qTPU

Hybrid Tensor Networks for Quantum-Classical Acceleration

Nathaniel Tornow, Emmanouil Giortamis, Dennis Sprokholt,
Christian B. Mendl, Pramod Bhatotia

Systems Research Group, TU Munich



OSDI'26, Seattle, USA

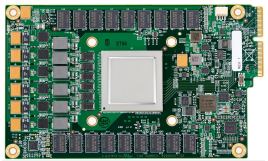
The age of accelerators

Accelerators power the growing need for computational resources

Classical accelerators

GPUs, TPUs, FPGAs, ASICs, …



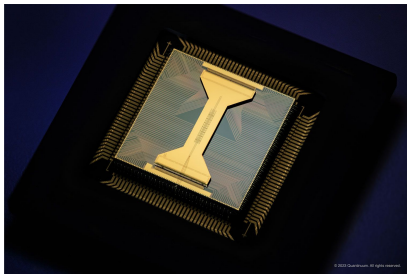






Quantum accelerators (QPUs)





Classical- vs. quantum-accelerators



Classical accelerators

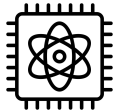


high throughput



highly scalable

Quantum accelerators





low throughput



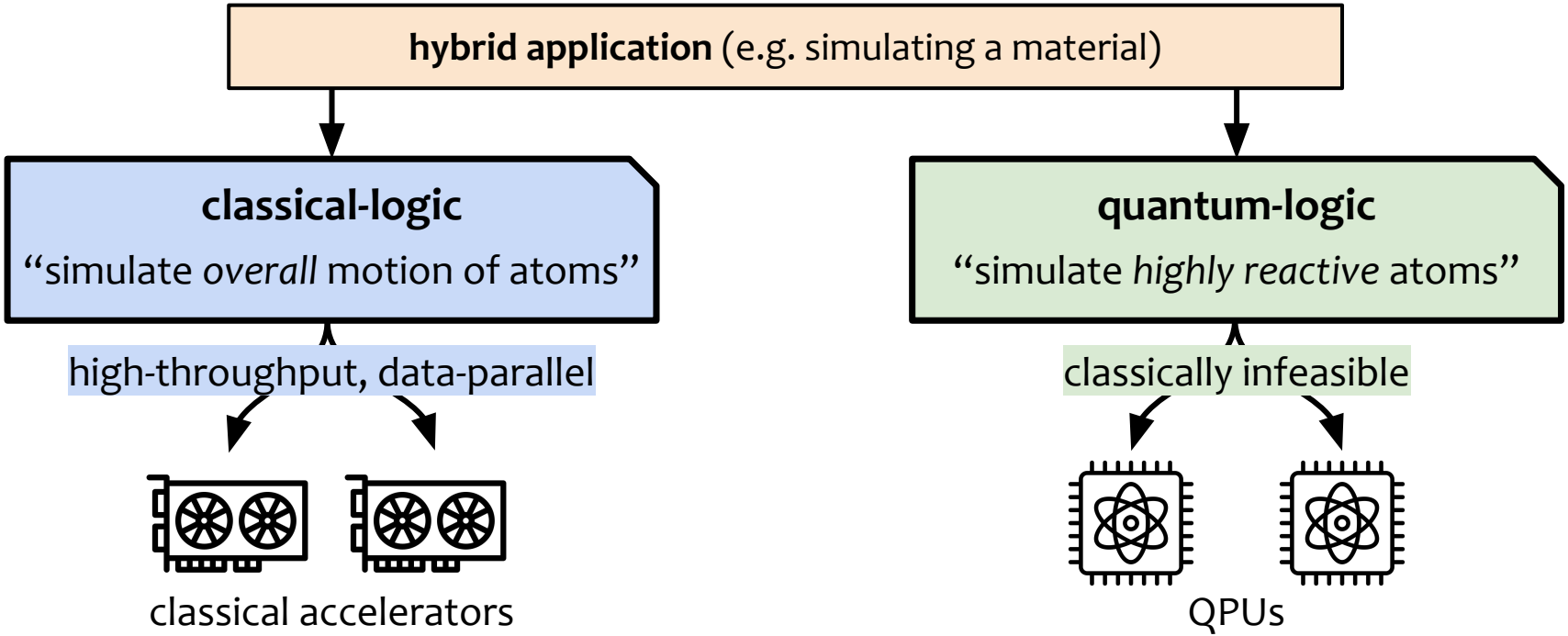
fragile, erroneous & small



express high correlations

How can we reap the benefits of both paradigms?

The need for hybrid compute



We require tight integration to leverage QPUs to their full potential

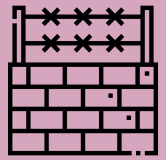
Limitations in designing hybrid applications

Fragmentation



Programming in two distinct paradigms

Fixed boundaries



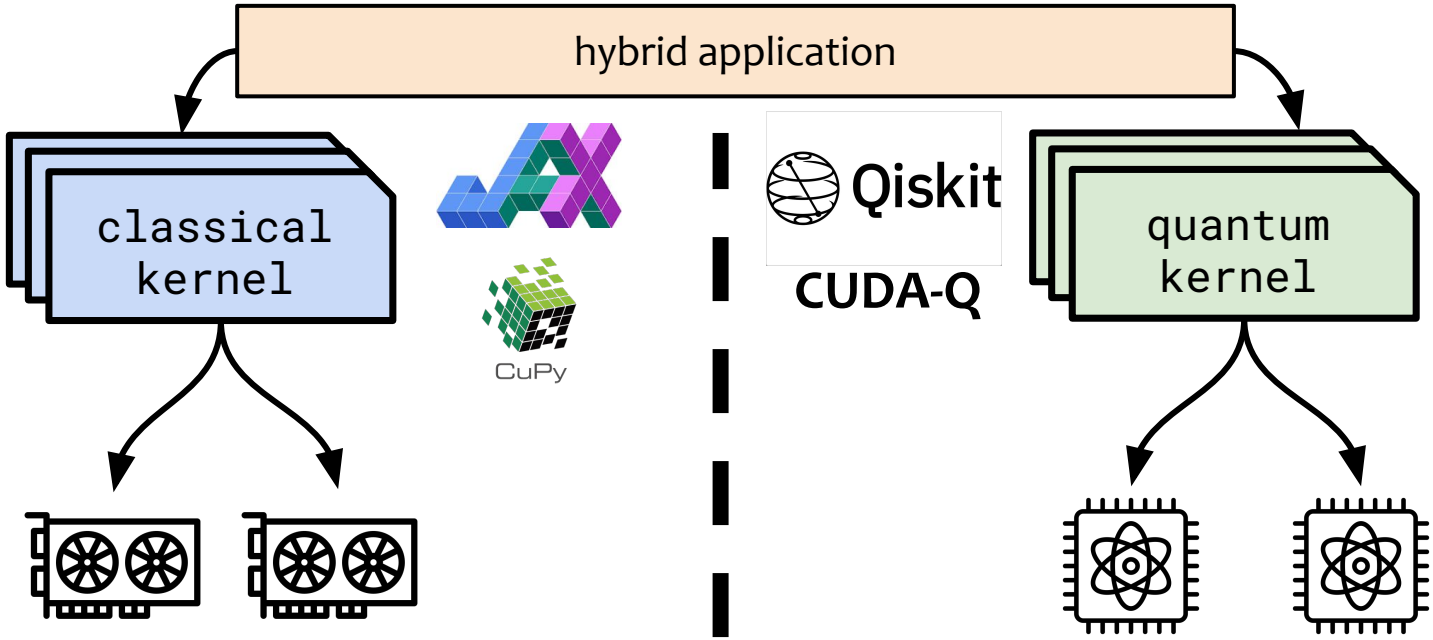
Quantum-classical boundary fixed *a priori*

Manual orchestration



Inefficient runtime

Limitation #1: Fragmentation



Programmers must reason in two distinct paradigms

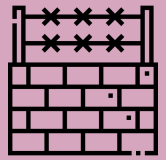
Limitations in designing hybrid applications

Fragmentation



Programming in two distinct paradigms

Fixed boundaries



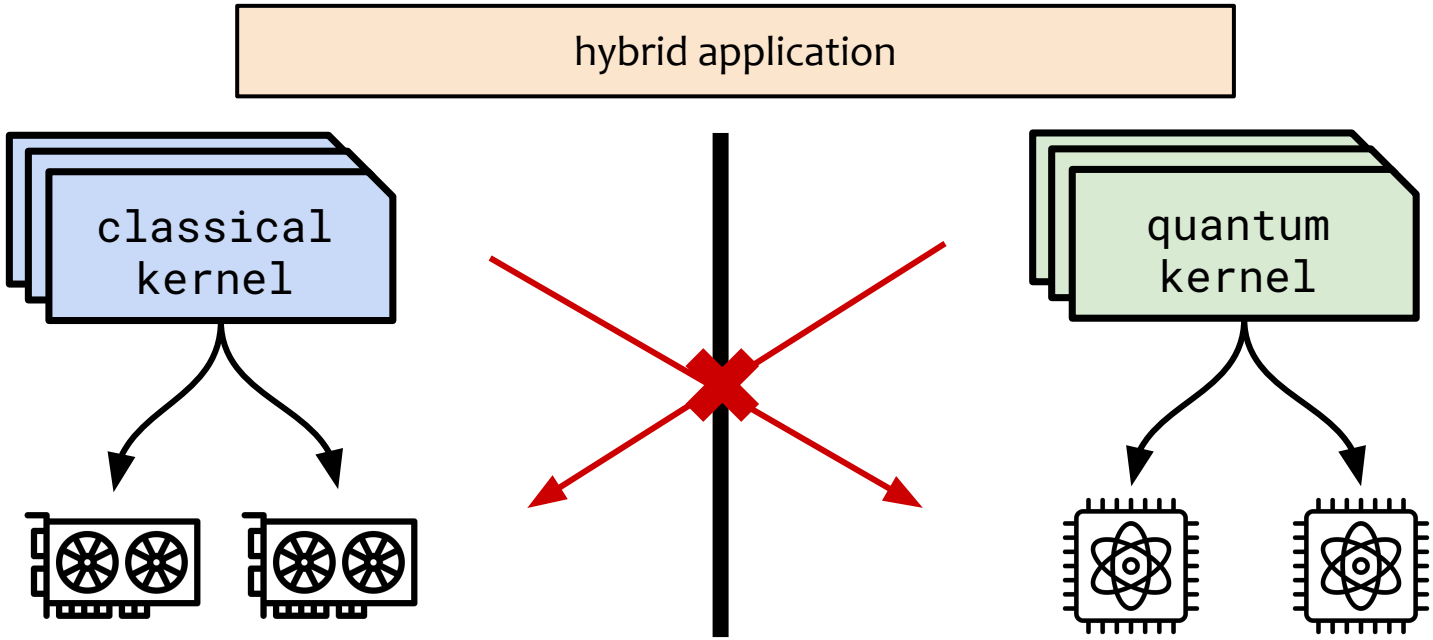
Quantum-classical boundary fixed *a priori*

Manual orchestration



Inefficient runtime

Limitation #2: Fixed boundaries



Quantum-classical boundary fixed *a priori*, preventing holistic optimizations

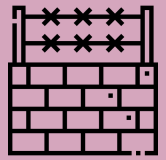
Limitations in designing hybrid applications

