A Case for Multi-Programming Quantum Computers

Poulami Das[*]
Swamit S. Tannu[*]
Prashant J. Nair[#]
Moinuddin Qureshi[*]

[*]Georgia Institute of Technology

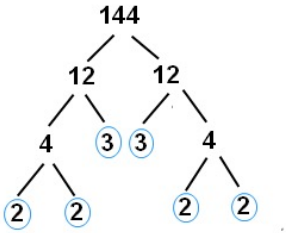
[#]The University of British Columbia

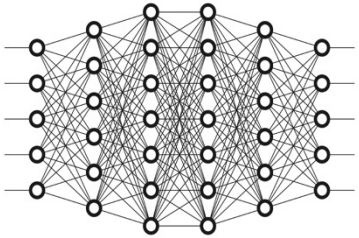
Why Quantum Computing?

- Quantum computers can solve hard problems that conventional computers cannot in a reasonable amount of time

Why Quantum Computing?

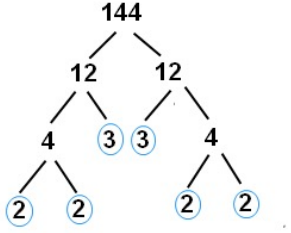
- Quantum computers can solve hard problems that conventional computers cannot in a reasonable amount of time

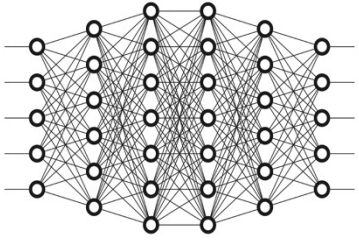




Why Quantum Computing?

- Quantum computers can solve hard problems that conventional computers cannot in a reasonable amount of time



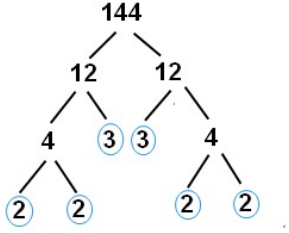


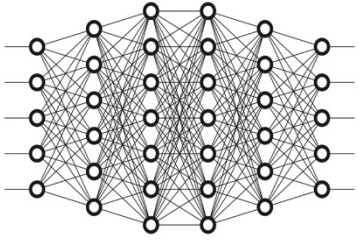
- Quantum computers with 50+ qubits already available!

Why Quantum Computing?

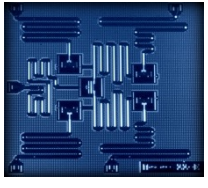
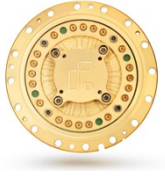
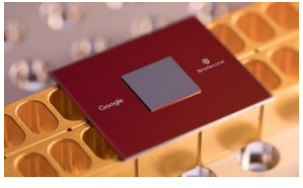
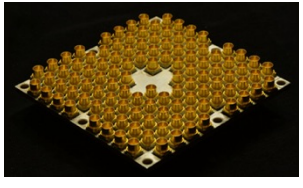
- Quantum computers can solve hard problems that conventional computers cannot in a reasonable amount of time





- Quantum computers with 50+ qubits already available!

Provider	IBM	rigetti	Google	intel
				
# Qubits	5, 16, 20, 53	19	53, 72	49

Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers

Noisy Intermediate Scale Quantum (NISQ) Computing

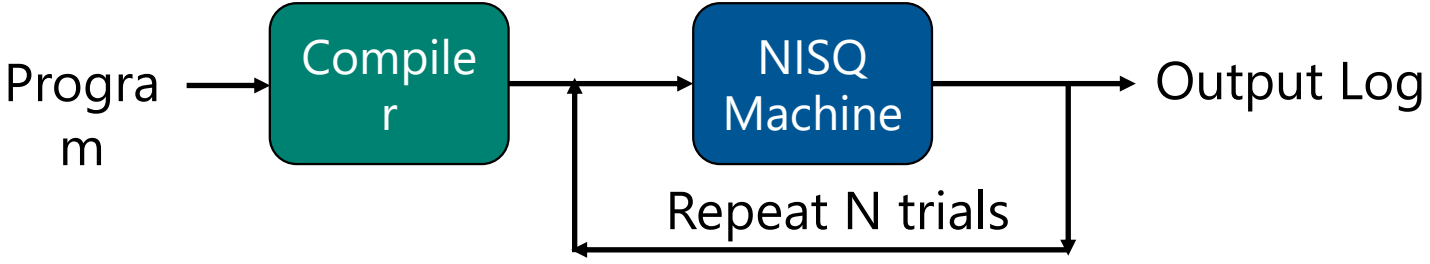
- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction

Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill

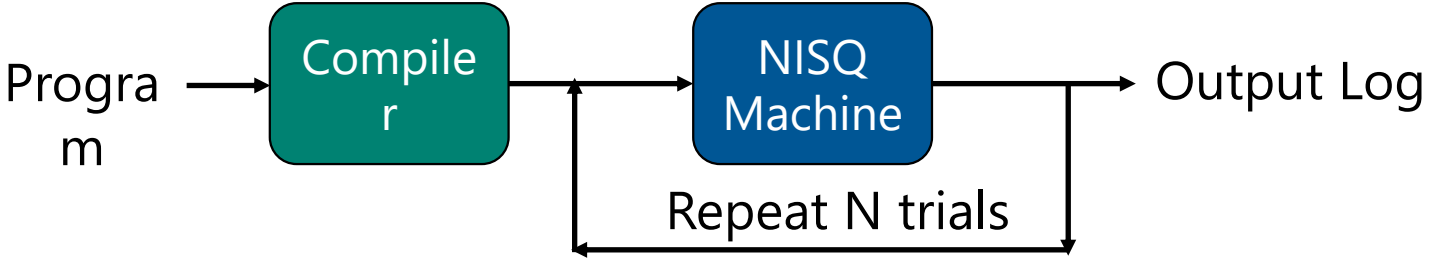
Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill



Noisy Intermediate Scale Quantum (NISQ) Computing

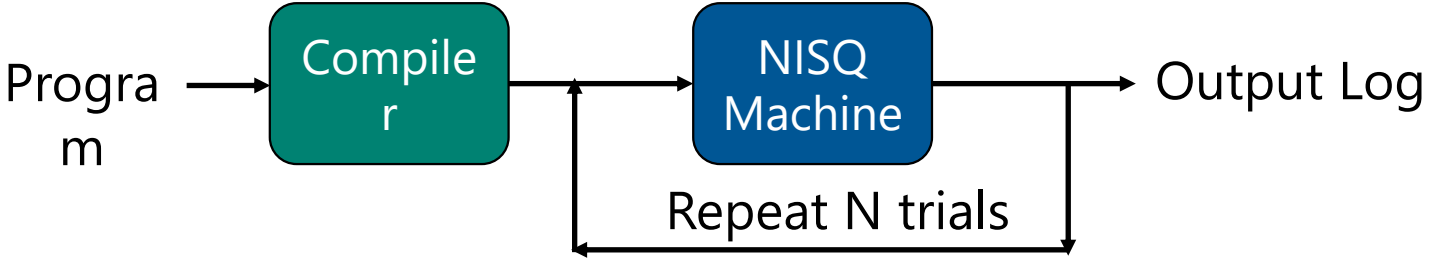
- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill



