Quantum in the Cloud: Characterization and Management of Resources

Gokul Subramanian Ravi[1],

Kaitlin N. Smith[1], Pranav Gokhale[2], Prakash Murali[3] and Frederic Chong[1,2]

1: University of Chicago **2:** Super.tech **3:** Princeton University

CCF-1730082/1730449, NSF Phy-1818914, DE-SC0020289, DE-SC0020331, NSF OMA-2016136, Q-NEXT DOE NQI Center, CIFellows (NSF 2030859) and IBM/CQE.

Talk Outline

Background and Characterization of Resources

Building a Resource Manager

Other Work: Variational Quantum Algorithms

Summary: Characterizing the Quantum Cloud

- Background: Machines are diverse and user demands are growing constantly: limited knowledge on best machine for any usecase + very long queuing times.

- Goal: Efficient management of cloud resources is critical → Understanding of job / machine characteristics is critical

- Study: Two-year analysis of EPiQC jobs executed on IBM machines academic: 20 machines, 6k jobs, 600k circuits, 10bil machine executions.

- Insights / Recommendations: Verification, Compilation, Machine Diversity, Resource Management, Queuing, Execution.

Gokul Ravi, Kaitlin N. Smith, Pranav Gokhale, Frederic T. Chong. Quantum Computing in the Cloud: Analyzing job and machine characteristics. IISWC 2021. Nov 2021.

Background

Quantum computing

Execution on the the quantum cloud

Quantum Computing

- Quantum information's ability to leverage superposition, interference, and entanglement gives significant advantages in cryptography, chemistry, optimization, and machine learning.

- Today's Noisy Intermediate-Scale Quantum (NISQ) devices have nearly 100 qubits and suffer multiple forms of error.

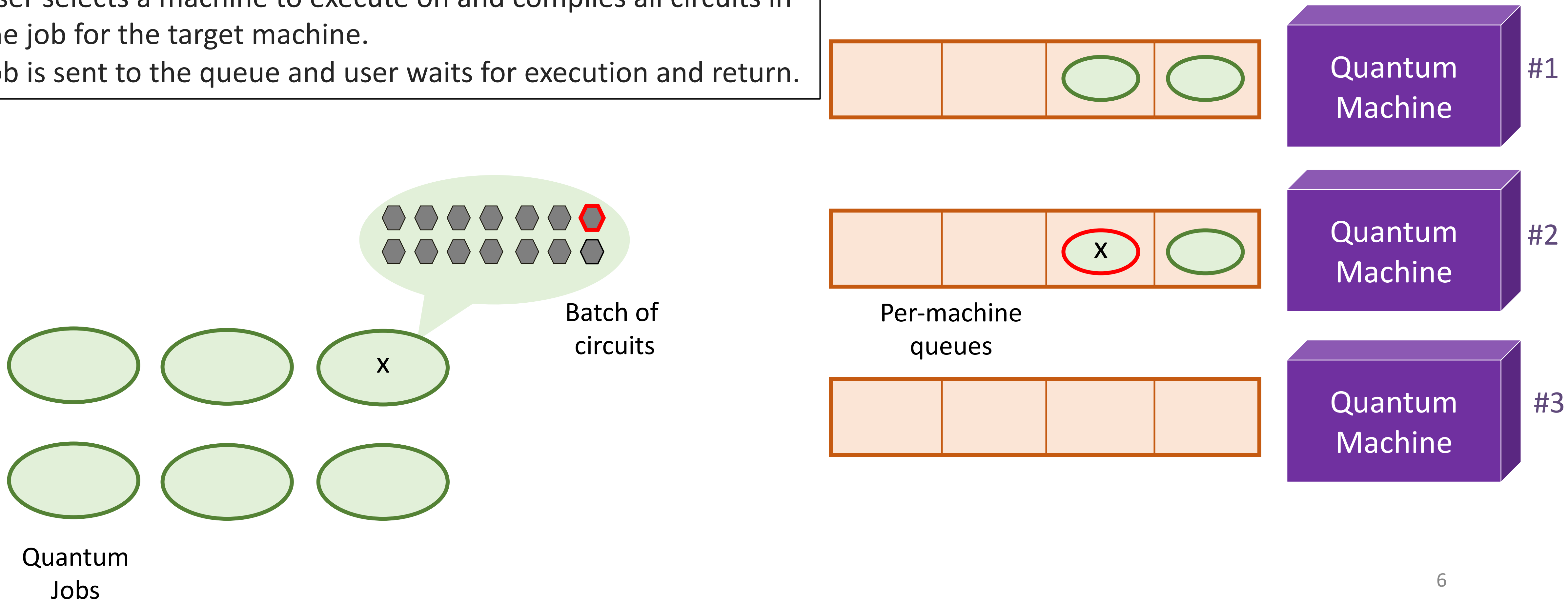
- Error rates are on the decrease (but significant) and devices with as many as 1000 qubits are on the horizon. The future of quantum computing is promising and demand is constantly growing.

- Quantum computing is still at a nascent stage and quantum computers are a rare and expensive resource and thus are primarily accessed world-wide via the cloud.

- Similar to classical HPC, efficient management of cloud resources is critical. Unlike classical HPC:
  - Quantum machines are significantly impacted by machine fidelity constraints,
  - Quantum circuits are currently low complexity, meaning that their execution/fidelity trends are "predictable".

Submitting jobs to the quantum cloud

- Access to the quantum cloud (modeled on IBM).
- Users create a batch of circuits to execute as a job – all circuits in a batch are executed back to back.
- User selects a machine to execute on and compiles all circuits in the job for the target machine.
- Job is sent to the queue and user waits for execution and return.



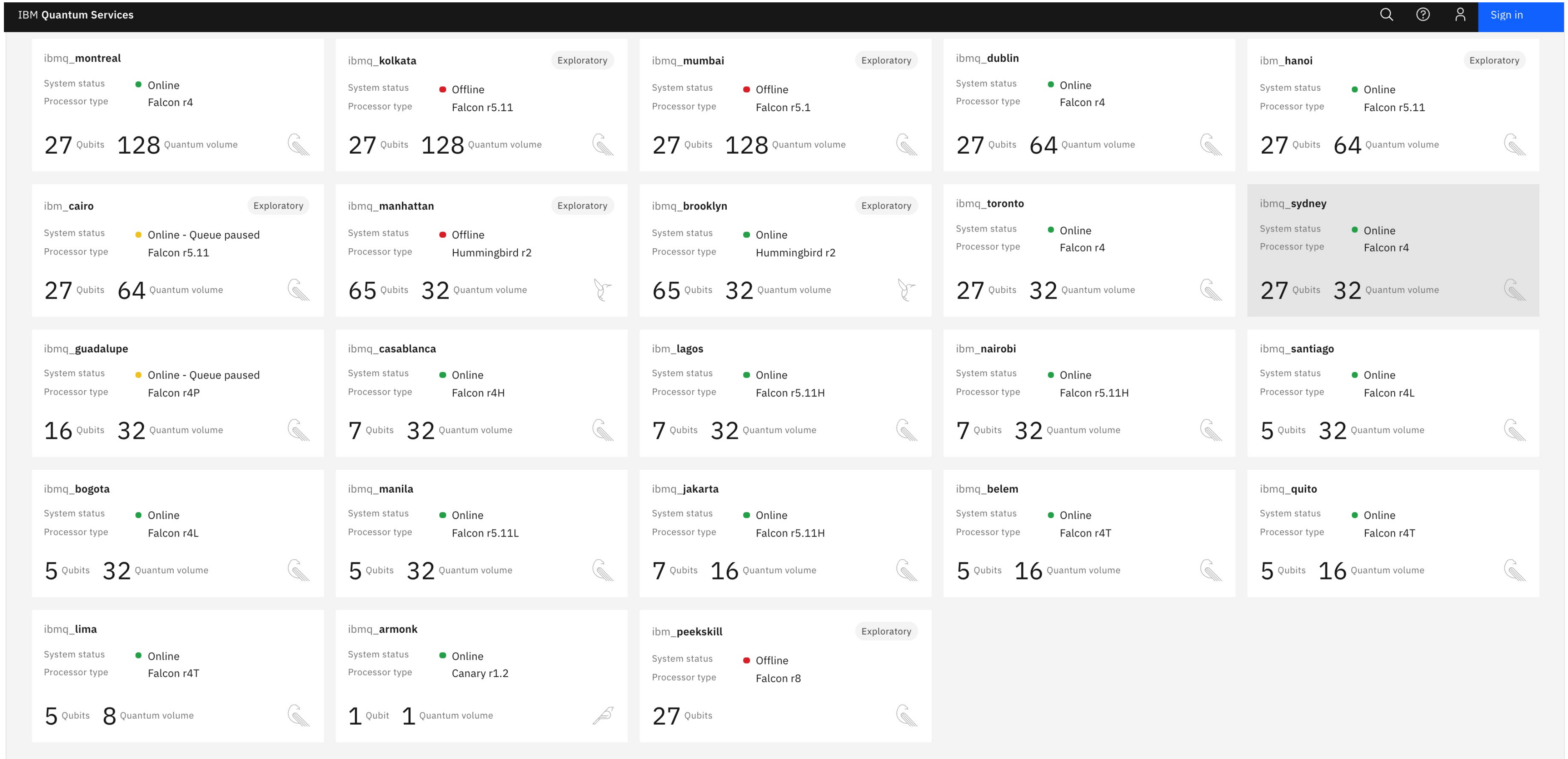
Understanding **quantum machines** and **jobs** are critical to efficiently manage the cloud.

Machines

Size / Topology / Error

IBM Quantum Cloud: Qubit (Size) diversity

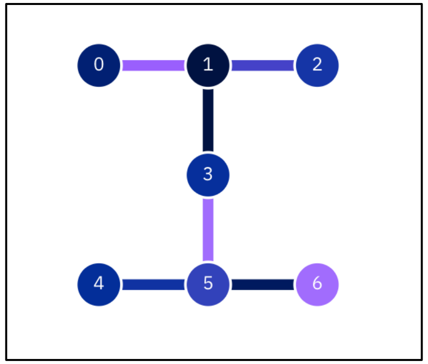
- Quantum machines are usually heterogeneous, coming in various sizes and configurations.
- Larger machines can execute larger applications but are limited by technology constraints

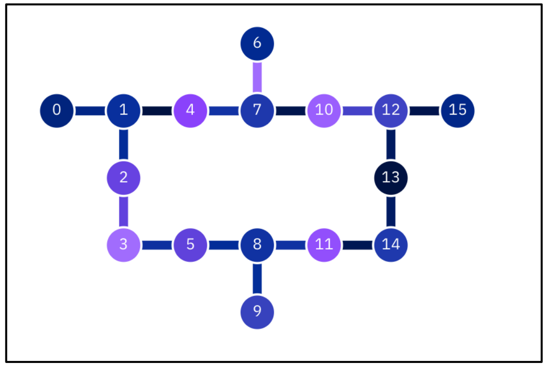


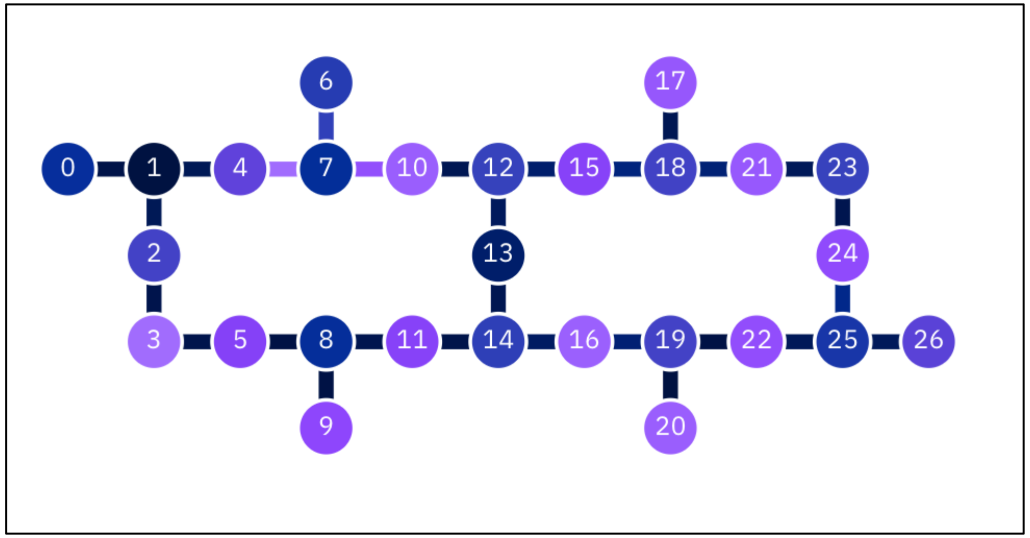
https://quantum-computing.ibm.com/

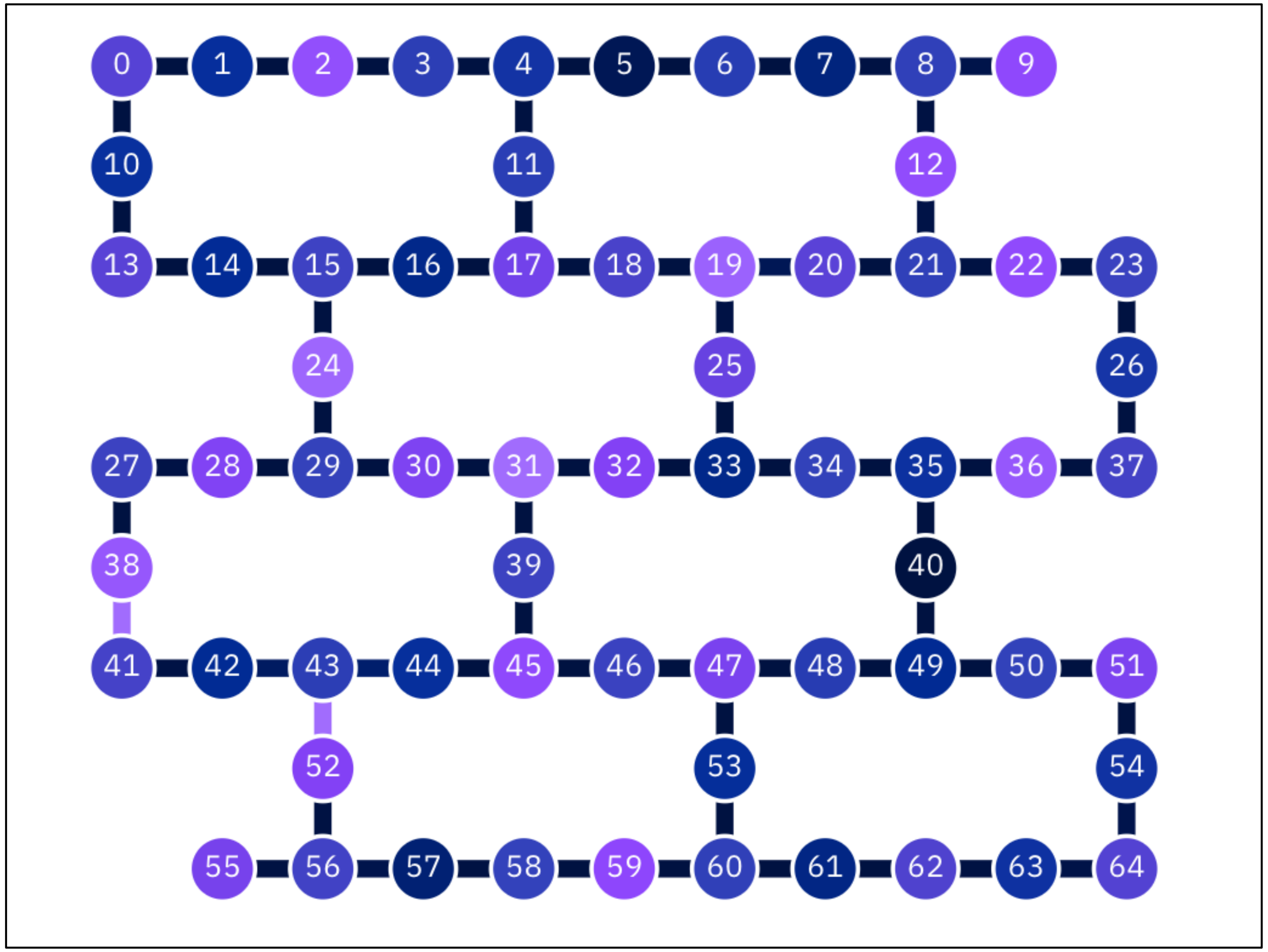
IBM Quantum Cloud: Topology Diversity

Noise constraints restrict connectivity and impacts topology of different sized machines, and thus their capability.



