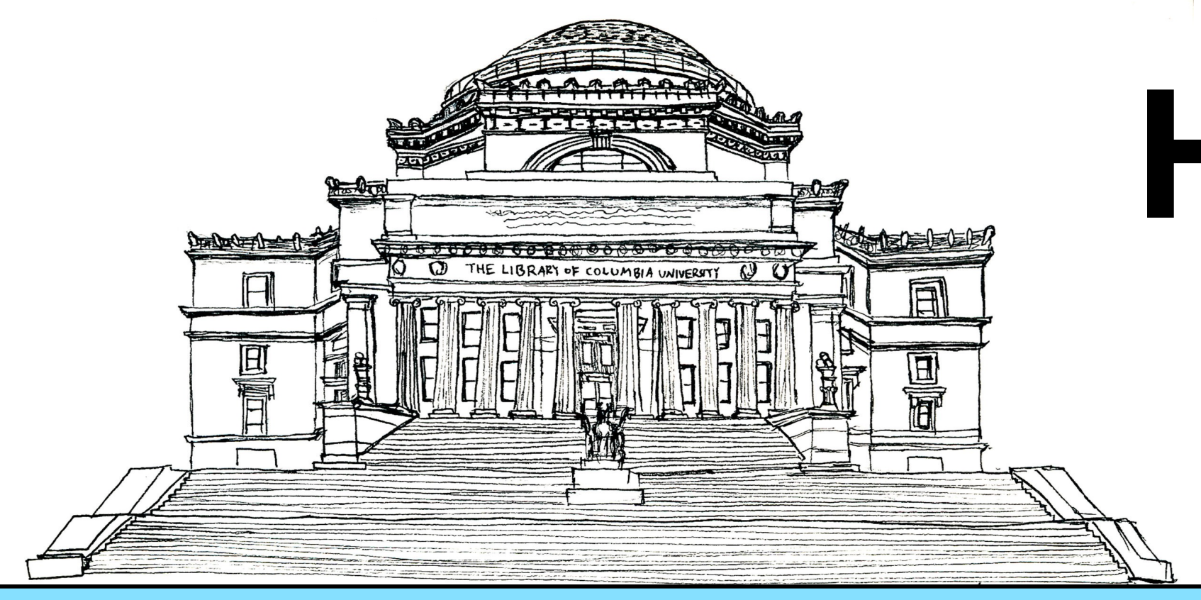


HyperQ - Virtual Machines for Quantum Computers

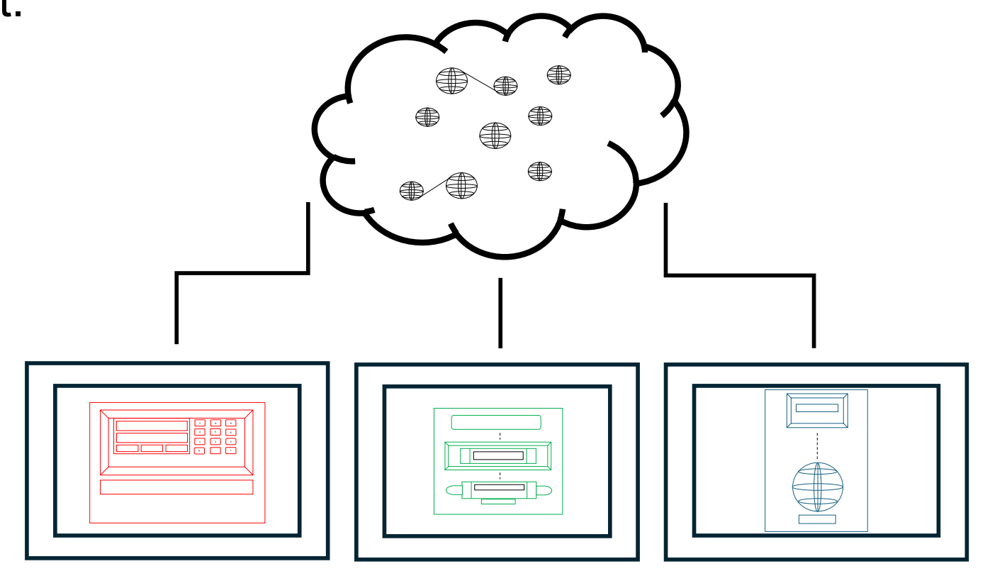
***Aaron Bordeaux*[1]**, *Jason Nieh*[2], *Runzhou Tao*[2], *Hongzheng Zhu*[2]