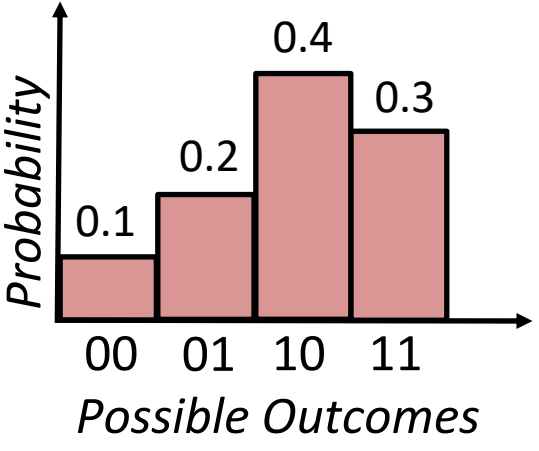
N is large, 8192 for IBM-Q16

Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill

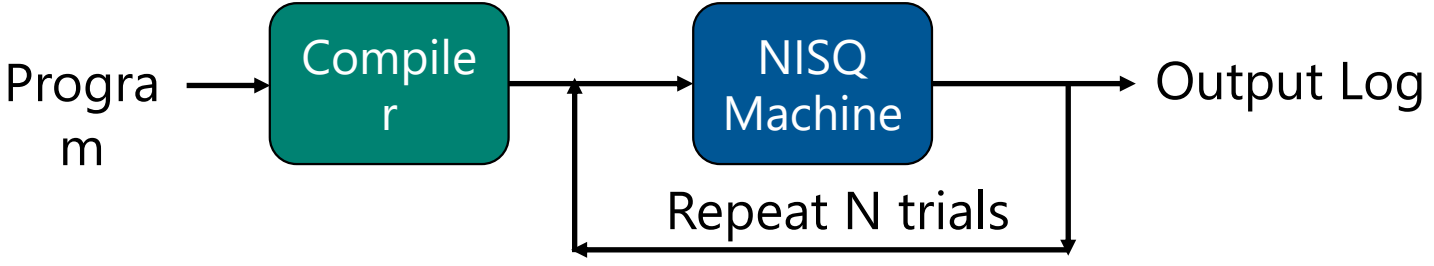


N is large, 8192 for IBM-Q16

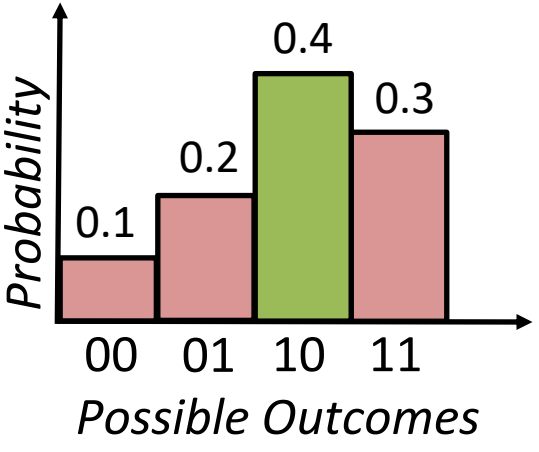


Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill

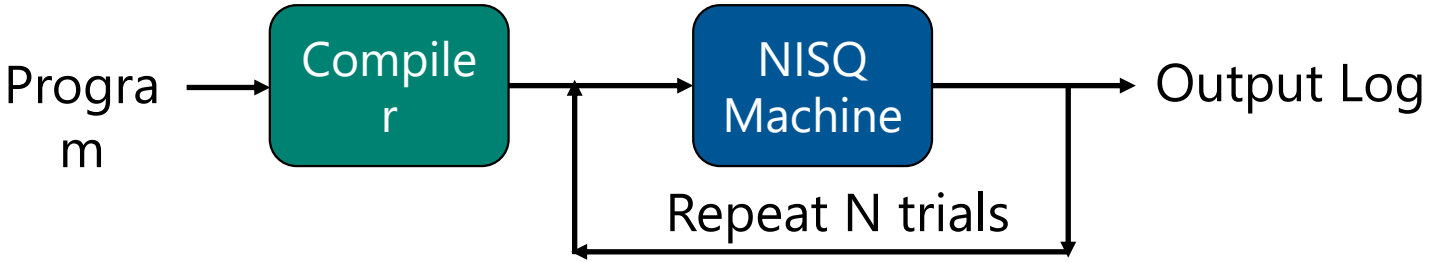


N is large, 8192 for IBM-Q16

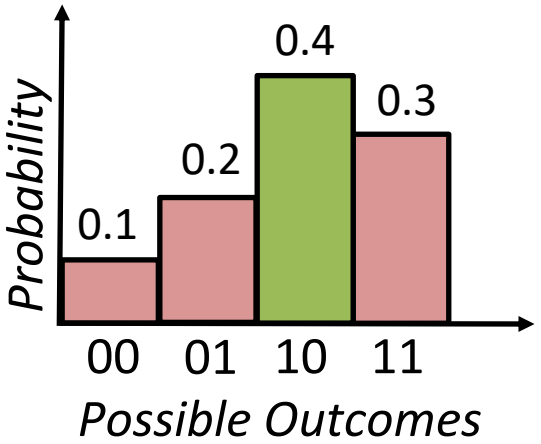


Noisy Intermediate Scale Quantum (NISQ) Computing

- Noise leads to high error rates on existing and near-term quantum computers
- Near term Quantum Computers too small for Quantum Error Correction
- Referred to as NISQ computers- John Preskill



N is large, 8192 for IBM-Q16



- Reliability measured using *Probability of Successful Trial (PST)*

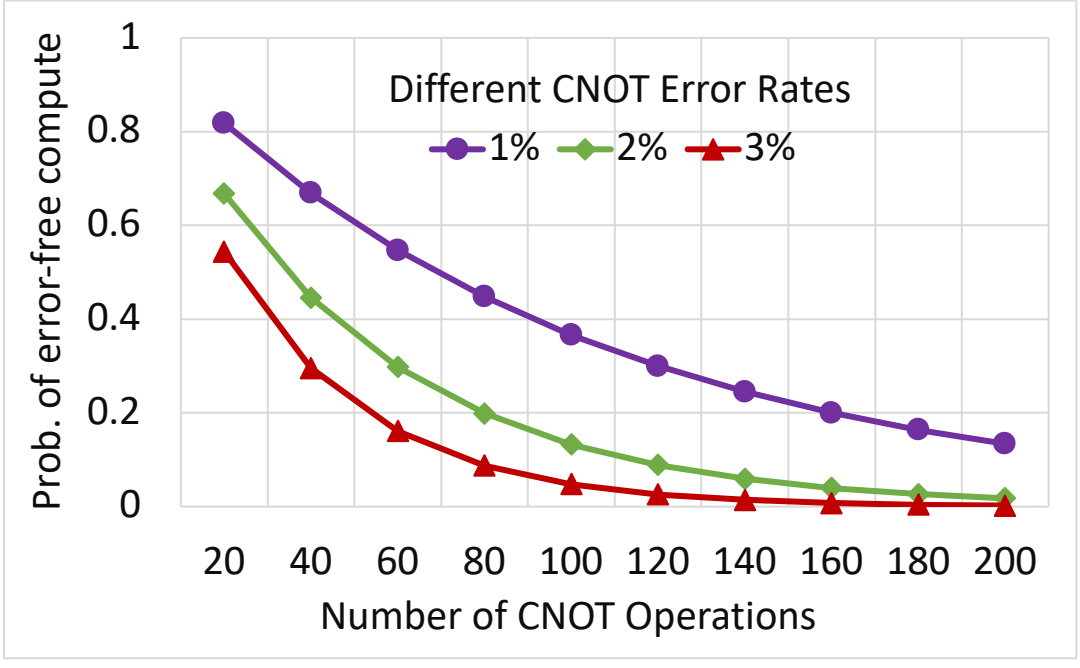
NISQ computers will be operated in the presence of noise and reliability is measured using the PST from multiple trials

Underutilization in NISQ Computers

Error rates limit the number of reliable operations

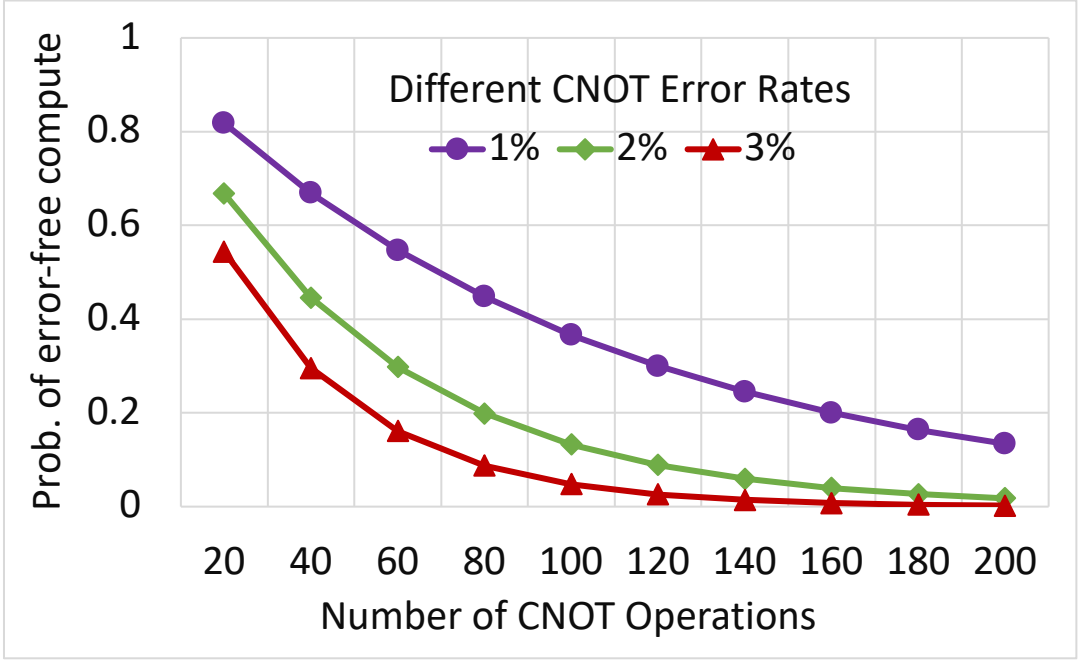
Underutilization in NISQ Computers

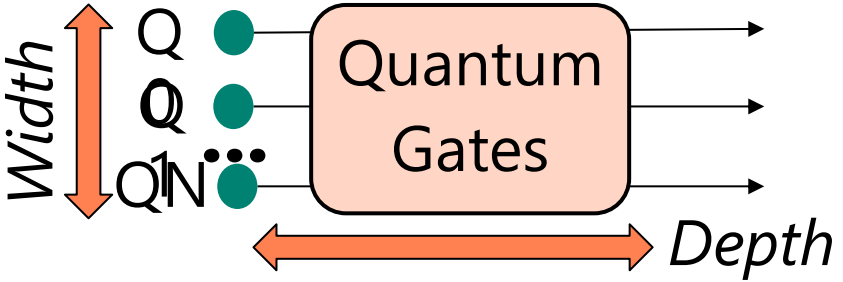
Error rates limit the number of reliable operations