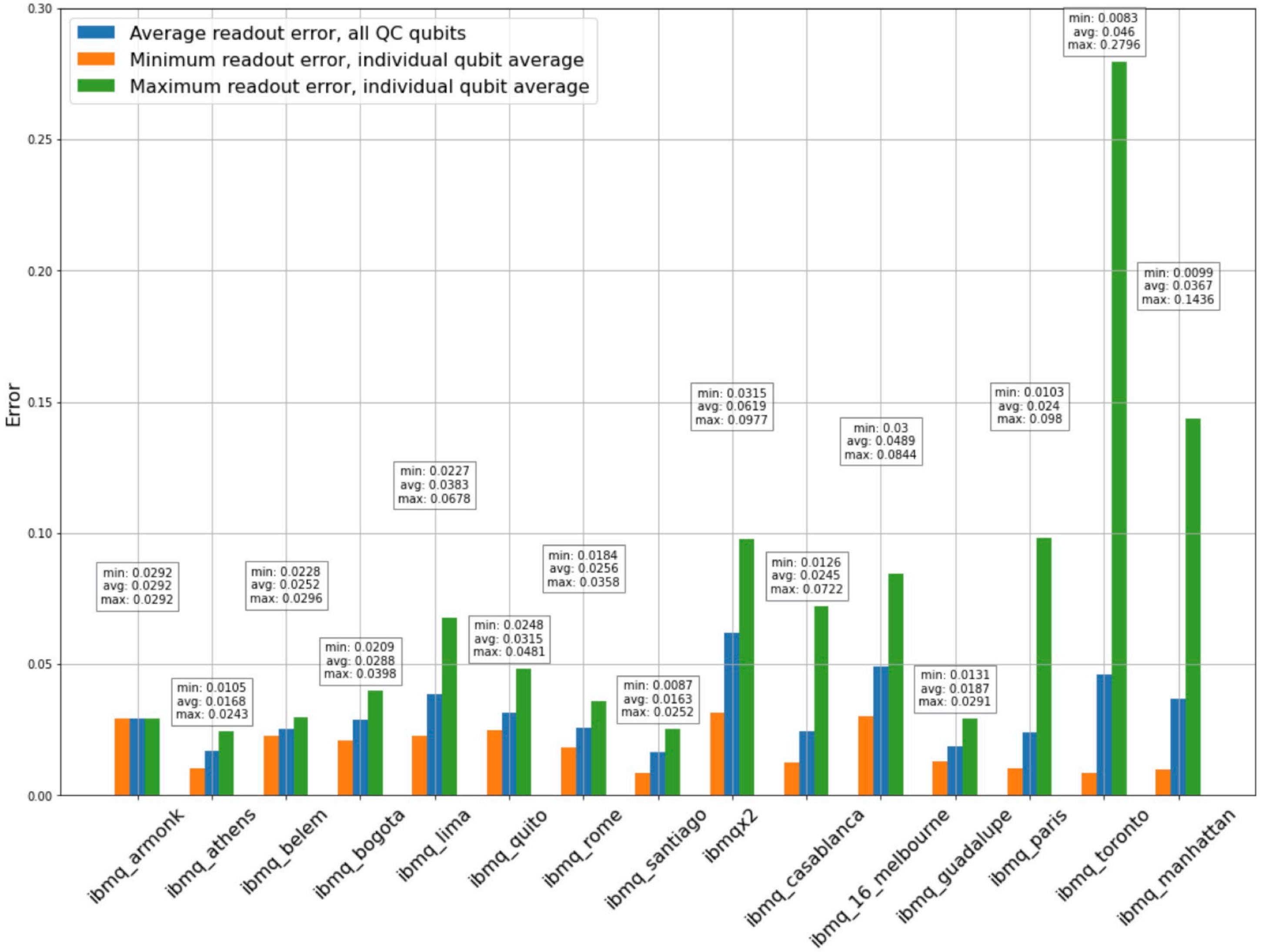






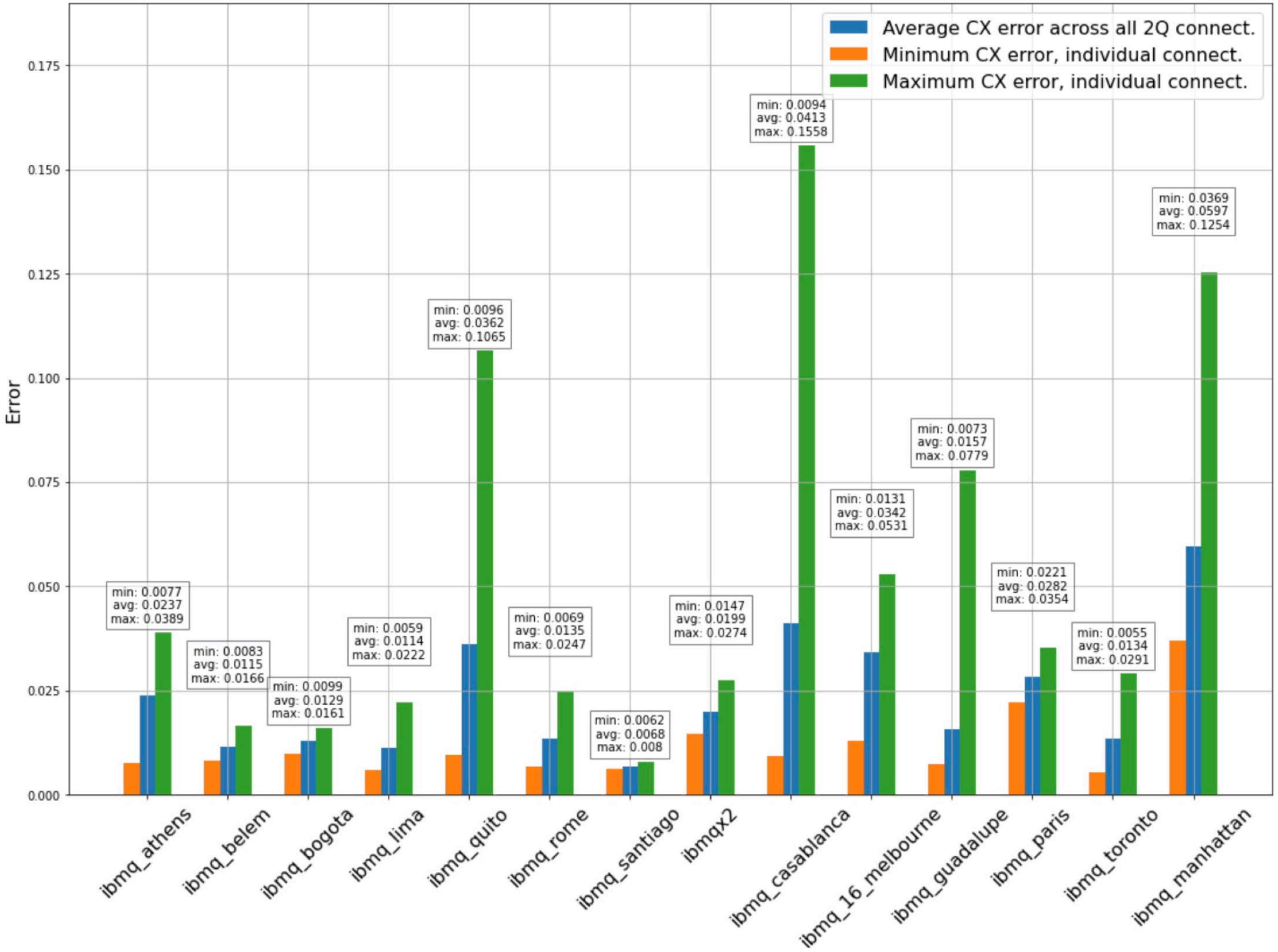
https://quantum-computing.ibm.com/

IBM Quantum Cloud: RO Error diversity (Spatial)



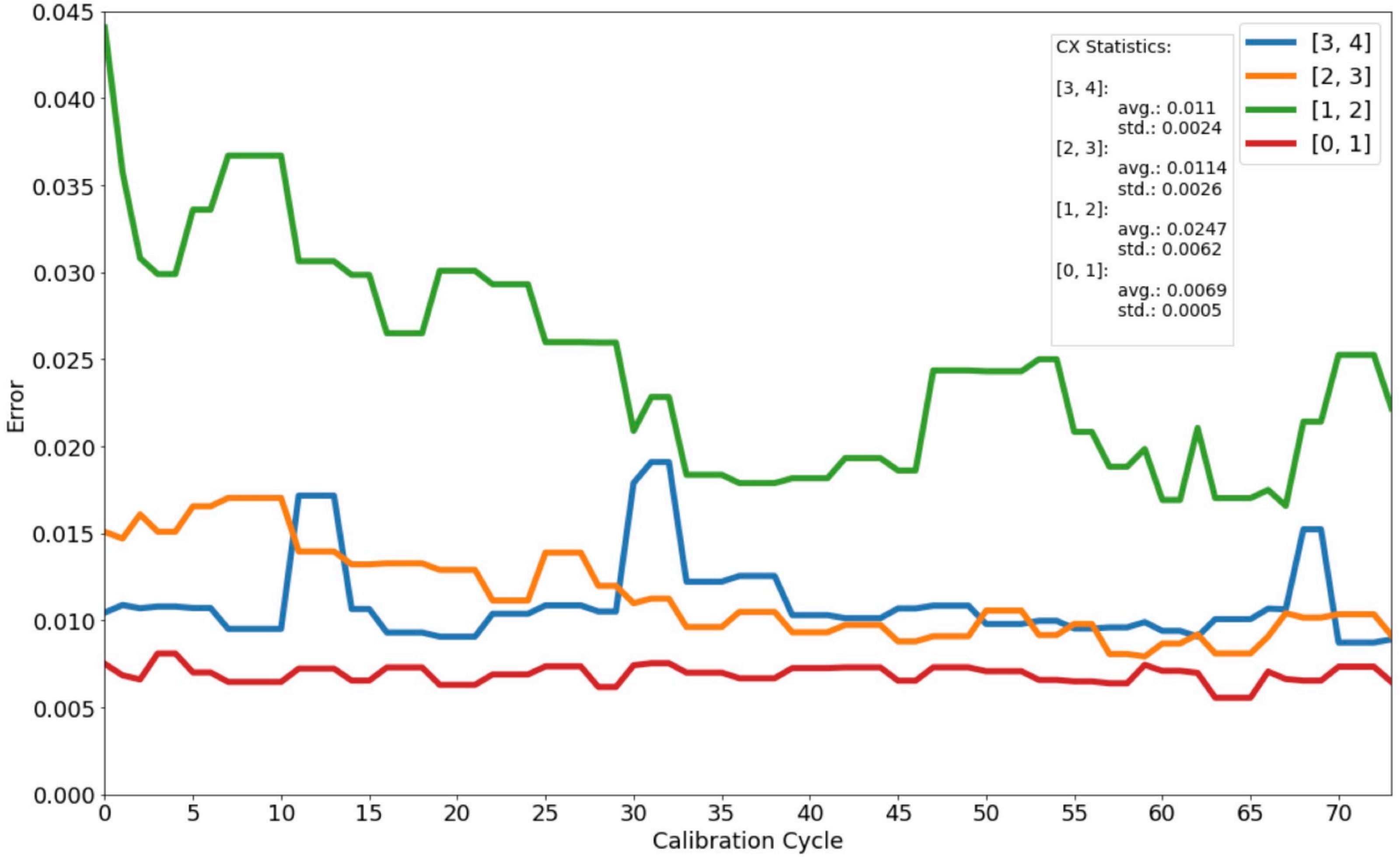
- Readout Error across IBM machines averaged over 2.5 months.
- Errors vary widely within machines as well as across machines.
- Worse-error qubits can be avoided on larger machines if low utilization.
- But average error rates are not necessarily well correlated with machine size.
- Readout errors can be corrected via measurement error mitigation but suffer from (re-)calibration vs accuracy trade-offs.

IBM Quantum Cloud: CX Error diversity (Spatial)



- CX Error across IBM machines averaged over 2.5 months.
- Machine connectivity constraints result in inserting SWAPs which lead to more CNOTs (and errors).
- Errors vary widely within machines as well as across machines.
- While worse-error qubits can be avoided in larger machines, avoiding qubits with good locality can lead to more SWAPs.
- Again, average error rates not well correlated with machine size.

IBM Quantum Cloud: CX Error diversity (Temporal)



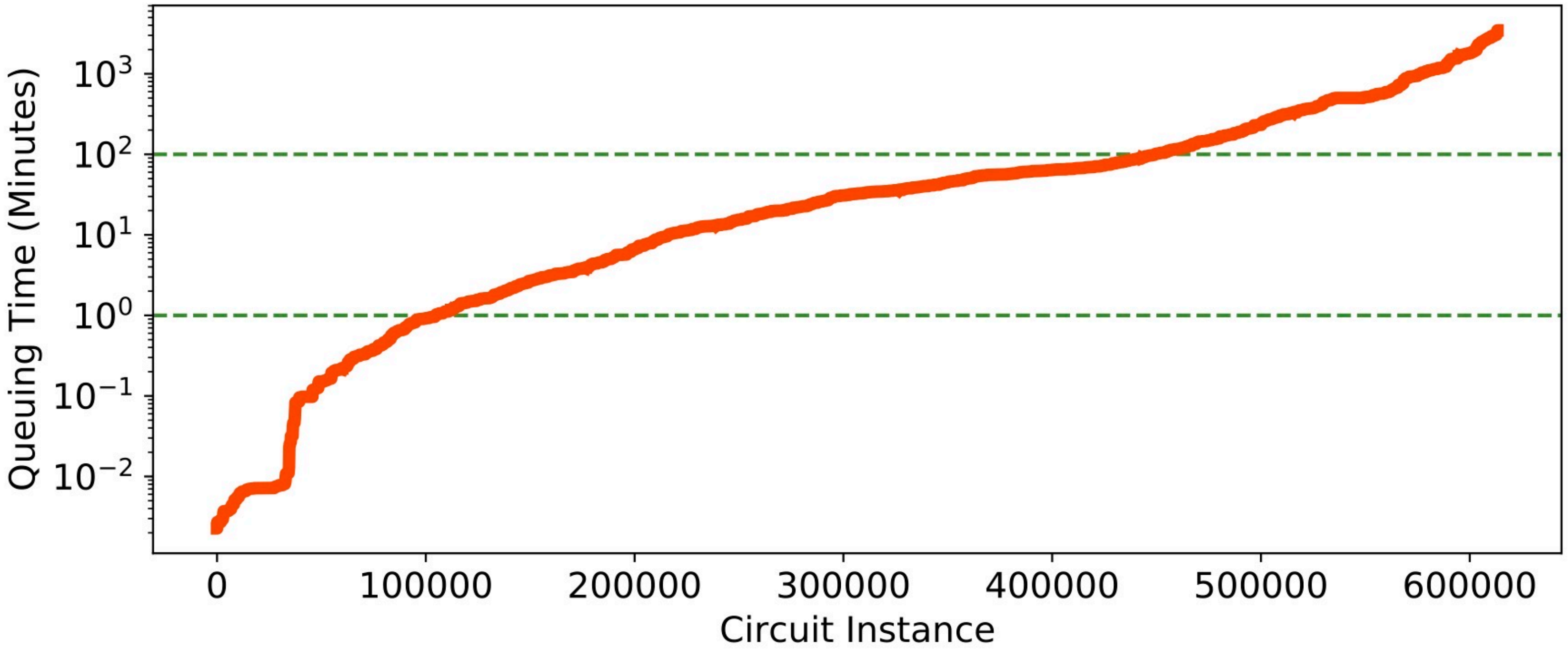
- CX Error on specific qubits on IBMQ Rome over 75 days / calibration cycles.
- Errors vary across calibration cycles meaning that both the optimal qubit set, as well as the application fidelity change over time.
- Error / qubit characteristics also drift within calibration cycles.

Thus, machine characteristics are challenging to comprehend and heavily influence the best machine for a usecase.

