Qonductor

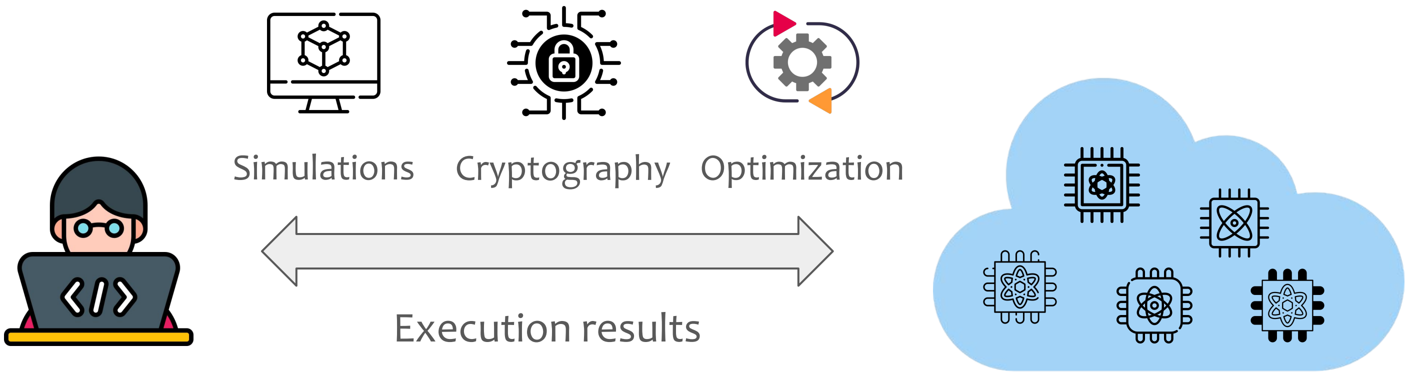
A Cloud Orchestrator for Quantum Computing

Emmanouil (Manos) Giortamis, Francisco Romão,
Nathaniel Tornow, Dmitry Lugovoy, Pramod Bhatotia