Underutilization in NISQ Computers

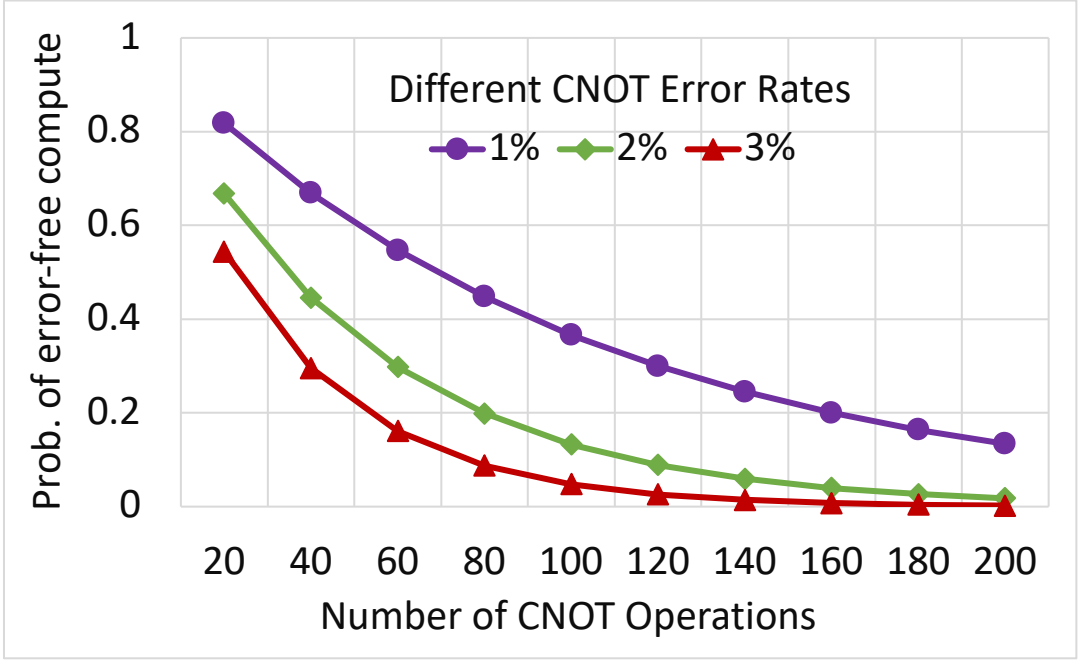
Error rates limit the number of reliable operations