Jobs on Machines

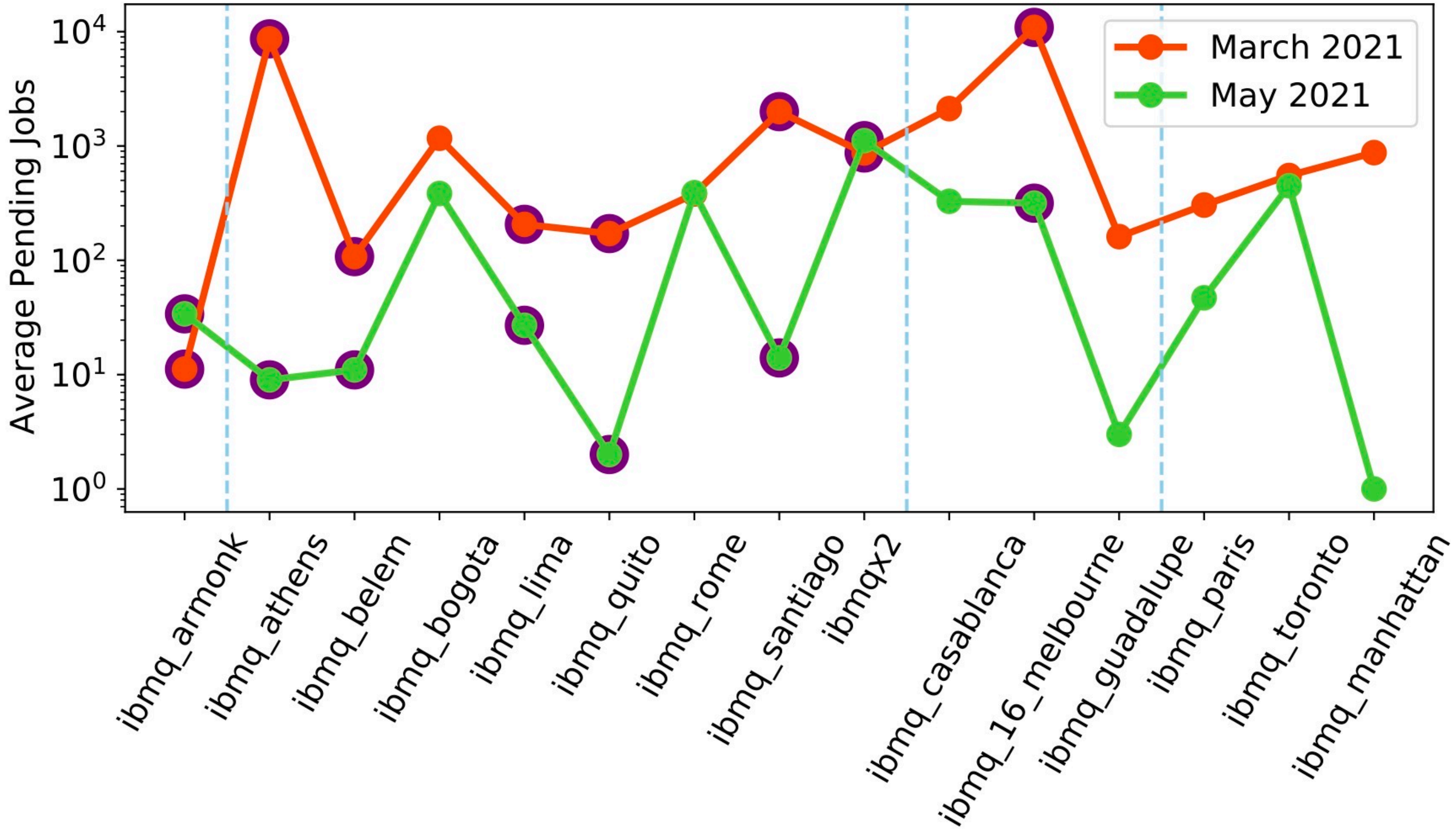
Queuing / Execution / Utilization / Fidelity

IBM Quantum Cloud: Queuing Times



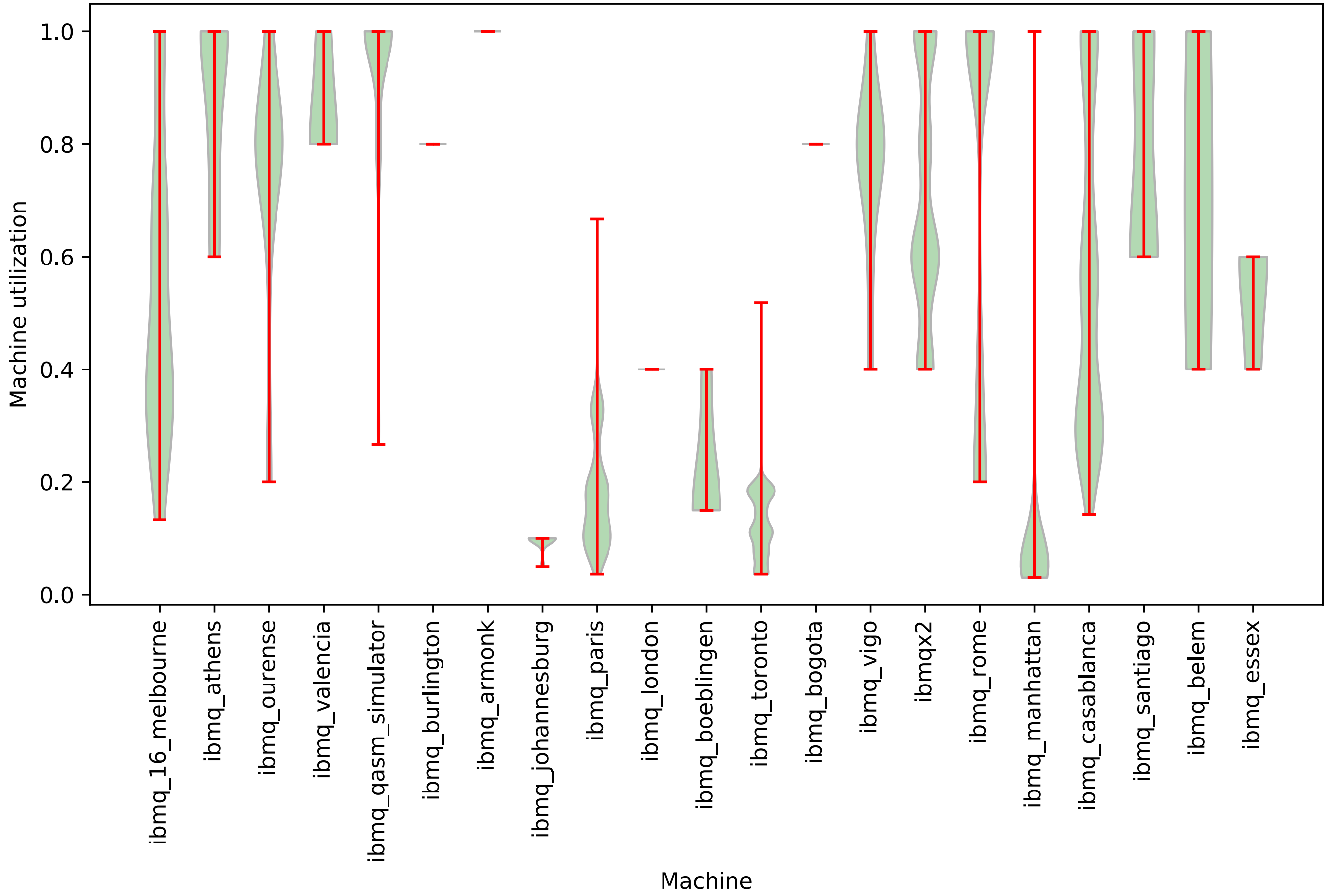
- Sorted queuing times from a two-year period.
- Green lines correspond to times of 1 minute and 2 hours.
- Median queuing time is around 1 hour.
- Over 25% of the circuits are queued for 2 hours or more.
- ~5% are queued for more than a day.

IBM Quantum Cloud: Per-machine trends



- Average pending jobs across different weeks, machines sorted by size.
- Publicly accessible machines are highlighted in purple.
- Public / older machines are in higher demand.
- Jobs are not distributed equally across machines.
- Distributions are not stable over time.
- Exacerbated by reservations?

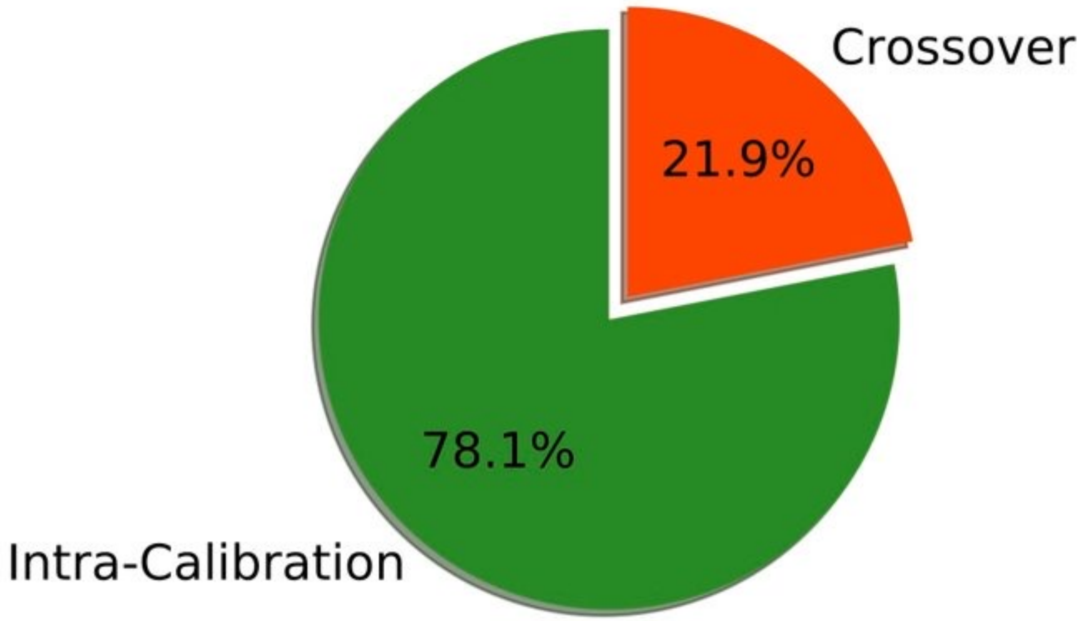
IBM Quantum Cloud: Machine Utilization



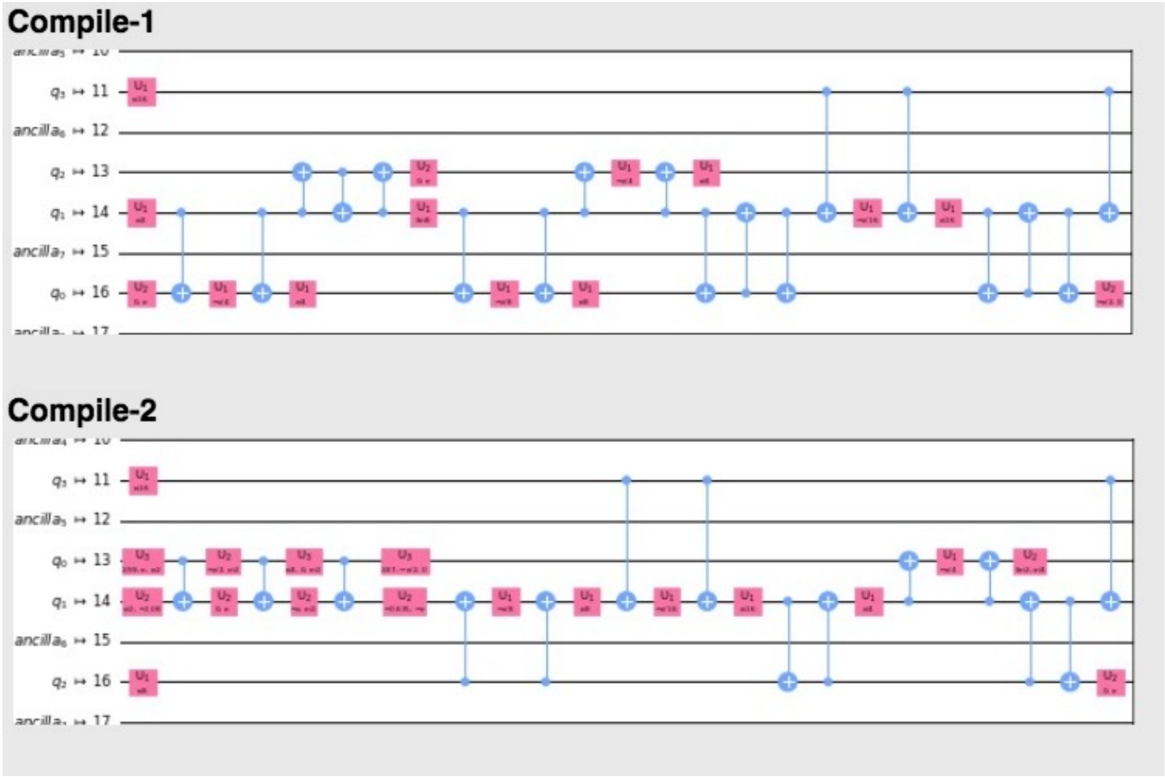
- Fraction of machine qubits that were used by the circuit.
- Utilization is lower on larger machines due to connectivity and bisection bandwidth.
- Large apps challenging to run due to connectivity constraints - increase depth / lower fidelity..
- Utilization non-uniform among same size machine. Choices made based on minimally understood heuristics.
- Usage choices often influenced by queued jobs instead of utilization - bad for throughput and overall fidelity.