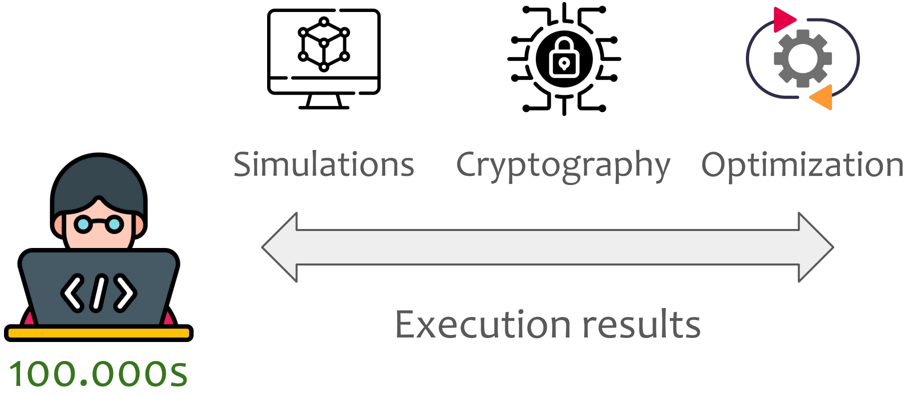
Technical University of Munich

ACM/IEEE SC'25, St. Louis, USA

Quantum cloud computing paradigm



Quantum cloud computing paradigm

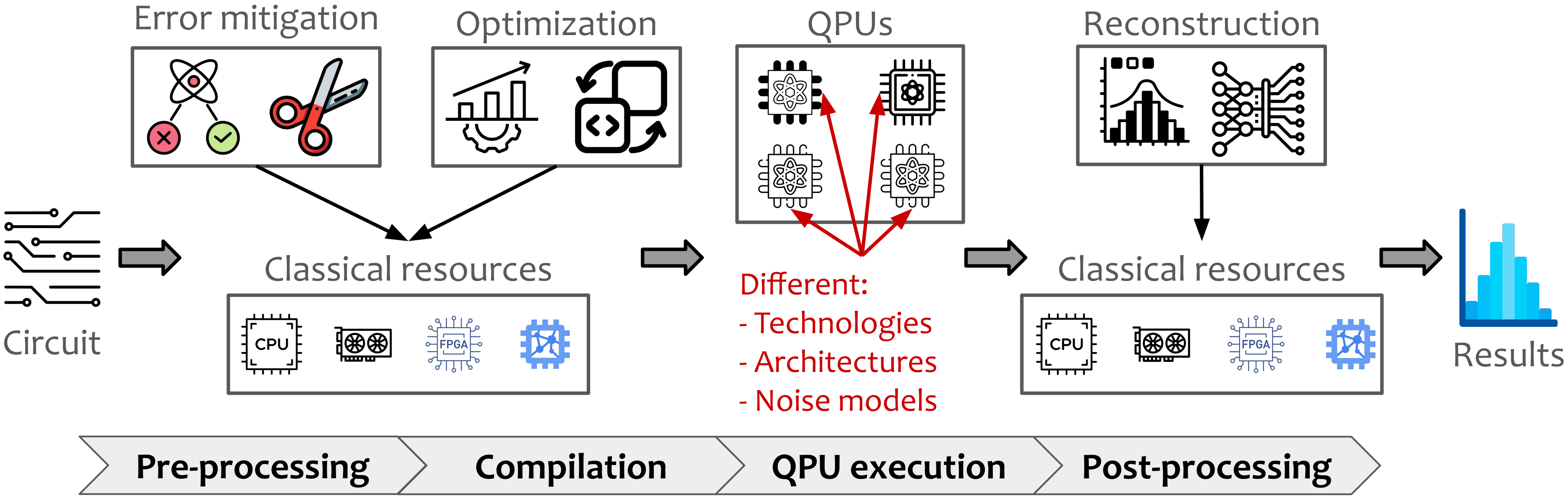


aws
Amazon Braket
Azure Quantum
QUANTINUUM
IONQ
IBM Q™
IQM Resonance

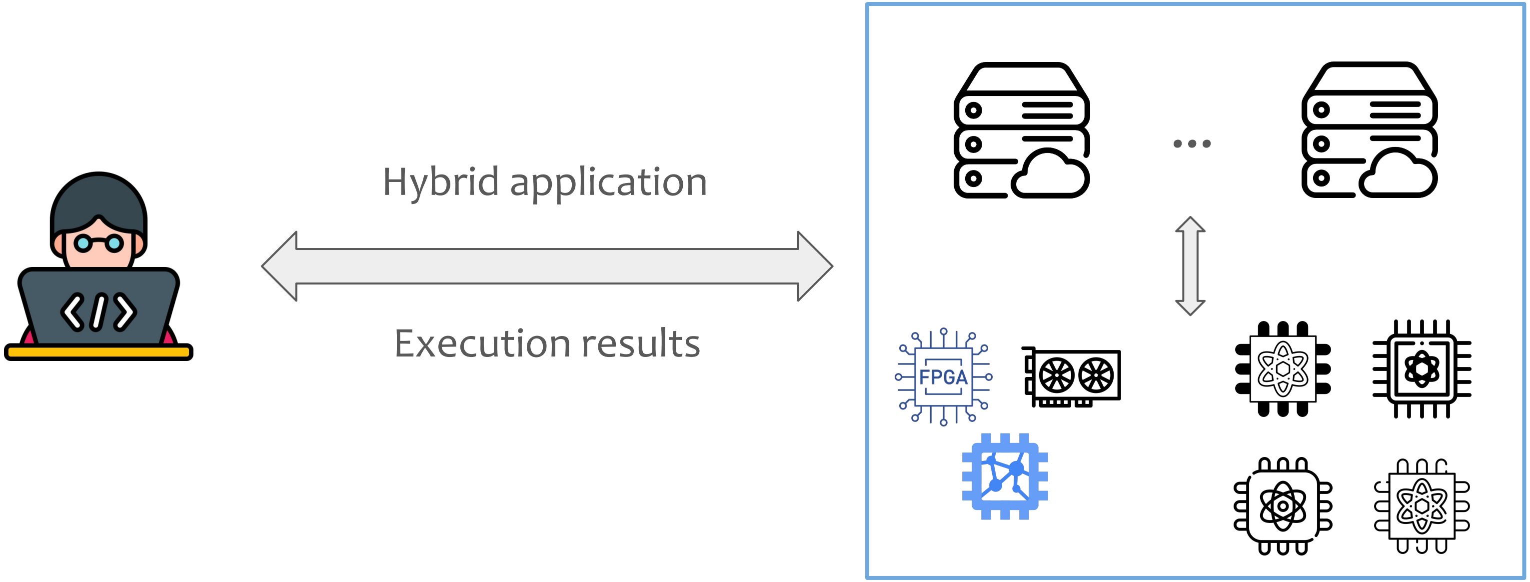
< 100 QPUs online

Obsolete model: Quantum applications require classical code & resources

Quantum applications are hybrid, really



Quantum computers as HPC accelerators



HPCQC: Tight integration of quantum computers in the HPC clusters

Our focus: Hybrid applications on HPCQC clusters

- Hybrid quantum-classical applications that require hybrid resources
- All types of resources are hosted on the cloud
- Quantum resources are scarce and heterogeneous

This programming & execution setting presents unique challenges

The HPCQC cloud faces three challenges

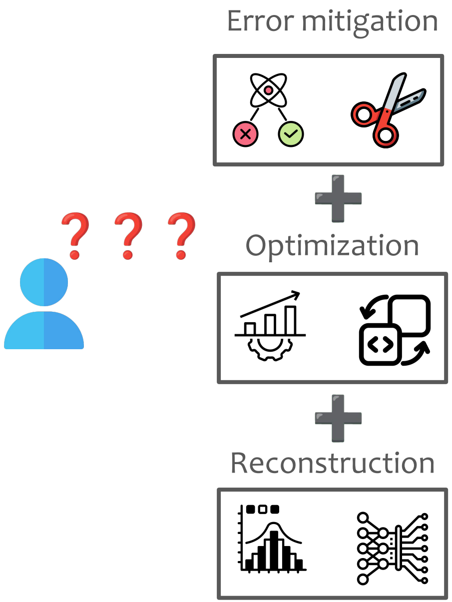
#1 Hybrid programming model

#2 Hybrid resource estimation

#3 Quantum cloud tradeoffs

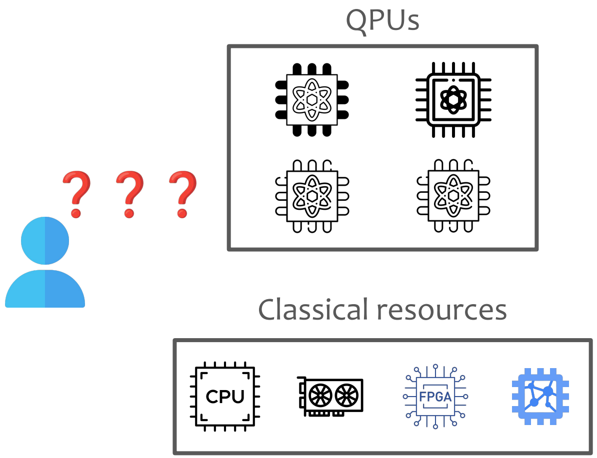
Challenge #1: Hybrid programming model

- Ad-hoc stitching of quantum & classical tasks
- No standardization in creating workflows
- Manual compute resource allocation



Challenge #1: Hybrid programming model

- Ad-hoc stitching of quantum & classical tasks
- No standardization in creating workflows
- Manual resource allocation

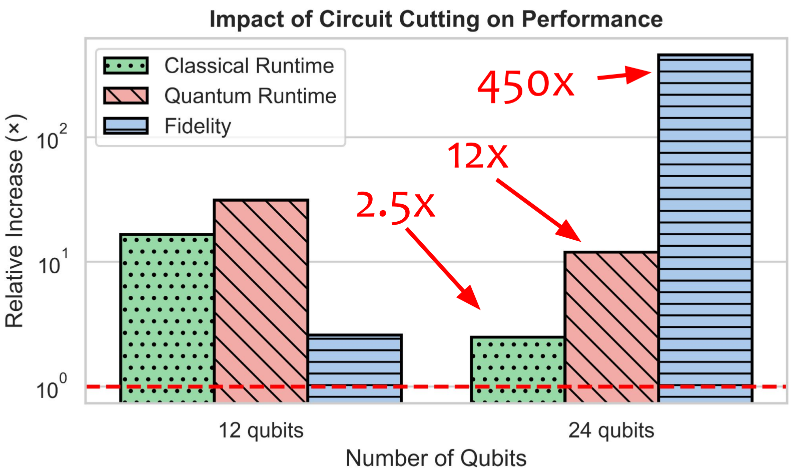


Problem: Ad-hoc hybrid workflow development & execution

Key idea: Hardware-agnostic APIs & hybrid workflow programming tools

Challenge #2: Hybrid resource estimation

- Error mitigation increases fidelity
- Typically, at a hybrid runtime cost
- The impact is (often) hard to compute

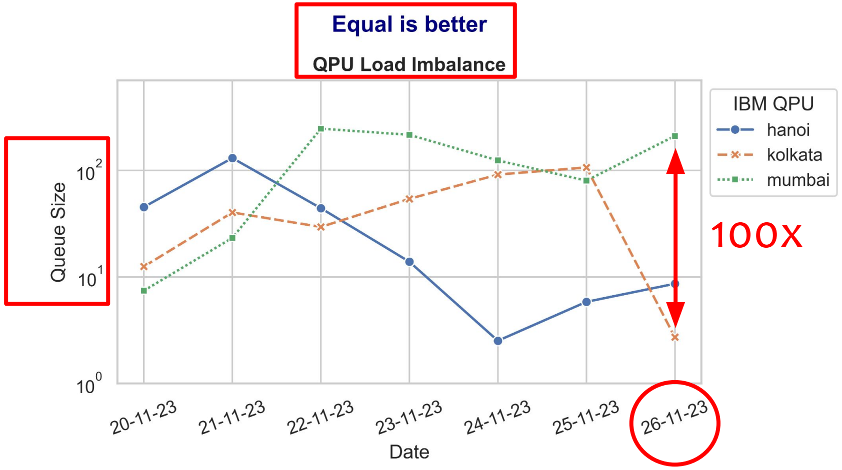


Problem: Classical and quantum runtime and fidelity tradeoff

Key idea: Hybrid resource and performance estimation

Challenge #3: Quantum cloud tradeoffs

- Manual QPU selection access model
- Users want maximum fidelity
- High-fidelity QPUs become hotspots



Problem: Fundamental tradeoff between fidelity and resource efficiency

Key idea: Scheduling that optimizes both fidelity and resource efficiency

The need for a hybrid cloud orchestrator

HPCQC application development and execution are hindered by programming complexity, hybrid resource management, and inherent quantum cloud tradeoffs

How to design an orchestrator for the hybrid quantum-classical cloud that is **programmable**, **performant**, and **resource-efficient**?