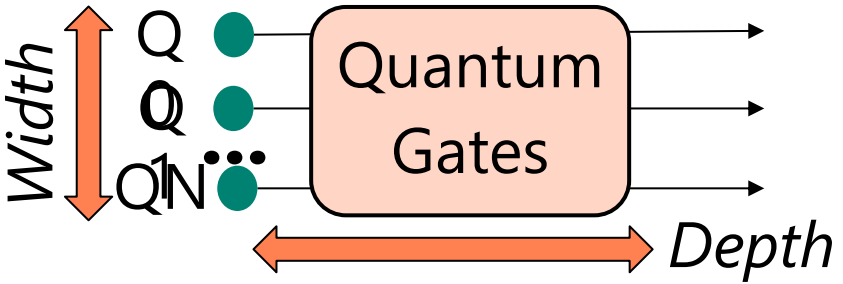


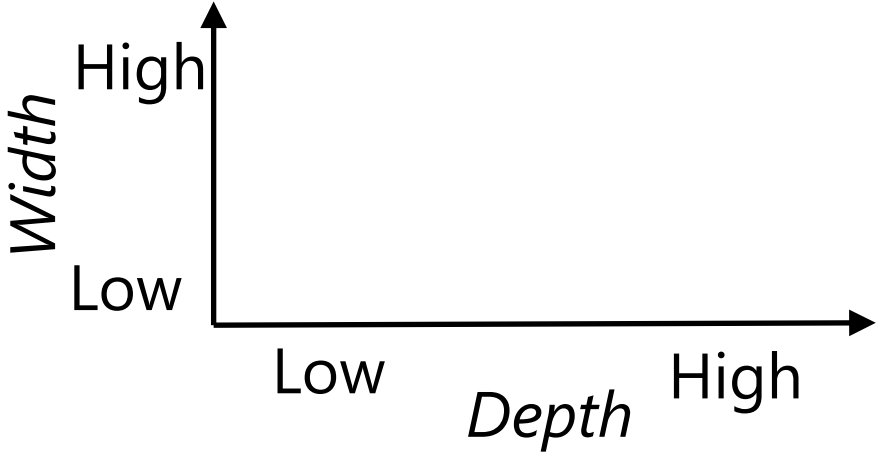


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

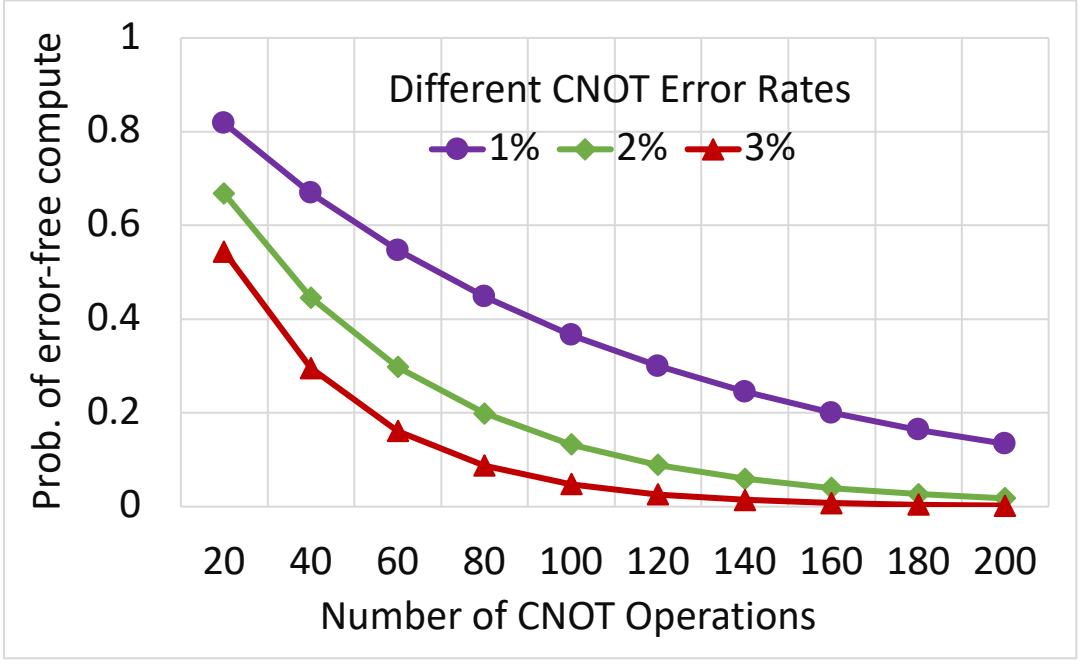


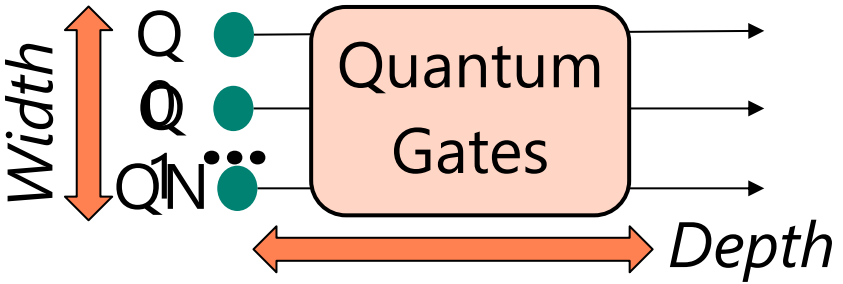


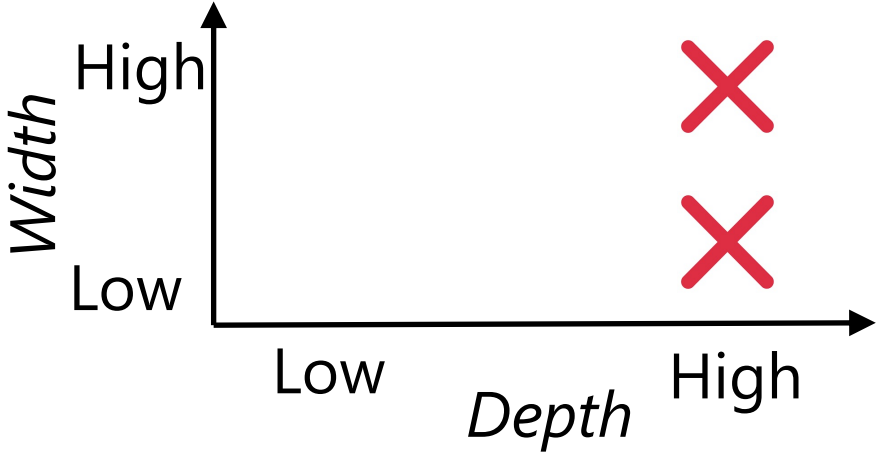


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

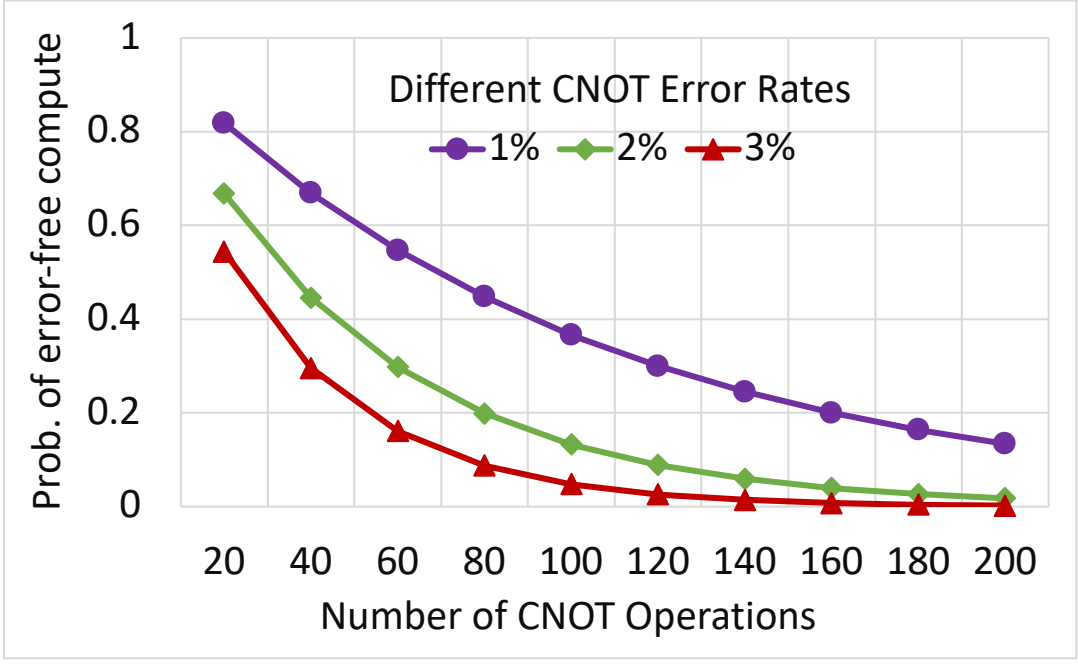


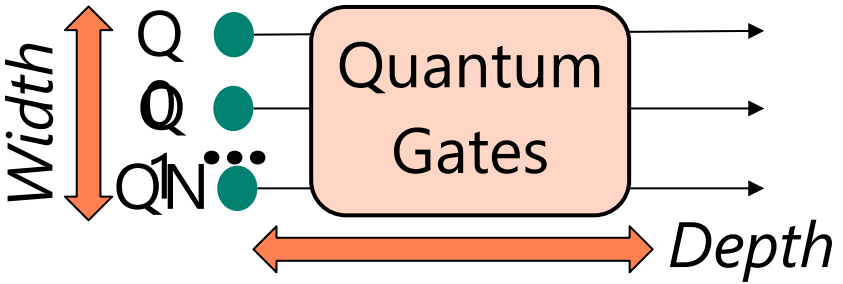