Fragmentation



Programming in two distinct paradigms

Fixed boundaries



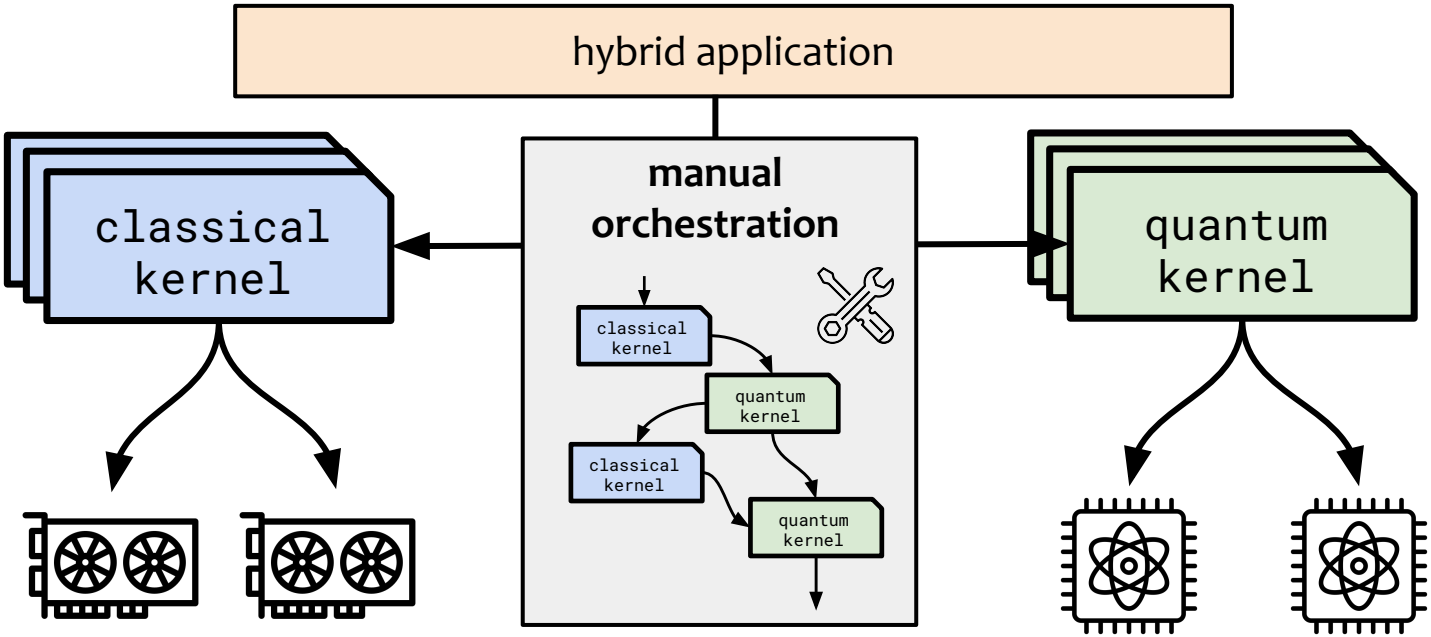
Quantum-classical boundary fixed *a priori*

Manual orchestration



Inefficient runtime

Limitation #3: Manual orchestration

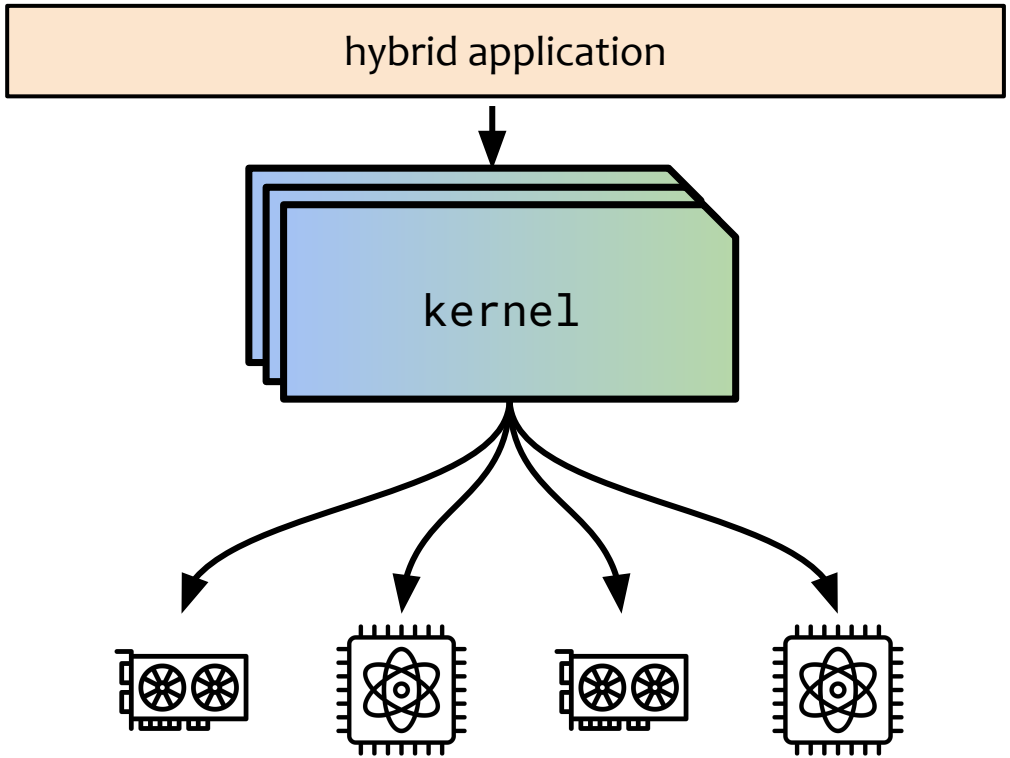


Explicit orchestration, leading to inefficiencies

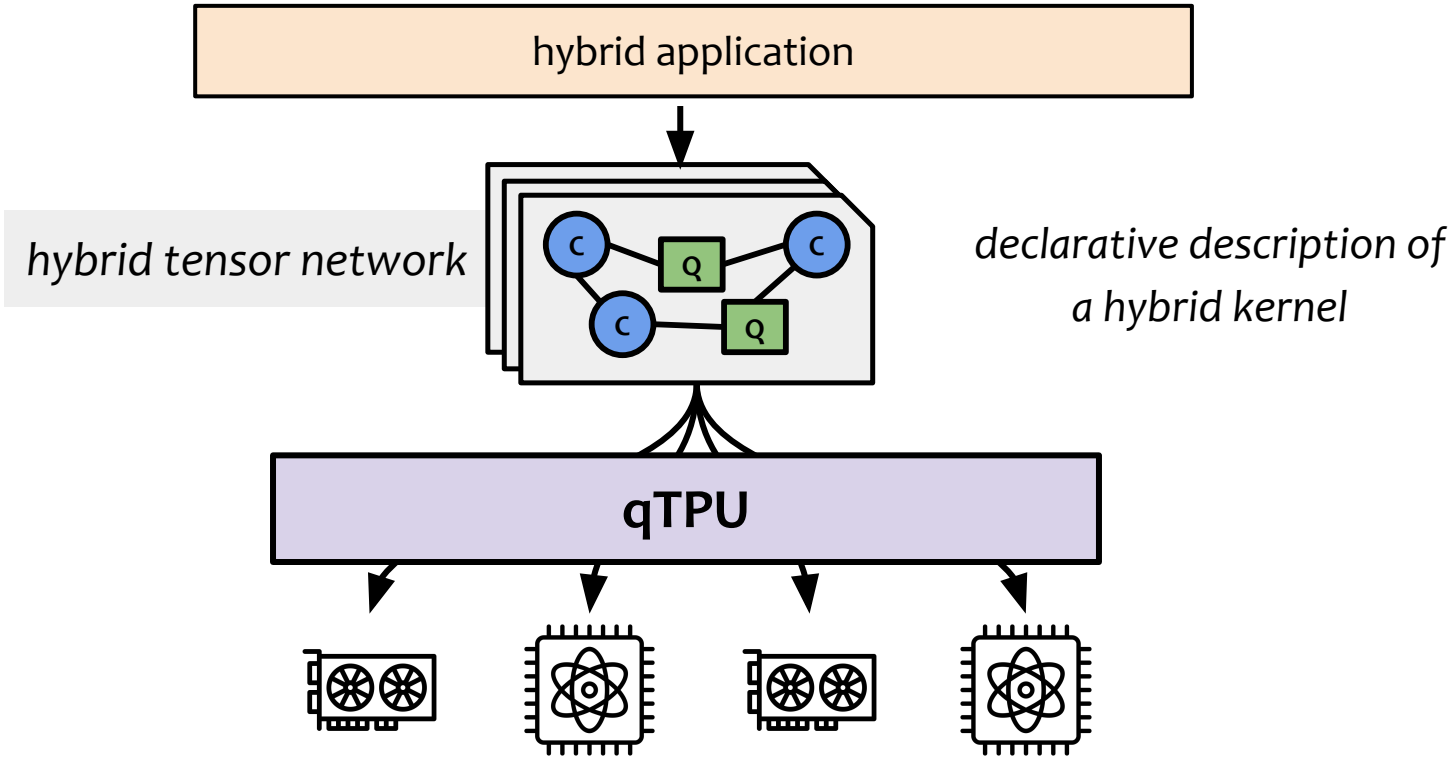
Research question

How can we design a concise and expressive computing paradigm that unifies quantum and classical computation?

Our approach



Our approach



Presenting qTPU

qTPU: A framework for unified, flexible and scalable hybrid computing

Core contributions:

Simple, declarative and unified **programming model** for hybrid computation

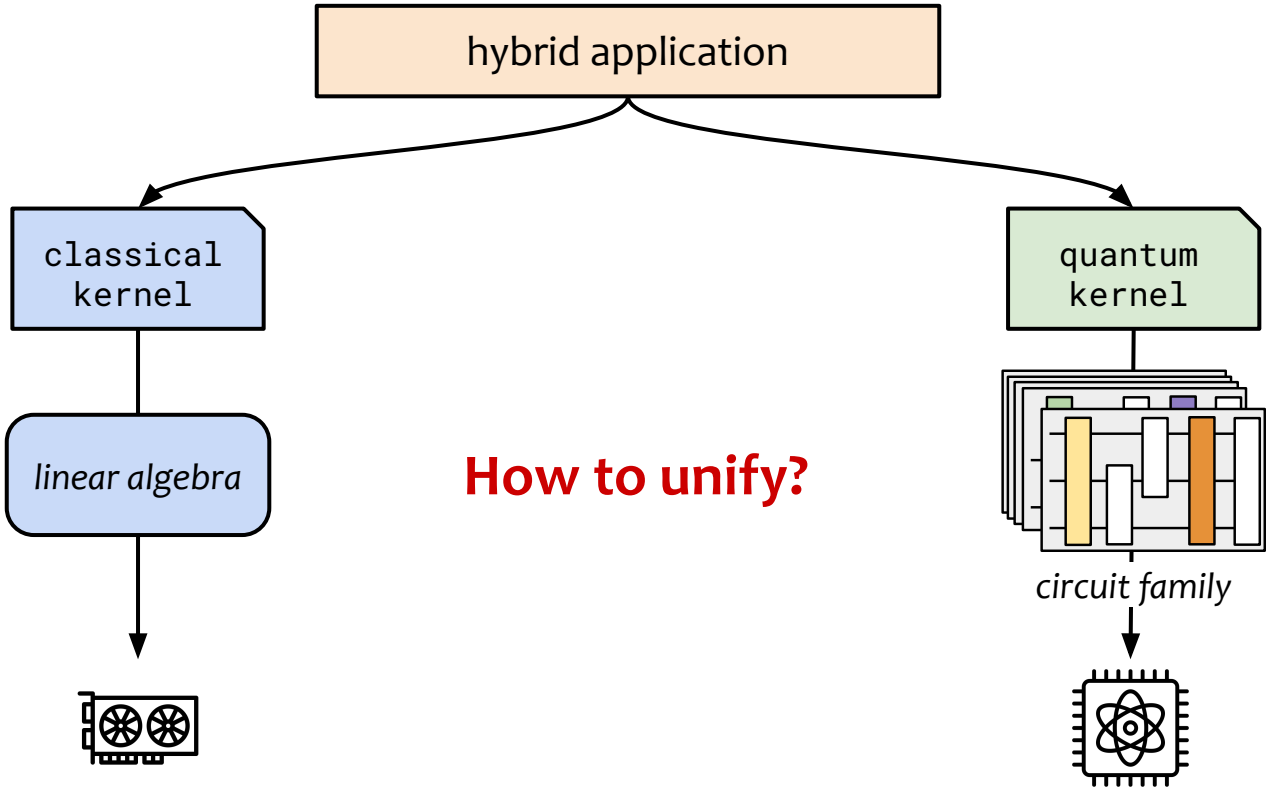
Compiler optimization across quantum-classical paradigms