IBM Quantum Cloud: Wait times => Errors

Jobs compiled for one calibration cycle
but executing in another

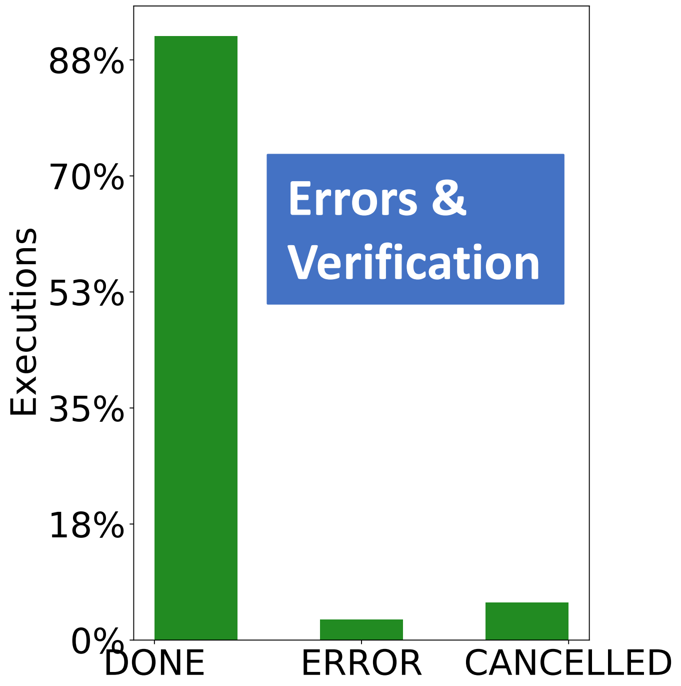


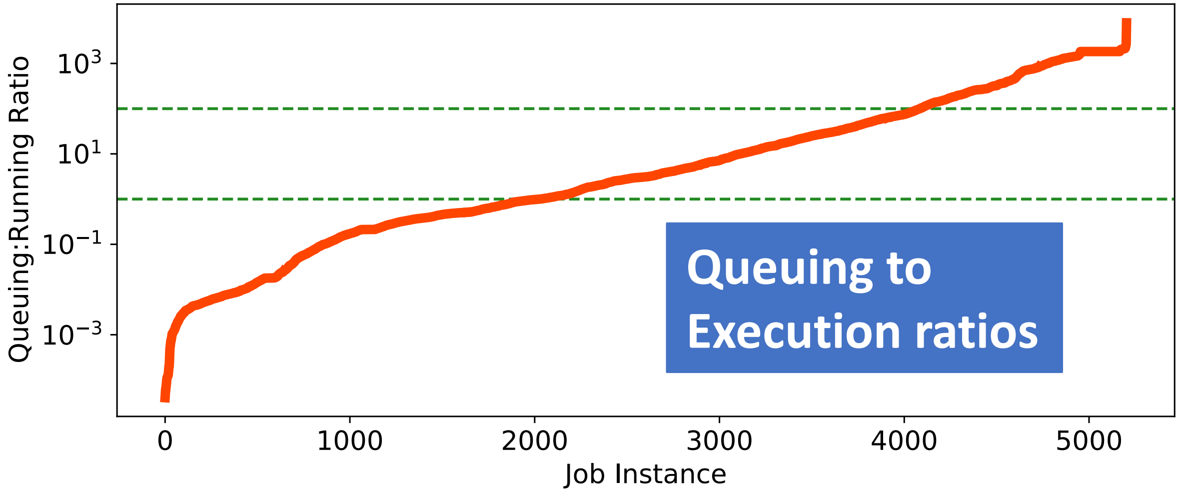
Noise-aware mapping over calibrations

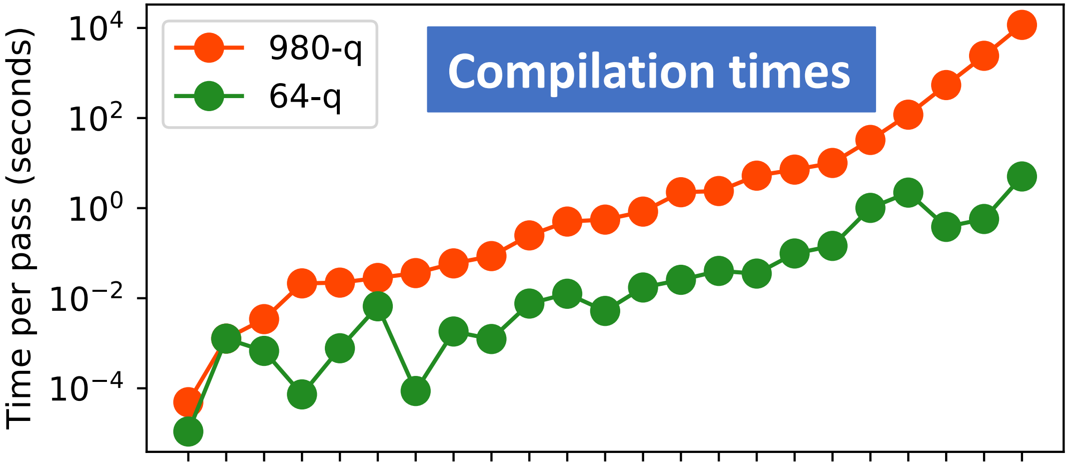


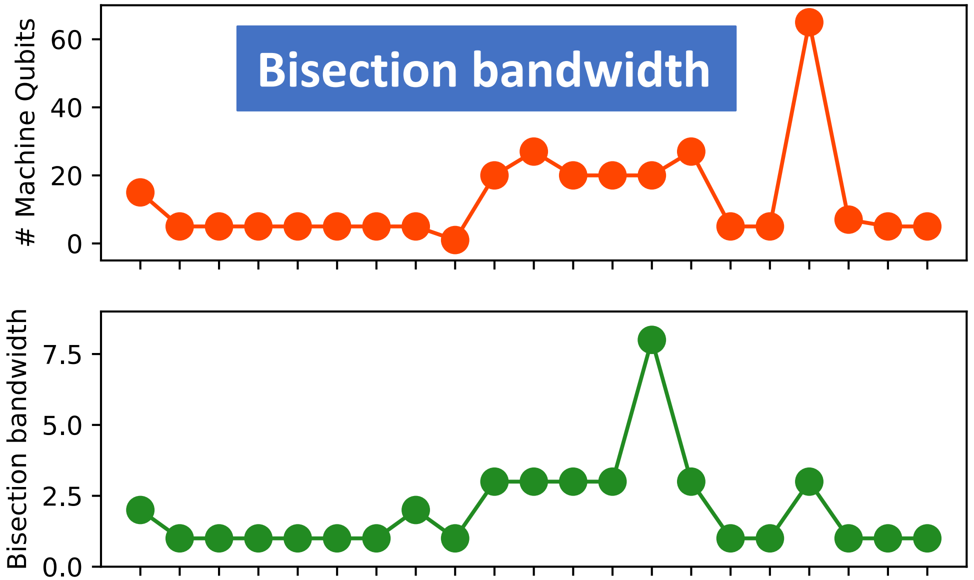
Thus, load is imbalanced and queuing times are often very long, influenced by sub-optimal machine-application mappings.

Other observations…









Takeaways

Comparisons / Recommendations / Diversity

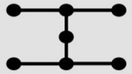
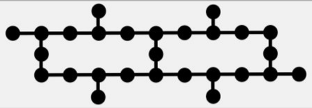
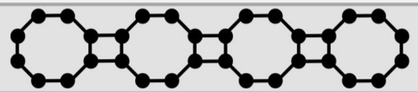
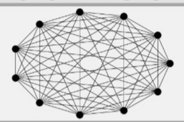
Contrasting against Classical Cloud

- **Incorrect executions:** While verification and incorrect executions are common in classical computing, the challenge in quantum computing is that verification techniques are still at a very nascent stage.
- **Compilation times:** Quantum compilation is deeply tied to the characteristics of the hardware - error rates, topology and its calibrated state. Compiled executables cannot be distributed independent of the machine.
- **Fidelity:** Choosing the right machine to maximize the fidelity is unique to quantum computing, and especially complex given the unique circuit interactions with the environment it executes in.
- **Queuing times:** While reducing queueing times is well researched in the classical domain, they are even more critical in the quantum domain because of the temporally changing characteristics.
- **Execution times:** Execution times in quantum are highly predicable compared classical computing. Overheads dominate quantum execution times. Higher predictability can allow for better scheduling policies.

Insights / Recommendations

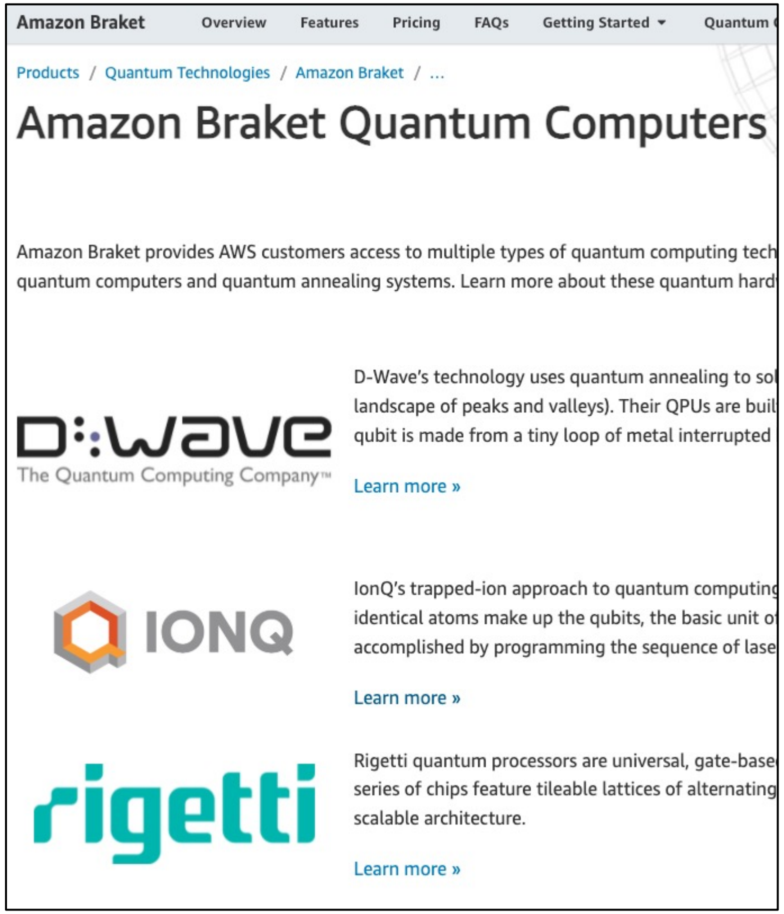
- **Verification:** As circuit complexity increases, so will the potential for mistakes and incorrect executions. Debugging and verification strategies are a must to maximize useful system utilization.
- **Compilation:** Compilation times are on the increase. Need to build more scalable compilation strategies, as well as potentially overlap some compilation tasks with the already long queuing times.
- **Machine Diversity:** Machine characteristics vary widely across machines and time. Heuristics for machine selection will be critical and require more study, especially with increasing application / machine complexity.
- **Queuing Times:** Load imbalance leads to widely varying queuing times reducing system throughput, and application fidelity. Predicting queuing times is critical especially with increased demand and competitive business models.
- **Execution Times:** NISQ-era execution times are likely to be highly predictable and mostly dependent on a few characteristics. Predicting execution time accurately amplifies the possibility of efficient scheduling.
- **Resource Management:** To maximize the overall system utilization/throughput and to improve application fidelity across users, machine-aware system wide management of resources should be explored.

Multiple vendors: More diversity

Machine	Qubits	Coherence Time (µs) (T1, T2)	Gate Times (µs) (1Q, 2Q, Rd)	Gate Errors (%) (1Q, 2Q, Rd)	Topology
IBM-Casablanca	7	91.21, 125.23	0.035, 0.443, 5.9	0.028, 0.83, 2.09	
IBM-Montreal	27	104.14, 86.88	0.035, 0.423, 5.2	0.052, 1.76, 1.96	
Rigetti-Aspen-9	32	31.1, 17.5	0.06, 0.16, 2*	0.6, 4.9, 5.84	
IonQ	11	>1e7, 2e5	10, 210, 100	0.28, 3.04, 0.39	
AQT	4	62, 37	0.03, 0.152, 1.02	0.083, 2.1, 1.25	

Tech Policy

Amazon joins race for quantum computer with new Caltech center


Amazon Braket Overview Features Pricing FAQs Getting Started Quantum

Products / Quantum Technologies / Amazon Braket / ...

Amazon Braket Quantum Computers

Amazon Braket provides AWS customers access to multiple types of quantum computing tech
quantum computers and quantum annealing systems. Learn more about these quantum hard

D:WAVE The Quantum Computing Company™

D-Wave's technology uses quantum annealing to sol
landscape of peaks and valleys). Their QPUs are buil
qubit is made from a tiny loop of metal interrupted
Learn more »

IONQ

IonQ's trapped-ion approach to quantum computing
identical atoms make up the qubits, the basic unit o
accomplished by programming the sequence of lase
Learn more »

rigetti

Rigetti quantum processors are universal, gate-base
series of chips feature tileable lattices of alternating
scalable architecture.
Learn more »

https://aws.amazon.com/braket/quantum-computers/

Talk Outline

Background and Characterization of Resources

<u>Building a Resource Manager</u>

Other Work: Variational Quantum Algorithms

Summary: Managing jobs/resources in the quantum cloud

- Background: Machines are diverse and user demands are growing constantly: limited knowledge on best machine for any usecase + very long queuing times.

- Goal: An automated job scheduling and resource allocation approach which achieves optimal trade-offs between fidelity, wait times, QOS specifications etc.

- Proposal:
  - Predicts fidelity trends across machines.
  - Estimates run times and, thereby, wait times.
  - Uses a utility function that is inherently able to prioritize fidelity improvements at low load, wait time reduction at high load and balanced otherwise.
  - Optimizes for other constraints such as QOS, machine recalibration, etc.

- Result: Reduce wait times by over 3x and improve fidelity by over 40% on different usecases.

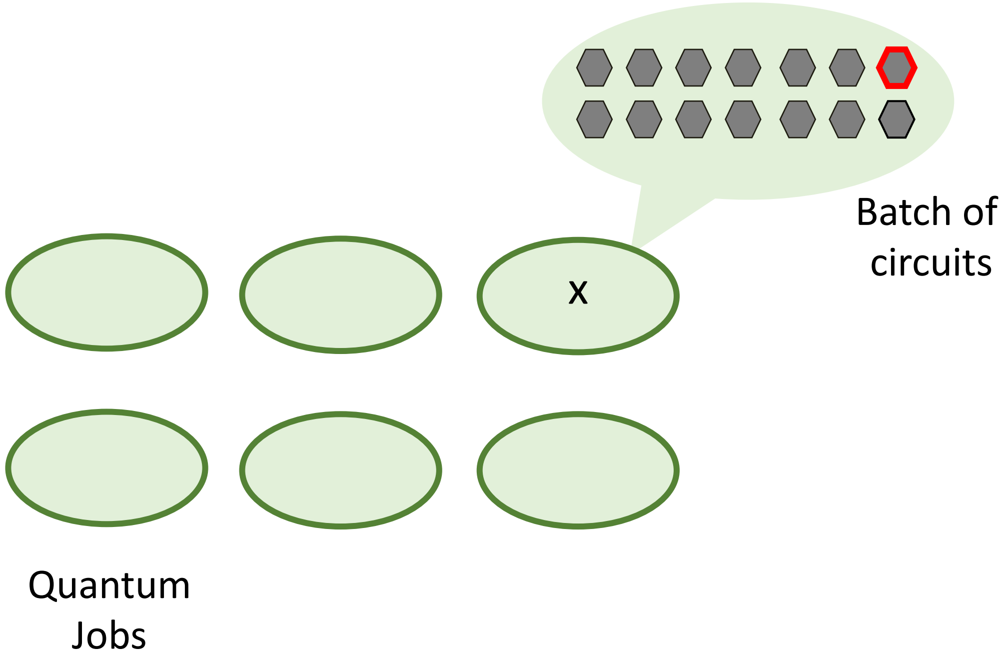
Gokul Ravi, Kaitlin Smith, Prakash Murali, Frederic T. Chong. Adaptive job and resource management for the growing quantum cloud. QCE 2021. Oct 2021.

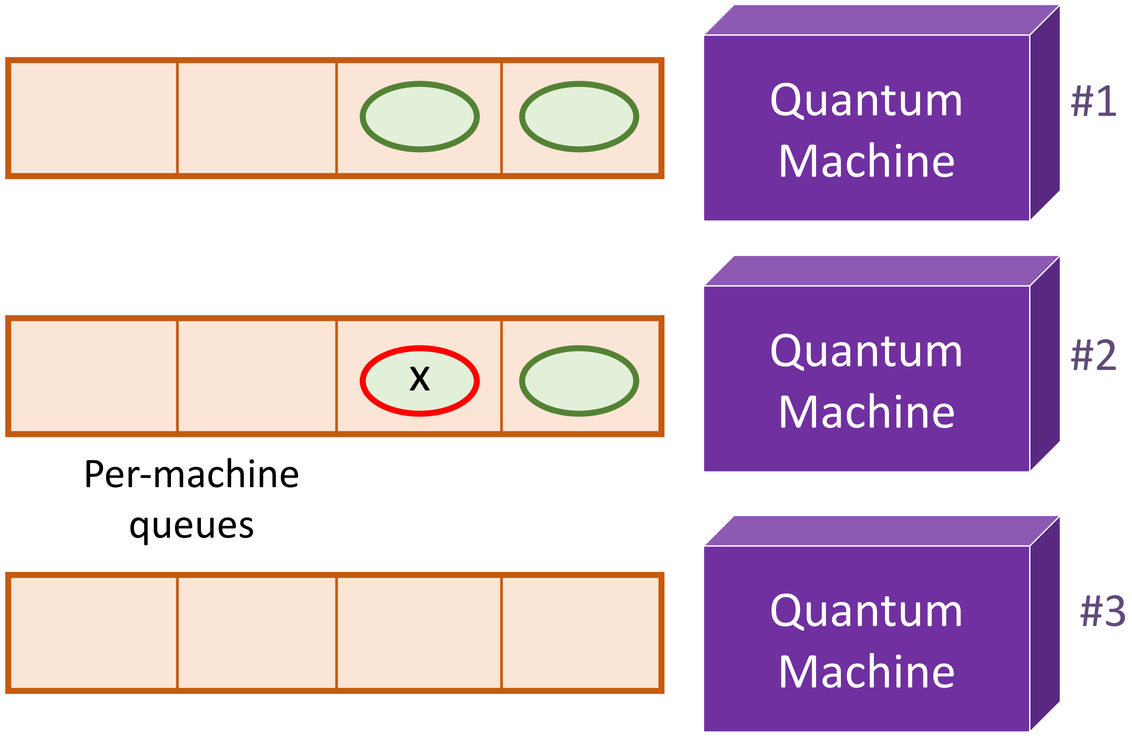
Design

Resource Manager / Queuing time estimator / Fidelity correlator / Utility function

Submitting jobs to the quantum cloud

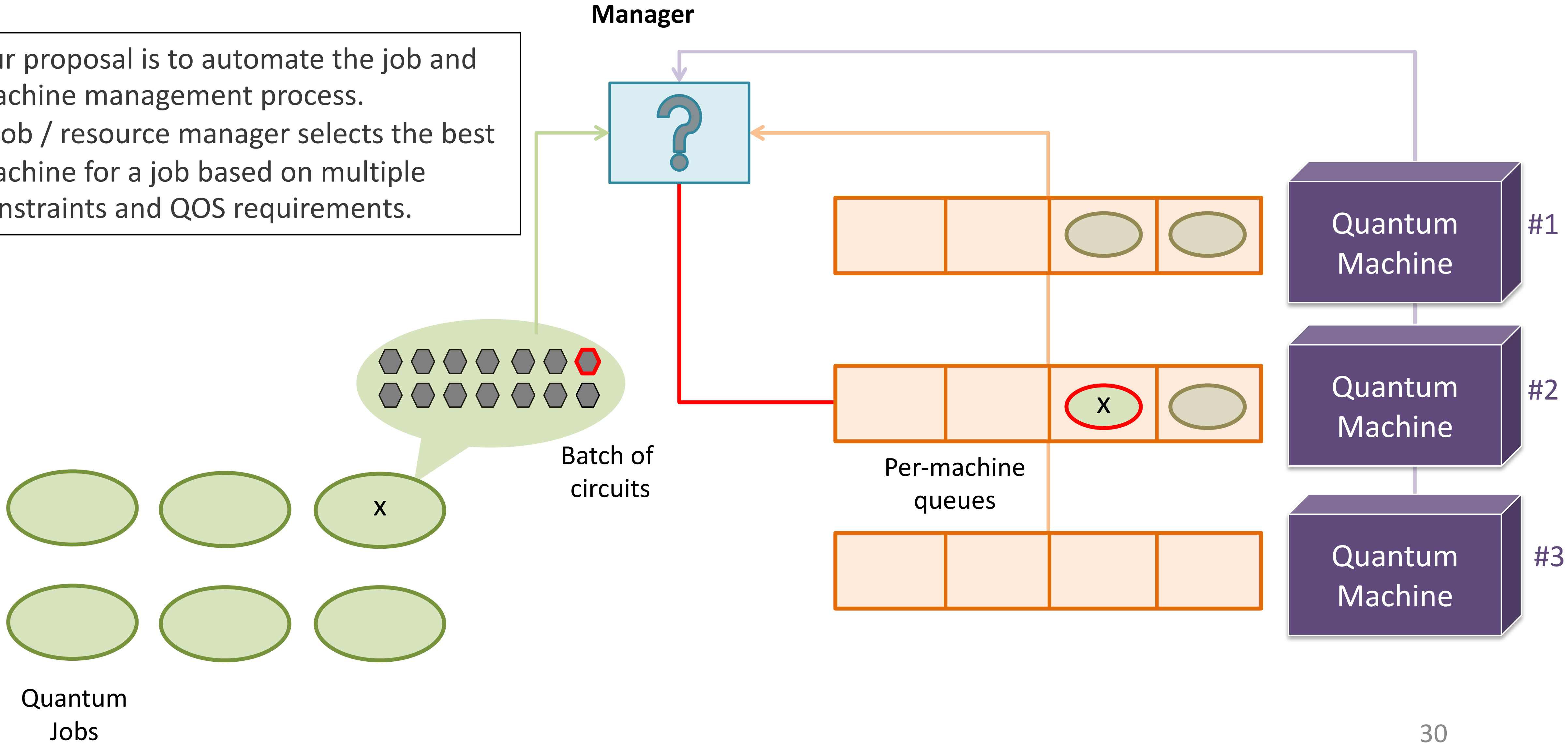
- Access to the quantum cloud (modeled on IBM).
- Users create a batch of circuits to execute as a job – all circuits in a batch are executed back to back.
- User selects a machine to execute on and compiles all circuits in the job for the target machine.
- Job is sent to the queue and user waits for execution and return.