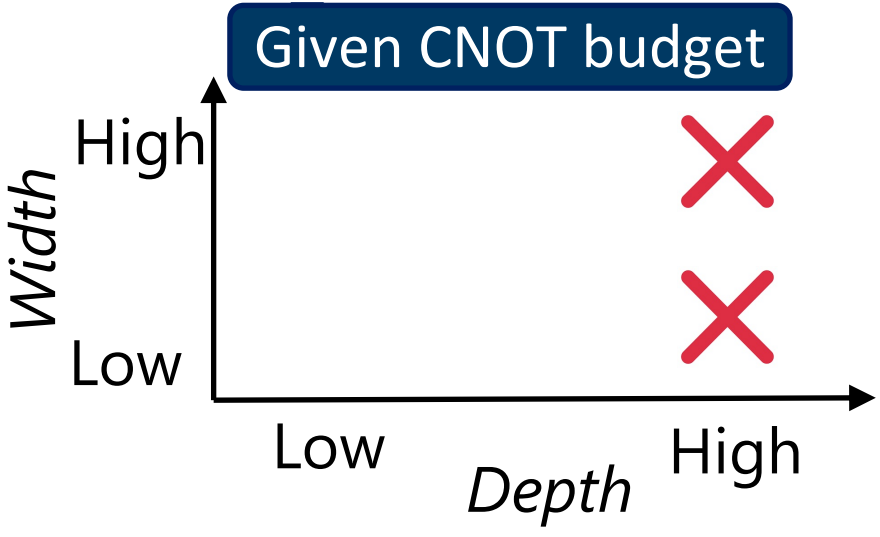


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

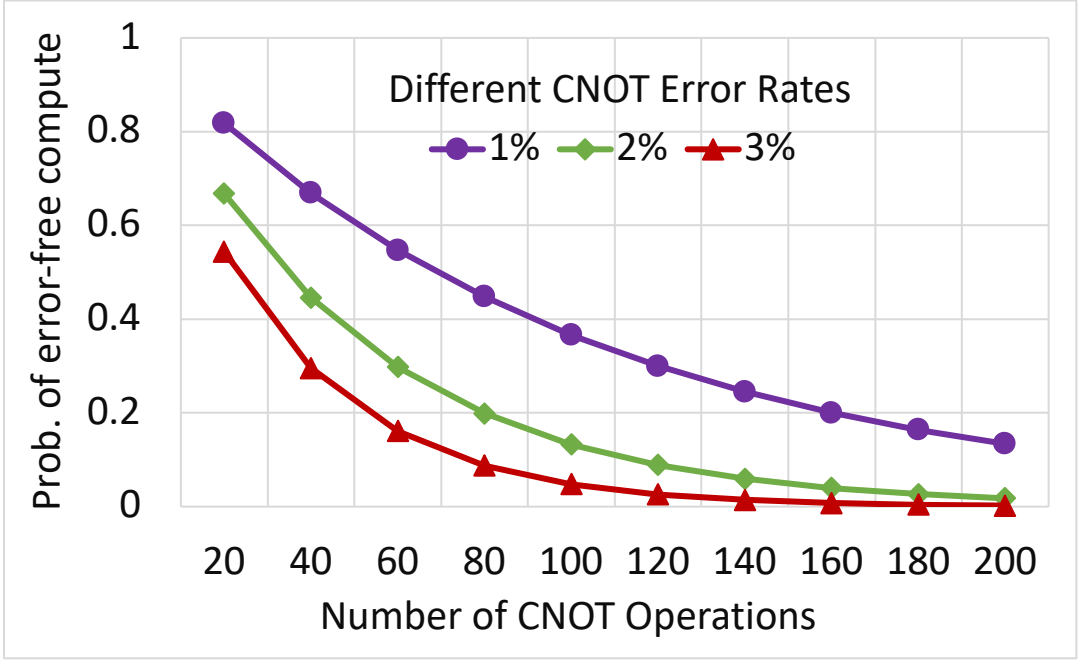


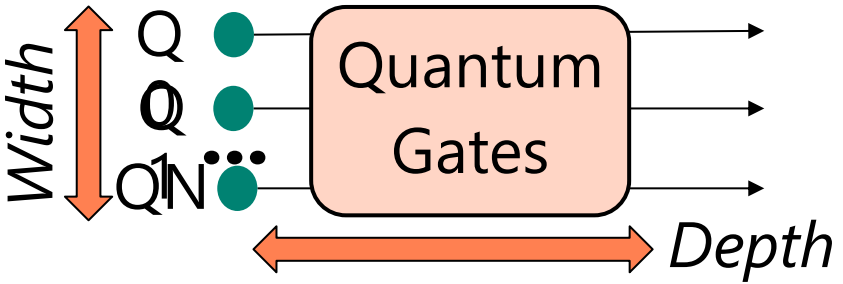


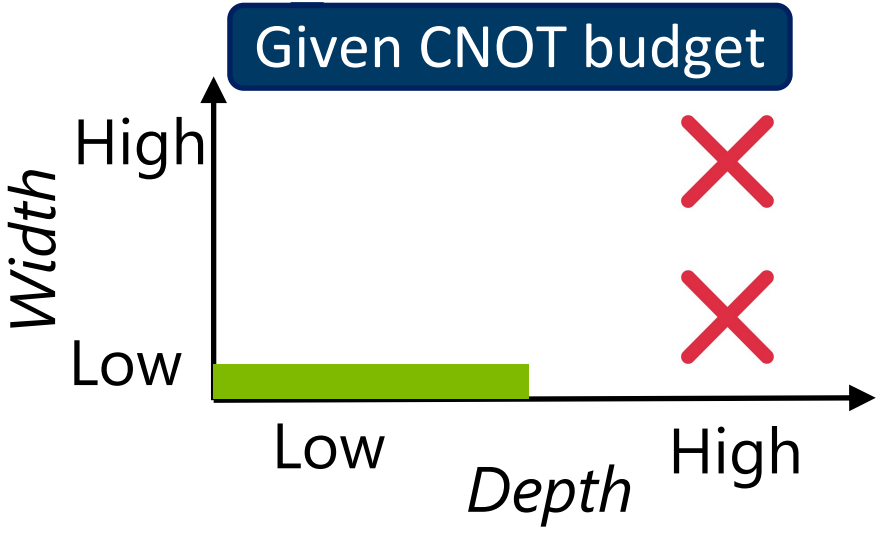


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

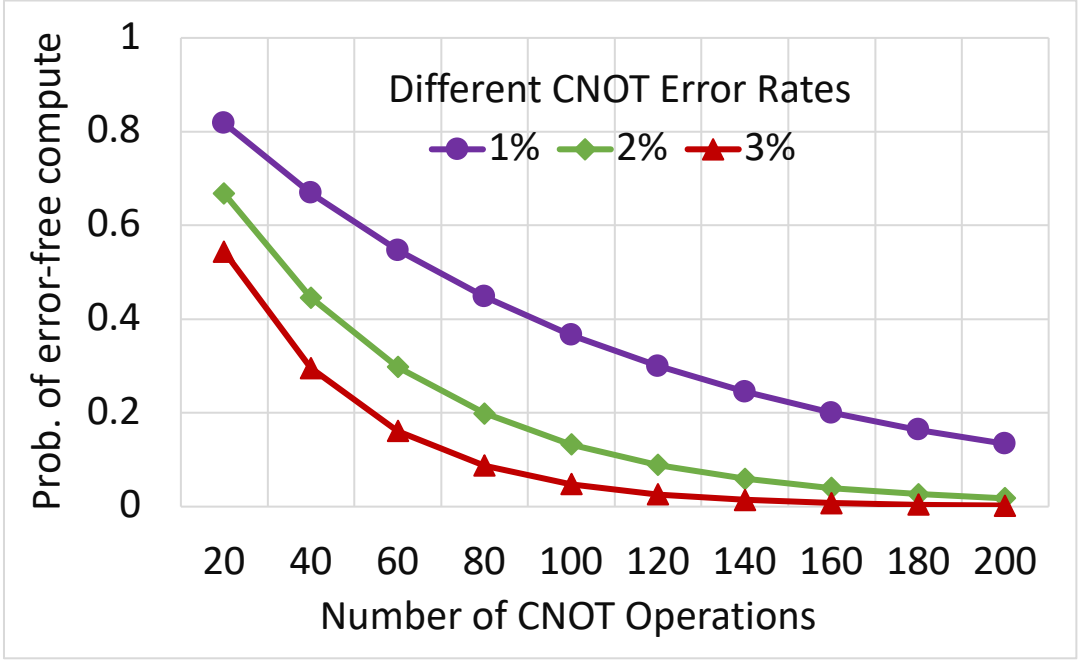


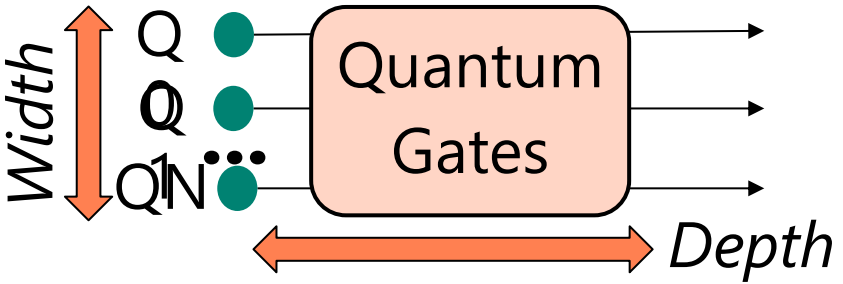


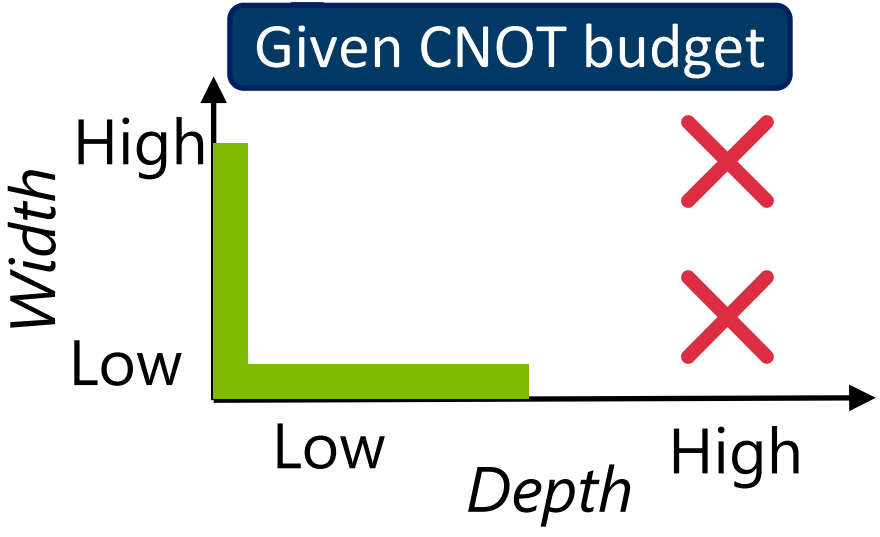