Scalable, unified runtime across QPUs and classical accelerators

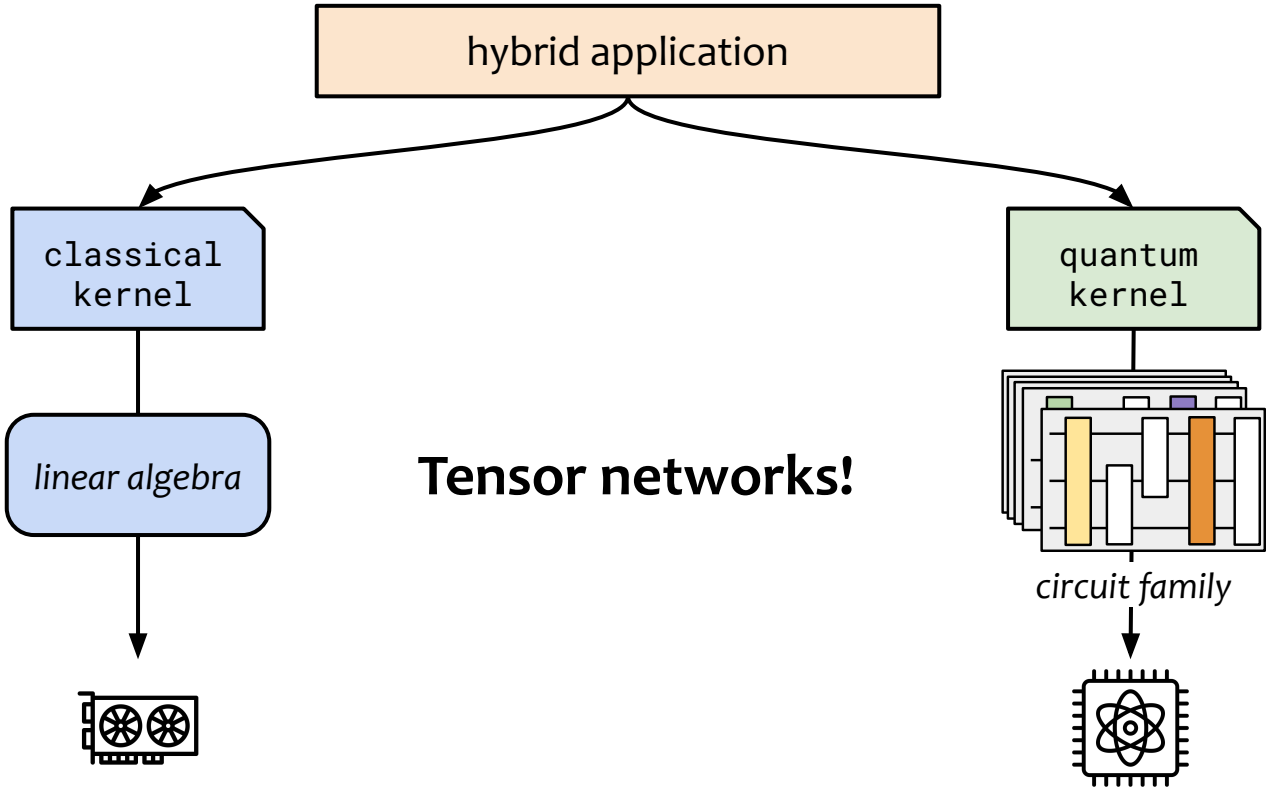
Outline

- Motivation
- Key idea: Hybrid Tensor Networks
- System design
- Evaluation

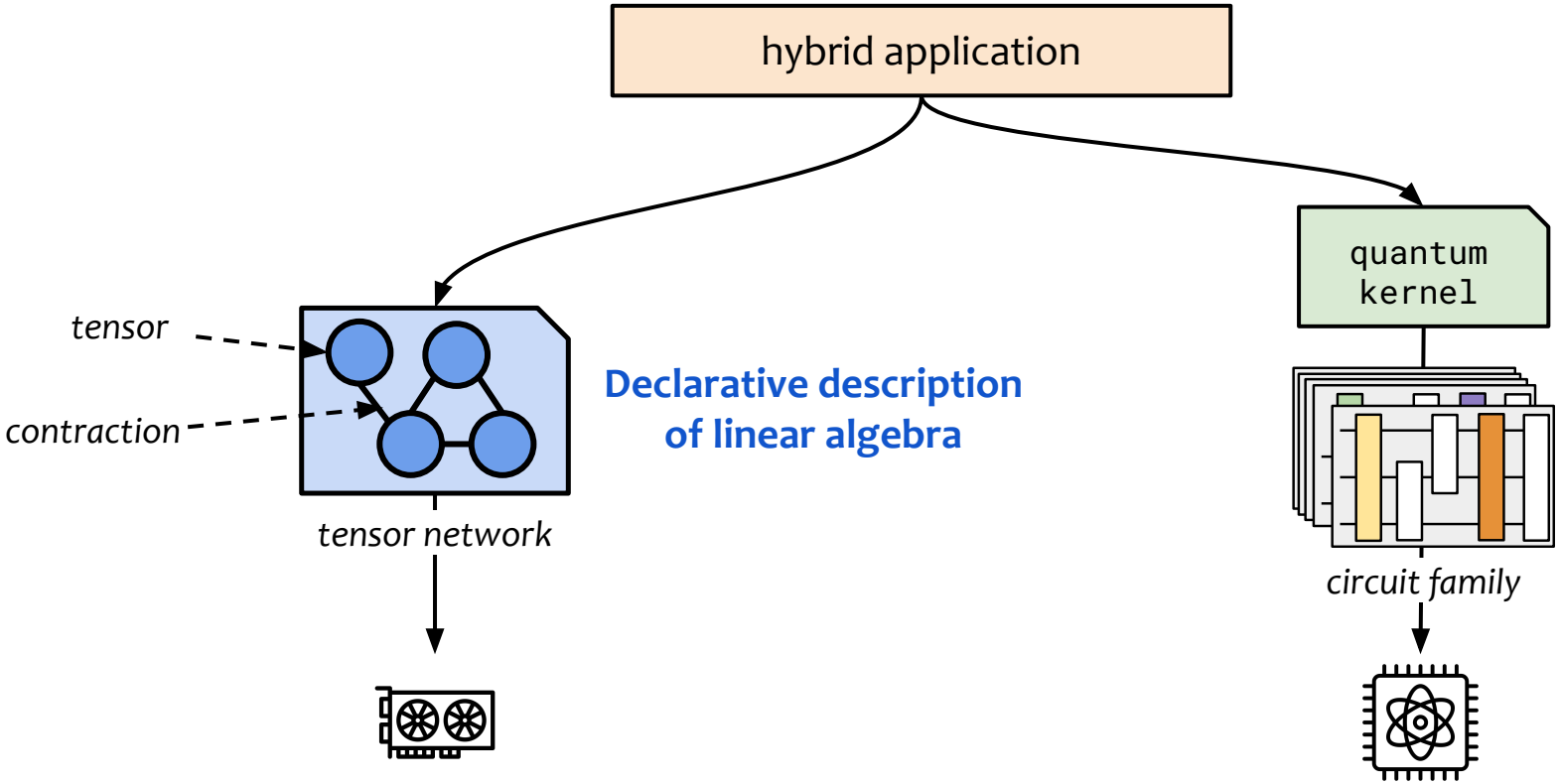
The core of a hybrid application



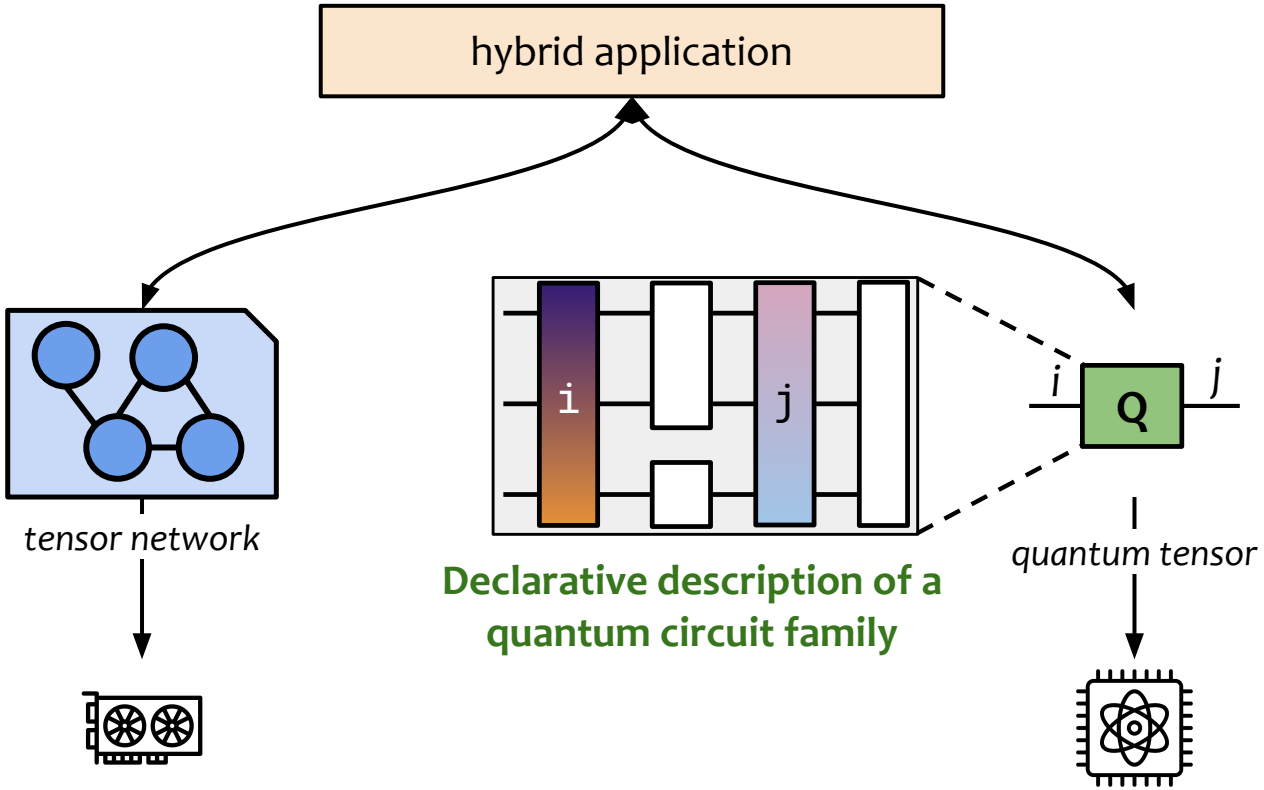
The common denominator: Tensor networks



The common denominator: Tensor networks

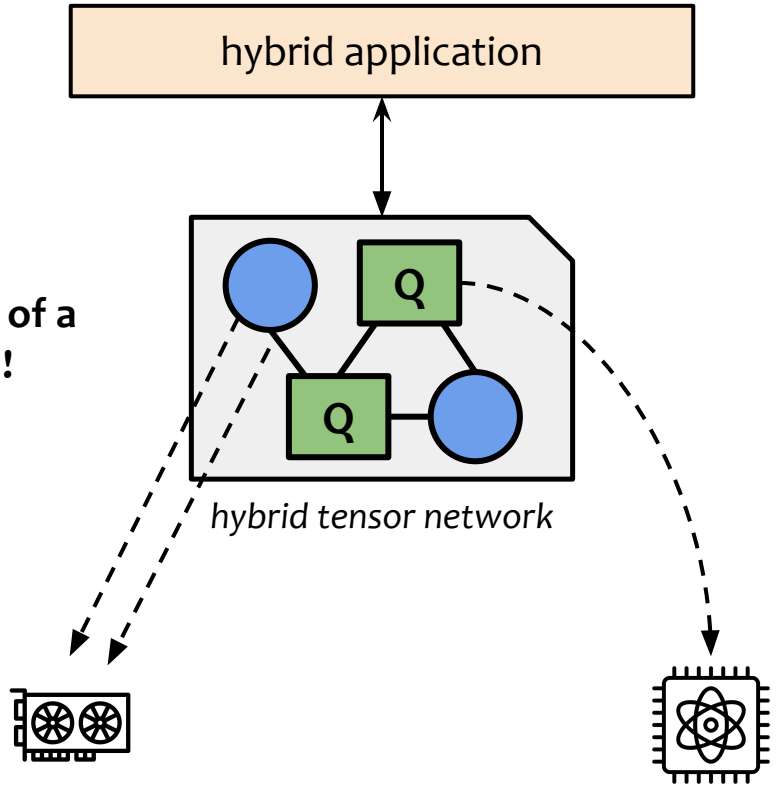


The common denominator: Tensor networks



Our approach: Hybrid Tensor Networks

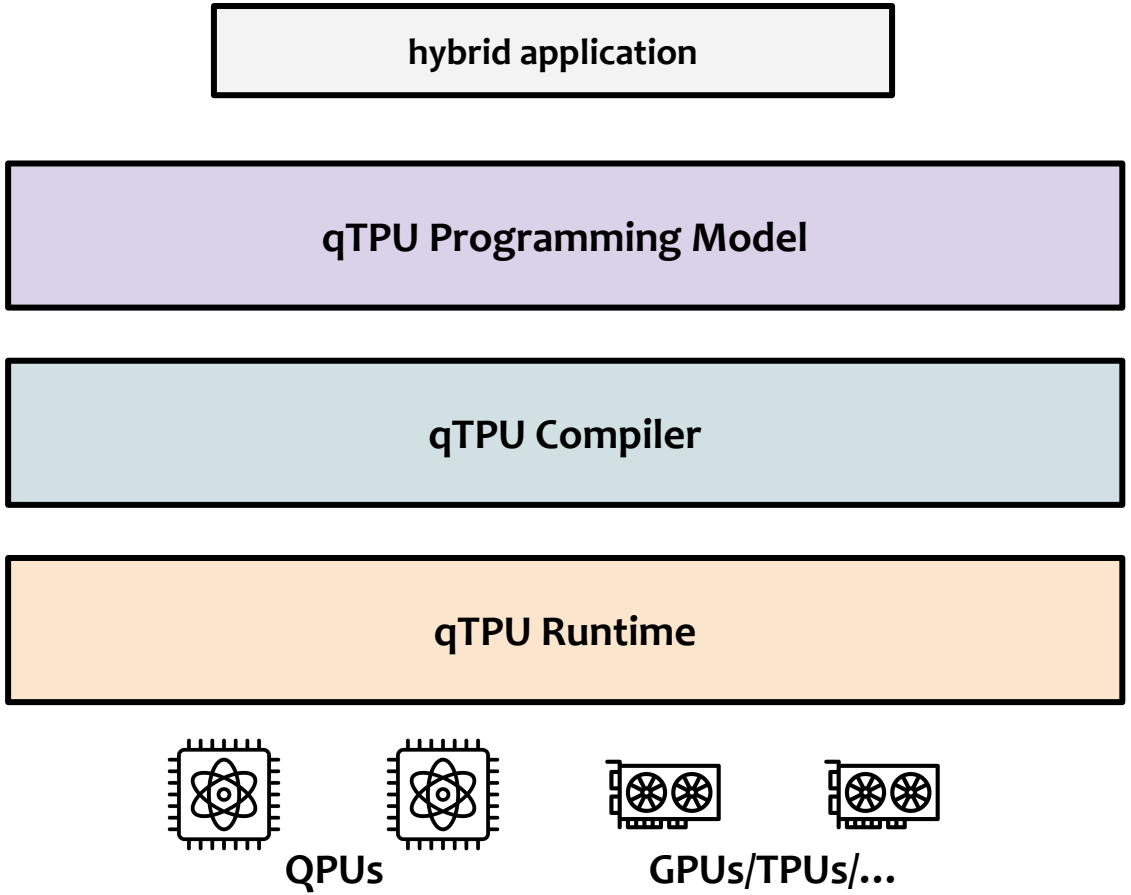
Declarative description of a hybrid computation!



