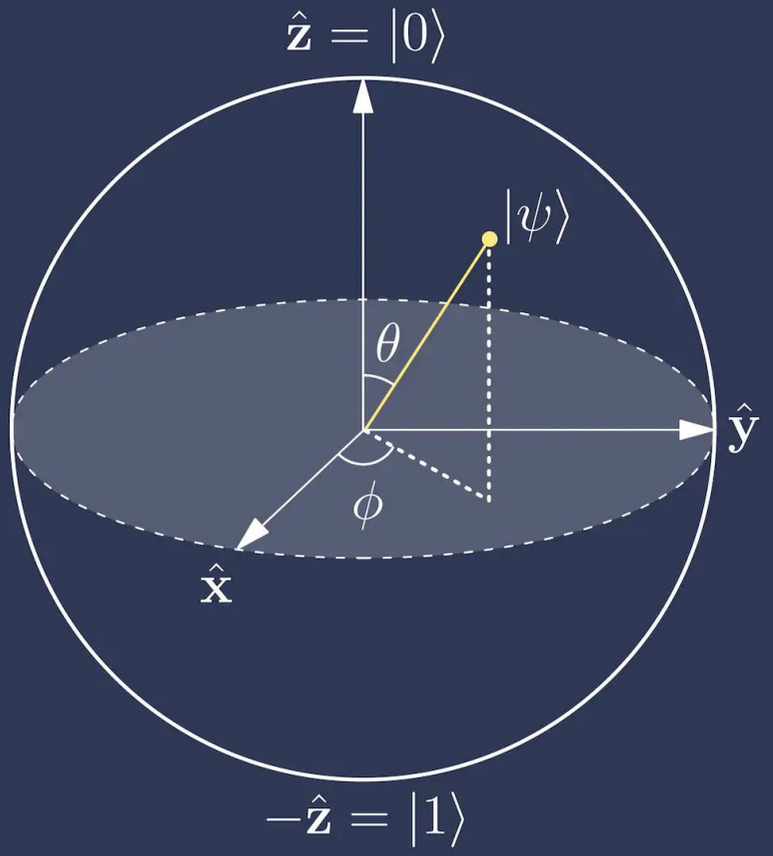
Quantum Virtual Machines

USENIX OSDI: Runzhou Tao and Hongzheng Zhu, University of Maryland, College Park;Jason Nieh, Columbia University; Jianan Yao, University of Toronto; Ronghui Gu, Columbia University

Paper Review - Reliable System Class

Background: "Qubit"