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

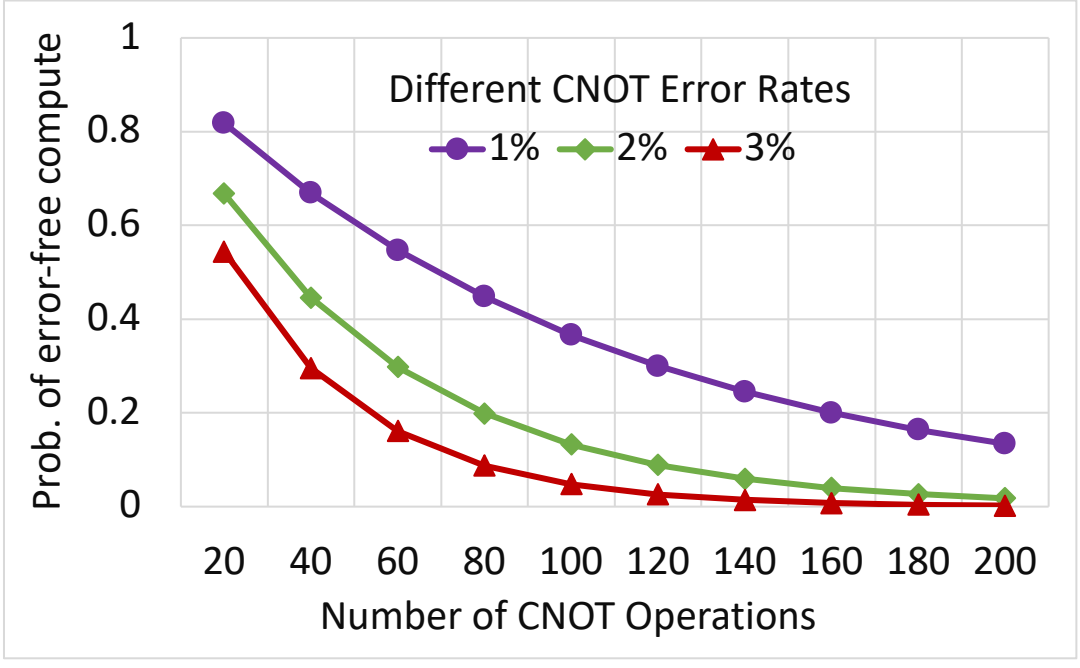


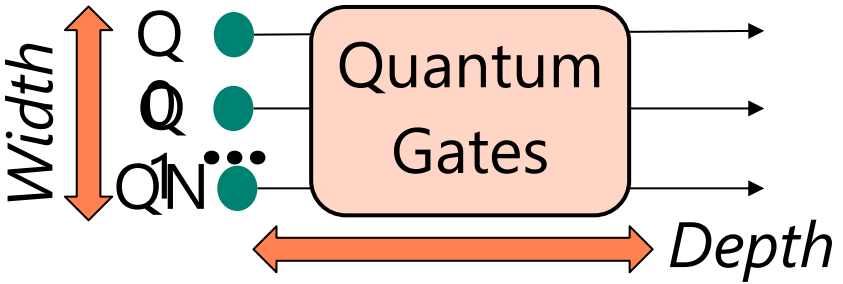


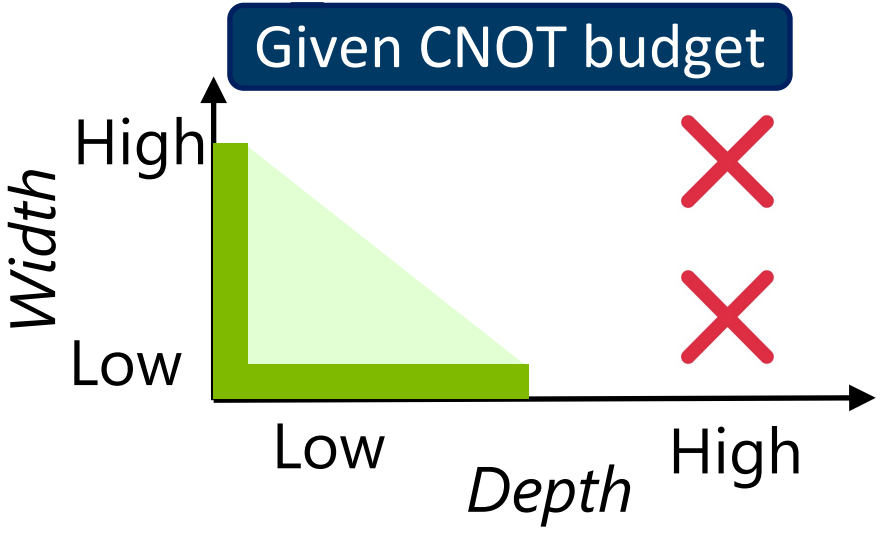


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

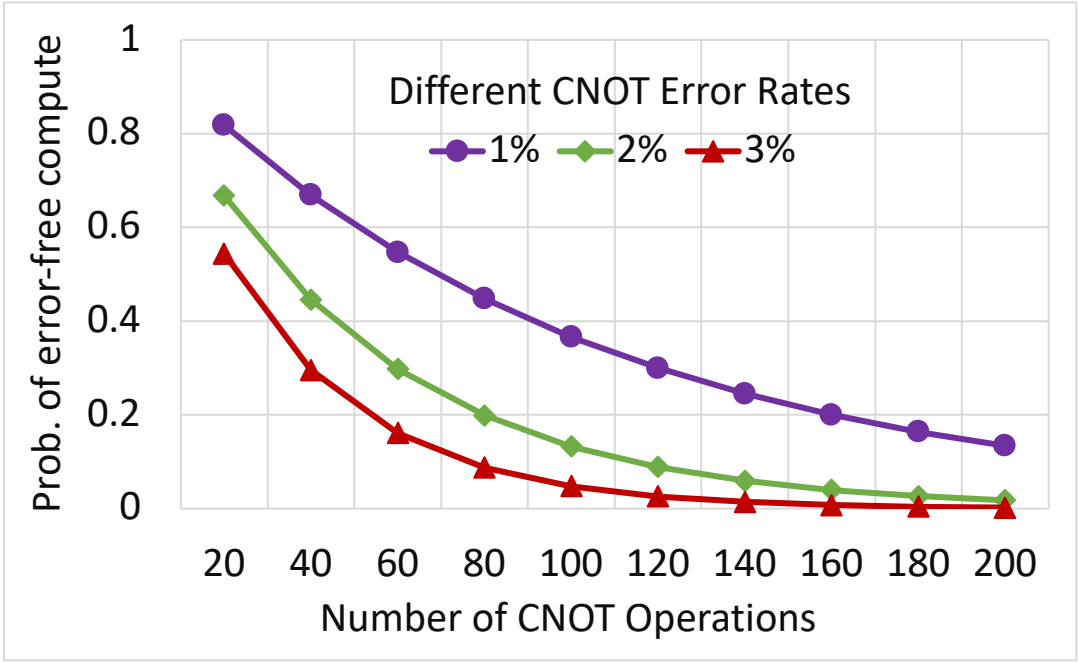


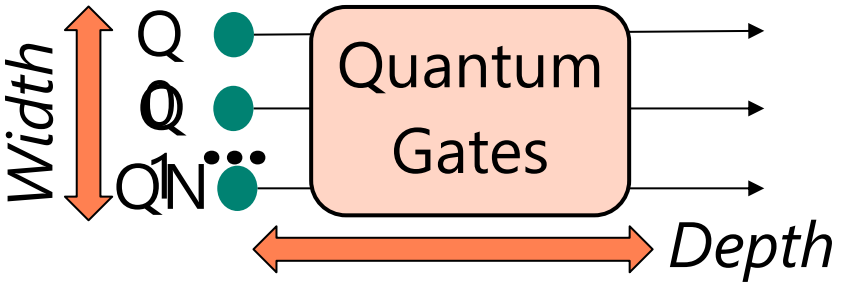


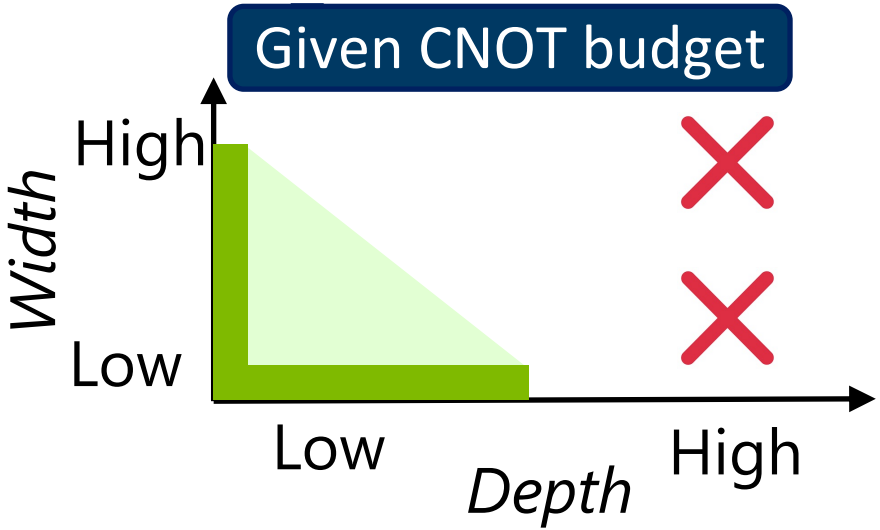


Underutilization in NISQ Computers

Error rates limit the number of reliable operations

