Observability Hints for Quantum Network Applications

Vaastav Anand, Antoine Kaufmann
Max Planck Institute for Software Systems

ABSTRACT

Quantum internet promises to enable a more powerful class of applications than those possible just with classical communication. Due to the heterogeneity of the quantum network, the continuous interaction between classical and quantum components, as well as the non-deterministic and probabilistic nature of quantum network applications, it is imperative that we develop mechanisms to track the performance of quantum network systems. In this paper, we argue for the inclusion of classical observability techniques as first class citizens in the design of the infrastructure powering the quantum system. We discuss the different challenges associated for enabling observability and discuss potential ways in which we can enable observability for quantum network systems.

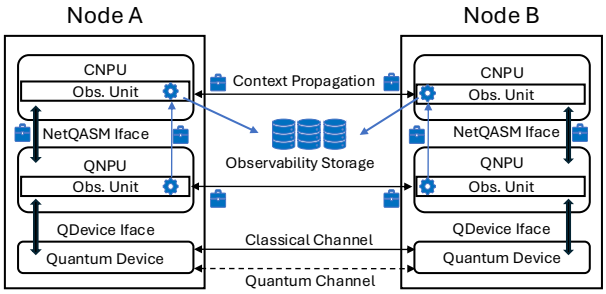

Figure 1: Augmented QNodeOS with observability units

1 MOTIVATION

Quantum Internet and quantum networks are slated to play an important role in the future of cloud and distributed computing by enabling a new class of networked applications that are impossible to be implemented using solely classical communication [13, 18]. Common examples of such applications include Quantum Blind Computation [2, 7], Delegated Quantum Computation [3], Quantum Key Distribution [14], among others. Despite the excitement surrounding these applications, demonstrations of these applications have been limited to ad-hoc software with quantum hardware specialized for executing specific tasks.

While quantum networked applications remain in its nascency, developers deeply care about the performance and efficacy of these applications. This need will only be further exacerbated in a true realization of a quantum internet, where the number of nodes communicating would be far greater than just two. Thus, the key requirement for effectively doing performance analysis is to have visibility into the internals of the system and to effectively track the execution across component boundaries.

Traditionally, observability in classical distributed systems provides insight into the performance and execution of the system that can then be used by developers to understand system behavior, isolate root causes of issues and incidents, and for optimizing the performance of the system. To provide this insight, observability makes use of its 3 core pillars — logs, metrics, and traces. Logs provide detailed information about the execution at the granularity of events deemed interesting by developers. Metrics continuously track important Service Level Objectives (SLOs) such as latency and throughput, allowing developers to set automated alerts and monitors for the key performance indicators of the system. Traces provide a cross-component detailed view of the execution at request-level granularity that span multiple nodes and components.

We propose that to enable performance analysis of quantum networked applications, observability must be treated as a first class citizen in the design and implementations of operating systems enabling quantum networked applications. While we discuss challenges associated with enabling observability in quantum networked systems, we believe these challenges are also applicable to quantum operating systems [9, 10] designed for the public cloud [1, 11, 12, 16]. We present a mock-up of a potential design of a dedicated observability unit in the context of the QNodeOS operating system [6] and discuss future steps associated with incorporating observability into quantum networked applications.

2 CHALLENGES

However, gaining visibility in quantum network applications is challenging for many reasons. First, these applications are inherently heterogeneous. They are composed of mixed classical and quantum tasks that depend on each other in different ways and often require both classical communication as well as quantum communication. Thus, we need to track observability not only across network boundaries but also across processor boundaries. Second, these applications are probabilistic and non-deterministic. The success of the application is often highly dependent on the quality and the fidelity of the entangled states generated. Thus, these applications may require many different executions to gain a meaningful result. As a result, more specialized metrics need to be calculated and tracked on a per-application basis. Third, there are a lot of sources of error that can degrade the performance of the system. These may be decoherence errors that cause the quantum state to decohere or collapse as well as errors in application of quantum gates or errors in performing measurements. Thus, the different sources must be accounted for and tracked during the execution which requires obtaining deep low-level visibility. Fourth, gaining insight into the quantum states is a difficult and resource consuming process. For example, performing quantum state tomography for even a two- qubit state requires performing measurements with different settings to get the full probability density matrix of the state. Thus, we may need to redesign observability techniques to make them more efficiently applicable for quantum networked systems.

