerate time to solution. In addition, each task may be allocated to a resource that supports acceleration based on quantum computing capabilities.

The coordination of these multiple QPU resources is orchestrated through the communication of instructions and data across the system. Message passing protocols, such as message passing interface (MPI) and parallel virtual machine (PVM), have been used previously to program distributed nodes and similar approaches seem highly feasible for these quantum-accelerated variants. However, defining interfaces for the protocols including data types and operations will be important for this approach. Anticipating future heterogeneous HPC nodes, these interfaces may not be developed in isolation but must include a diversity of computational accelerators including GPUs, FPGAs, tensor processing units, and neuromorphic processors.[1]

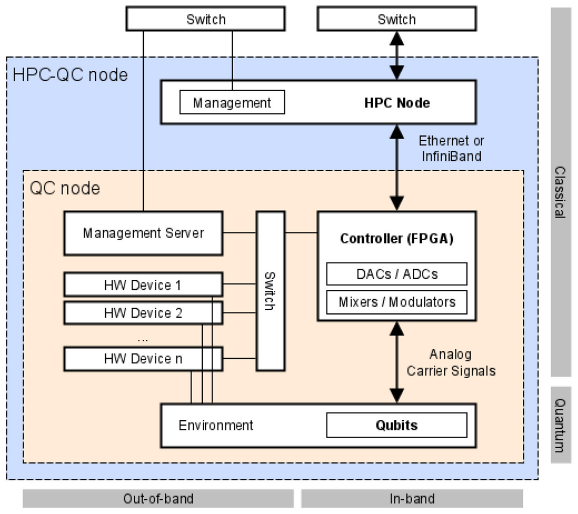

FIGURE 3. A component diagram representing the microarchitecture of a HPC-QC node with a common interconnect as depicted in Figure 2(b). The diagram shows the major components needed for the operation of a QPU within the HPC node infrastructure. Individual components are grouped into so-called out-of-band and in-band scopes and are placed on the left-hand and right-hand side of the figure, respectively. The QPU, which contains the qubits and is capable of processing quantum information, is depicted at the lower part, whereas classical information processing components are shown in the upper part of the figure. Several hardware (HW) devices control the QPU environment, which has a direct effect on qubit properties and thus the quality of execution of instructions.

We mention a third macroarchitecture for hybrid computing that extends the distributed design toward a distributed quantum computing system. As shown in Figure 2(c), this approach uses a quantum interconnect as a significant advance in the computational capability of these distributed QPUs as entangling operations across computational workloads would be possible. This design open up many new application possibilities by permitting entangling operations between nodes. However, such advances are expected to introduce additional requirements for synchronous quantum communication and we defer that analysis to elsewhere.

Designing and forecasting the performance of these hybrid HPC systems will require significant system-level modeling and simulation efforts. In the absence of widely available production-ready QPUs, subsystem-level virtualization is needed to simulate execution of quantum algorithms using classical processors (embedding virtual QPUs into a classical HPC system by making some of the nodes run QPU simulators). We anticipate upcoming exascale HPC platforms will prove critical for simulating system models of hybrid HPC systems including the conventional CPU as well as QPU operations. In addition, the flexibility provided by hybrid virtualization will permit dynamic reconfiguration of the HPC platform in which the granularity, composition, and connectivity of the classical and quantum computing units can be fine tuned to a given quantum/classical workload. Importantly, the ultimate realization of these systems is likely to evolve in stages and the necessary specialized classical hardware components, e.g., FPGA controllers, may be gradually integrated as the systems progress toward full functionality.

MICROARCHITECTURE DESIGN

The node design presented above must account for the conventional CPU and memory systems (for data processing and resource management) as well as the real-time control electronics (typically state-of-the-art FPGA circuits) and digital-to-analog/analog-to-digital converters that comprise the QPU. As shown in Figure 3, the microarchitecture of this hardware stack may be viewed as an encapsulated execution unit. The role of this subsystem is implementation of QPU instructions and facilitating in-band signaling. Outputs from the execution unit to the quantum register are control channels for modulating independent carrier signals of each register element via analog upsampling/downsampling stages (mixers or modulators). The carrier frequencies depend on the specific quantum computing platform used and range from microwave (few gigahertz) for superconducting qubits to optical frequencies (hundreds of terahertz) for trapped-ion qubits. Thus, the execution unit translates digital instructions into analog pulses and this translation may be tuned with respect to the instruction set architecture as well as the physical condition of the quantum register.