[1]*Princeton University*, [2]*Columbia University*

Background + Motivation

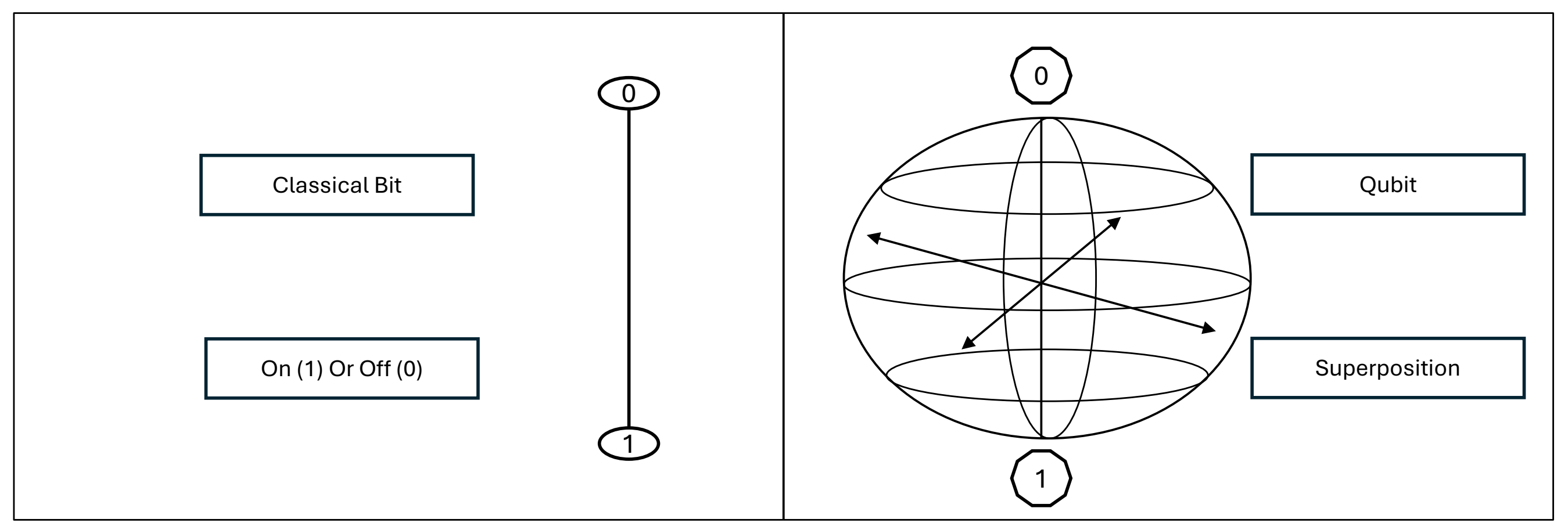
Project Description

- Quantum cloud computing provides on-demand access to quantum computers, utilizing Noisy Intermediate-Scale Quantum (NISQ) devices. Despite the high demand and limited availability of quantum computers, the current software infrastructure is primitive, leading to inefficiencies and long wait times. Quantum programs are processed one at a time, with most programs utilizing only a fraction of available qubit.