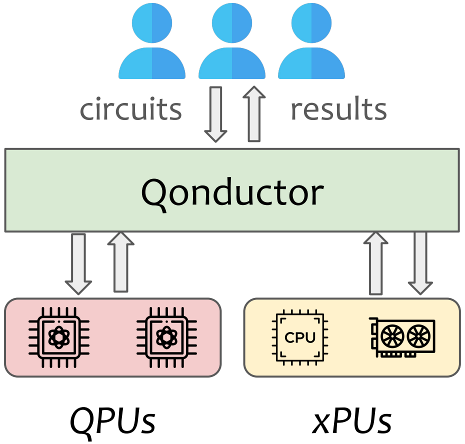
Our proposal: Qonductor

Qonductor: A Cloud Orchestrator For Quantum Computing

An orchestrator designed for hybrid applications running on hybrid resources

Core contributions:

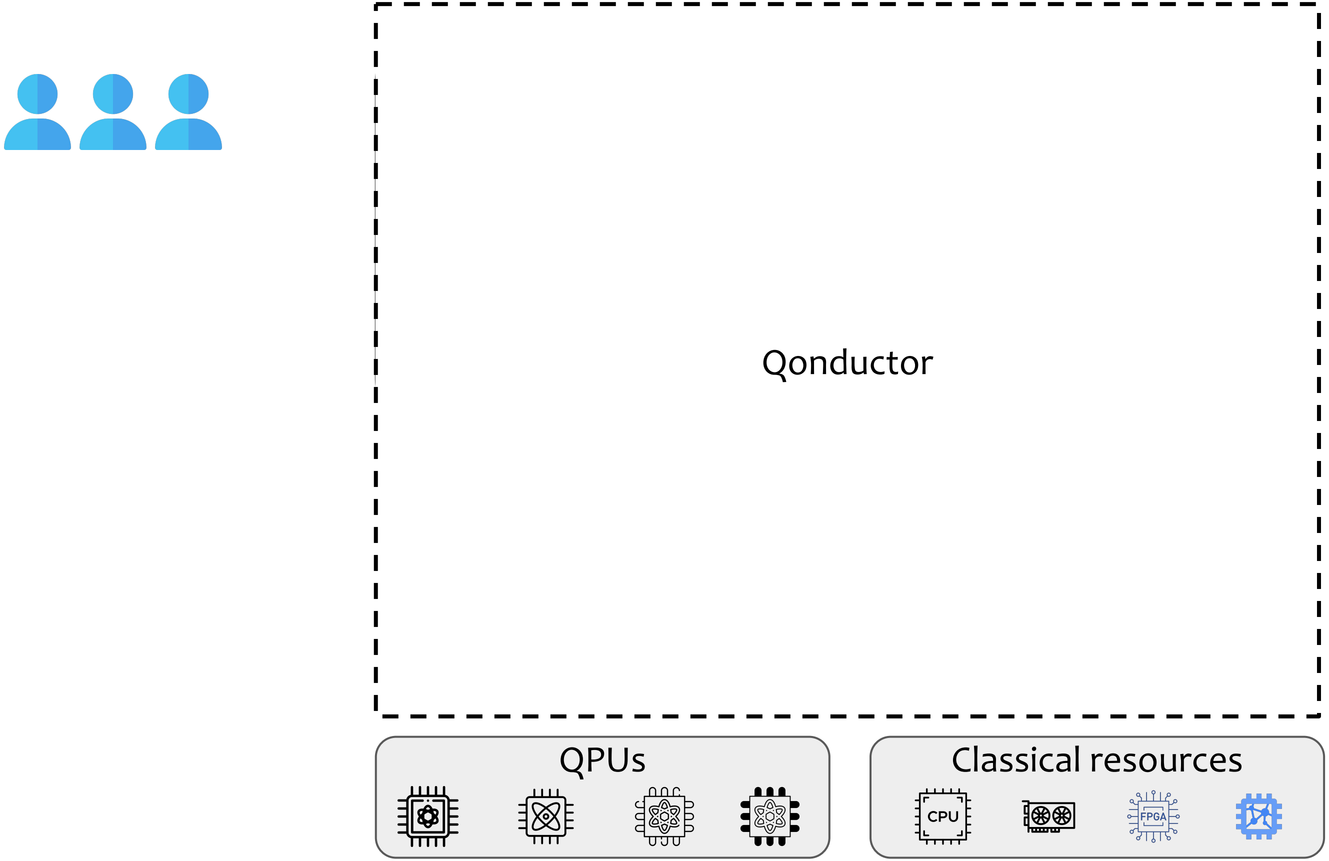
- Hardware-agnostic APIs and hybrid workflow tools
- Hybrid resource estimation for resource allocation
- Hybrid scheduling that manages cloud tradeoffs