3 OBSERVABILITY IN THE QUANTUM AGE

To overcome the challenges, we need to design dedicated observability units that can track the execution across heterogeneous

components and collect relevant metrics associated with the applications. We present our vision of native observability support in quantum networked applications within the context of the design of QNodeOS [6], the first functional general purpose operating system designed for quantum network applications. The observability augmented version of QNodeOS is shown in Figure 1. The original QNodeOS design consists of three parts, (i) a Classical Network Processing Unit (CNPU) that handles the execution of the classical part of the user program, (ii) a Quantum Network Processing Unit (QNPU) that executes the quantum part of the user program and manages the underlying resources provided by the quantum device - namely entanglement and qubits, and (iii) a quantum device that provides the quantum resources. We augment the design by modifying the CNPU and QNPU with dedicated observability units that are responsible for providing the three pillars of observability.

3.1 Metrics

For quantum network systems, there are a variety of possible metrics developers might need to track. There are two different classes of metrics that need to be measured and monitored - (i) classical metrics, and (ii) quantum metrics.

Classical Metrics. Classical metrics represent the class of metrics that can be measured using classical techniques and do not require any quantum measurements. Common examples of these metrics include end-to-end latency, server throughput, resource utilization, and power usage. Diagnosing issues requires collecting classical metrics about the quantum aspects of the system. For latency, the developers might additionally want to measure the latency (or waiting time) for generating 1 entangled pair of qubits. For throughput, possible measures include throughput of the number of qubits generated by a quantum device, throughput for the number of entangled pairs generated per second, throughput of the total number of successful entanglement swapping (or teleportation) attempts. For utilization, a sensible metric might be the number of physical qubits utilized per second or the number of logical qubits utilized per second. Utilization metrics might also include metrics about the link utilization of the classical and quantum communication channels between two neighboring nodes. These metrics would need to be tracked by both the CNPU and QNPU.

Quantum Metrics. Quantum metrics are metrics pertaining to the quantum state and require quantum solutions. For example, the most popular example of a quantum metric is fidelity. In a quantum network application, a developer might want to track different fidelity metrics. Fidelity metrics could include the fidelity of the initial states, fidelity of the entangled states, or fidelity of the final computation result. Other quantum metrics include metrics that measure the quality of the entanglement. For example, entanglement entropy [4] measures the quality of the entanglement of a pure state. Some quantum applications even have pre-requisites on the quality of entanglement before execution can happen. For example, the E91 protocol [8] for entanglement-based quantum key distribution requires that the prepared entangled state be a state such that it violates the CHSH inequality [5]. As not all entangled states violate the inequality, it becomes imperative that the quality of the entanglement must be measured. These metrics would need to be tracked by the QNPU.

A key difference between classical and quantum metrics is that quantum metrics are often second order metrics. This is because, to calculate a single value for these metrics, we require multiple measurements, sometimes in the order of thousands, as is the case when using quantum tomography [17] for calculating the fidelity of the entangled state. Thus, we also want to track metrics about these quantum metrics, such as number of measurements per calculation of the metric and resources spent calculating the metric. Moreover, we might need to use sampling strategies to overcome the high resource requirements of measuring quantum metrics.

3.2 Tracking Execution

Traces. Traditionally, traces track the execution of a request through the different components of the system. In classic settings, tracing requires propagating a request-specific context across the various component boundaries often demarcated by individual deployment units [15]. However, for quantum network systems, we are operating in a heterogeneous setting with two different processor components providing different capabilities. Thus,to obtain complete executions, we need to now propagate the execution context across both the processor boundaries in addition to the network boundaries to get a full picture of the execution of the system.

To correctly provide tracing capabilities, we propose modifying the CNPU and QNPU design in two ways. First, the CNPU of one node must propagate the context to the CNPU of the other node to establish context propagation over the classical communication channel between the two CNPUs. Second, the execution context must be propagated through the CNPU-QNPU channel which can then further be propagated and attached to the communication with the QNPU of a neighboring node. Propagating the context into the QNPU allows attaching information about the local quantum execution to the trace of the application as well as ensuring that all possible entanglement generation requests and communication with neighboring nodes is correctly attributed to the application. This context propagation is important, as it allows us to correctly attribute usage of quantum resources for different applications especially if there are multiple applications executing over the quantum network at the same time.