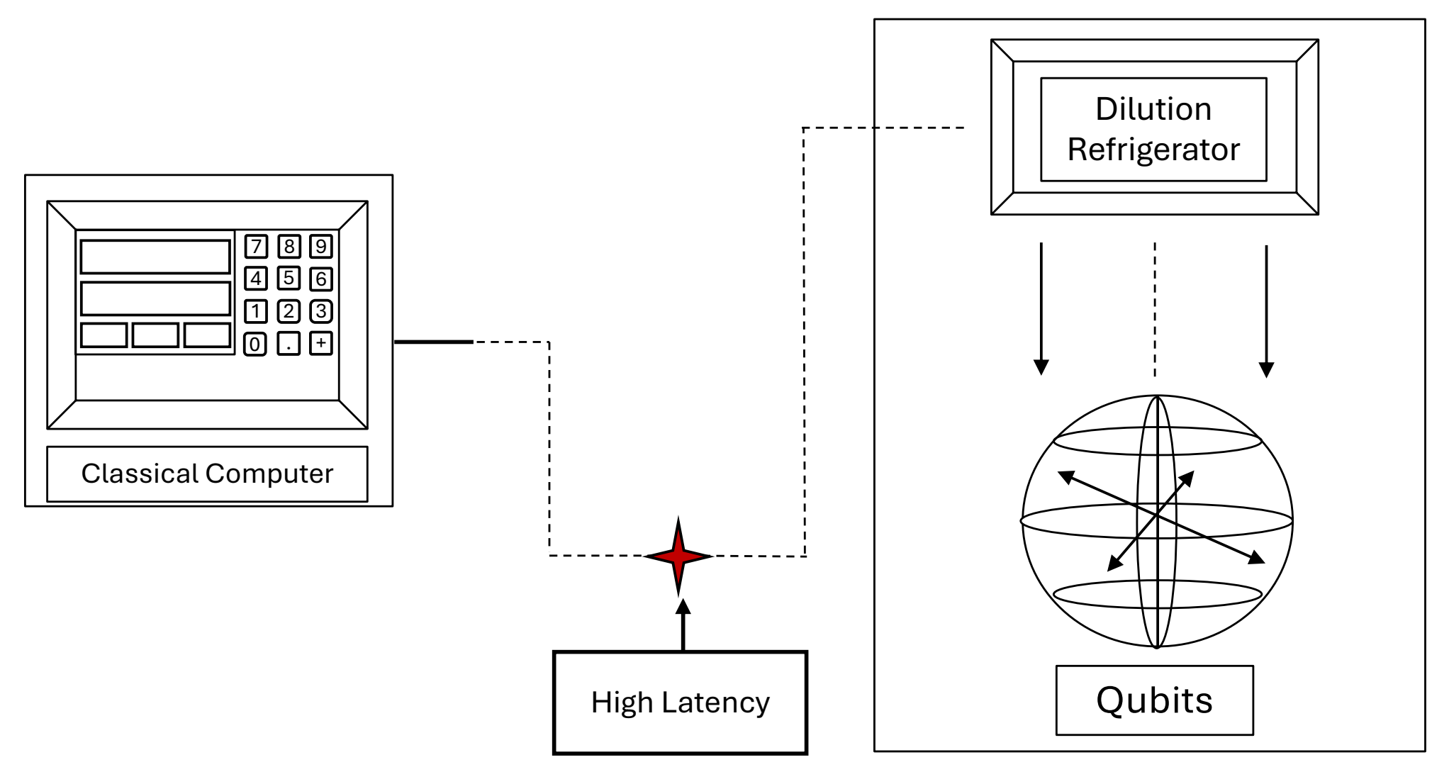
Qubits

Qubits, or quantum bits, are the fundamental units of information in quantum computing, capable of representing and processing data as both 0 and 1 simultaneously due to the principles of superposition and entanglement.



Problems

- **Inefficient Resource Utilization:** Quantum programs are run sequentially on quantum computers, causing underutilization since most programs use only a small fraction of the available qubits.
- **High Latency:** Due to high demand and the limited availability of quantum computers, users often experience long wait times for their quantum programs to be processed and results returned.
- **Primitive Software Infrastructure:** Unlike classical cloud computing, which leverages virtual machines (VMs) for efficient multi-tenant resource utilization, quantum cloud computing lacks such advanced virtualization infrastructure. The current system processes quantum programs one by one without the capability to run multiple programs simultaneously, leading to inefficiencies.