For the macroarchitectures in Figure 2, quantum execution units are internal to the QPU.[7] Presently, FPGAs have proven to be highly effective implementations of the execution units needed to implement instructions on prototype QPUs. In addition to being reconfigurable, FPGAs offer relatively low-latency responses. Figure 3 shows a quantum-enabled node that contains multiple components next to the execution unit to manage and control the operation of the QPU. Due to the large number of environmental effects (e.g., electromagnetic fields, vibrations, temperature, etc.) on the qubits and the complexity of controlling the environment to a suitable precision for reliable operation, a management server and several additional hardware (HW) devices are needed within the QC node. These components can be ascribed to the out-of-band management, which is well known from system management of classical computing systems.

Currently, FPGAs are the most common hardware used to interface between the HPC and QC node. They sit physically close to the qubits providing commands locally. The device controls the analog systems that directly control the qubits. The most important feature of FPGAs, as mentioned above, is their programmability. However, access to these devices is unavailable currently to algorithm developers. One way to increase access and improve performance would be to allow computational access directly on the FPGA level. Operations such as parameter updates and logical while-loops could be programmed into the system at the FPGA or similar device level and as a result, improve the latency bottleneck caused by the delays in the movement of data between the entire stack of the HPC and QC system.[9]

Modulated carrier signals resemble the most fundamental qubit operations, which are commonly defined as pulses in the software stack. Pulses are the building blocks for quantum logic gates, which by themselves are building blocks for quantum circuits. Quantum gates (operations) can be classified in single-qubit and multi-qubit gates as well as the measurement operation. A measurement on a qubit retrieves one bit of classical information in a digital form. Typical single-qubit gate durations range from several nanoseconds for superconducting qubits to microseconds for trapped-ion qubits, respectively, whereas multiqubit gates and measurements are orders of magnitude longer due to their complexity in both technologies.

A measurement-based feedback control of qubits during the execution of a quantum circuit is enabled by conditional gates, for which their condition depends on a logic input derived from the result of a measurement (same or different qubit) and/or additional classical bits as inputs. This quantum feedback sets the requirement on the latencies of digital communication channels throughout the real-time control electronics stack and might be difficult to achieve, e.g., in case of systems with a large number of superconducting qubits.

For quantum circuits without conditional gates, the latency requirement is relaxed (high-latency) because a single circuit is executed as an immutable sequence of circuits. Due to its wide availability and simplicity, currently communication channels between the compute node and the real-time electronics are accomplished via an Ethernet link. Low latency communication channels might be realized by different cable-based standards, e.g., InfiniBand, which are common in networking communications of HPC environments. This allows remote direct memory access into the memory of the FPGA, which significantly reduces the communication latency overhead for executing quantum circuits. Furthermore, these channels still allow to share resources by creating networks of HPC nodes and QC hardware systems. Even further integration of the QC hardware into HPC infrastructure by placing the real-time control electronics on the same device might enable the usage of wire-based near-range communication protocols, e.g., PCI Express or NVLink, in case even lower latencies and high data rates are required.

PROGRAMMING MODELS

The software stack for enabling QC integration with HPC will require extensibility and modularity.[10] Existing variability in quantum technologies (superconducting, trapped-ion, etc.) from competing hardware vendors (Google, IBM, Rigetti, Honeywell, etc.) requires customization at all levels of abstraction. Here we distill those demands into a number of service interfaces for programming, compilation, execution, and control of QPUs. Service interfaces provide a means for deploying complex software systems that grow organically as new hardware, programming, control, and compilation techniques are developed by the community. Ultimately, we advocate for a hardware-agnostic approach that decomposes software into extensible interfaces for host-to-control integration, compiler, language, and algorithmic libraries,[11] as illustrated in Figure 4.