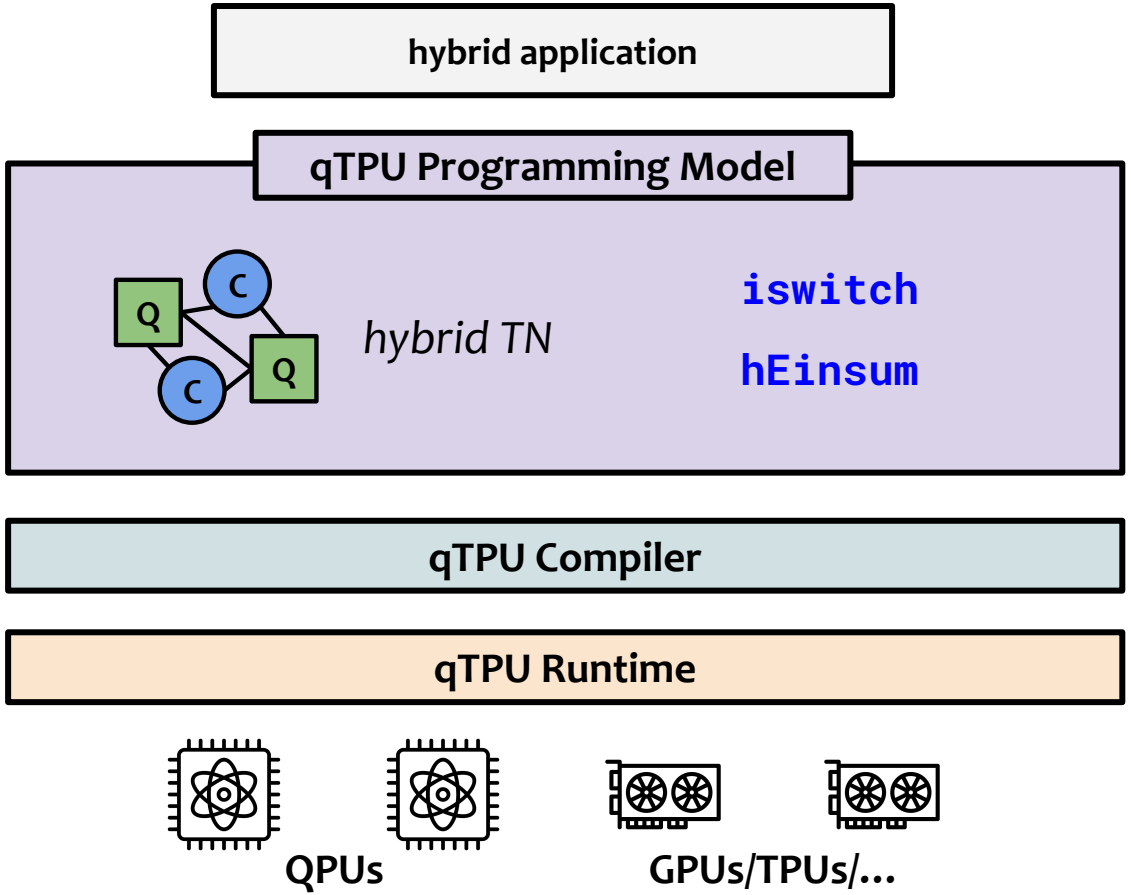
Outline

- Motivation
- Key idea: Hybrid Tensor Networks
- System design
- Evaluation

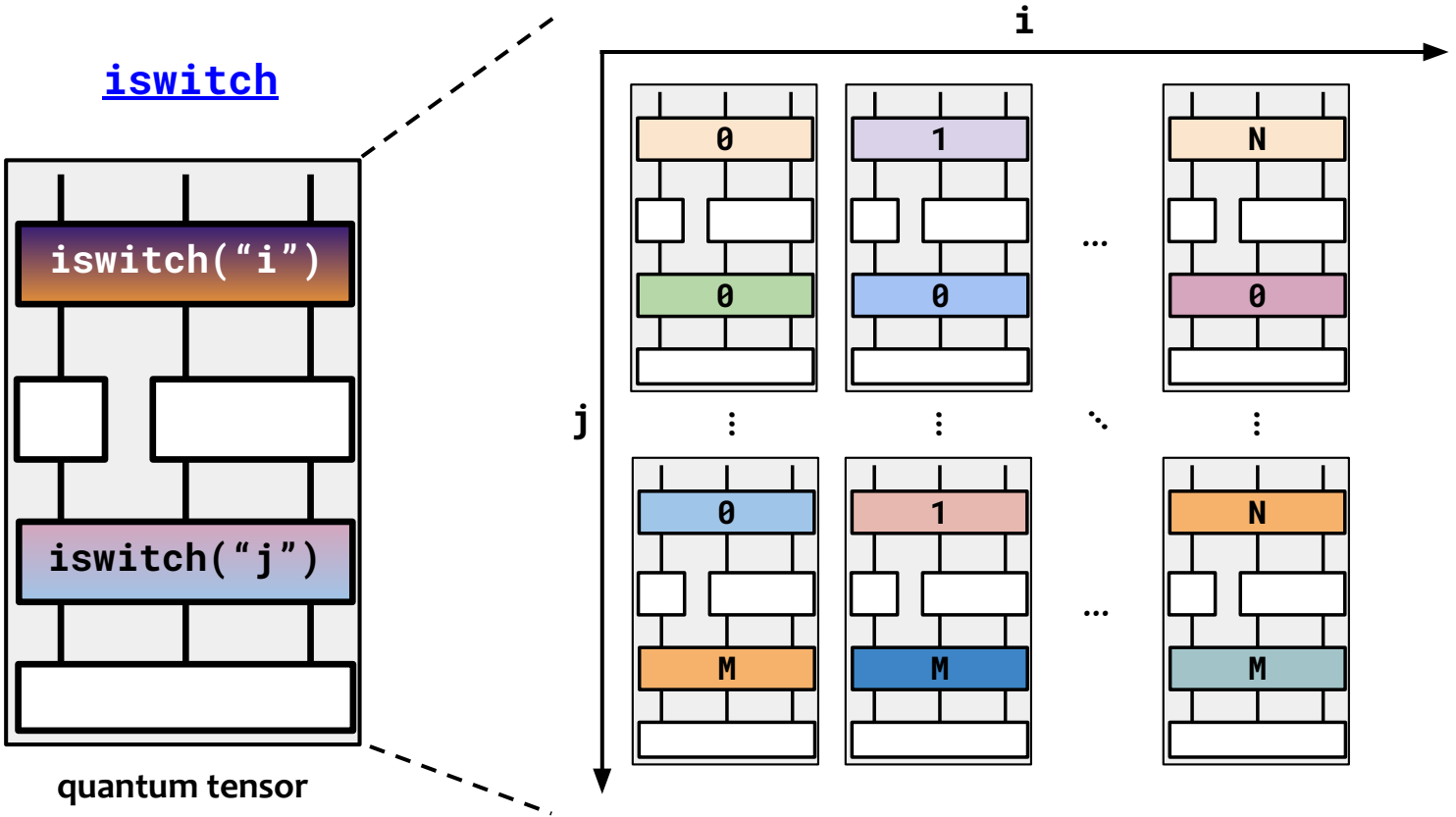
System overview



System overview

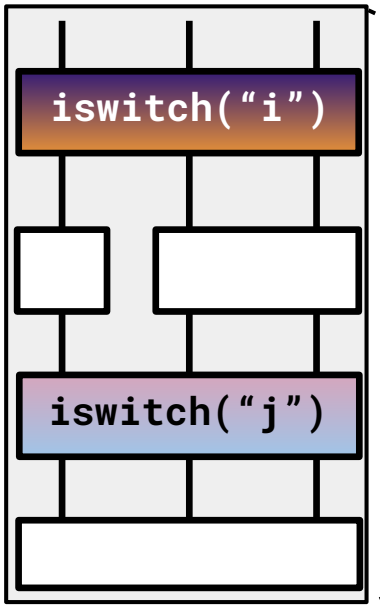


The qTPU Programming Model



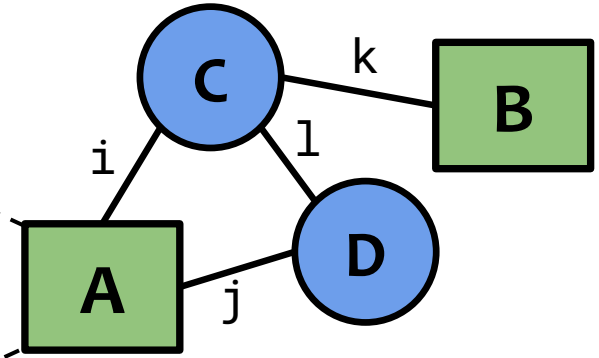
The qTPU Programming Model

iswitch



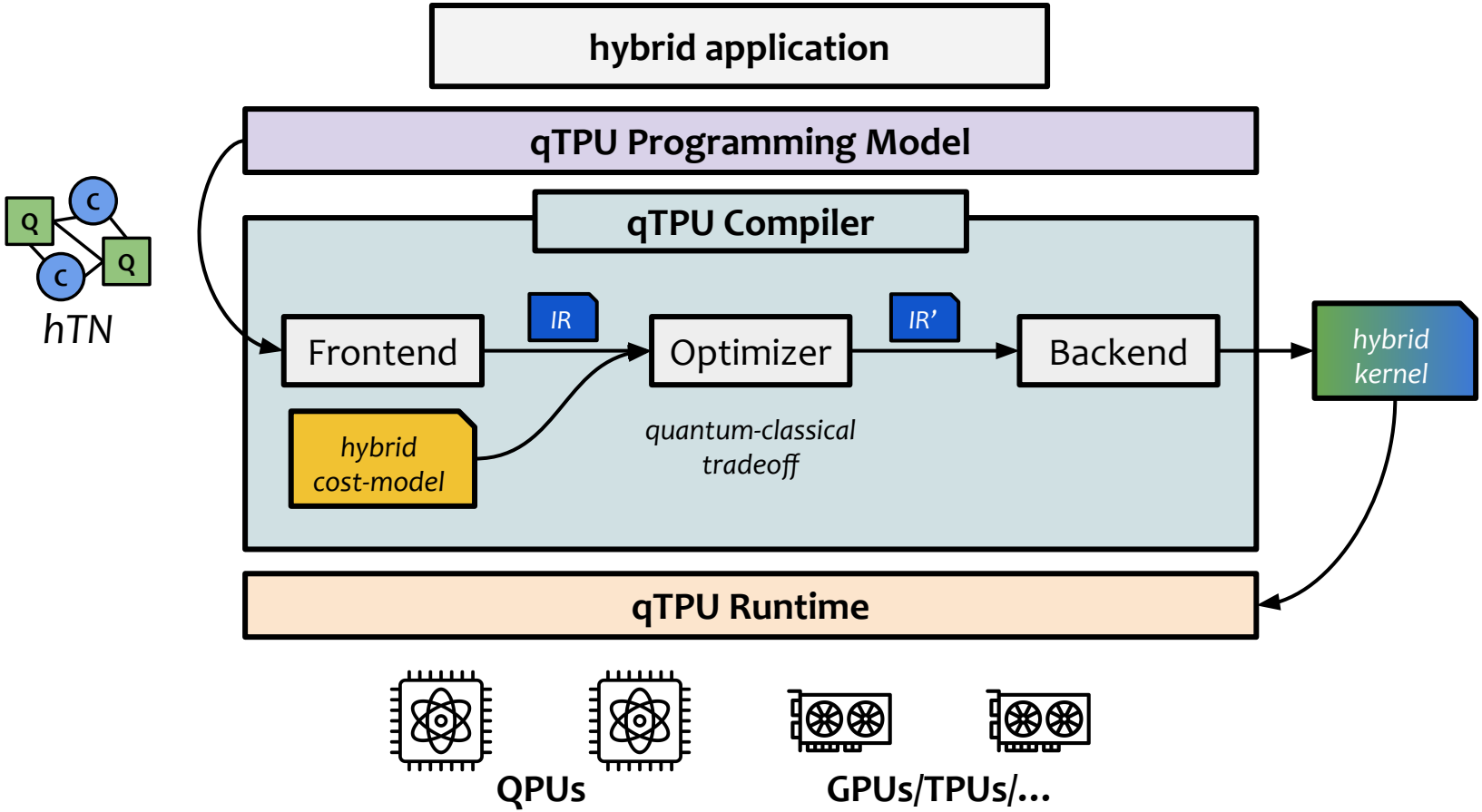
quantum tensor

hEinsum

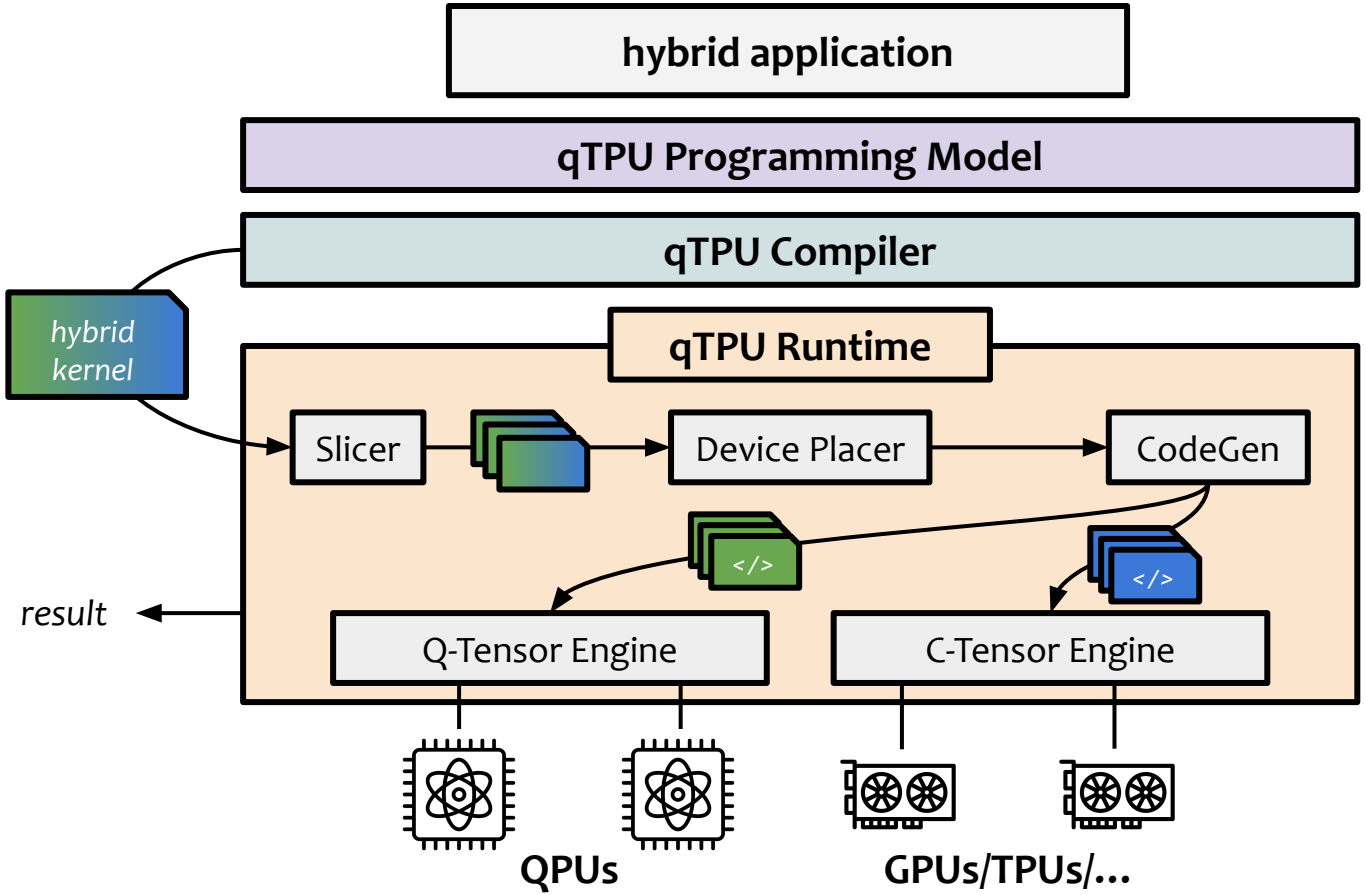



```
hEinsum(
    "ij,k,ilk,jl->",
    A,B,C,D
)
```