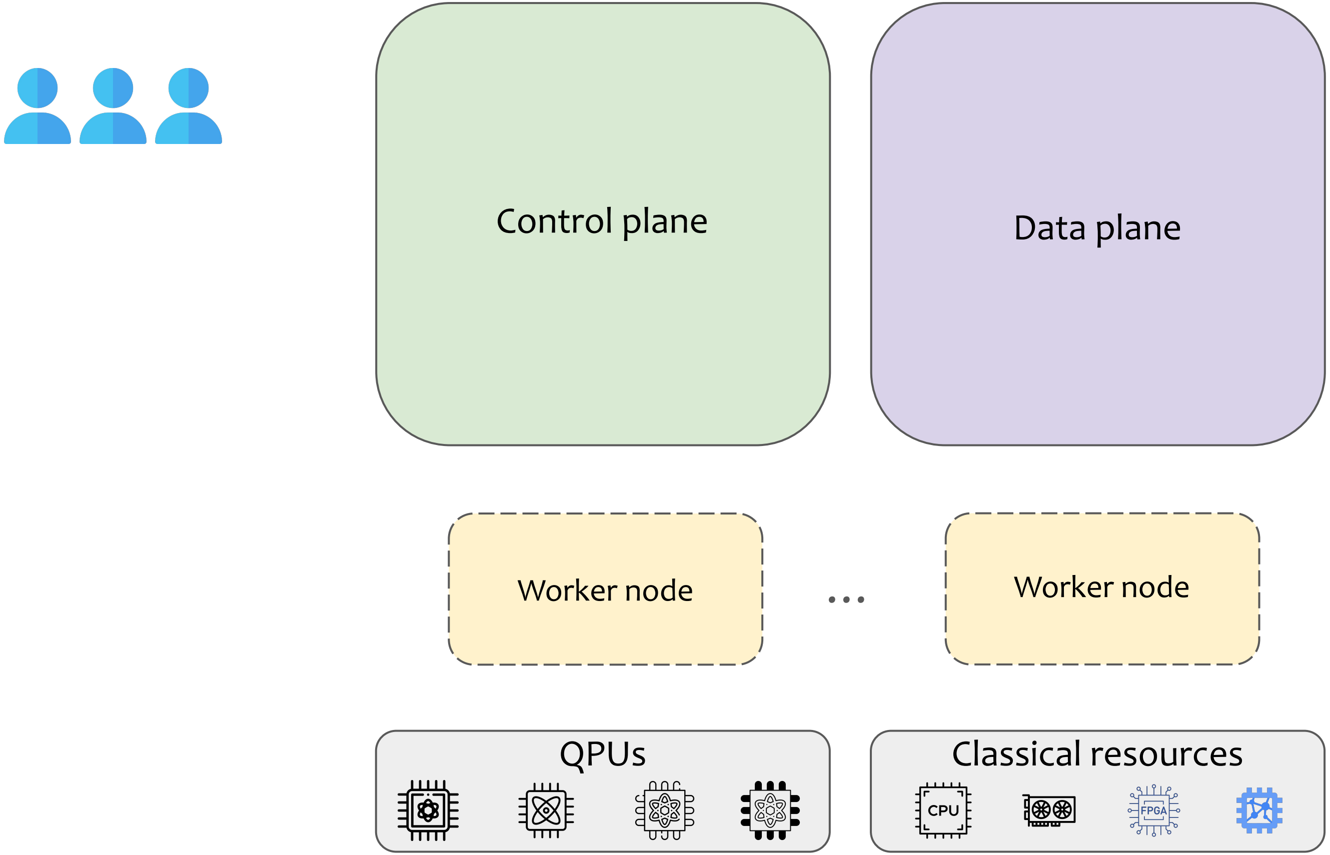
Outline

- ~~Introduction & motivation~~
- System design
  - System overview
  - System Components
- Evaluation

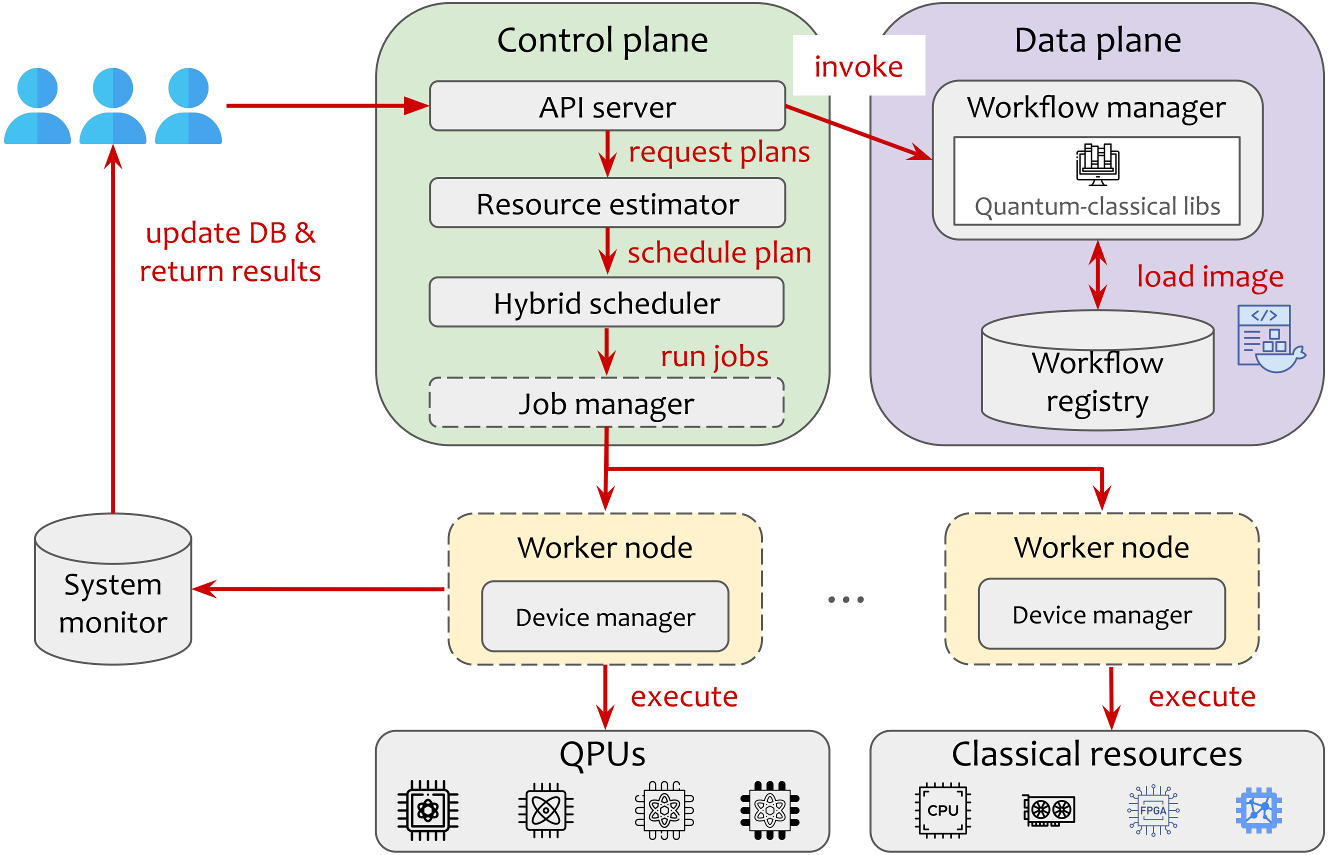
High-level system overview



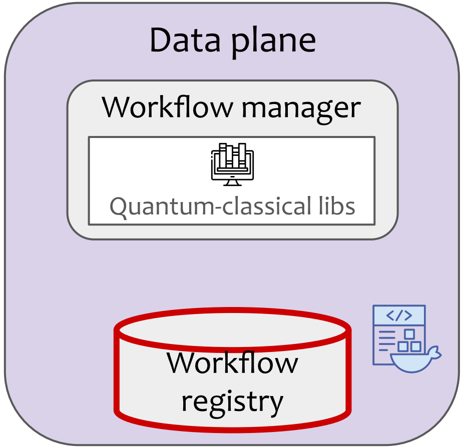
High-level system overview



System overview and workflow



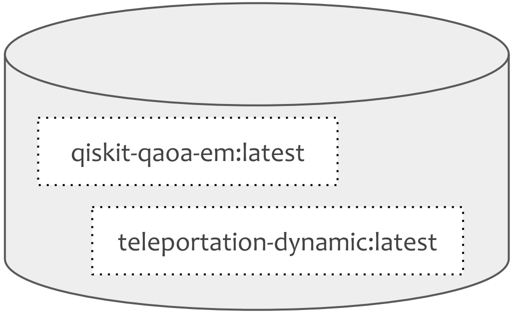
Qonductor data plane



Qonductor workflow registry

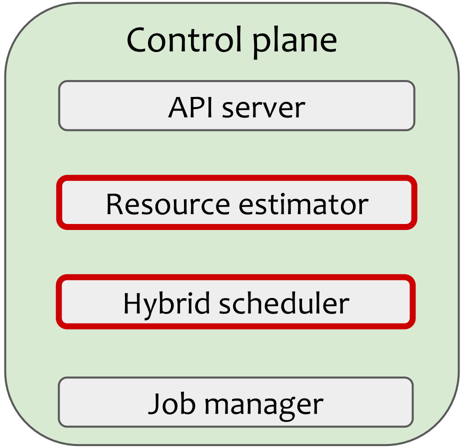
- Qonductor provides common libraries
  - Quantum: Algorithms, error mitigation, etc.
  - Classical: Simulation, post-processing, etc.
- Users can create, store, and reuse images
  - Example: Max-cut on bipartite graph QAOA
- Qonductor provides a workflow registry
  - Example: QFT w/ 50 qubits and circuit knitting