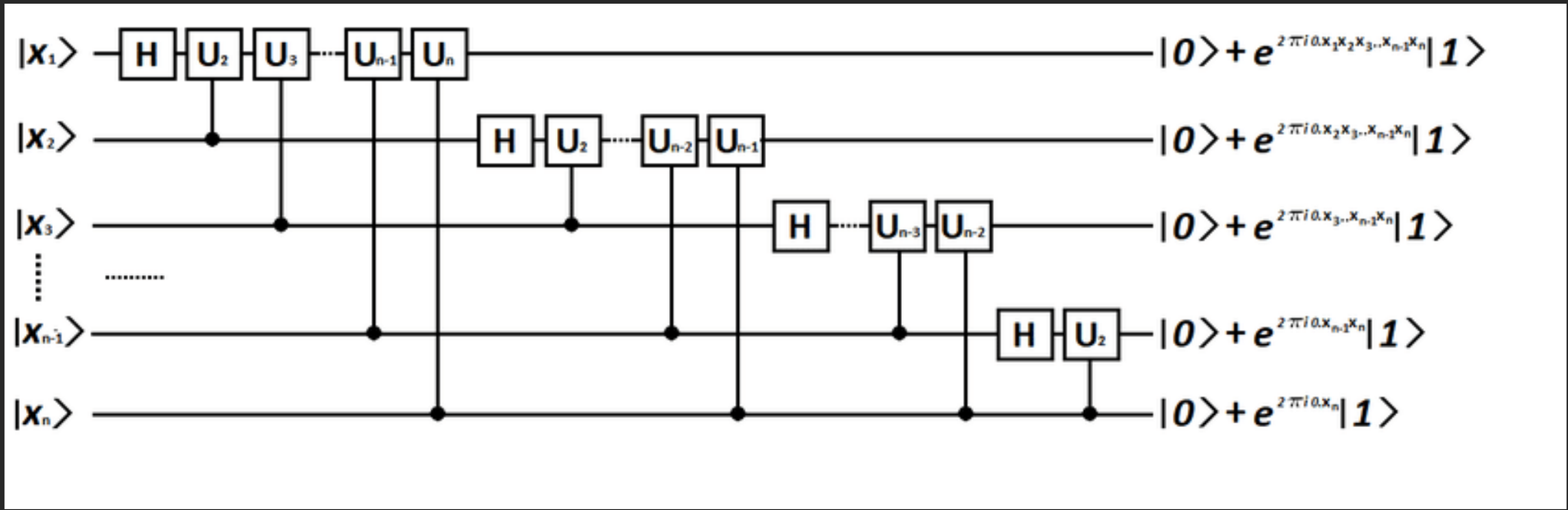


Qubit

/'kju:bɪt/

Basic unit of
quantum information

Background: "Quantum Circuits"



Problem

- Very few quantum processing units
- High demand (Paper: 3T, Now: 3.9T programs run)
- High latency: Job queue sometimes is **days** long
- Most programs/circuits use a fraction of the available qubits
- There is no QRAM

Proposed solution

- Target IBM quantum processors
- Provide "virtualization"
- Allow dynamic multiprogramming
- Improve utilization
- Lower latency
- Preserve **fidelity**
- Minimal Overhead

Quantum compiling

1. Translate circuit to use the available topology of the qubit connections.
2. *Approximate* gates that are not available.

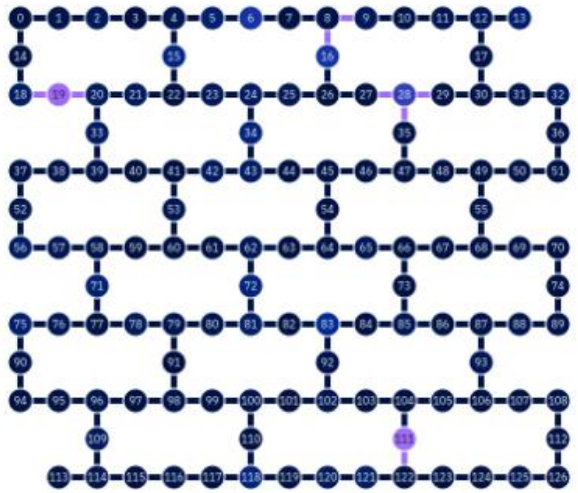


Figure 2: IBM 127-qubit Eagle machine [50].