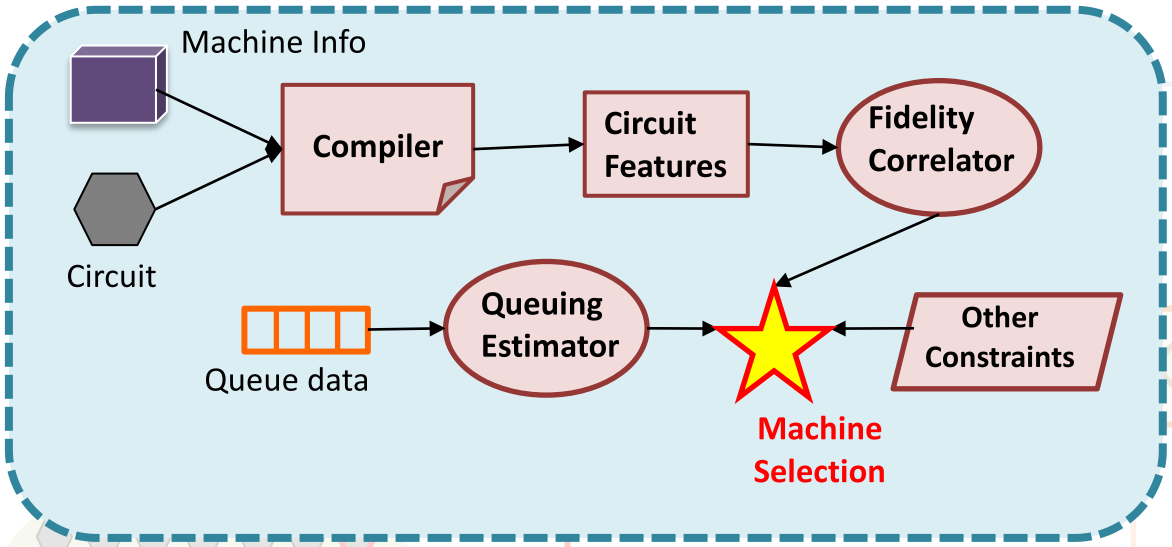


Quantum Cloud Resource Manager

- Our proposal is to automate the job and machine management process.
- A job / resource manager selects the best machine for a job based on multiple constraints and QOS requirements.



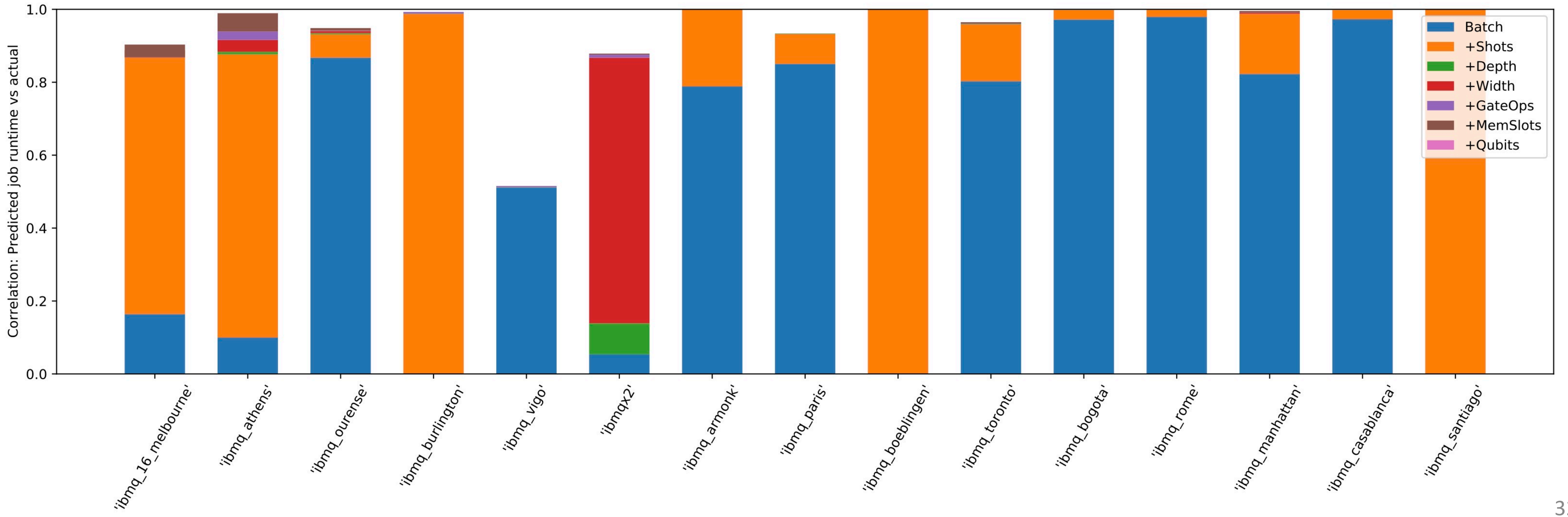
Manager in action



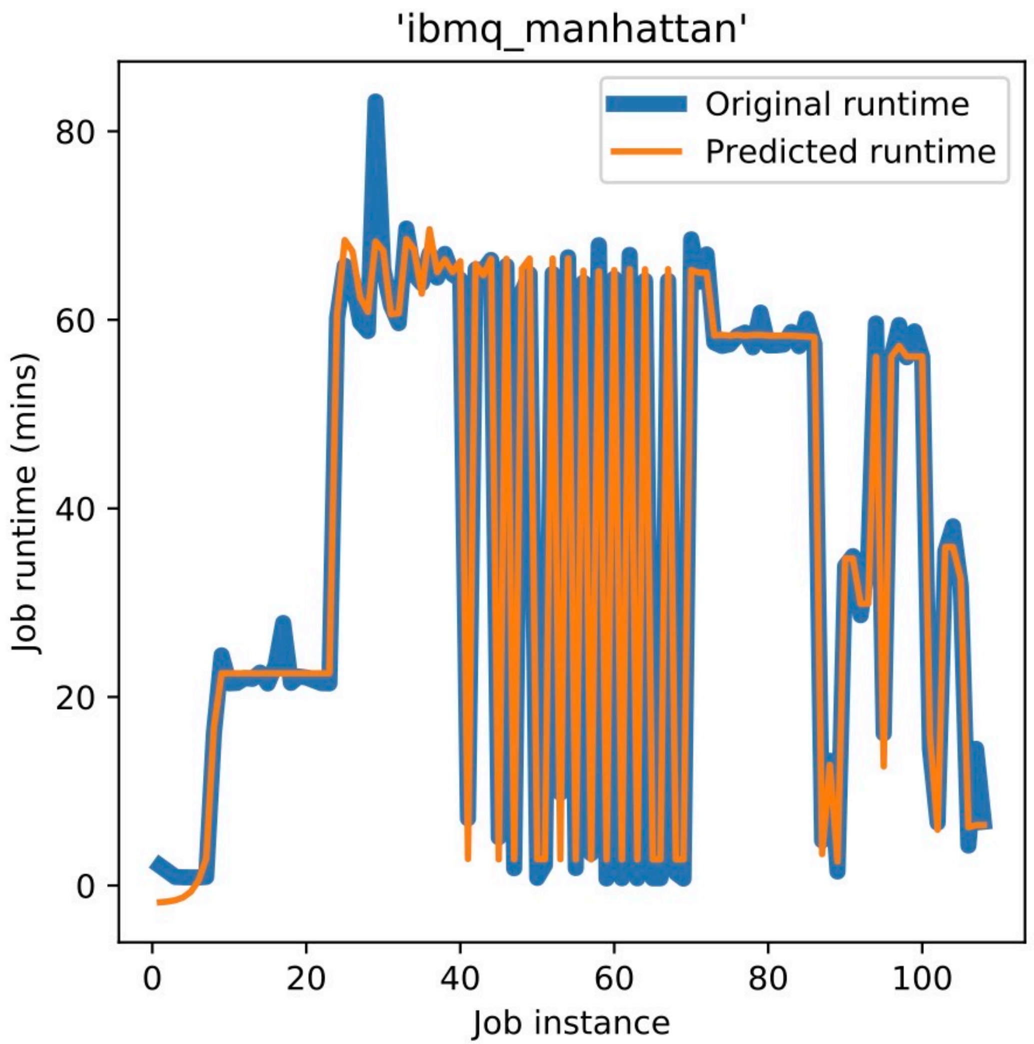
1. A job's representative QC is compiled for all suitable machines.
2. Post-compilation features of the circuit for each machine are passed to the fidelity correlator.
3. Correlator provides a correlation between circuit features and expected fidelity on each machine.
4. Queuing information, job size and number of shots are used to predict wait times on each machine.
5. Other constraints like QOS requirements and calibration schedules are considered.
6. Machine is selected and any uncompiled circuits in the job are compiled to the selected target.
7. Job joins the machines queue and waits for execution.
8. Scheduler can provide inputs to optimally space out machine recalibrations.

Queuing Estimator: Job features to predict runtime

- Correlation of job features vs actual runtimes: correlation is 0.95 or above on almost all machines.
- The major contributor is the batch size, i.e. the number of circuits in the job.
- A second contributor is the number of shots, influential when the batch size is low.
- Other factors like depth, width and memory slots have limited influence

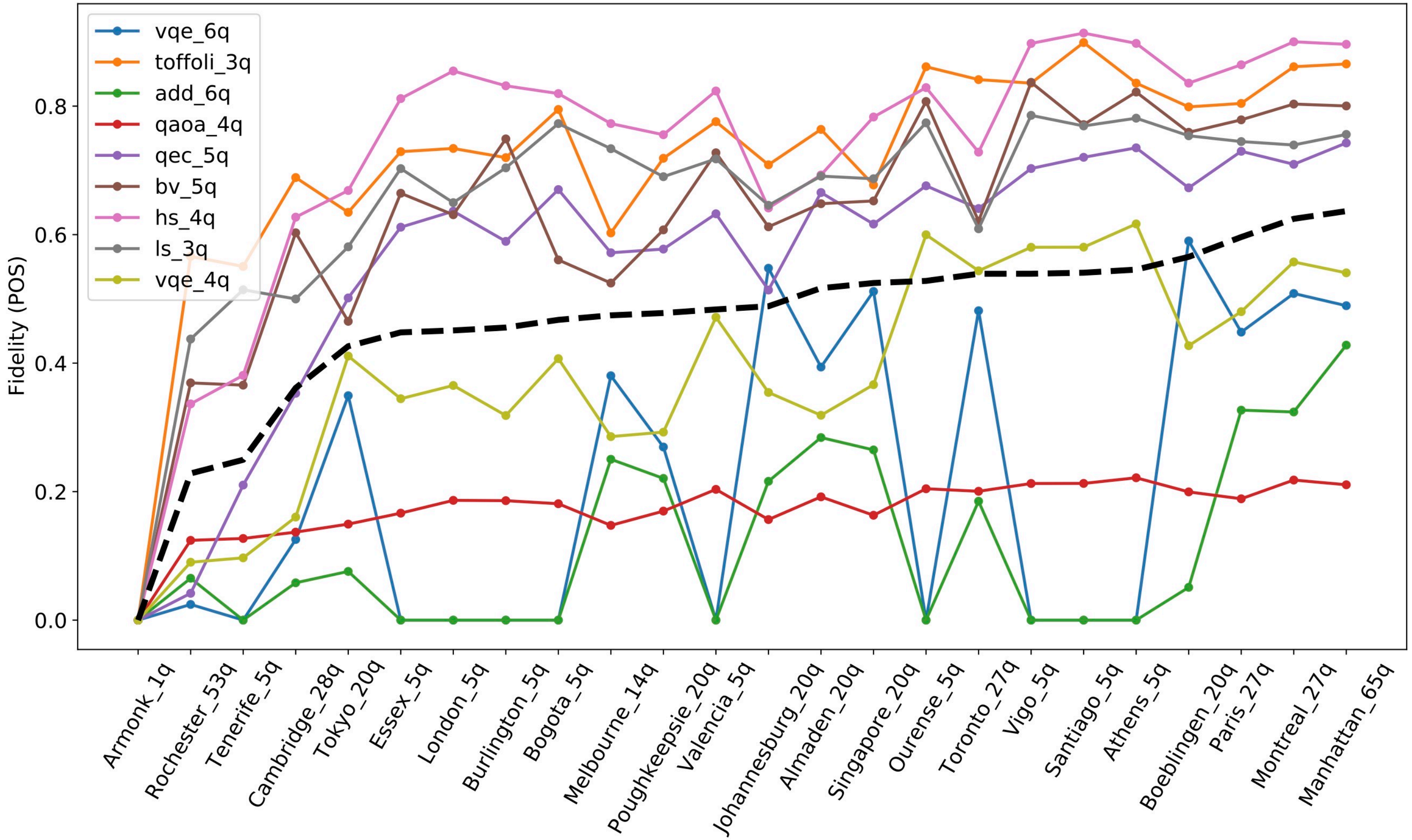


Predicted vs Actual Runtime



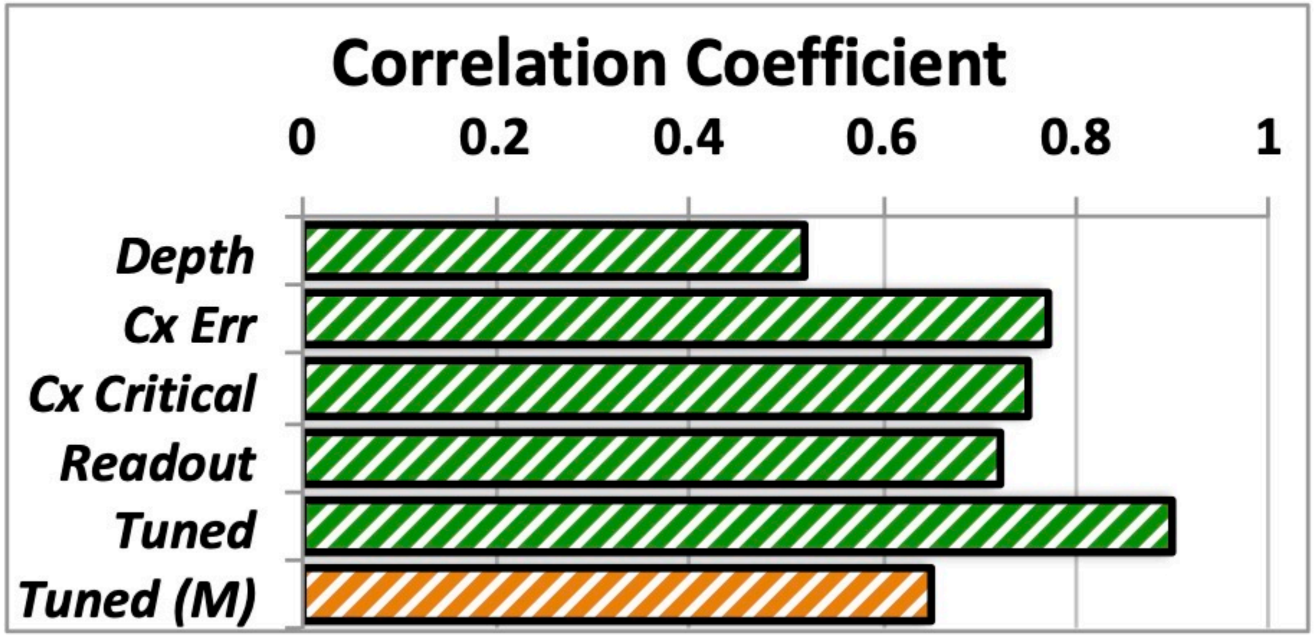
- Actual vs Predicted runtimes for different jobs on IBMQ Manhattan.
- While machine and job characteristics can vary widely, application runtimes remain predictable.
- Runtime predictions can then be accumulated to obtain queuing times.