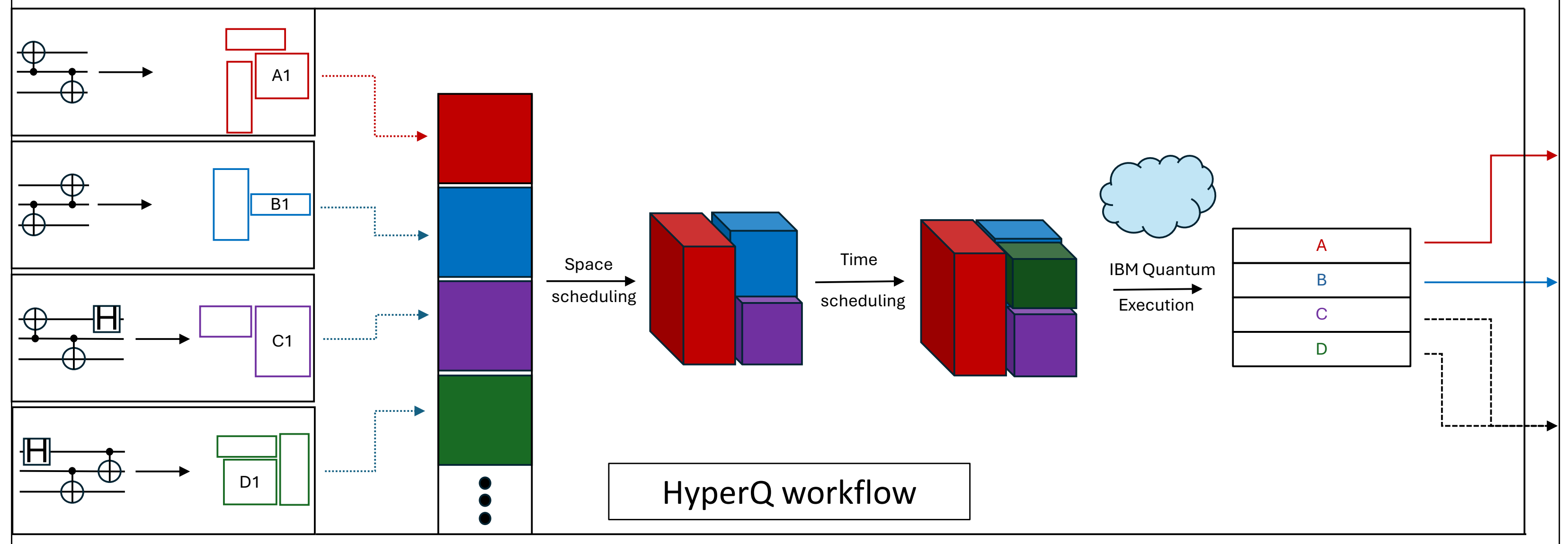
Motivations

- **Improve Resource Utilization:** By introducing virtualization, multiple quantum programs can run simultaneously on the same quantum computer, utilizing more of the available qubits and resources efficiently.
- **Reduce Latency:** Allowing multiple quantum programs to run concurrently can significantly decrease the wait times for users to get their results.
- **Enhance Software Infrastructure:** Virtualization can provide isolation between programs, ensuring that simultaneous execution does not lead to interference or reduced fidelity in results.