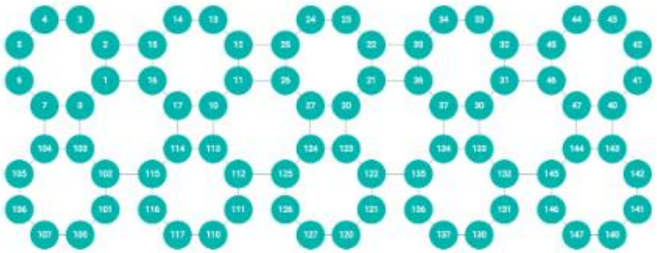
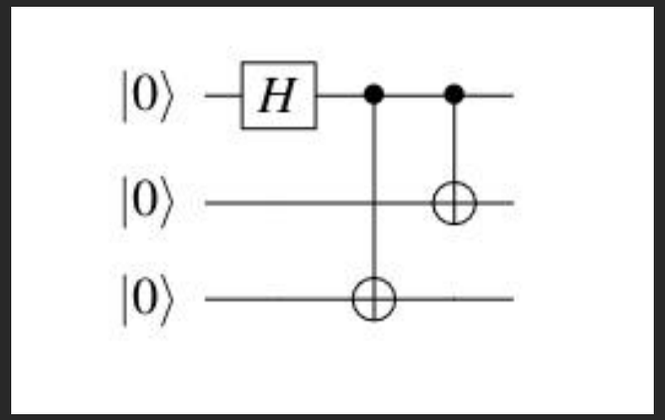
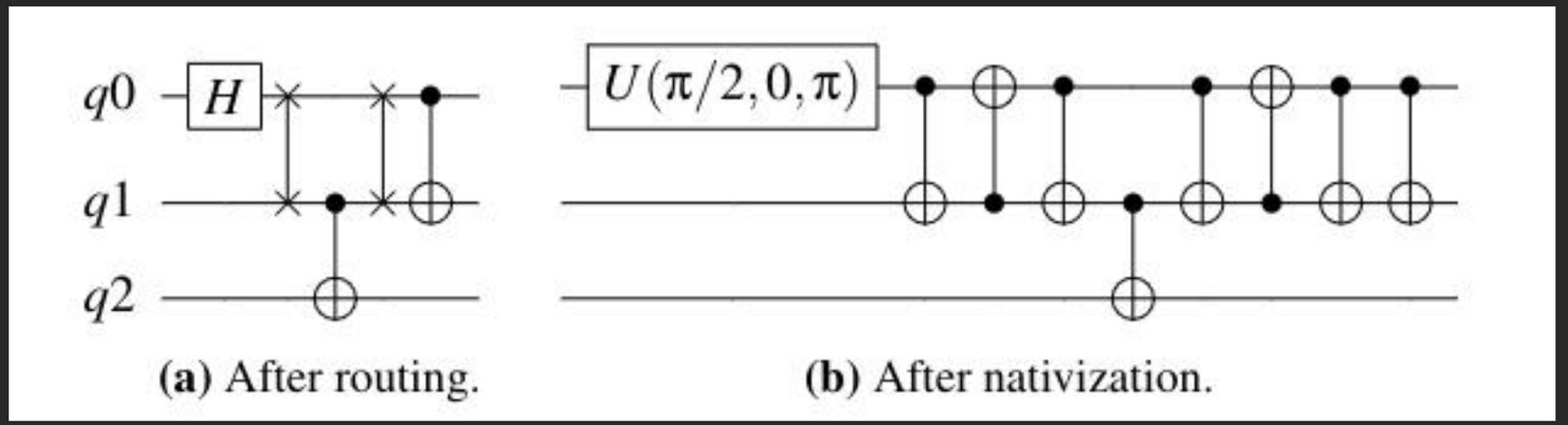


Figure 3: Rigetti 80-qubit Aspen M-3 machine [1].