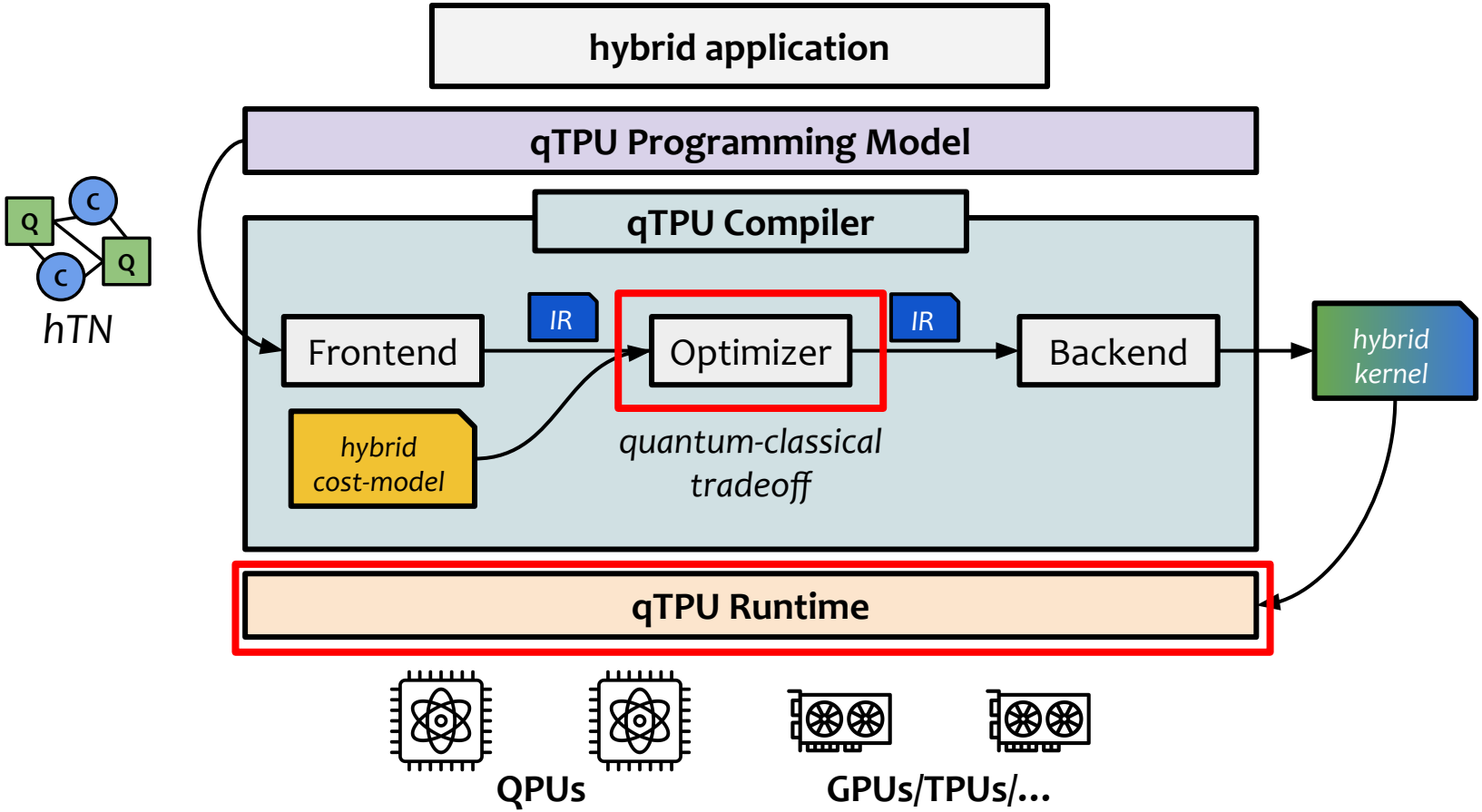
System overview



System overview



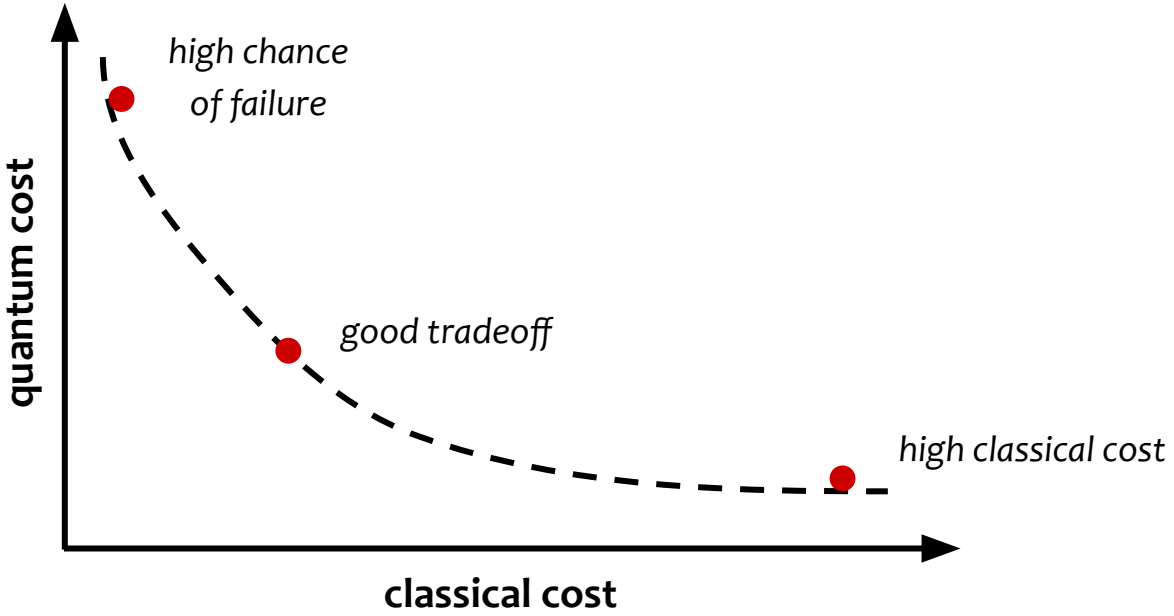
System overview



Optimizer

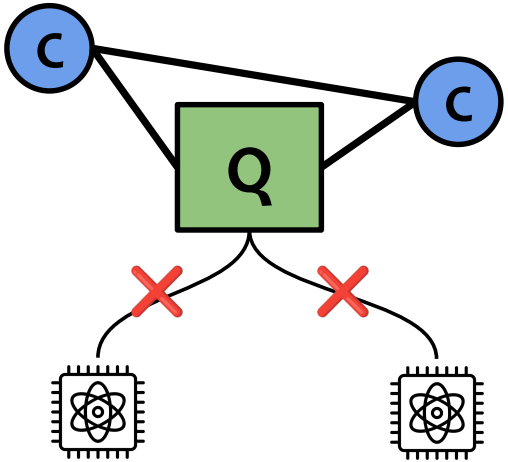
Goal: Find an optimal tradeoff between **quantum error** and **classical cost**

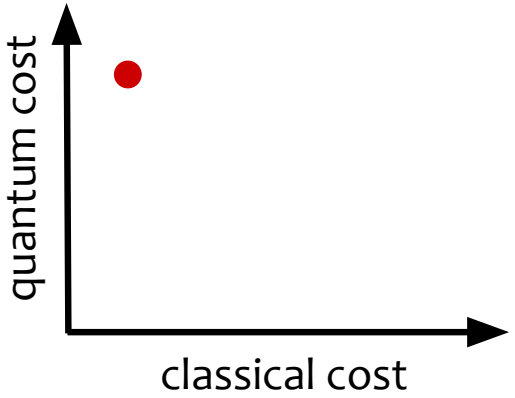
probability of error in executing on a QPU



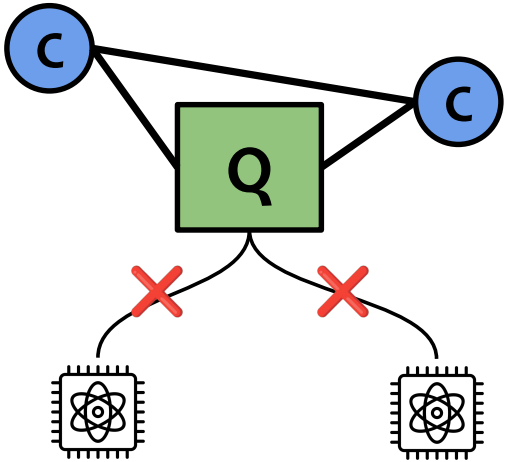
number of FLOPs on a classical accelerator