Fidelity trends across machines



- Fidelity of 9 benchmarks on the 26 simulated quantum machines.
- Machines are sorted by this average app fidelity.
- Fidelity trends exist - Athens / Manhattan often perform better but are sometimes application dependent.
- Correlation isn't purely related to the size of the machines - Athens and Santiago are 5q machines.
- Potential macroscopic trends within machine behavior but not simple enough to be naively captured.

Fidelity Correlator using circuit characteristics



- Fidelity correlator uses four features of circuit compiled to a particular quantum machine.
  - Circuit Depth,
  - Avg. CX error over the circuit,
  - Avg CX in the circuit critical path,
  - Readout errors.
- Model is a product of linear terms: $Fn=\Pi(ai+bi*xi)$, where Fn is the fidelity, xi is the feature and ai, bi are the tuned coefficients.
- Model is tuned on a pre-collected training set.
- Figure shows correlation between actual application fidelity and the tuned model, as well as with each feature.

Utility Function

- Maximizing the function should result in a job schedule that provides a good balance between fidelity and queuing time *(at any load)*

- The function should also account for QOS requirements and the impact of calibrations and stale compilations on the utility of the machine.

- Beyond the above (not pursued here), the utility function could account for user priorities, improved machine utilization etc.

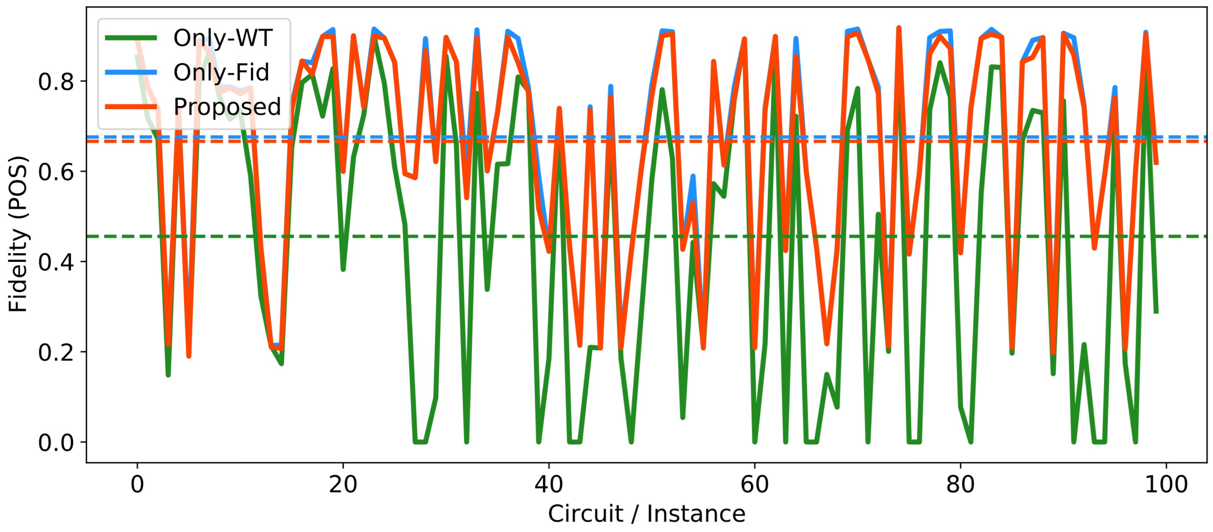
- We use a balanced linear equation of the form $\Sigma(ai*xi)$. xi (features) are between 0 and 1 and ai (coefficients) can be tuned empirically.

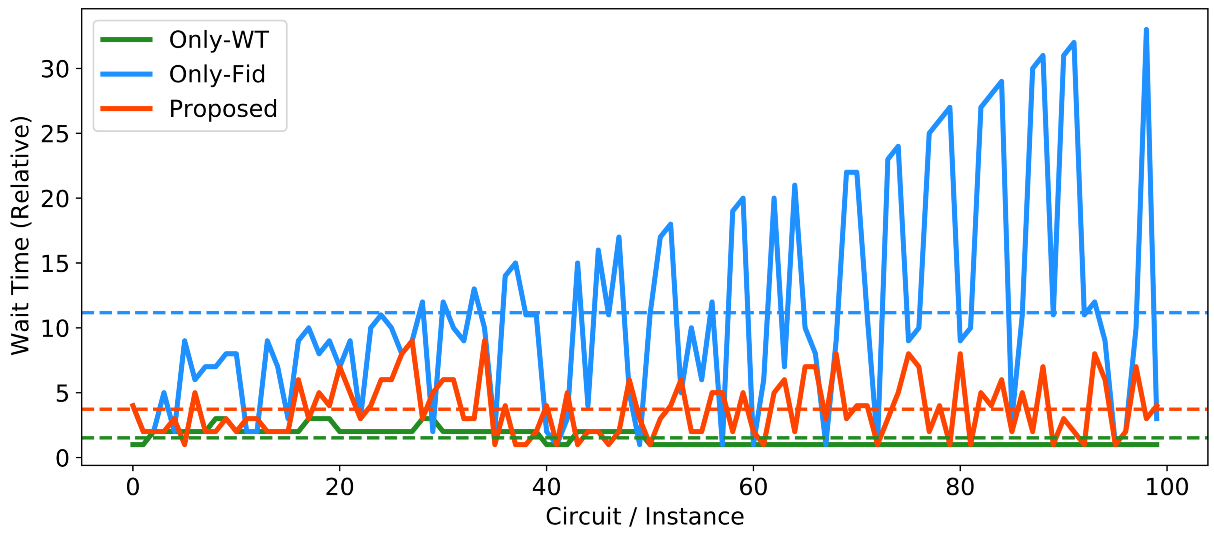
Evaluation

Studies / Limitations

Optimizing for Fidelity / Wait Time (Low load)

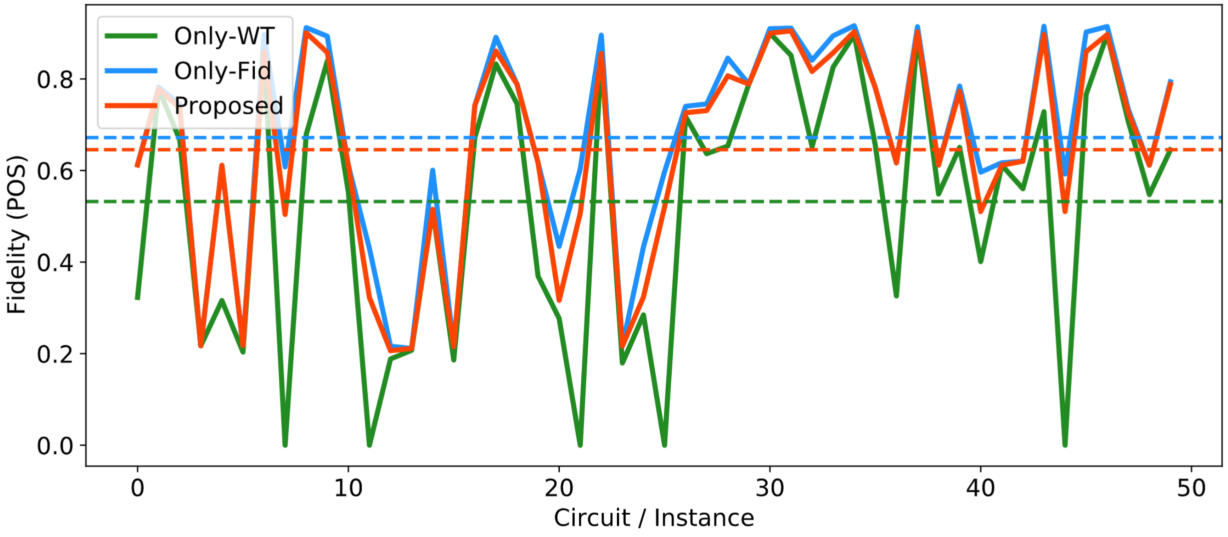
Low load Ideal: Choose highest fidelity machines, since the queuing times are not significant and thus best results are worth the short wait.

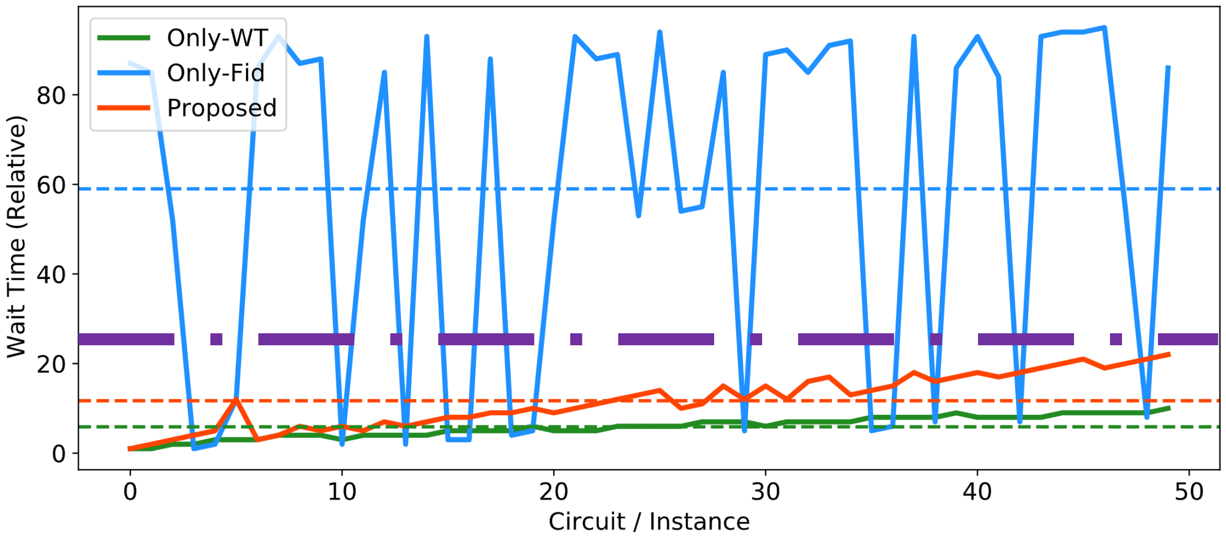




Incorporating QOS (WT < 25)

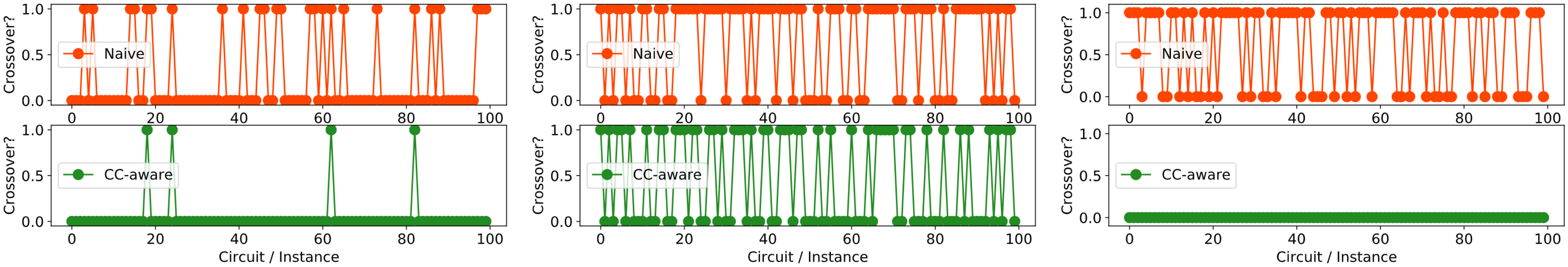
Fidelity and wait times for a QOS which tolerates wait times of up to "25". The strict bound means that Proposed approach sacrifices about 5% of maximum fidelity but still achieves 20% higher fidelity than the Only-WT approach.





Avoiding Calibration Crossovers

- Machine-aware compilation on an old calibration cycle is not optimal for execution on a new calibration cycle.
- CC-aware approaches schedule near-calibration jobs on machines with low queuing time.
- At high load this is insufficient, and crossovers are alleviated through a staggered calibration approach.



(a) Low Load **(b) High Load** **(c) Staggered Calibration (High Load)**

Limitations / Future work

- Behaviors observed are a partial consequence of IBM and user policies.
- Does not optimize for user priorities, machine utilization, drift, dynamic changes to system load etc.
- Compilation across multiple machines does not scale. Identify execution characteristics which can be estimated without compilation to shortlist machines.
- Improve fidelity correlation and execution time prediction models to achieve tighter bounds.
- Explore staggered calibration policies based on observing queuing times, job arrival patterns etc.
- Current work uses a simulated loaded system. Real-world testing for practical use.

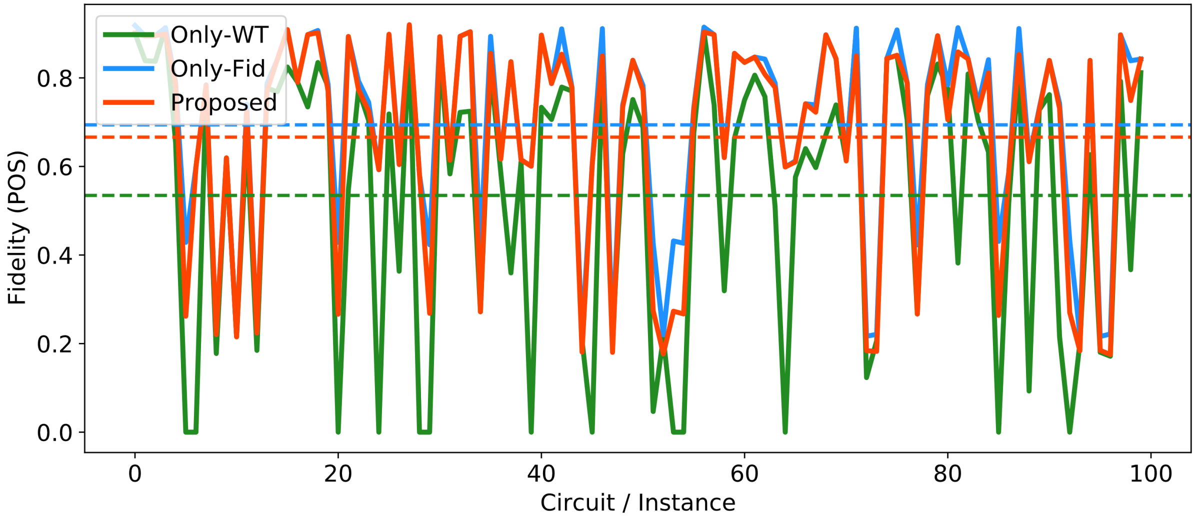
Thank you!

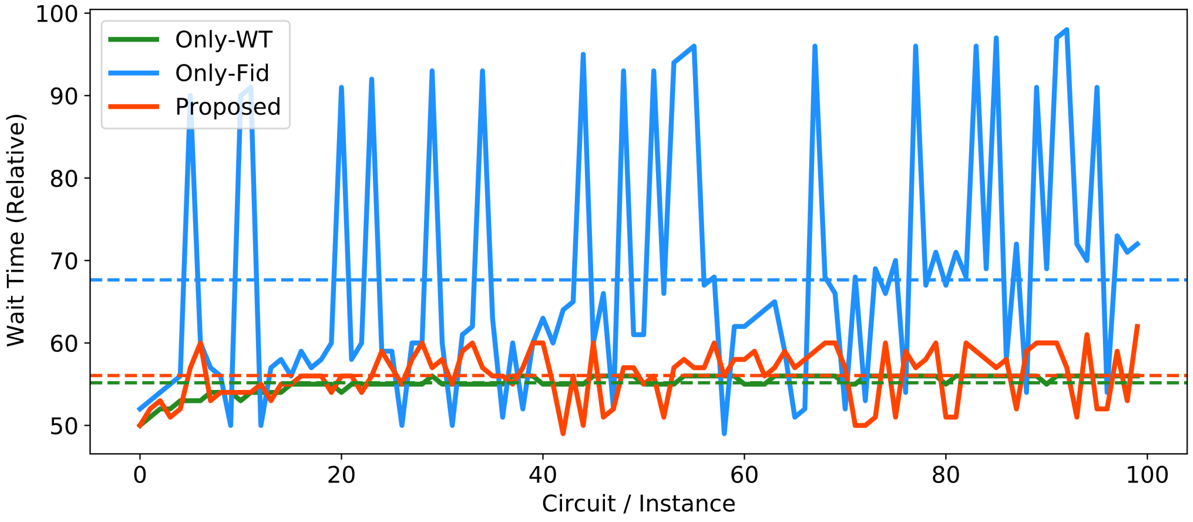
gravi@uchicago.edu

Backup

Evaluation: Optimizing for Fidelity and Wait Time
High load

High load Ideal: Accept some loss in fidelity to achieve reasonable queuing times, though we would still want thefidelity to be substantial enough for realistic benefits.