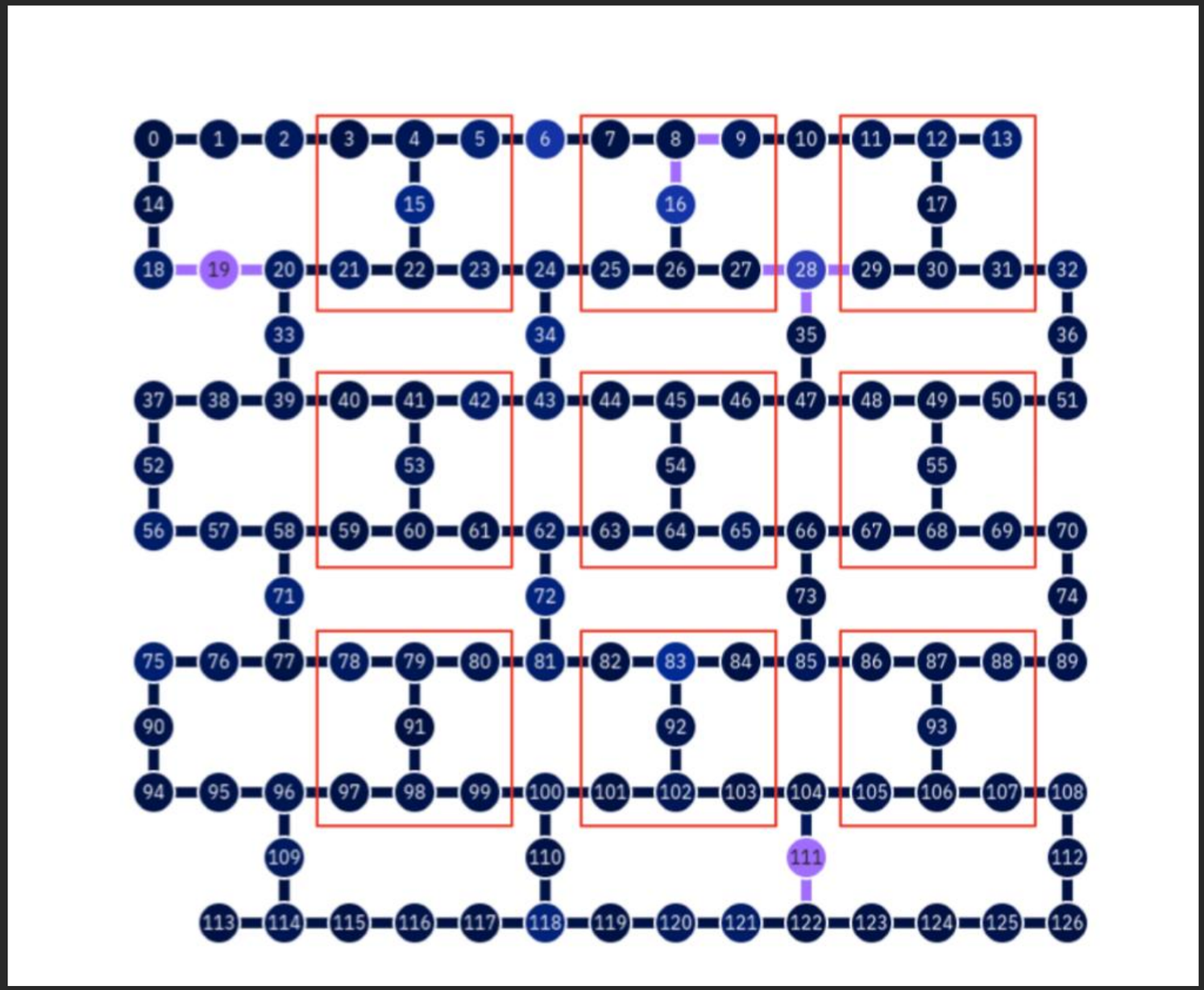
Quantum compiling: Examples





(a) After routing. **(b)** After nativization.

qVM – new target backend



Scheduling

1. Space aware scheduling
2. Time scheduling
3. Noise-aware scheduling

Evaluation: Benchmark

- QASMBench small and medium category
- Two benchmark sets from QASMBench: small-only and small&med
- small-only benchmark: 29 program workloads and each is repeated 5 times (145 total)
- small&med benchmark: 49 program workloads and each is repeated 4 time (196 total)

Evaluation: Configs

- IBM Quantum
- HyperQ
- HyperQ space+time
- HyperQ noise aware

Throughput

job arrival	configuration	small-only	small&med
all-at-once	IBM Quantum	456 s	683 s
	HyperQ	54 s	178 s
	improvement factor	8.4x	3.8x
	HyperQ space+time	47 s	139 s
	improvement factor	9.7x	4.9x
	HyperQ noise aware	64 s	176s
	improvement factor	7.1x	3.9x
poisson	IBM Quantum	456 s	683 s
	HyperQ	143 s	230 s
	improvement factor	3.2x	3.0x
	HyperQ space+time	143 s	203 s
	improvement factor	3.2x	3.4x
	HyperQ noise aware	152 s	223 s
	improvement factor	3.0x	3.1x

Table 3: Throughput for HyperQ versus IBM Quantum.

Utilization

job arrival	configuration	small-only	small&med
all-at-once	IBM Quantum	3.3%	7.8%
	HyperQ	28%	35%
	improvement factor	8.6x	4.4x
	HyperQ space+time	35%	46%
	improvement factor	11x	5.8x
	HyperQ noise aware	23%	35%
	improvement factor	7.2x	4.4x
poisson	IBM Quantum	3.3%	7.8%
	HyperQ	10%	26%
	improvement factor	3.2x	3.3x
	HyperQ space+time	10%	28%
	improvement factor	3.2x	3.6x
	HyperQ noise aware	9.7%	26%
	improvement factor	3.0x	3.4x

Table 4: Utilization for HyperQ versus IBM Quantum.