Quantum	Classical
VQE, QAOA	MPS simulator
QFT, QPE	REM
DD, ZNE	Circuit knitting

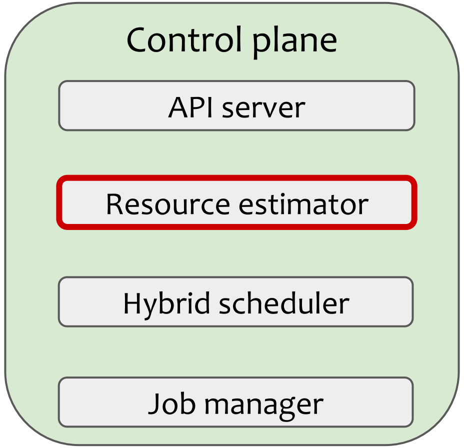


Libraries and workflow registry enable faster hybrid workflow generation

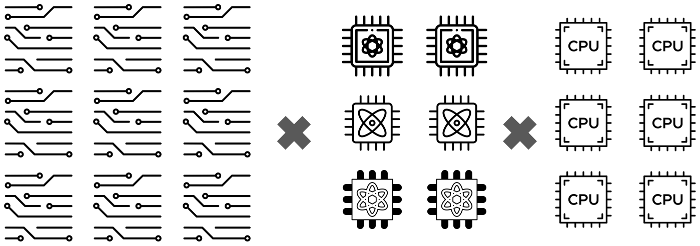
Qonductor control plane



Qonductor control plane



Qonductor resource estimator challenges



Quantum circuits x QPUs x classical resources

Challenge #1: Computational complexity and vast search space

Qonductor resource estimator challenges

Probabilistic Error Cancellation $\langle O \rangle_{mitigated} = \gamma \sum_{k=1}^{M} p'_k \cdot \text{sign}(\gamma_k) \cdot \langle O \rangle_k$

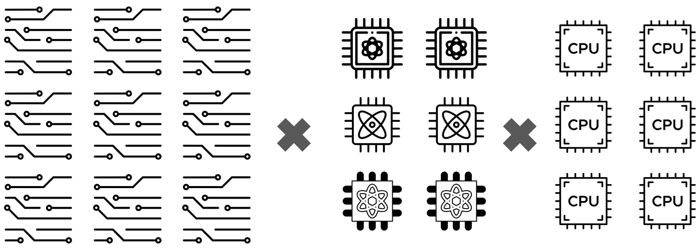
Hard to compute!

Zero-Noise Extrapolation $\langle O \rangle_{mitigated} = \frac{\lambda_2}{\lambda_2 - 1} \langle O \rangle(1) - \frac{1}{\lambda_2 - 1} \langle O \rangle(\lambda_2)$

Formulas for estimating the error-mitigated observables

Challenge #2: Fidelity is mathematically hard to model

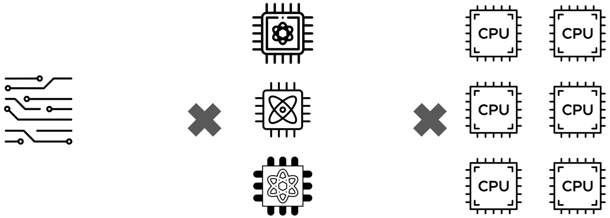
Resource estimator approaches



- For techniques that generate 100s of circuits, use the worst-case one
- For QPUs, use the models (e.g. IBM Falcon) as templates for compilation

Approach #1: Limit search space using templates

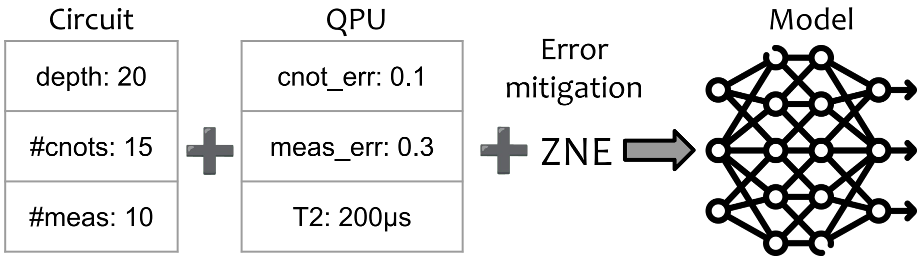
Resource estimator approaches



- For techniques that generate 100s of circuits, use the worst-case one
- For QPUs, use the models (e.g. IBM Falcon) as templates for compilation

Approach #1: Limit search space using templates

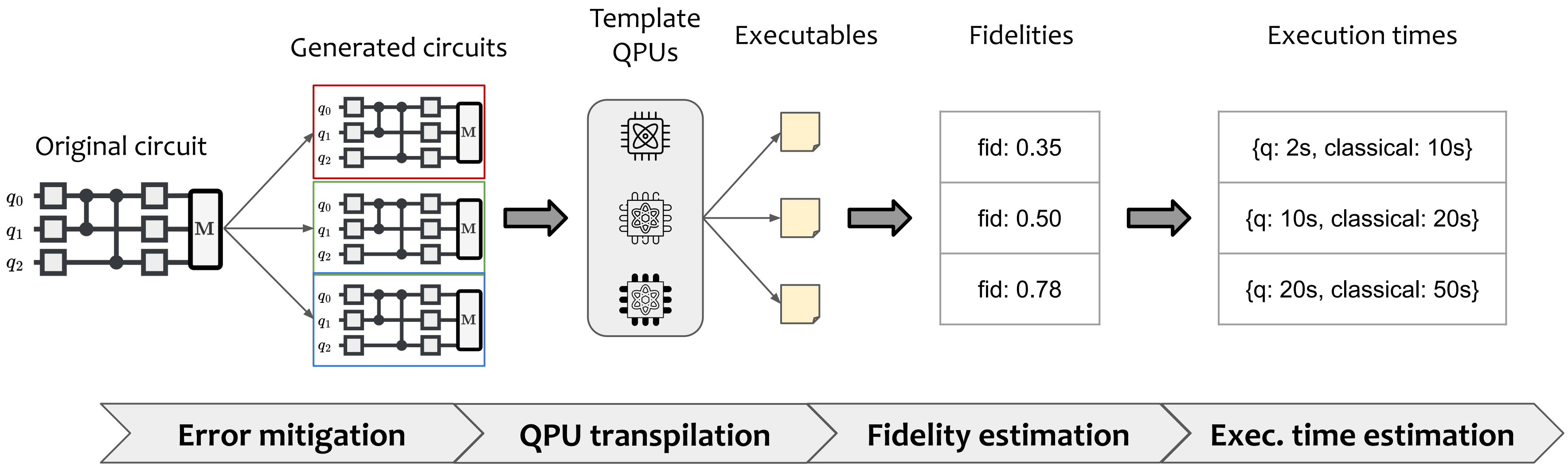
Resource estimator approaches



- Train the model with 1000s of runs, w/ and w/o error mitigation, extract metrics
- Use circuit & QPU properties and the error mitigation technique(s) as features

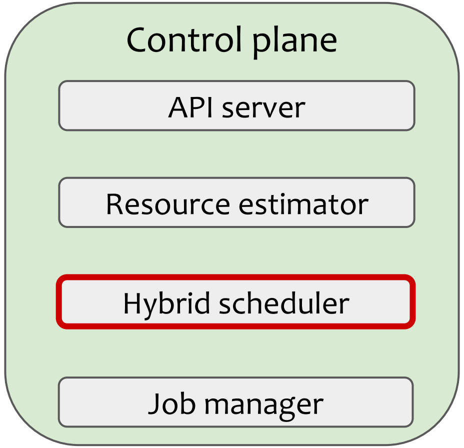
Approach #2: Use a machine learning model to predict performance

Resource estimator workflow



The output fidelity & execution times are used by the scheduler!