Experimental Setup

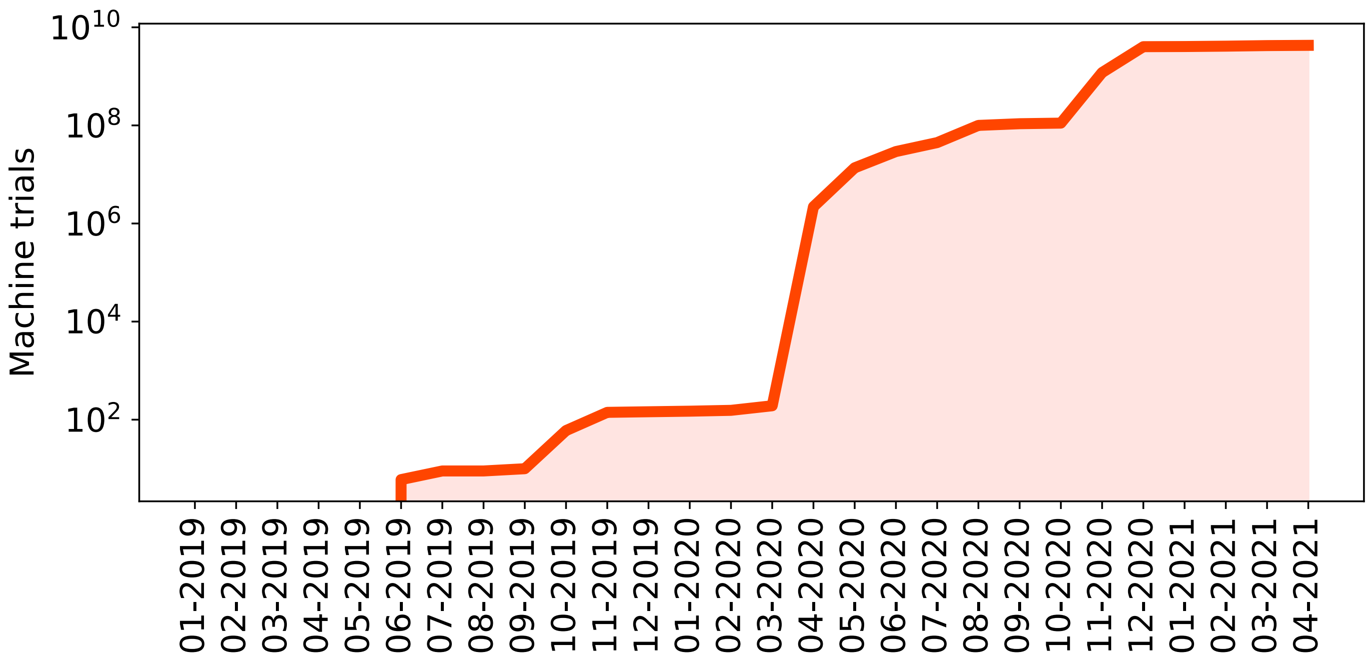
- Compilation: IBM Qiskit, Noise aware compilation
- Cloud Simulation: 26 device models of IBM quantum machines + different load distributions of low, high and random queuing jobs / times across the machines.
  - Low load: < 10% of maximum queuing on each machine,
  - High load: 50-100%,
  - Random Load: 1-100%
- Benchmarks: Toffoli, Hidden Subgroup problem, Bernstein-Vazirani, Linear Solver, QAOA, VQE, Repetition Code Encoder, RC Adder.
- Metrics: Probability of Success, Queuing Time
- Comparisons: a) Only Wait Times, b) Only Fidelity

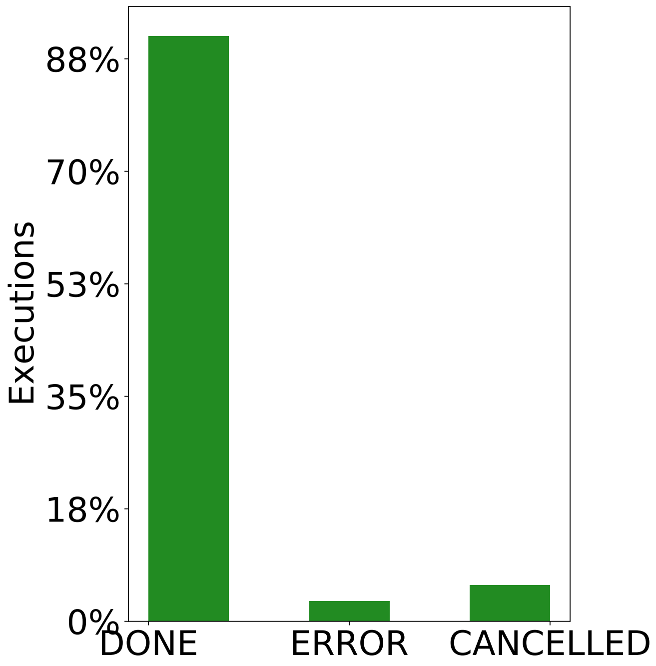
Scope of the study

- **Data:** Our study has focused on data collected across IBM's fleet of quantum computers, from over a two year period in an academic setting.
- **Queuing:** The queuing data is generally applicable to all users of the IBM quantum systems over the studied period. Increasing demand for machines is not limited to IBM machines.
- **Execution:** The execution data is less closely tied to the specifics of the quantum circuits being run and is more tied to the size of the jobs - all of which again are applicable to all users of these systems.
- **Fidelity:** The general impact of calibration, noise characteristics, constraints of device connectivity etc on application fidelity, are not restricted in anyway to the specific circuits executed in this study.
- **Device:** Insights are useful to all superconducting devices which are are limited in connectivity, more noisy and require frequent recalibration and generally extrapolate to other devices like Trapped-Ion.

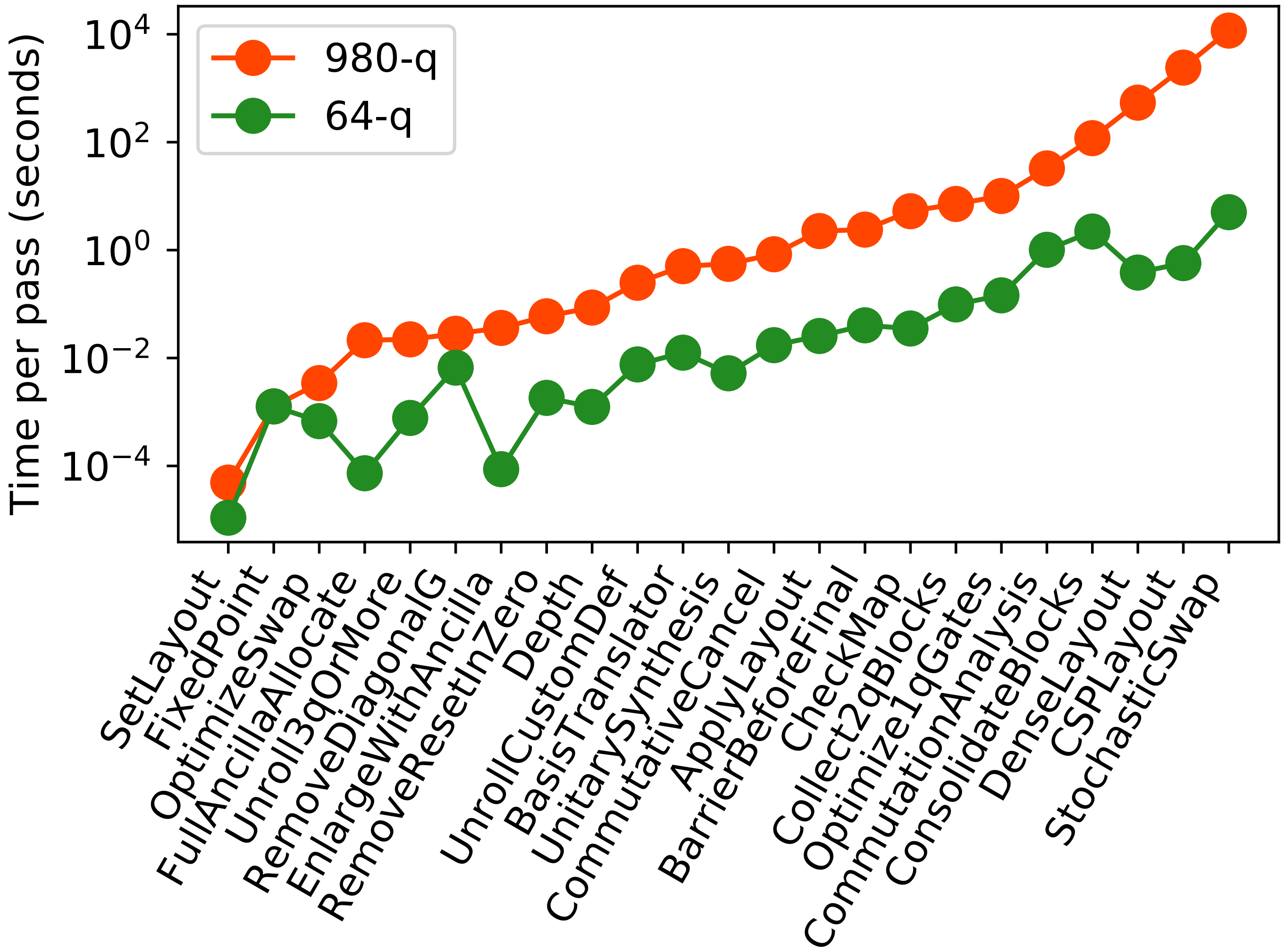
IBM Quantum Cloud: Overview

- Sorted queuing times from a two-year period.
- Green lines correspond to times of 1 minute and 2 hours.





IBM Quantum Cloud: Compilation Times



- Sorted queuing times from a two-year period.
- Green lines correspond to times of 1 minute and 2 hours.
- Median queuing time is around 1 hour.
- Over 25% of the circuits are queued for 2 hours or more.
- ~5% are queued for more than a day.

Why this study?

- Growing scarcity of quantum resources in the cloud, as the demand consistently increases.
- Quantum machines in the cloud are limited and the number of users and “jobs” submitted to are drastically growing.
- Contention trends will continue to worsen until the cost of building large and reliable quantum computers becomes more easily surmountable.
- Similar to classical HPC, efficient management of cloud resources is critical.
- Unlike classical HPC:
  - Quantum machines are significantly impacted by machine fidelity constraints,
  - Quantum circuits are on the lower end of the complexity spectrum, meaning that their execution /fidelity trends are “predictable”.

Details of our study

- 20 IBM Quantum Computers, over a two-year period up to April 2021 in an academic research setting.
- 6000 jobs run on these quantum machines, which encompass over 600,000 quantum circuits.
- Each circuit run for multiple trials on the quantum machines – data gathered over 10 billion machine executions.

Design: Utility Function

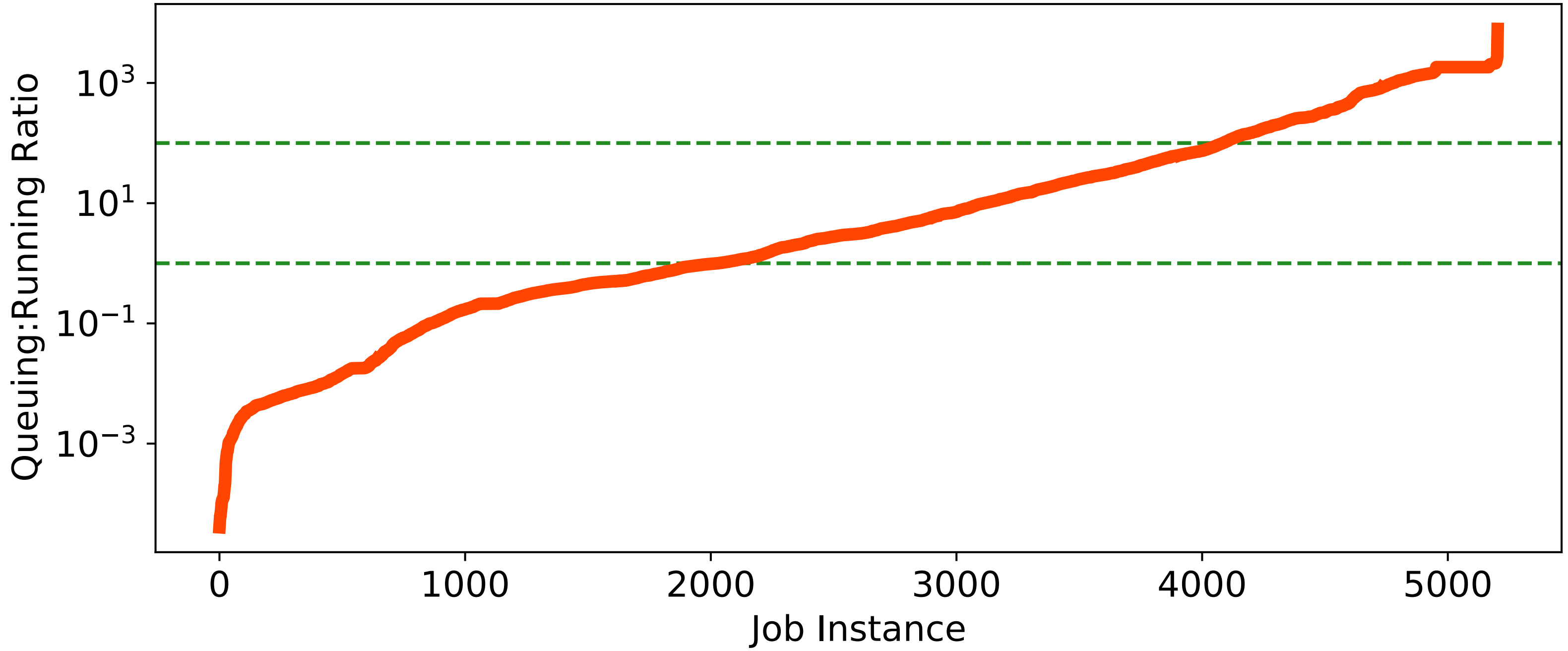
- Maximizing the function should result in a job schedule that provides a good balance between fidelity and queuing time *(at any load)*

- The function should also account for QOS requirements and the impact of calibrations and stale compilations on the utility of the machine.

- Beyond the above (not pursued here), the utility function could account for user priorities, improved machine utilization etc.

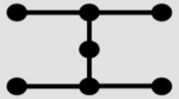
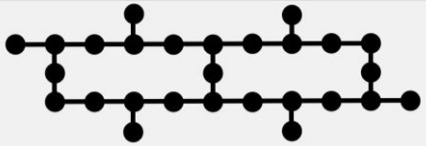
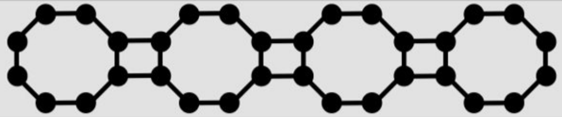
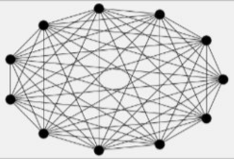
- We use a balanced linear equation of the form $\Sigma(a_i * x_i)$. x_i (features) are between 0 and 1 and a_i (coefficients) can be tuned empirically.