Language-level constructs for QPUs are enabled by pulse-level instruction that execute a robust quantum intermediate representation (QIR) by translating domain specific or stand-alone languages into hardware actions. For N languages and M backends, the QIR enables $N + M$ implementations and avoids writing a parser-to-device implementation for all available languages and backends. Custom digital or analog instructions may then be mapped directly through the IR to a control system instantiation. The QIR uniquely represents these quantum expressions at a level that is higher than the native QPU instructions and can present itself in a number of forms: 1) an in-memory object model defined as an *n-ary* tree that is uniquely

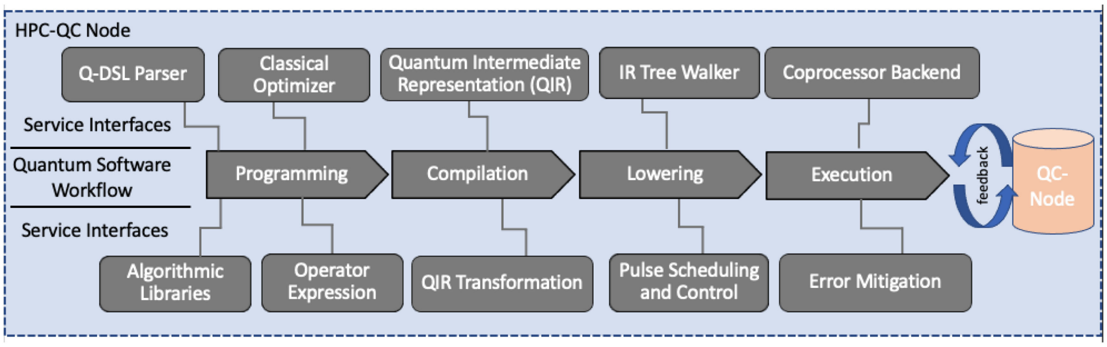

FIGURE 4. Decomposition of the software architecture required for quantum-HPC integration into a series of workflow steps, each exposing a unique set of service interfaces. This architecture maximizes the flexibility, modularity, and extensibility of the suggested integration strategy. Here we show the workflow decomposed into programming, compilation, IR lowering, and execution components. Each component exposes a series of service interfaces intended for the implementation of concrete use cases.

situated for analysis and transformation; 2) a graph that demonstrates quantum instruction control-flow; 3) a human-readable assembly string; or 4) a persisted bitcode or binary file.

The QIR provides an extension point for host-to-control interaction. Each QPU vendor provides a unique control system for the execution of quantum instructions. The interaction may be managed through a service interface that exposes the quantum instruction API for specific QPUs and maps QIR expressions to native instruction sets. These interfaces may be further configured for application uses cases, hardware support, and quantum compilation that address quantum resource optimization, placement and routing, and error mitigation and QEC. The transformations take as input an instance of the QIR and output a modified instance.

High-level programmability of the CPU-QPU hybrid system needs performant language approaches that permit creation of new quantum algorithmic primitives.[12] Software protocols for quantum acceleration in HPC environments will also need to be compatible with existing HPC applications, tools, compilers, and parallel runtimes. This implies the need for hybrid software approaches to provide system-level languages, like C++, with appropriate bindings to higher level application languages like Python or Julia. Providing an infrastructure based on C++ will provide the performance necessary to integrate future QPUs and enable in-sequence instruction execution for utilities and methods that require fast-feedback. A C++-like language is multiparadigm (object-oriented, functional, etc.) and enables integration with existing HPC workflows. Of course, these language extensions will benefit from a service-oriented approach, whereby agnostic compiler technologies, like Clang, parse the quantum expressions from domain specific languages. Algorithmic interfaces that support this uniform programming model for hybrid execution can encapsulate both common programming concerns and remove the need for programmers to rewrite code from scratch for every compilation target.

Leveraging existing classical HPC programming models should enable the programmability of the distributed macroarchitectures described above while providing flexibility to accommodate the diversity of CPU-QPU interfaces. For distributed systems, a hierarchical programming model across system, node, and device interfaces can leverage conventional approaches in tandem with the quantum programming model described here. For example, in a distributed memory system with multiple nodes each having a dedicated QPU, the well-known MPI+X model may distribute work across nodes via the MPI. Extensions to the runtime infrastructure can map programs onto remote or integrated execution models. Single-circuit and ensemble-circuit applications represented in the QIR can be submitted for execution in a remotely hosted QPU paradigm, while the execution of individual quantum instructions may be designed to support the fast-feedback, in-sequence applications.