Qonductor control plane



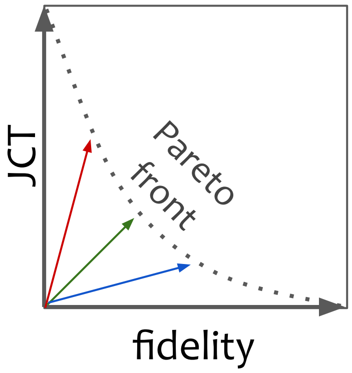
Qonductor quantum scheduler challenges

Schedule	Fidelity	JCT
schedule_0	0.54	55s
…		
schedule_*n*	0.38	10s

Systematic exploration of schedule possibilities is NP-hard

Challenge #1: Large schedule exploration space to explore

Qonductor quantum scheduler challenges



Optimization problems with conflicting objectives are Pareto-optimal

Challenge #2: Schedules can be equivalent and not better than others

Quantum scheduler solutions

Minimize waiting times

Minimize error

$$\min \quad f_1(x) = \frac{1}{N}\sum_{i=1}^{N}\left(w_{x_i} + \sum_{k=1}^{N} t_{kx_k}\,[x_i = x_k]\right), \qquad f_2(x) = \frac{1}{N}\sum_{i=1}^{N}(1 - f_{ix_i})$$

$$\text{s.t.} \quad q_i - s_{x_i} \le 0, \quad 1 \le x_i \le Q, \quad \forall i = 1,..,N \qquad (1)$$

Constraints

- Formulate optimization objectives, use NSGA-II to generate solutions
- NSGA-II explores the solution space in parallel and is robust against local optima

Approach #1: Multi-objective optimization with genetic algorithm

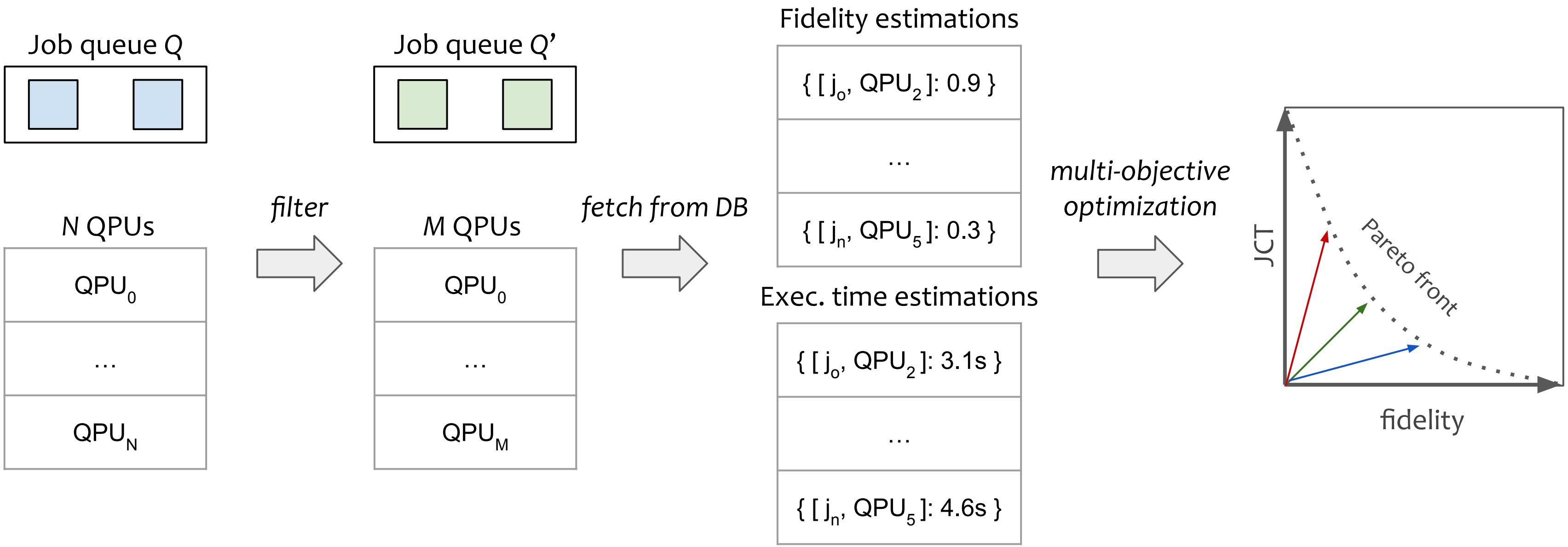
Quantum scheduler solutions

$$w_i(x) = \frac{(f_i^{max} - f_i(x))/(f_i^{max} - f_i^{min})}{\sum_{m=1}^{M} (f_m^{max} - f_m(x))/(f_m^{max} - f_m^{min})}$$

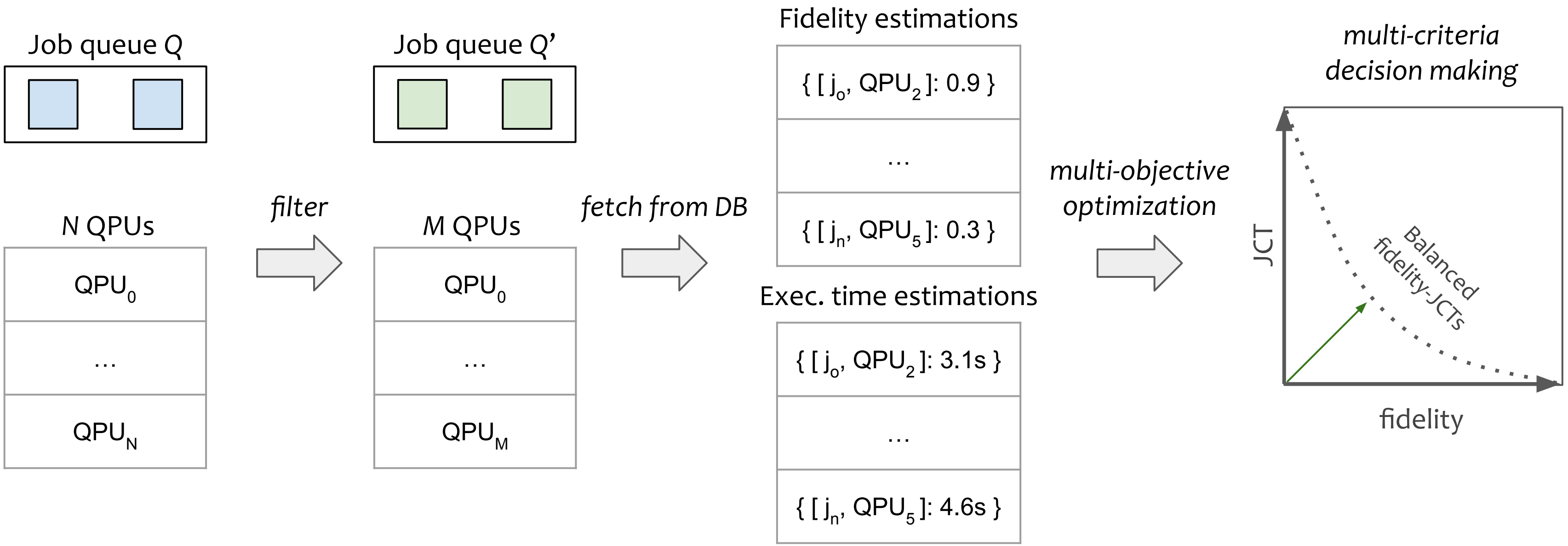
- Users set the preference for fidelity, JCTs, or balanced
- Our pseudo-weights measure the relative importance within the Pareto front
- Select the solution w/ pseudo-weights closest to user's preferences