Optimizer

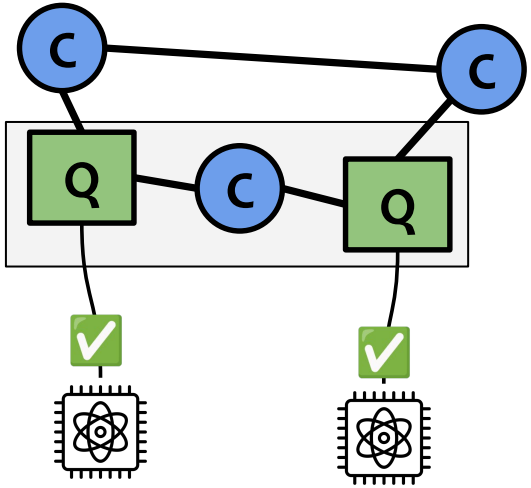


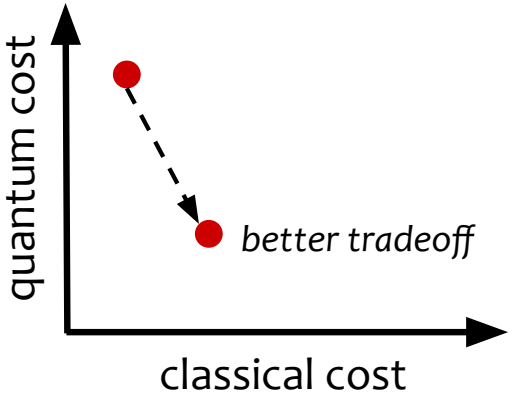


Optimizer

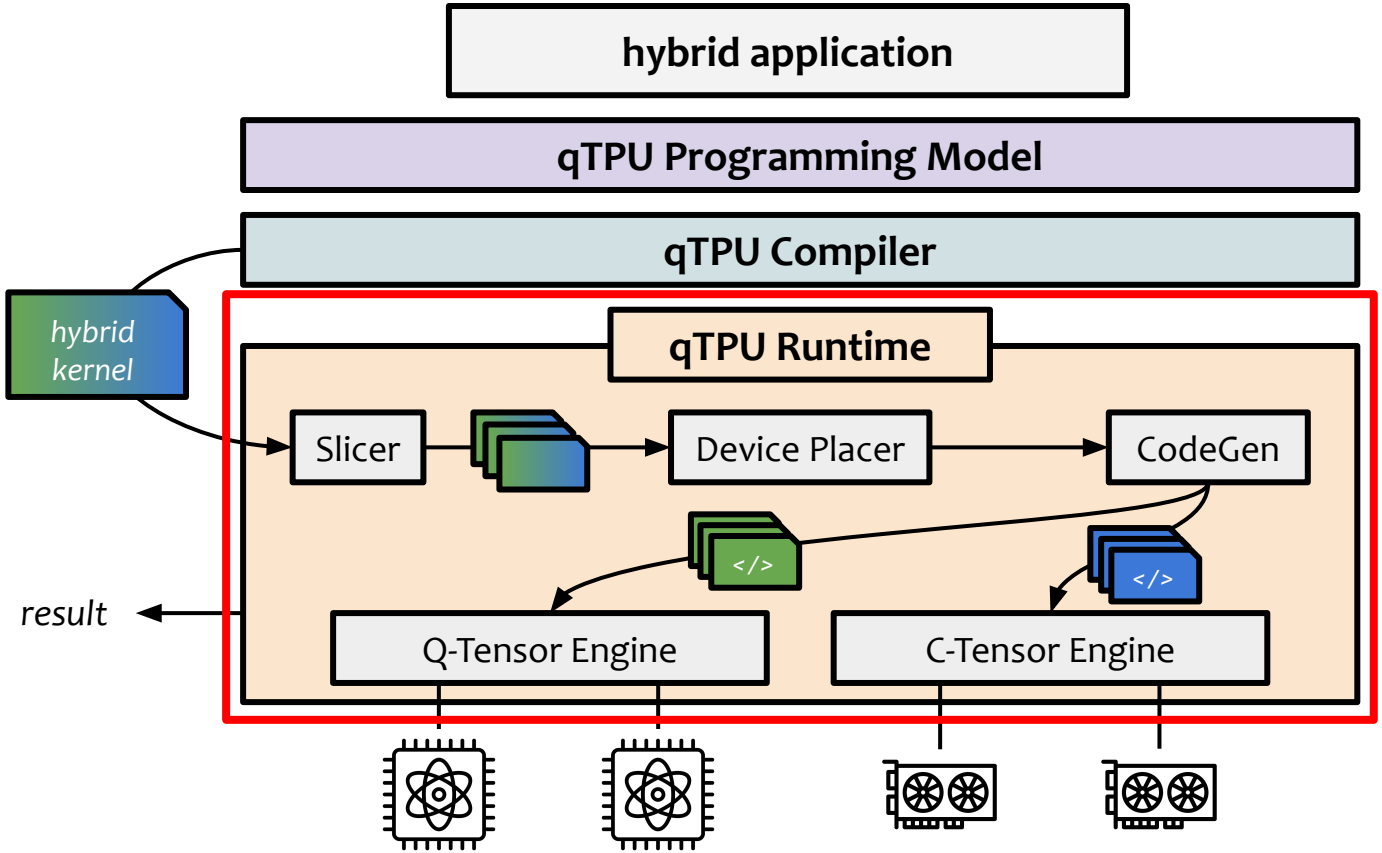

C
Q
C


optimize

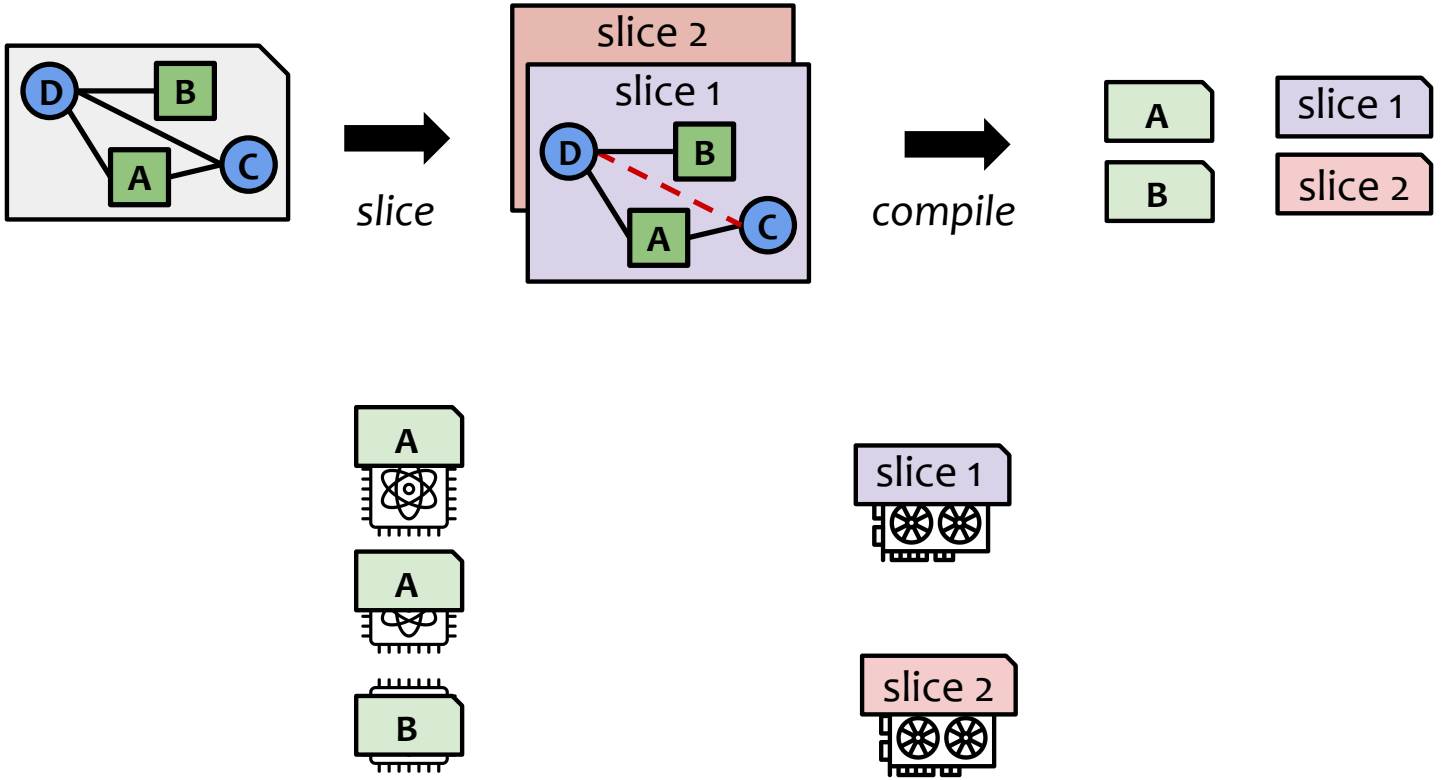

C
C
Q
C
Q


quantum cost
better tradeoff
classical cost

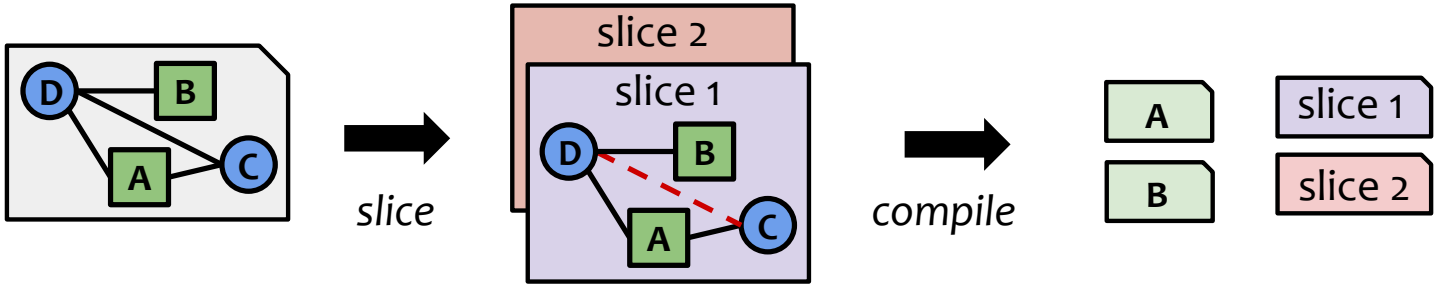
System overview

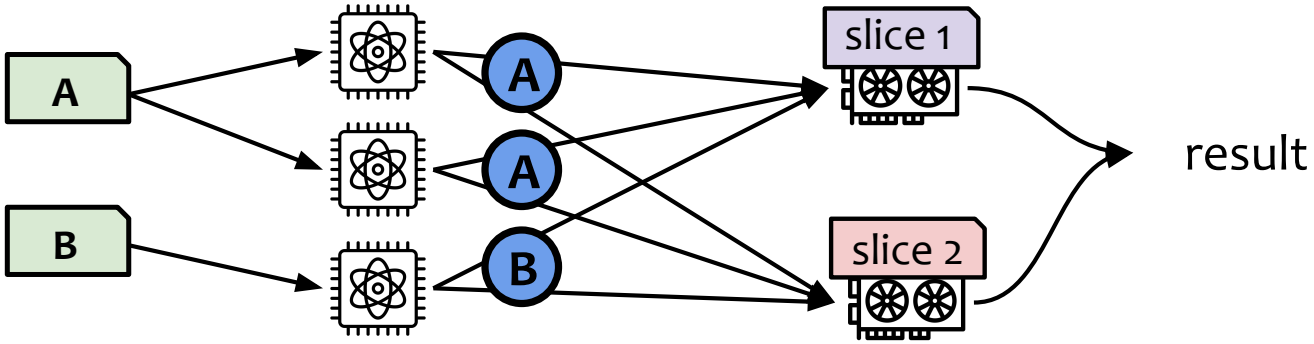


Runtime


D
B
A
C
slice
slice 2
slice 1
D
B
A
C
compile
A
B
slice 1
slice 2
A
A
B
slice 1
slice 2
QPUs
classical accelerators

Runtime


D
B
A
C
slice
slice 2
slice 1
D
B
A
C
compile
A
B
slice 1
slice 2


A
B
A
A
B
slice 1
slice 2
result
QPUs
classical accelerators

Outline

- Motivation
- Key idea: Hybrid Tensor Networks
- System design
- Evaluation

Evaluation

Evaluated along three dimensions:

#1

Programming model & end-to-end case studies:
Does qTPU simplify and accelerate composed hybrid workflows?

#2

Compiler quality:
Can qTPU explore a quantum-classical trade-off effectively?

#3

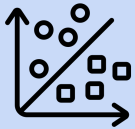
Runtime efficiency:
Does qTPU's runtime scale across QPUs and classical accelerators?

Implementation

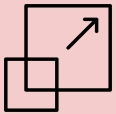
End-to-end implementation based on Qiskit and NVIDIA CUDA QUANTUM.

Three case-studies:

Hybrid ML



Scalable Hybrid Compute
(circuit cutting)



Error Mitigation



Methodology

Experimental setup:

Classical

- **NVIDIA A40 GPU, 48 GB HBM3**
- 2× AMD EPYC 9654 96-core CPUs
- 1.5 TB RAM

Quantum

IBM Marrakesh (IBM Heron r2)

- 156 qubits
- **Error:** 10^{-4} - 10^{-3}

Benchmarks: Widely-used quantum workloads

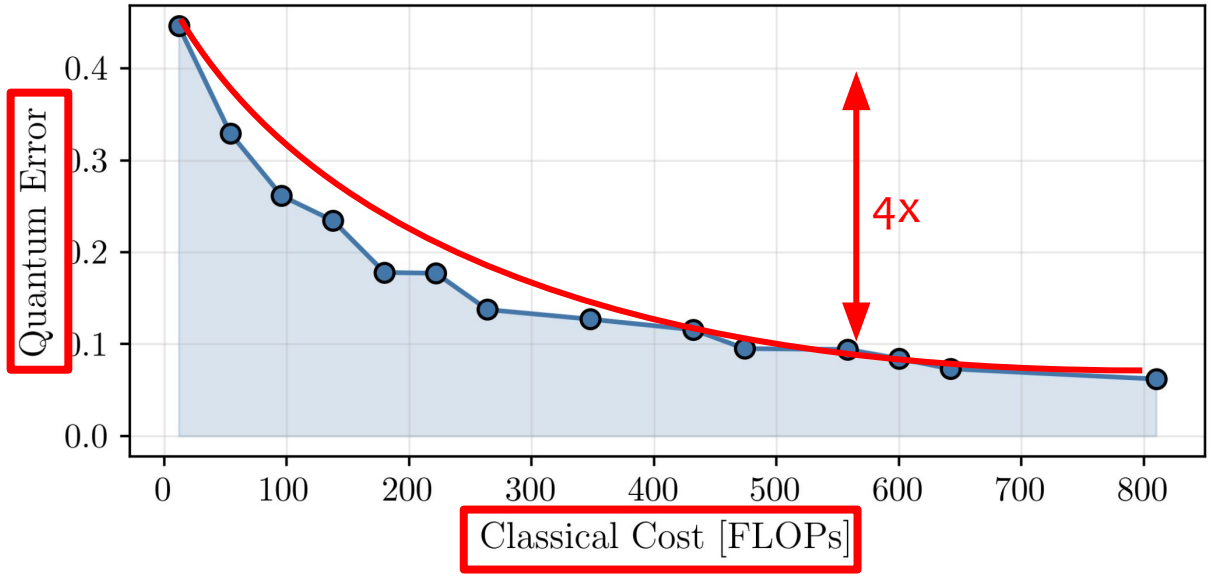
Metrics: Quantum Error & Classical Cost (FLOPs)

Holistic quantum-classical optimization

Can qTPU holistically optimize a hybrid program trading off quantum- and classical-cost?

QNN 100-qubit circuit

Error model: IBM Marrakesh



qTPU enables a flexible tradeoff between quantum cost and classical cost

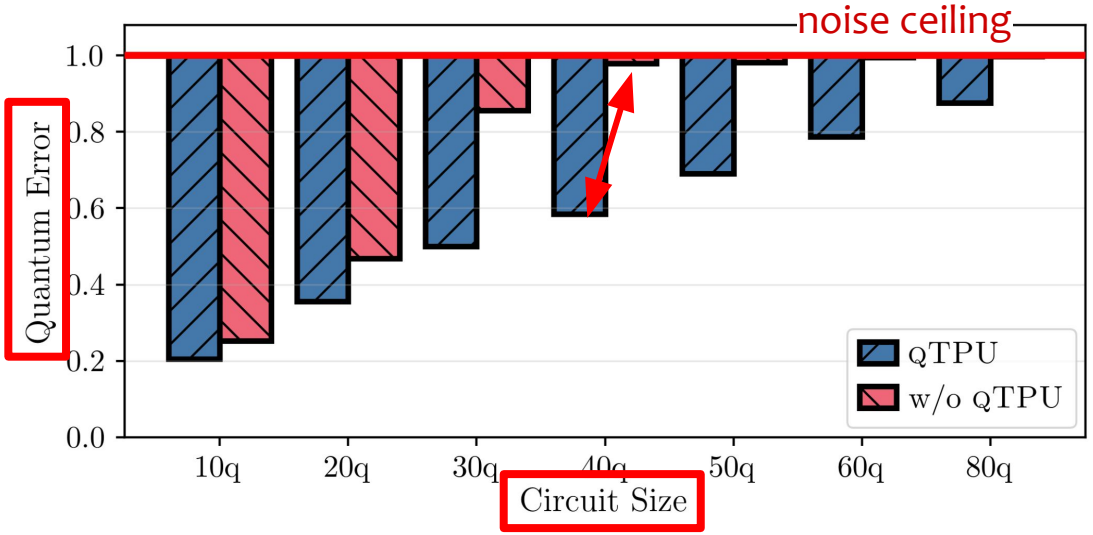
Error-reduction on real QPUs

Does qTPU's error reduction hold on real QPUs at scale?