Design

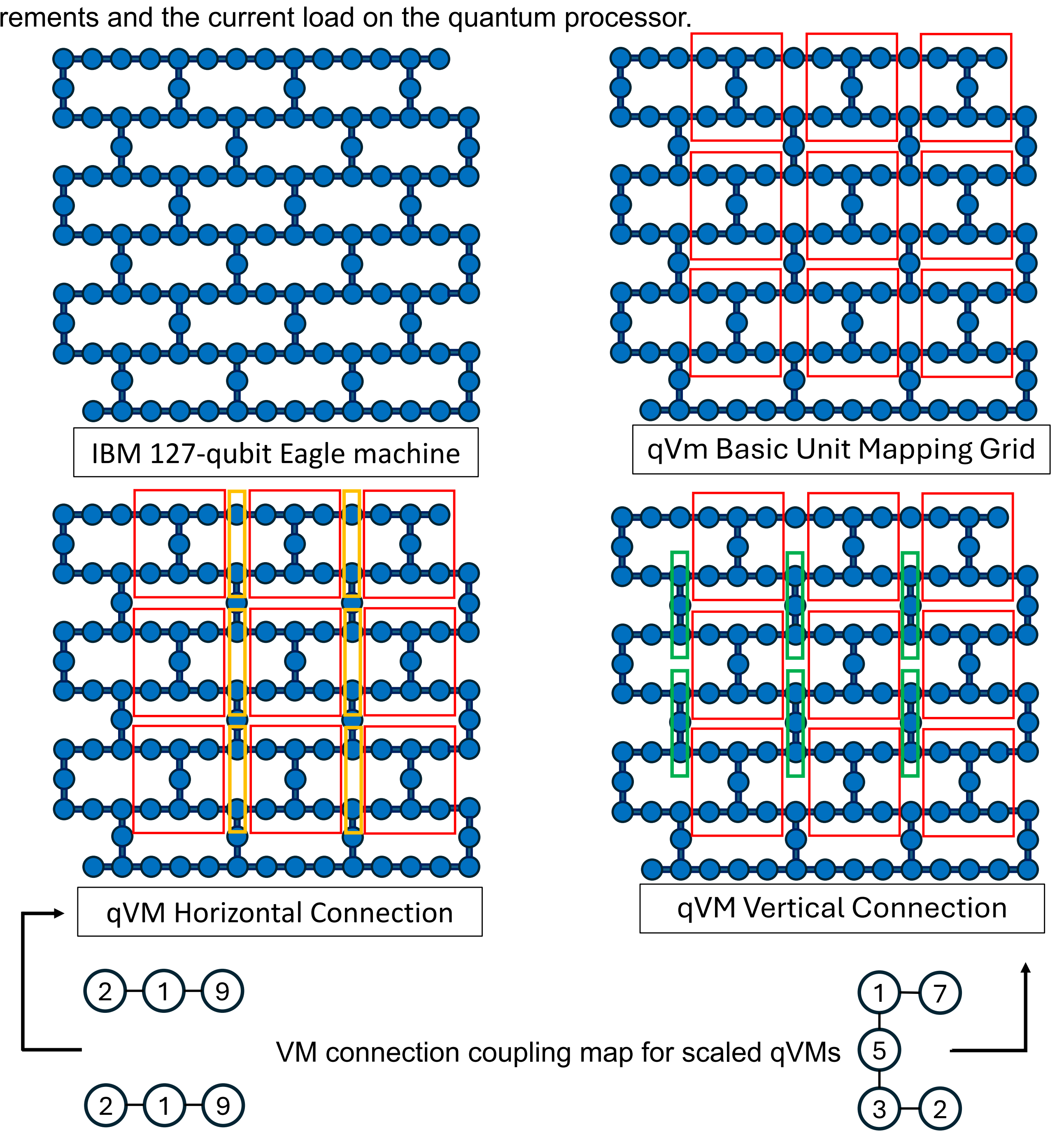
HyperQ Design

qVM (Quantum Virtual Machine): is a virtualized abstraction of quantum hardware that allows multiple quantum programs to run concurrently on a single physical quantum processor. It mimics the architecture and gate sets of the underlying quantum hardware.