Approach #2: Multiple-criteria decision making with pseudo-weights

Quantum scheduler workflow



Quantum scheduler workflow



The scheduler balances the tradeoff between fidelity and JCTs

Outline

- ~~Introduction & motivation~~
- ~~System design~~
  - ~~System overview~~
  - ~~System Components~~
- Evaluation

Evaluation

- Implemented based on Kubernetes and the Qiskit quantum SDK

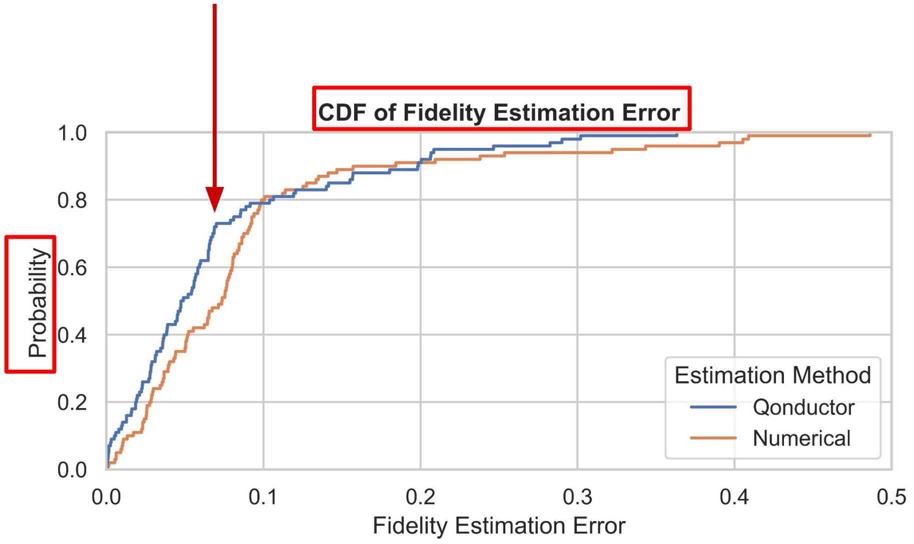
- **Main research questions (RQs):**
  - **RQ1:** What is the resource estimator's accuracy in predicting performance?
  - **RQ2:** What is the quantum scheduler's performance w.r.t. fidelity and JCTs?
  - **RQ3:** What is the quantum scheduler's load-balancing performance?

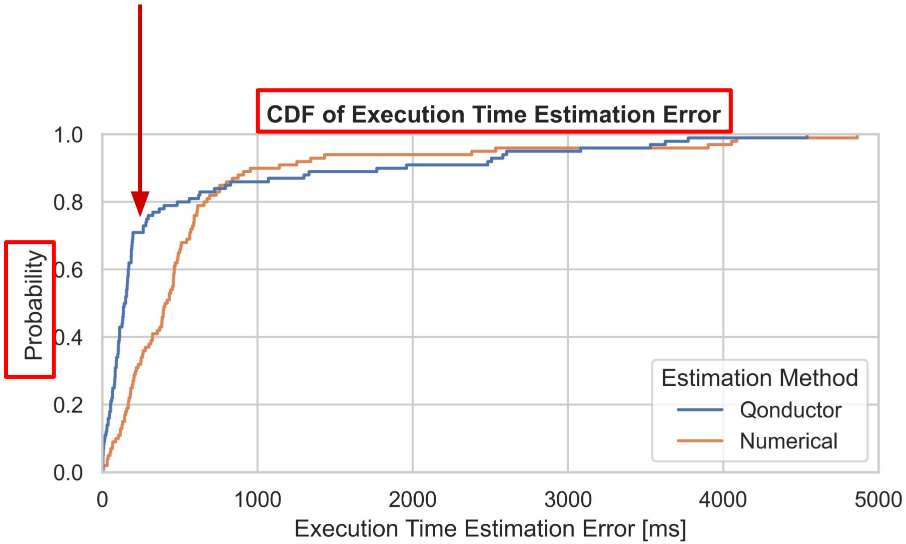
Many more results in the paper!

Evaluation methodology

- **Setup:** IBM 7-qubit and 27-qubit Falcon QPUs, 64-core AMD EPYC 7713P
- **Benchmarks:** SOTA quantum applications: QAOA, VQE, QFT, GHZ & W states, etc.
- **Dataset:** We collect over **70.000** circuits and **~7000** runs on the IBM cloud
  - We also monitor IBM job queues to simulate **real arrival rates**
- **Cloud emulator:** We simulate real cloud workloads based on our dataset
- **Baselines**
  - First-come-first serve (FCFS), numerical performance estimation methods
- **Metrics**
  - **Fidelity** (Higher is better)
  - **JCT** (Lower is better)
  - **Utilization** (Higher is better)