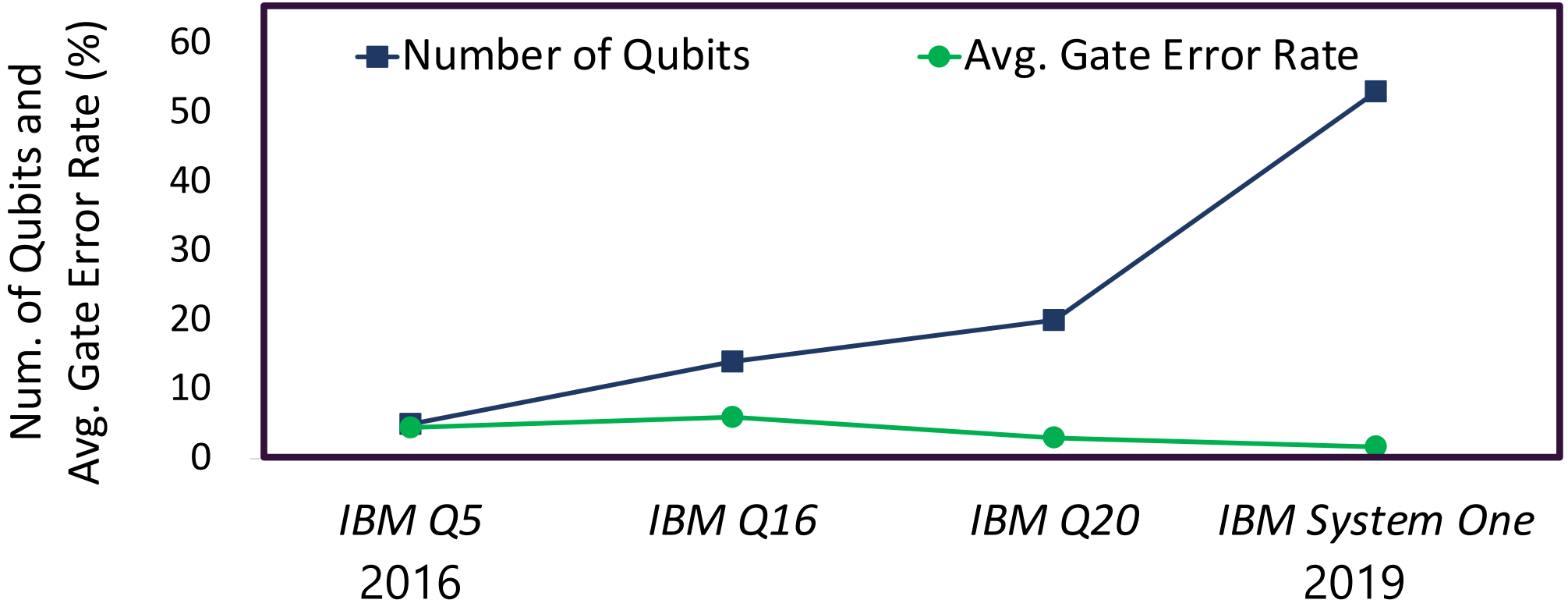




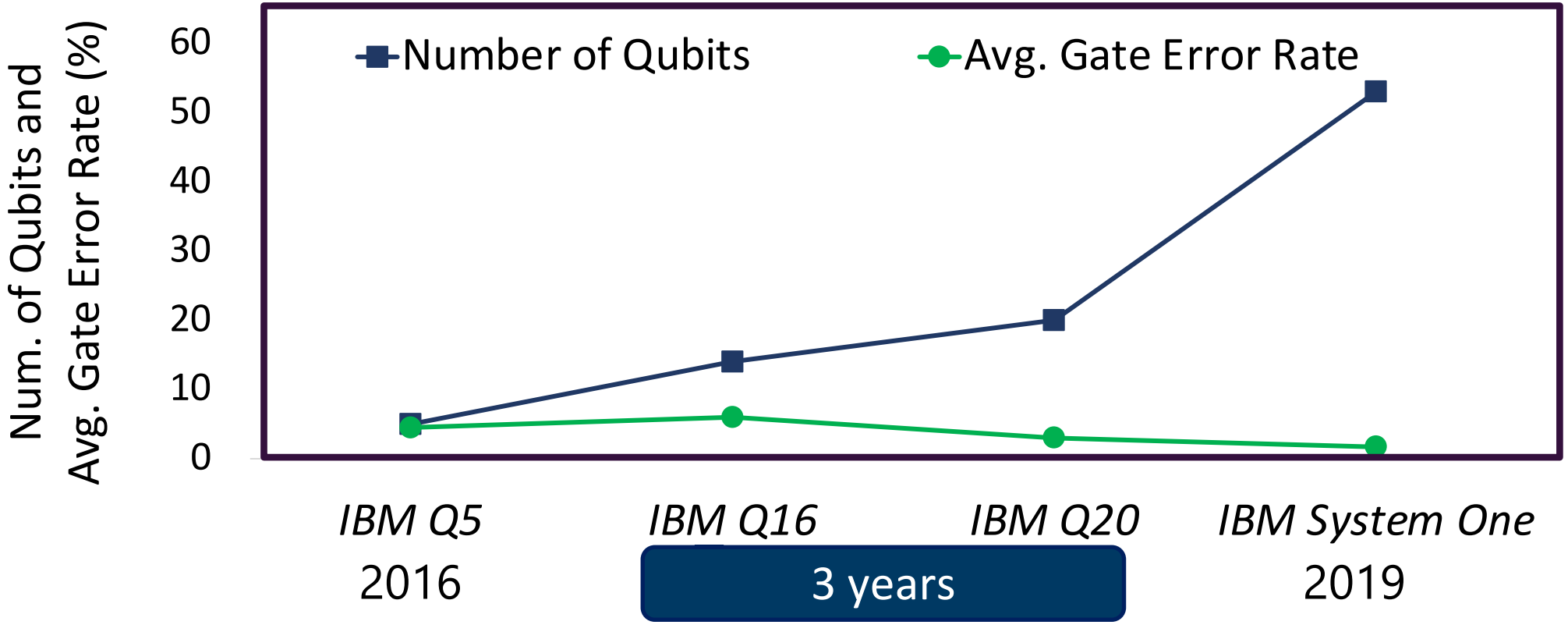


Programs may not use all the qubits leaving unused resources

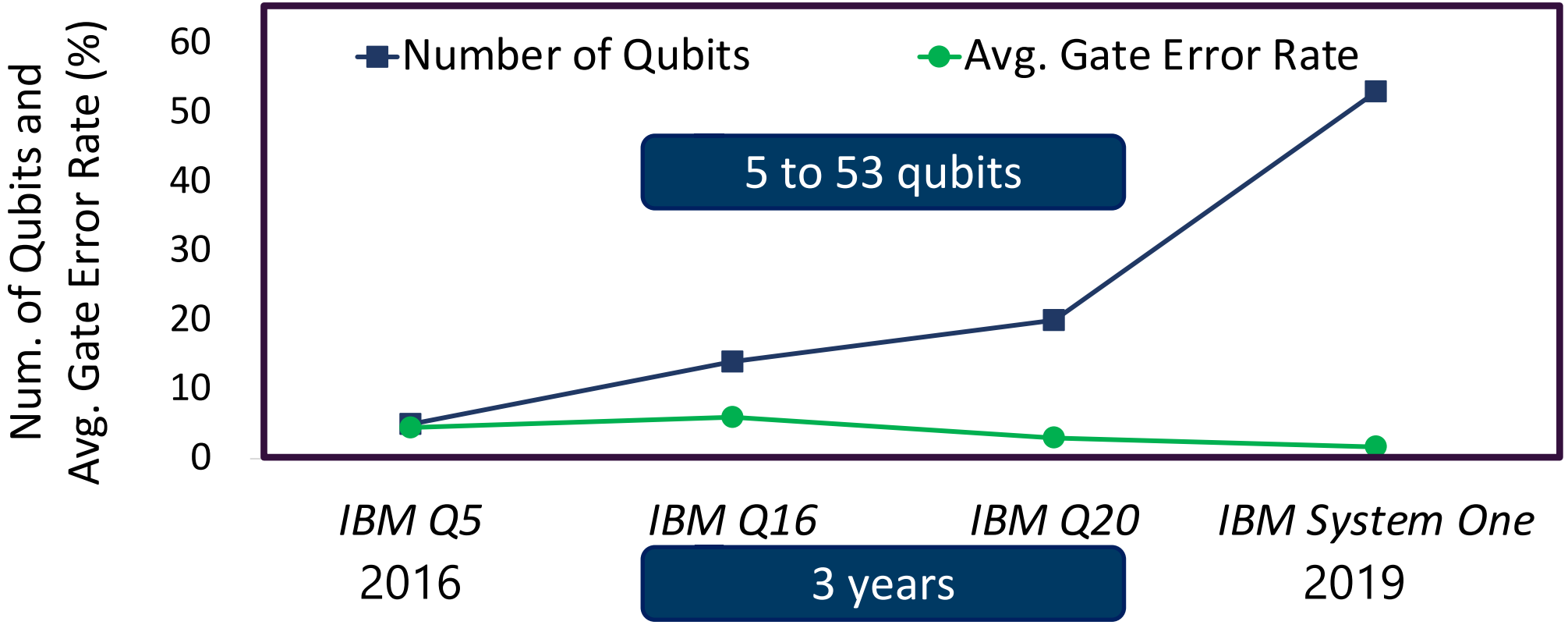
Quantity vs. Quality Gap



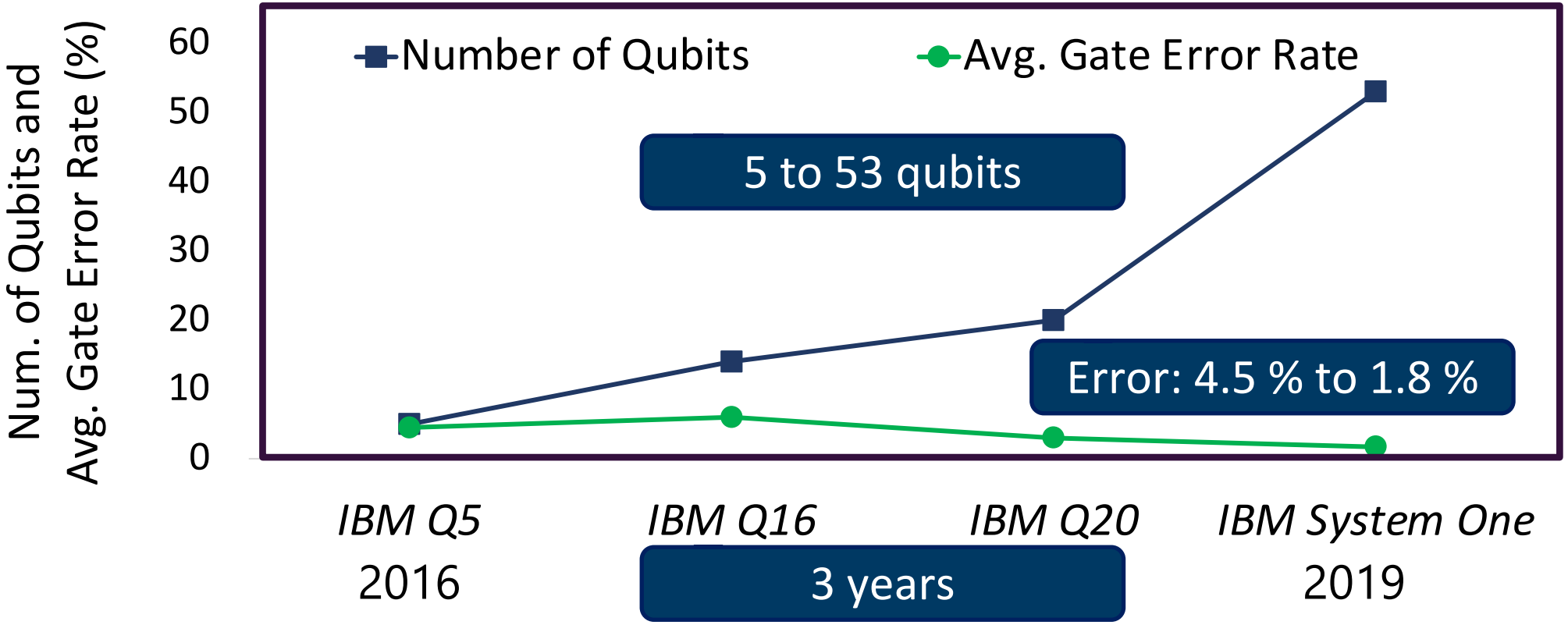
Quantity vs. Quality Gap



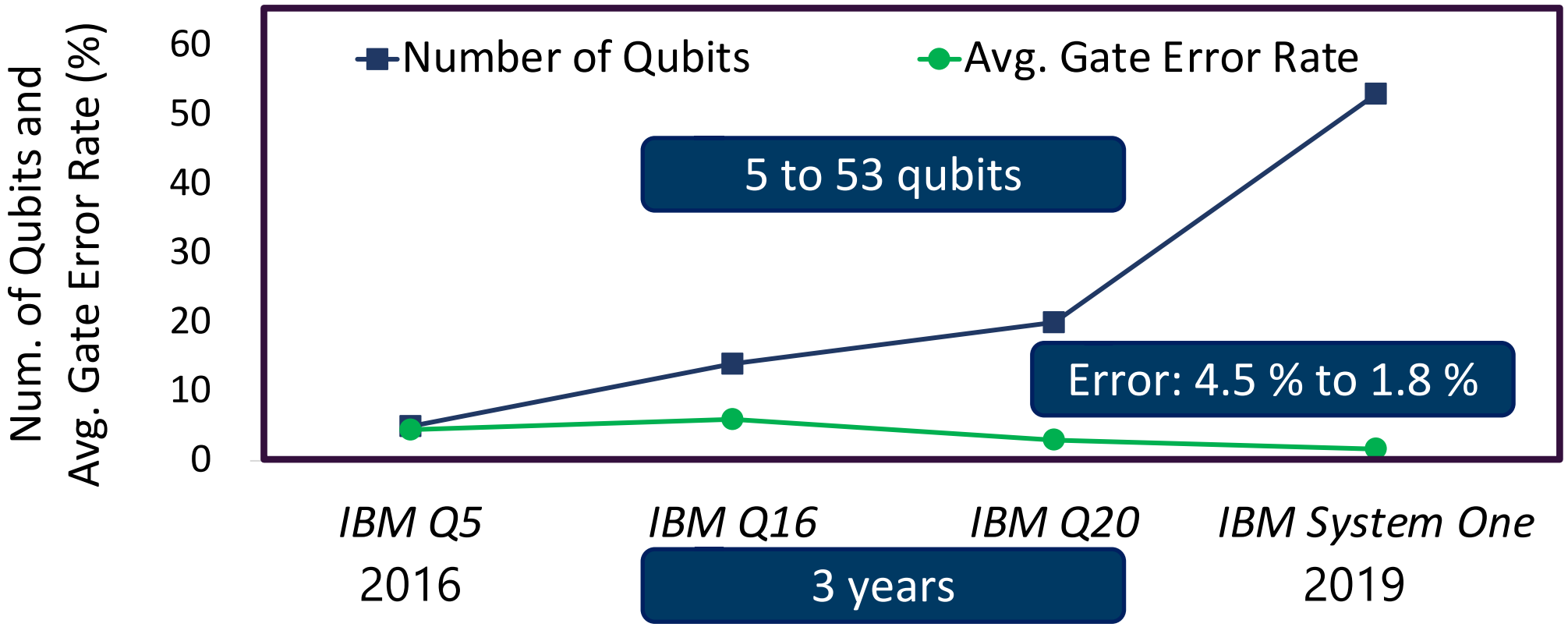
Quantity vs. Quality Gap



Quantity vs. Quality Gap



Quantity vs. Quality Gap