OUTLOOK AND CHALLENGES

Our prospective on the synergy between applications, architectures, and infrastructure for HPC and QC has highlighted several near-term opportunities for hardware and software integration. In identifying requirements on communication latency derived from different application contexts, we have articulated some of the technologies needed to advance current QCs toward integration with HPC nodes. We have discussed the different possible macroarchitectures for those hybrid HPC systems and we have detailed the microarchitecture and principal execution units of the quantum-accelerated nodes. The programming models and software interfaces were outlined to manage the diverse and growing capabilities of QCs while also leveraging existing HPC workflows.

A notable conclusion from our analysis is that the existing requirements for integration of QCs with HPC systems are well within reach of current technologies. While many technical challenges remain to complete such integration, the overall feasibility of the proposal has proven sound. Among these challenges are the need to engineer cryogenic controls for low-latency communication for in-sequence processing, the need to decompose device controls into low- and high-latency decisions, the need to standard interfaces for communication and controls, and the necessity to provide service interfaces that promote common methods for programming and resource management.

Alongside these system-level concerns are the ongoing needs to improve device characteristics, including register size, gate fidelities, coherence times, and others, as well as the overall scaling of these resources to limit that support quantum computational advantage. The selection of target applications and application data will prove powerful in identifying the system resources needed to realize such goals and provide even better expectations for near-term demonstration of quantum computing.

Finally, we note that our perspective has intentionally excluded the concern of fault-tolerant operation. Although in-sequence processing applications and QEC specifically are necessary precursors to such capabilities, we have neglected the concurrent management and control of both logical and physical resources during computation that will be required by these more advanced systems. We anticipate that the future regime of fault-tolerant quantum computing will raise many additional challenges beyond those considered here.

The rapidly developing field of quantum computing harbors many exciting advances for computation and application science. By leveraging modern HPC systems, these promises are more likely to be realized sooner and with greater effort. Now is an exciting time for quantum computing and computing more generally.

ACKNOWLEDGMENTS

This article has been authored by UT-Battelle, LLC, under Contract DE-AC05-00OR22725 with the U.S. Department of Energy (DOE). The U.S. government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article, or allow others to do so, for U.S. government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doepublic- access-plan).

This work was supported in part by the Department of Energy, Office of Science, Early Career Research Program. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

REFERENCES

1. J. S. Vetter *et al.*, “Productive computational science in the era of extreme heterogeneity: Report for doe ASCR workshop on extreme heterogeneity tech,” Rep. U.S. DOE Office Sci., 2018. [Online]. Available: URL https://www.osti.gov/biblio/1473756
2. T. S. Humble, H. Thapliyal, E. Muñoz-Coreas, F. A. Mohiyaddin, and R. S. Bennink, “Quantum computing circuits and devices,” *IEEE Des. Test*, vol. 36, pp. 69–94, 2019.
3. F. Arute *et al.*, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, pp. 505–510, 2019.
4. H. S. Zhong *et al.*, “Quantum computational advantage using photons,” *Science*, vol. 370, pp. 1460–1463, 2020.
5. Y. Wu *et al.*, “Strong quantum computational advantage using a superconducting quantum processor,” 2021, *arXiv:2106.14734*.
6. A. Aspuru-Guzik *et al.*, “ASCR Workshop on Quantum Computing for Science,” 2015. [Online]. Available: URL https://www.osti.gov/biblio/1194404
7. K. A. Britt and T. S. Humble, “Quantum accelerators for high-performance computing systems,” *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, pp. 1–13, 2017.
8. A. J. McCaskey, E. F. Dumitrescu, D. Liakh, M. Chen, W. C. Feng, and T. S. Humble, “A language and hardware independent approach to quantum—classical computing,” *SoftwareX*, vol. 7, pp. 245–254, 2018.
9. J. R. Cruise, N. I. Gillespie, and B. Reid, “Practical quantum computing: The value of local computation,” 2020, *arXiv:2009.08513*.
10. A. McCaskey, E. Dumitrescu, D. Liakh, and T. Humble, “Hybrid programming for near-term quantum computing systems 2018,” in *Proc. IEEE Int. Conf. Rebooting Comput.*, 2018, pp. 1–12.