Architecture

- **Qubit Partitioning**: Divide the physical qubits of a quantum processor into smaller subsets, each allocated to a different qVM. This allows multiple qVMs to run simultaneously on different qubit subsets.
- **Dynamic Allocation**: Adjust the number of qubits allocated to each qVM based on the program's requirements and the current load on the quantum processor.


IBM 127-qubit Eagle machine
qVm Basic Unit Mapping Grid
qVM Horizontal Connection
qVM Vertical Connection
2
1
9
1
7
5
3
2
2
1
9
VM connection coupling map for scaled qVMs

Scheduling Algorithms

- **Space Scheduling:** Allocate qVMs to non-overlapping sets of qubits, enabling parallel execution of multiple qVMs.
- **Time Scheduling:** Schedule multiple qVMs to run sequentially on the same qubits, optimizing the use of idle qubits and reducing latency.

Evaluation + Results + Conclusion

Evaluation of qVM Virtual Machines

To evaluate the effectiveness of HyperQ, various performance metrics were assessed from both the cloud provider's and the user's perspectives. The evaluation focused on improvements in throughput, utilization, and latency when using HyperQ compared to the baseline IBM Quantum Platform.

Throughput

Throughput was quantified by obtaining the actual runtime for each job on the quantum hardware, as reported by the IBM platform. The total runtime for each configuration was summed, and throughput was calculated by dividing the number of programs executed by the total runtime.

	small-only	small&med
IBM Quantum	362 s	614 s
HyperQ space only	58.8 s	236 s
improvement factor	6.16x	2.60x
HyperQ	58.8 s	197 s
improvement factor	6.16x	3.12x

Table 3. Throughput for HyperQ versus IBM Quantum

Throughput Results:

HyperQ significantly outperformed the baseline IBM Quantum Platform, showing an improvement factor of up to six times for the small-only benchmark and more than three times for the small & med benchmark

Utilization

Utilization was defined as active qubit second / total qubit second. An approximation was made by assuming that all qubits used by a program are active for the entire duration of the program's runtime on the quantum hardware.

	small-only	small&med
IBM Quantum	3.26%	7.81%
HyperQ space only	21.7%	28.8%
improvement factor	6.66x	3.69x
HyperQ	21.7%	37.2%
improvement factor	6.66x	4.76x

Table 4. Utilization for HyperQ versus IBM Quantum

Utilization Results:

HyperQ achieved nearly seven times better utilization for the small-only benchmark and five times for the small & med benchmark, demonstrating its ability to pack more qVMs together efficiently.

Latency

Latency included the time taken for program compilation, submission to the cloud service, queue waiting time, and execution time. With HyperQ, additional scheduling and aggregation steps were considered. A conservative measure of queue waiting time assumed an initially empty queue.

	small-only					small&med				
	Compile	Schedule	Queue	Run	Total	Compile	Schedule	Queue	Run	Total
IBM Quantum	0.040	N/A	180	3.13	183	0.223	N/A	295	3.13	298
HyperQ space only	0.041	0.118	28.2	3.26	31.6	0.160	0.367	112	6.37	119
HyperQ	0.041	0.213	28.9	3.26	31.7	0.160	0.831	95.7	9.36	106

Table 5. Average latency (seconds)

Latency Results:

HyperQ reduced average latency by up to six times by decreasing queue waiting times and optimizing the scheduling process. This significant reduction was primarily due to faster consumption of program workloads by running them simultaneously.

Conclusion

HyperQ is a pioneering system that introduces Quantum Virtual Machines to enhance resource utilization and reduce latency in quantum cloud computing. By leveraging the repeating qubit regions of quantum machines, HyperQ defines qVMs using an architecture-specific quantum instruction set and virtual qubit topology. This allows quantum programs to be compiled for specific-sized virtual machines and then binpacked together for simultaneous execution on real hardware, without runtime virtualization overhead.

PRINCETON UNIVERSITY

COLUMBIA UNIVERSITY

COLUMBIA ENGINEERING
The Fu Foundation School of Engineering and Applied Science

CS@CU

amazon