Latency

job arrival	configuration	small-only						small&med					
		compile	schedule	queue	run	total	improve	compile	schedule	queue	run	total	improve
all-at-once	IBM Quantum	0.04	N/A	226	3.2	229	N/A	0.22	N/A	338	3.5	342	N/A
	HyperQ	0.04	0.14	26	3.6	30	7.6x	0.16	0.40	96	5.1	101	3.4x
	HyperQ space+time	0.04	0.26	22	5.2	28	8.2x	0.16	0.70	76	6.6	83	4.1x
	HyperQ noise aware	0.04	0.14	33	3.6	37	6.2x	0.16	0.49	96	5.0	101	3.4x
poisson	IBM Quantum	0.04	N/A	156	3.2	159	N/A	0.22	N/A	244	3.5	248	N/A
	HyperQ	0.04	0.04	1.2	2.5	3.7	43x	0.16	0.29	7.0	5.0	12	20x
	HyperQ space+time	0.04	0.04	1.2	2.5	3.7	43x	0.16	0.34	4.4	5.0	9.7	26x
	HyperQ noise aware	0.04	0.04	1.4	2.6	4.0	40x	0.16	0.34	9.6	4.7	15	17x

Table 5: Average latency for HyperQ versus IBM Quantum (seconds).

Fidelity

configuration	all-at-once avg L1	poisson avg L1
IBM Quantum	0.55	0.55
HyperQ	0.55	0.57
HyperQ space+time	0.64	0.57
HyperQ noise aware	0.54	0.50

Table 6: Average fidelity for HyperQ versus IBM Quantum.

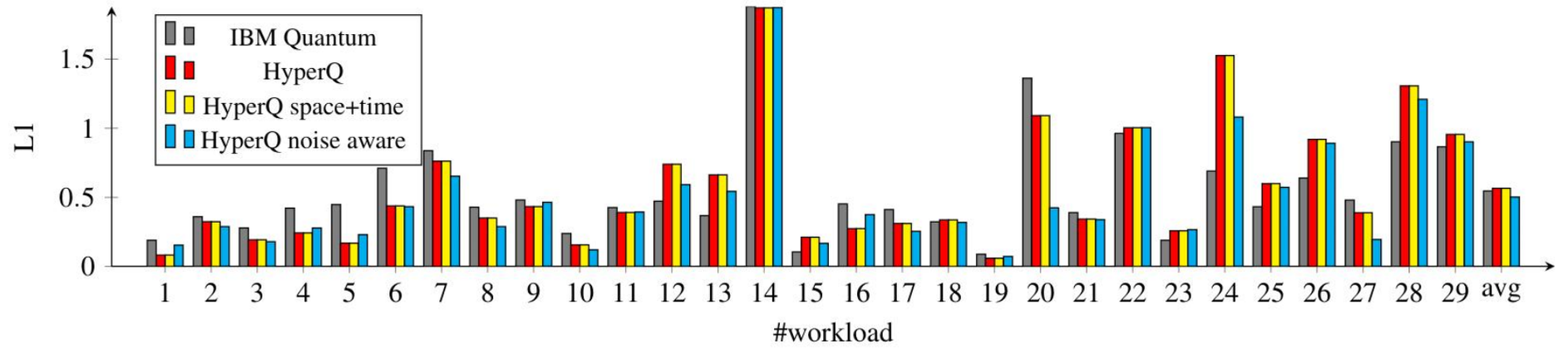


Figure 10: Fidelity for HyperQ versus IBM Quantum per workload with Poisson job arrival.

Conclussion / Advantages over Related Work

- Achieved fidelity, high utilizations and low latency.
- Managed to improve fidelity.
- No need for source code or recompilation.
- Dynamic scheduling.
- Can use the qiskit backend optimizations, unlike custom compilers.