11. A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, and T. S. Humble, "XACC: A system-level software infrastructure for heterogeneous quantum—classical computing," *Quantum Sci. Technol.*, vol. 5, 2020, Art. no. 024002
12. A. McCaskey, T. Nguyen, A. Santana, T. Kharazi, D. Claudino, and H. Finkel, "Extending C++ for Heterogeneous Quantum-Classical Computing," *ACM Trans. Quantum Computing*, vol. 2, 2021, Art. no. 6.

TRAVIS S. HUMBLE is the Deputy Director of the Department of Energy's Quantum Science Center, a Distinguished Scientist, and the Director of the Quantum Computing Institute, at Oak Ridge National Laboratory, Oak Ridge, TN, USA. He is leading the development of new quantum technologies and infrastructure to impact the DOE mission of scientific discovery through quantum computing. He is the Editor-in-Chief for *ACM Transactions on Quantum Computing*, an Associate Editor for *Quantum Information Processing*, and Co-Chair of the IEEE Quantum Initiative. He holds a joint faculty appointment with the University of Tennessee Bredesen Center for Interdisciplinary Research and Graduate Education working with students on energy-efficient computing solutions. Contact him at humblets@ornl.gov.

ALEXANDER MCCASKEY is a Research Scientist with the Beyond Moore Computing Group within the Computer Science and Mathematics Division, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA. He leads the Systems and Software research theme within the ORNL Quantum Computing Institute. His research is focused on programming models, compilers, and languages for heterogeneous quantum-classical computing. He is the Lead for a number of open-source quantum software projects, including the XACC system-level quantum framework and the qcor quantum-classical C++ compiler platform. McCaskey received a B.Sc. degree in physics and mathematics from the University of Tennessee in 2010, and an M.Sc. degree in physics from the Virginia Polytechnic and State University in 2014. Contact him at mccaskeyaj@ornl.gov.

DMITRY I. LYAKH is a Research Scientist with the Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA, with interests in hybrid high-performance quantum/classical computing. He is focused on enabling high-accuracy, large-scale quantum many-body simulations on the leadership heterogeneous HPC systems for applications in quantum chemistry, condensed matter physics, and quantum computing. In the context of quantum computing, he is a coauthor of the scalable tensor network based quantum circuit simulator, called TN-QVM, which can efficiently simulate quantum circuits on GPU-accelerated HPC platforms. He is also the primary developer of the scalable tensor network processing library, called ExaTN, which serves as a computational backend in the TN-QVM code. In general, his long-term interests are in maximally leveraging high-performance computing for quantum device modeling as well as enabling new quantum-accelerated scientific computing workflows. Contact him at liakhdi@ornl.gov.

MEENAMBIKA GOWRISHANKAR is a Graduate Research Assistant with the University of Tennessee Bredesen Center and is part of the Quantum Science Center, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA. She is working on the application and analysis of variational quantum algorithms with interests in quantum error correction for near-term quantum devices. Contact her at gowrishankam@ornl.gov.

ALBERT FRISCH is a Senior Research Engineer at Alpine Quantum Technologies (AQT), Innsbruck, Austria, with main focus on architecture and software development for ion-trapped quantum computers across the full stack, from cloud services to hardware control. His interests are in orchestrating automated calibration tasks, system health checks, and parameter monitoring. Frisch studied quantum optics and atomic physics and received a Ph.D. degree from the University of Innsbruck in ultracold quantum gases of erbium in 2014. Since 2017, he has been engaged in the hardware and software development for quantum computing at IBM working on room-temperature control electronics and the software development kit Qiskit. Contact him at albert.frisch@aqt.eu.

THOMAS MONZ is the Co-Founder and CEO at Alpine Quantum Technologies (AQT), Innsbruck, Austria, a quantum computer company focusing on trapped ions, and a Senior Scientist at the University of Innsbruck, Innsbruck, Austria. At the University of Innsbruck, he focused on the engineering aspects and scalability of ion-trap-based quantum computing. His team realized the first 19″ rack-mounted ion-trap quantum computer. With this device, he set a new record (as of 2021) for genuine multiparticle entanglement, demonstrating a 24-qubit GHZ state. His current efforts are focusing on 24/7 operation of this device, extending the capabilities to 50 qubits, and integrated ion-trap quantum computers into HPC infrastructure. Monz received a Ph.D. degree in experimental physics with a focus on the realization of quantum algorithms, verification, and validation of quantum computers, and general ion-trap-based quantum technologies. Contact him at thomas.monz@uibk.ac.at.