IBM Quantum Cloud: Queuing to Execution Ratios

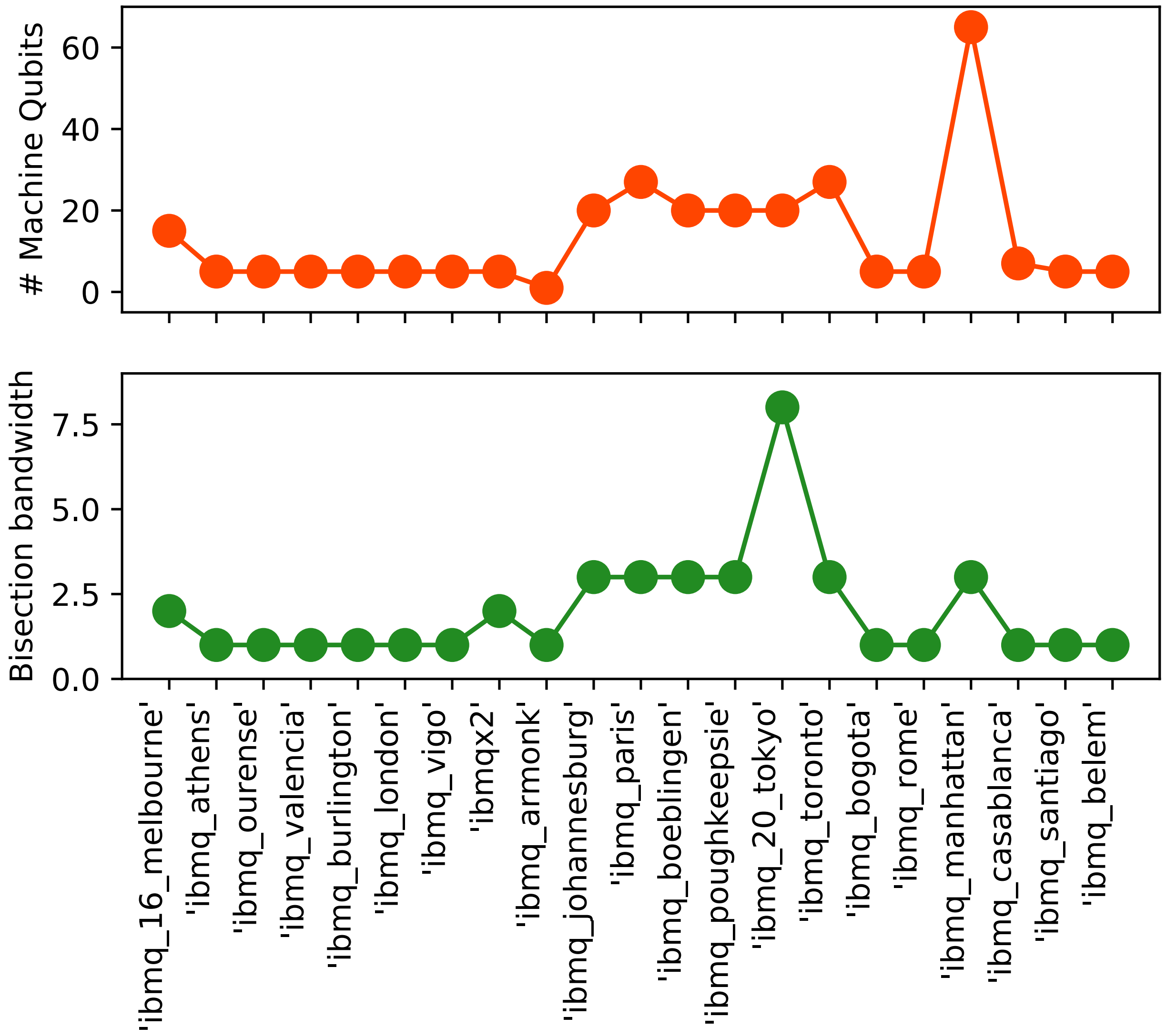


- Sorted queuing times from a two-year period.
- Green lines correspond to times of 1 minute and 2 hours.
- Median queuing time is around 1 hour.
- Over 25% of the circuits are queued for 2 hours or more.
- ~5% are queued for more than a day.

Multiple vendors: More diversity

Machine	Qubits	Coherence Time (µs) (T1, T2)	Gate Times (µs) (1Q, 2Q, Rd)	Gate Errors (%) (1Q, 2Q, Rd)	Topology
IBM-Casablanca	7	91.21, 125.23	0.035, 0.443, 5.9	0.028, 0.83, 2.09	
IBM-Montreal	27	104.14, 86.88	0.035, 0.423, 5.2	0.052, 1.76, 1.96	
Rigetti-Aspen-9	32	31.1, 17.5	0.06, 0.16, 2*	0.6, 4.9, 5.84	
IonQ	11	>1e7, 2e5	10, 210, 100	0.28, 3.04, 0.39	
AQT	4	62, 37	0.03, 0.152, 1.02	0.083, 2.1, 1.25	

IBM Quantum Cloud: Bisection Bandwidth



- CX Error across IBM machines averaged over 2.5 months.
- Machine connectivity constraints result in inserting SWAPs which lead to more CNOTs (and errors).
- Errors vary widely within machines as well as across machines.
- While worse-error qubits can be avoided in larger machines, avoiding qubits with good locality can lead to more SWAPs.
- Again, average error rates not well correlated with machine size.

Managing jobs/resources in the quantum cloud

- Background: Machines are diverse and user demands are growing constantly: limited knowledge on best machine for any usecase + very long queuing times.

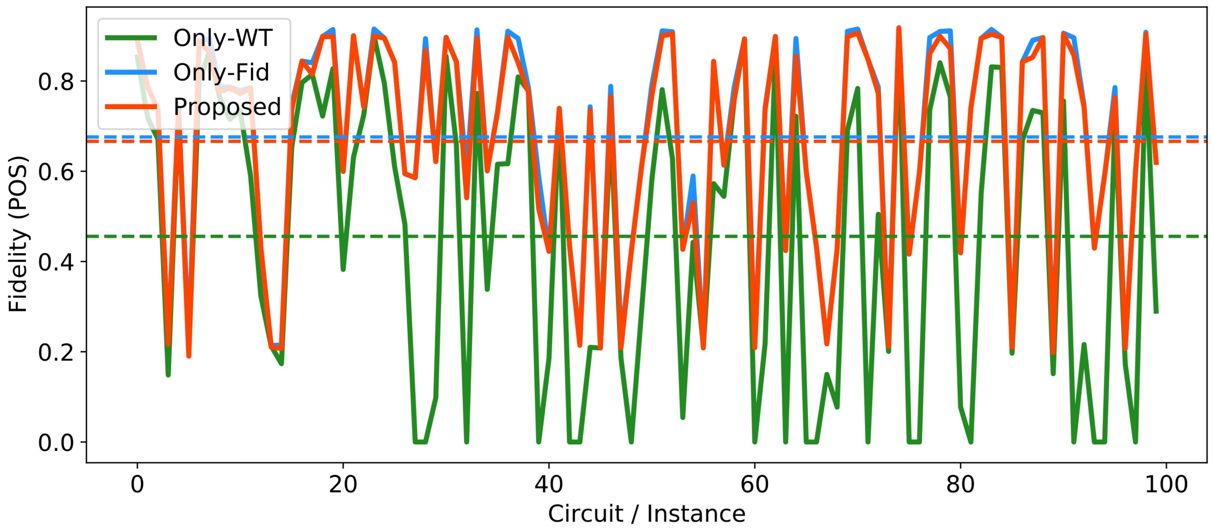
- Goal: An automated job scheduling and resource allocation approach which achieves optimal trade-offs between fidelity, wait times, QOS specifications etc.

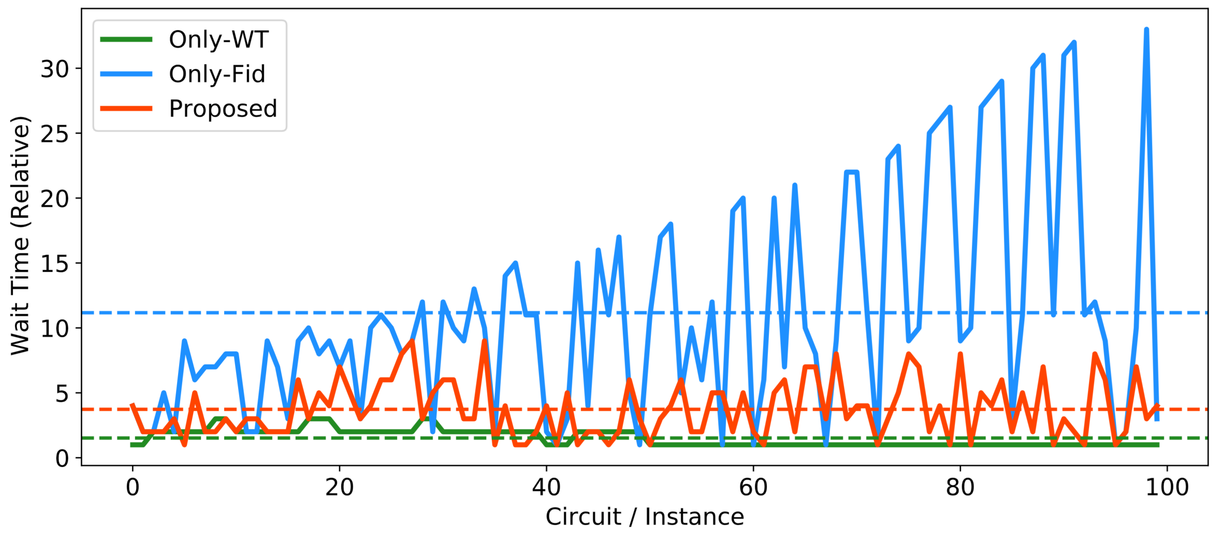
- Proposal:
  - Predicts fidelity trends across machines.
  - Estimates run times and, thereby, wait times.
  - Uses a utility function is inherently able to prioritize fidelity improvements at low load, wait time reduction at high load and balanced otherwise.
  - Optimizes for other constraints such as QOS, machine recalibration, etc.

- Result: Reduce wait times by over 3x and improve fidelity by over 40% on different usecases.

Evaluation: Optimizing for Fidelity and Wait Time
Low load

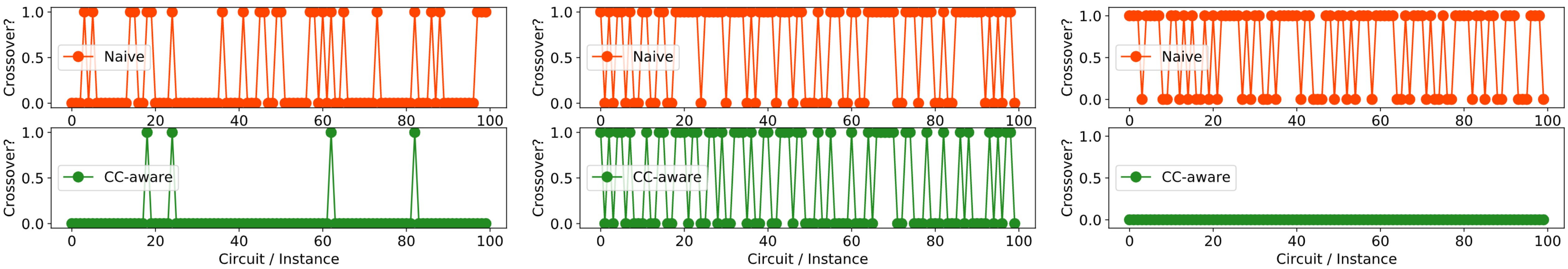
Low load Ideal: Choose highest fidelity machines, since the queuing times are not significant and thus best results are worth the short wait.





Evaluation: Avoiding Calibration Crossovers

- Machine-aware compilation on an old calibration cycle is not optimal for execution on a new calibration cycle.
- CC-aware approaches schedule near-calibration jobs on machines with low queuing time.
- At high load this is insufficient, and crossovers are alleviated through a staggered calibration approach.

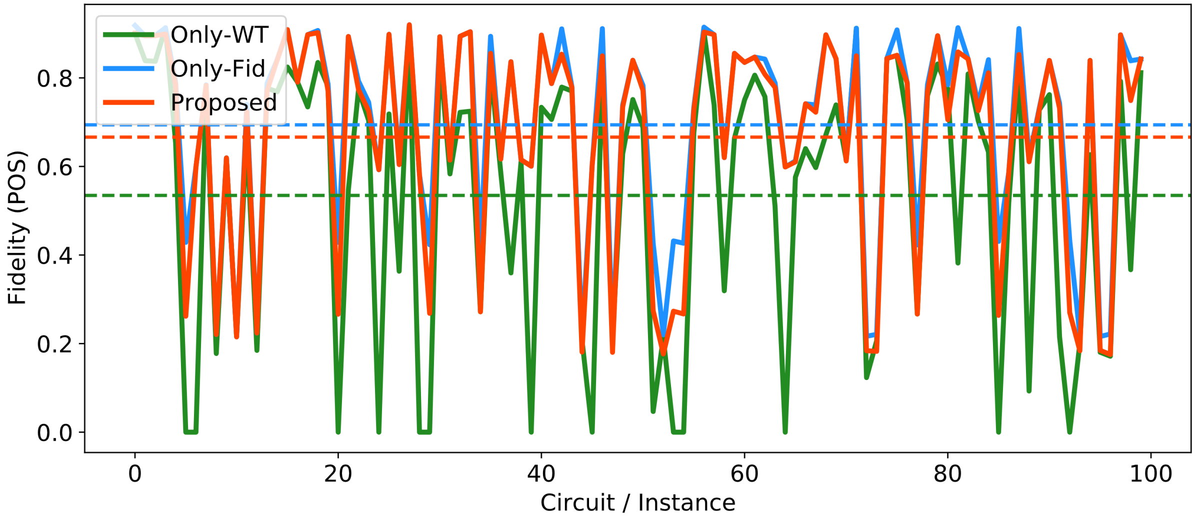


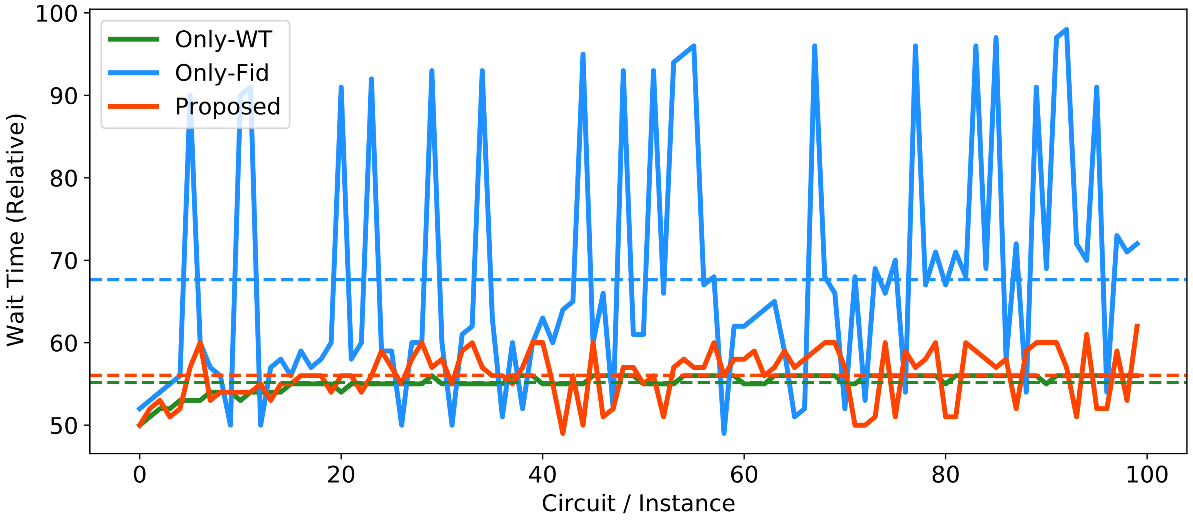
(a) Low Load **(b) High Load** **(c) Staggered Calibration (High Load)**

Evaluation: Optimizing for Fidelity and Wait Time

High load

High load Ideal: Accept some loss in fidelity to achieve reasonable queuing times, though we would still want thefidelity to be substantial enough for realistic benefits.





Experimental Setup

- Compilation: IBM Qiskit, Noise aware compilation

- Cloud Simulation: 26 device models of IBM quantum machines + different load distributions of low, high and random queuing jobs / times across the machines.
  - Low load: < 10% of maximum queuing on each machine,
  - High load: 50-100%,
  - Random Load: 1-100%

- Benchmarks: Toffoli, Hidden Subgroup problem, Bernstein-Vazirani, Linear Solver, QAOA, VQE, Repetition Code Encoder, RC Adder.

- Metrics: Probability of Success, Queuing Time

- Comparisons: a) Only Wait Times, b) Only Fidelity

Benchmarks

- Toffoli:A 3-input gate which performs logical AND be-tween two controls bits and writes onto the target bit.
- Hidden Subgroup Problem:Captures problems like factor-ing, discrete logarithm, graph isomorphism, and the shortestvector problem. It is implemented for 4 qubits.
- Bernstein-Vazirani:BV guarantees the return of the bitwiseproduct of some input with a hidden string [13]. BV isimplemented using 5 qubits.
- Linear Solver:Solver for a linear equation utilizing 3 qubits.
- Quantum Approximate Optimization Algorithm:QAOA [20]is implemented atop a parameterized circuit called an ansatzand we use one instance of a hardware efficient QAOA ansatzas the benchmark. We use QAOA ansatz for 4 qubits.
- Variational Quantum Eigensolver:The goal of this algo-rithm [30] is to variationally find the lowest eigenvalue of agiven problem matrix. We implement VQE on a hardware-efficient SU2 ansatz [6] and use one instance as the bench-mark. We construct the ansatz for 4 qubits (4 reps / fullentanglement) and 6 qubits (3 / SCA).
- Quantum Repetition Code Encoder:A repetition code en-coder which introduces redundancy to the encoding that canbe exploited for error detection [32] (5 qubits).
- Ripple Carry Adder:We implemented a linear-depth, 2 bitripple-carry adder quantum circuit that uses 6 qubits based onthe structure described in [15].