IBM Marrakesh (IBM Heron r2)

QNN circuits (10-80 qubits)

Baseline: bare-metal



qTPU improves fidelity by 9x on average

Conclusion

Can we build a concise and expressive hybrid computing paradigm?

qTPU: A framework for unified, flexible and scalable hybrid computing

- Simple, declarative and unified programming model for hybrid computation
- Optimization across quantum-classical paradigms (*tradeoff!*)
- Scalable execution across QPUs and classical accelerators



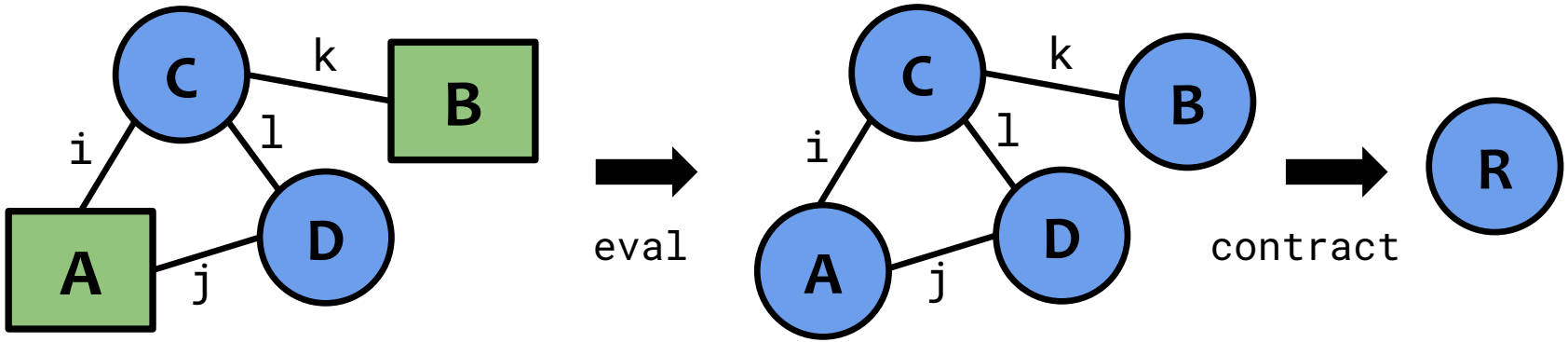


github.com/TUM-DSE/qtpu

nathaniel.tornow@tum.de

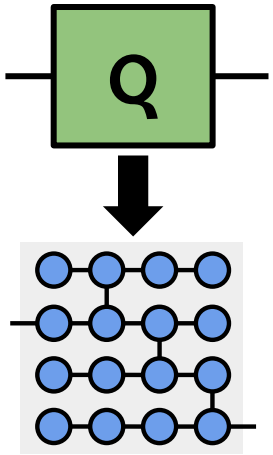
Backup

The qTPU Programming Model

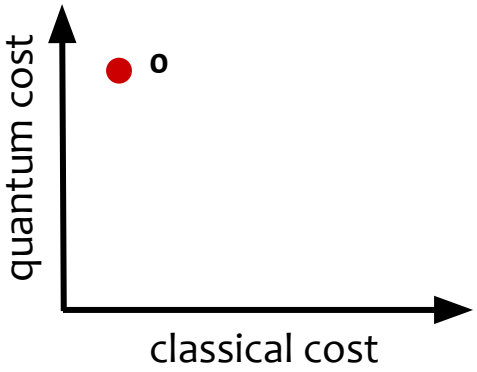



```
R = hEinsum("ij,k,ilk,jl->",A,B,C,D)
```

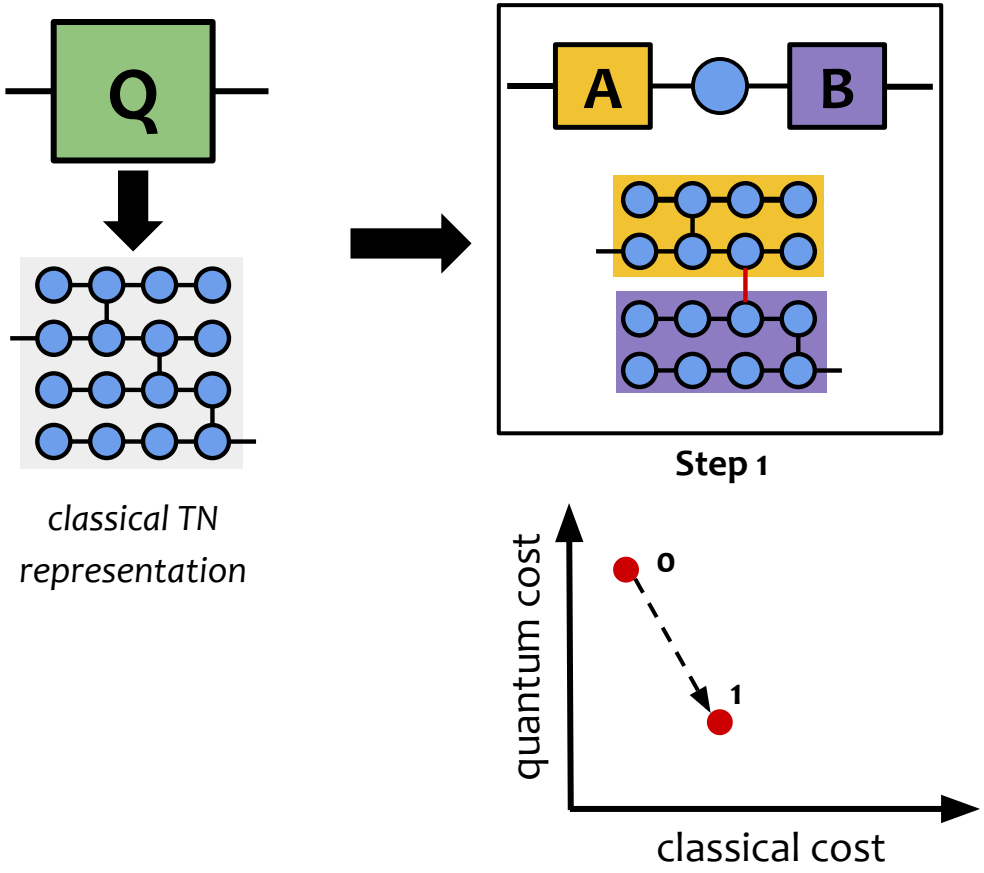
Iterative optimization workflow



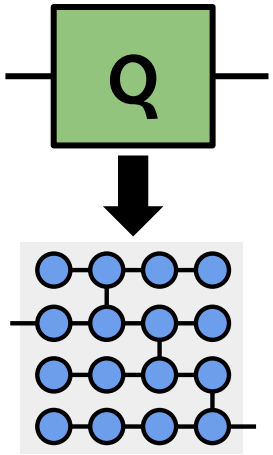
classical TN representation



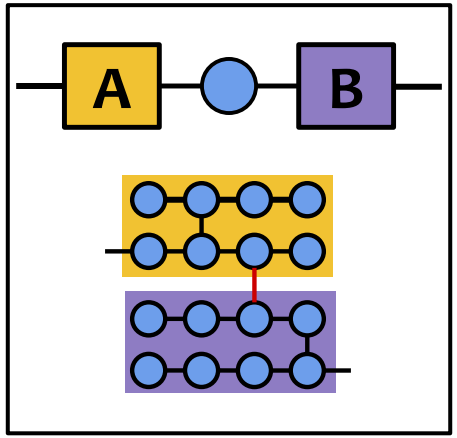
Iterative optimization workflow



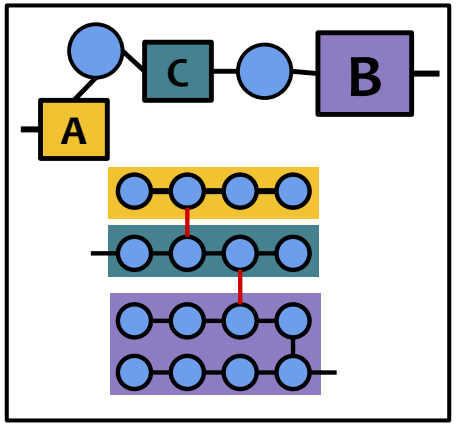
Iterative optimization workflow



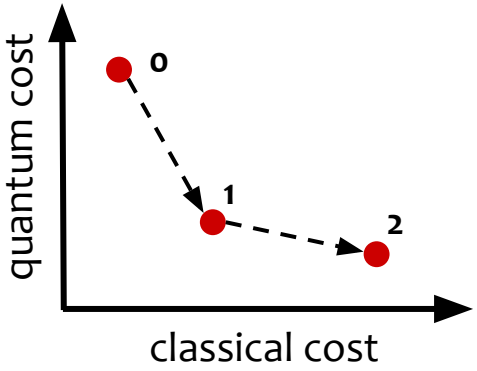
classical TN representation



Step 1



Step 2



iterate until convergence

Programming model

Explicit

```
@qpu
def quantum(x, W, o):
    q = alloq()
    ops1(q, x)
    ops2(q, W)
    ops3(q, o)
    measure(q)

@gpu
def classical(Z, V): return matmul(V, Z)

def predict(X, W, V, O):
    circuits = [
        quantum(x,W,o)
        for x in X
        for o in O
    ]
    Z = qpu.run(circuits)
    Y_pred = classical(Z, V)
    return Y_pred
```

Declarative

```
@quantum_tensor
def qtensor(X, W, O):
    q = alloq()
    iswitch(“i”, q, ops1(X))
    ops2(q, W);
    iswitch(“k”, q, ops3(O))
    measure(q)

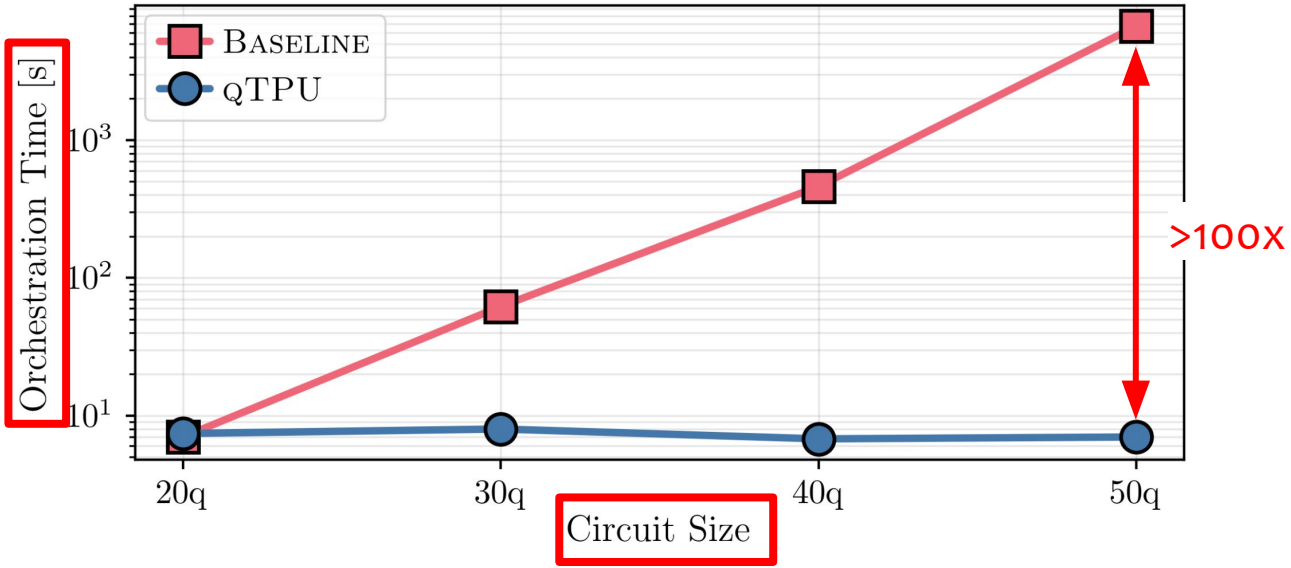
def predict(X, W, V, O):
    return hEinsum(
        “ij,jk->ij”,
        V, qtensor(X,W,O)
    )
```

Impact on orchestration overhead

Does qTPU simplify and accelerate end-to-end quantum-classical workloads?

Benchmark:
Distributed, error-mitigated hybrid ML application

Baseline: *SOTA frameworks*



qTPU eliminates orchestration overheads from quantum-classical workloads