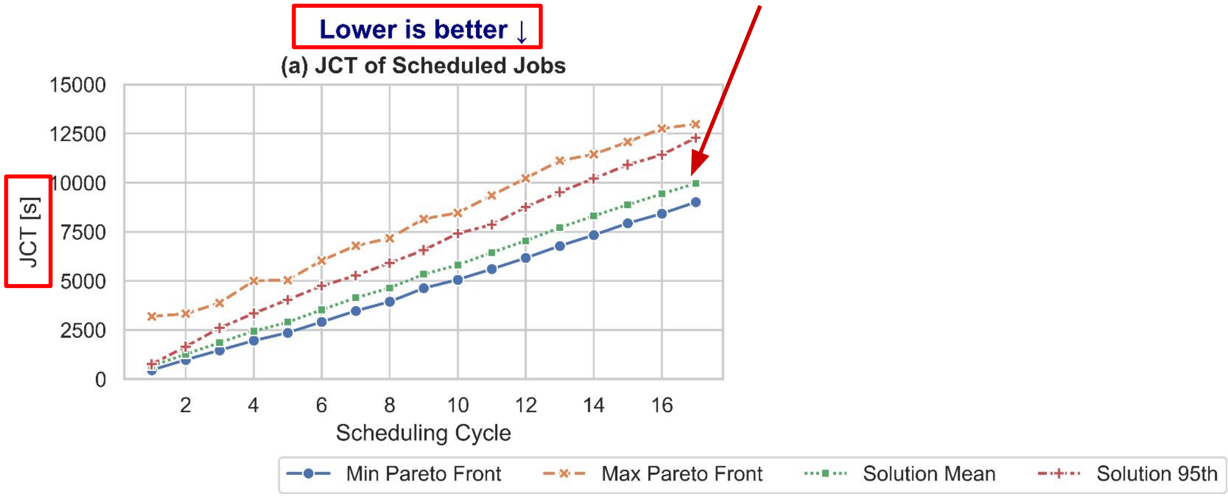
RQ #1: Resource estimator accuracy



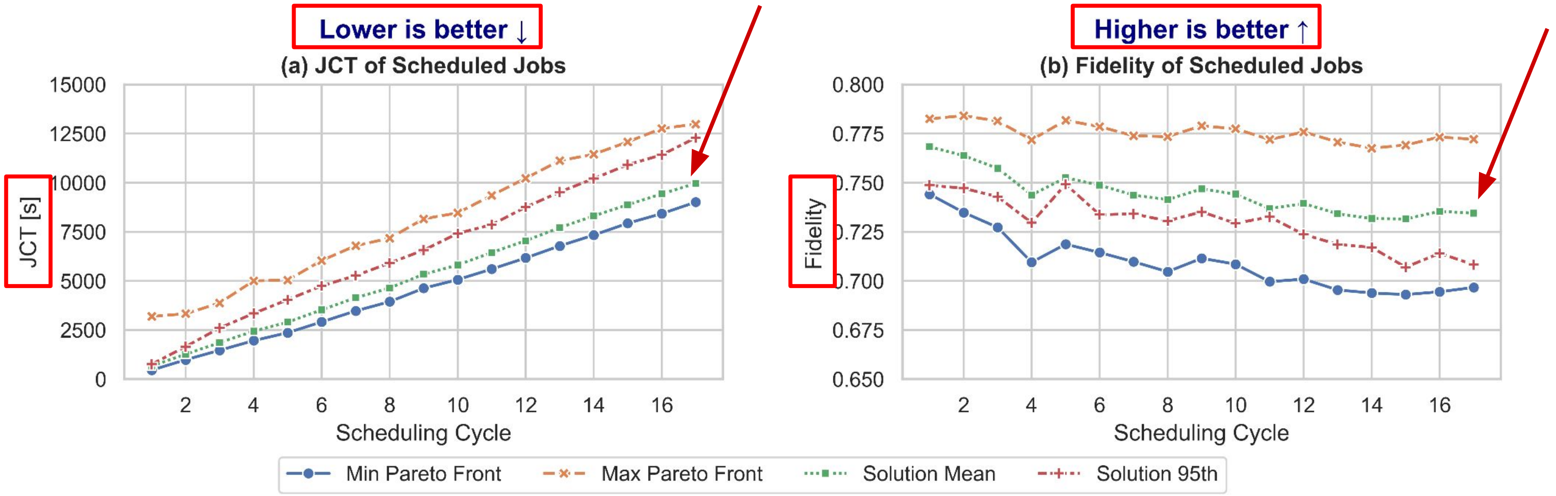


Qonductor achieves 11-50% higher accuracy compared to ad-hoc methods

RQ #2: Quantum scheduler performance

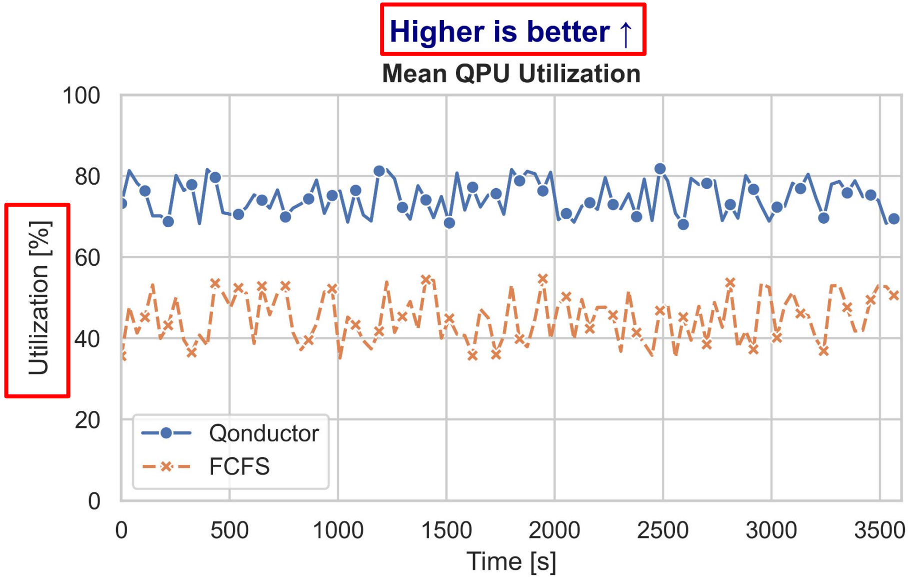


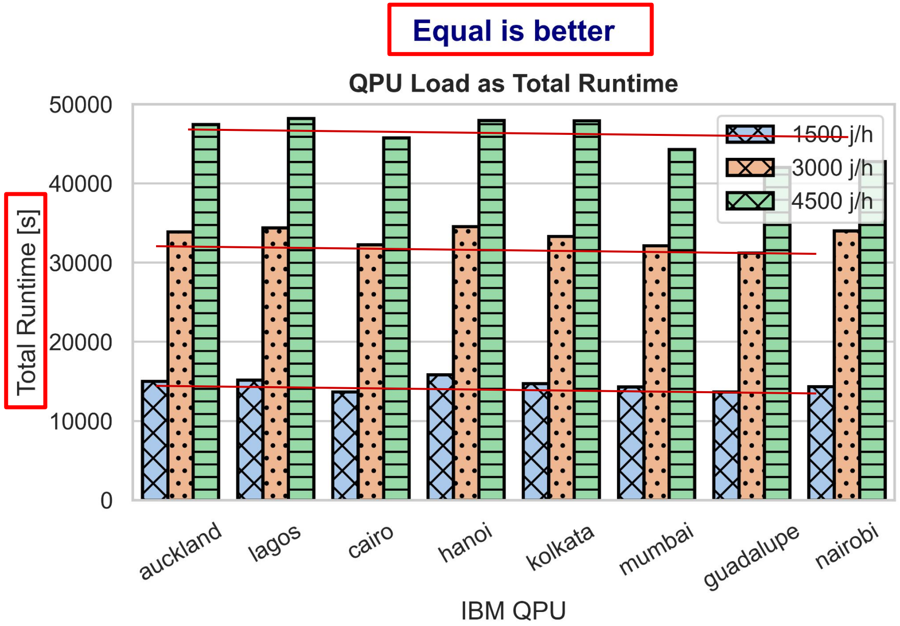
RQ #2: Quantum scheduler performance



Qonductor achieves ~48% lower JCTs for ~3% fidelity loss

RQ #3: Quantum scheduler load balancing





Qonductor achieves 66% higher QPU utilization with <16% load difference

Conclusion

- Hybrid application development & execution on the cloud face unique challenges:
  - Programming and execution complexity
  - Hybrid techniques and resources that affect classical and quantum performance
  - QPU load imbalance with high JCTs
- Qonductor: A Cloud Orchestrator For Quantum Computing
  - Hardware-agnostic APIs and hybrid workflow management
  - Hybrid resource estimation that predicts classical and quantum performances
  - Pareto-optimal multi-objective scheduling for QPU load balance and lower JCTs

emmanouil.giortamis@tum.de

github.com/manosgior/Qonductor-SC25