Logs. Traditionally, logs capture discrete events within the system. For quantum network systems, there are two distinct sources of events that the user might want to capture – (i) events from CNPU, i.e. the execution of the classical part of the program, and (ii) events from QNPU, i.e. the execution of the quantum part of the program. However, not all events in the two processor units might be specific to a given user program or application. Some of them might simply be housekeeping events for the specific processing unit. Thus, both types of events need to be tracked as well as easily differentiated.

4 CALL-TO-ACTION

The lack of visibility into the performance and execution of quantum applications has the potential to completely block the quantum internet revolution. We believe we have only scratched the surface of the power that observability techniques can provide in the design and maintenance of quantum network systems. We argue that the quantum systems community should work together in developing solutions and techniques that can truly unleash the full power of

observability and help in the fulfillment of the immense promise of quantum internet.

REFERENCES

[1] Amazon. Amazon braket. Accessed August 2024 from https://aws.amazon.com/braket/.

[2] Stefanie Barz, Elham Kashefi, Anne Broadbent, Joseph F Fitzsimons, Anton Zeilinger, and Philip Walther. Demonstration of blind quantum computing. *science*, 335(6066):303–308, 2012.

[3] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th annual IEEE symposium on foundations of computer science*, pages 517–526. IEEE, 2009.

[4] Pasquale Calabrese and John Cardy. Entanglement entropy and quantum field theory. *Journal of statistical mechanics: theory and experiment*, 2004(06):P06002, 2004.

[5] John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed experiment to test local hidden-variable theories. *Physical review letters*, 23(15):880, 1969.

[6] Carlo Delle Donne, Mariagrazia Iuliano, Bart van der Vecht, Guilherme Maciel Ferreira, Hana Jirovská, Thom van der Steenhoven, Axel Dahlberg, Matt Skrzypczyk, Dario Fioretto, Markus Teller, et al. Design and demonstration of an operating system for executing applications on quantum network nodes. *arXiv preprint arXiv:2407.18306*, 2024.

[7] P Drmota, DP Nadlinger, D Main, BC Nichol, EM Ainley, Dominik Leichtle, A Mantri, Elham Kashefi, R Srinivas, G Araneda, et al. Verifiable blind quantum computing with trapped ions and single photons. *Physical Review Letters*, 132(15):150604, 2024.

[8] Artur K Ekert. Quantum cryptography based on bell's theorem. *Physical review letters*, 67(6):661, 1991.

[9] Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, and Pramod Bhatotia. Qos: A quantum operating system. *arXiv preprint arXiv:2406.19120*, 2024.

[10] Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, Dmitry Lugovoy, and Pramod Bhatotia. Orchestrating quantum cloud environments with qonductor. *arXiv preprint arXiv:2408.04312*, 2024.

[11] Google. Quantum ai. Accessed August 2024 from https://quantumai.google/.

[12] IBM. Ibm quantum. Accessed August 2024 from https://www.ibm.com/quantum.

[13] H Jeff Kimble. The quantum internet. *Nature*, 453(7198):1023–1030, 2008.

[14] Wen-Zhao Liu, Yu-Zhe Zhang, Yi-Zheng Zhen, Ming-Han Li, Yang Liu, Jingyun Fan, Feihu Xu, Qiang Zhang, and Jian-Wei Pan. Toward a photonic demonstration of device-independent quantum key distribution. *Physical Review Letters*, 129(5):050502, 2022.

[15] Jonathan Mace, Ryan Roelke, and Rodrigo Fonseca. Pivot tracing: dynamic causal monitoring for distributed systems. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, page 378–393, New York, NY, USA, 2015. Association for Computing Machinery.

[16] Microsoft. Azure quantum cloud service. Accessed August 2024 from https://azure.microsoft.com/en-us/products/quantum.

[17] Matteo Paris and Jaroslav Rehacek. *Quantum state estimation*, volume 649. Springer Science & Business Media, 2004.

[18] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):eaam9288, 2018.