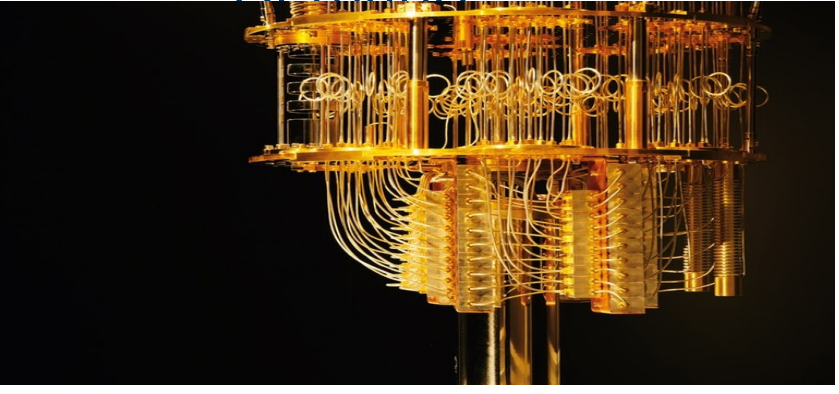
With current scaling of error rates, it is difficult to use all the qubits

Demand for Limited NISQ Resources

High
Demand

Demand for Limited NISQ Resources

High Demand

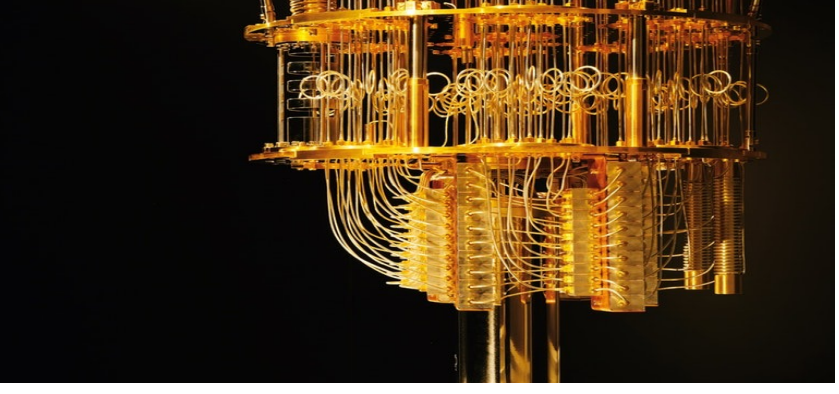


Expensive and Limited

Demand for Limited NISQ Resources

High Demand

Increasing Gap

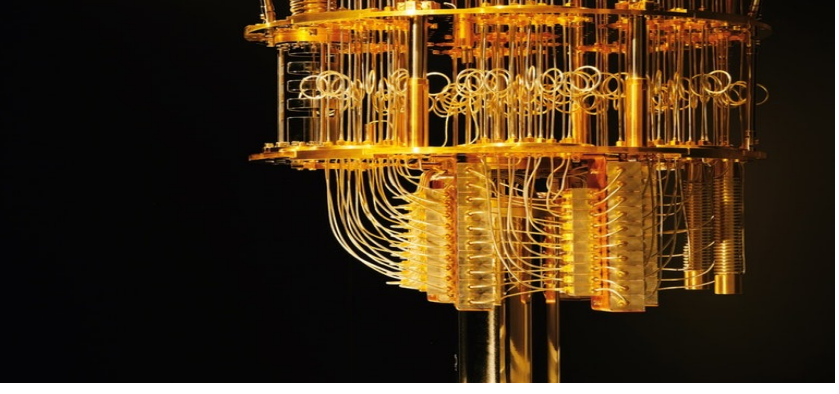


Expensive and Limited

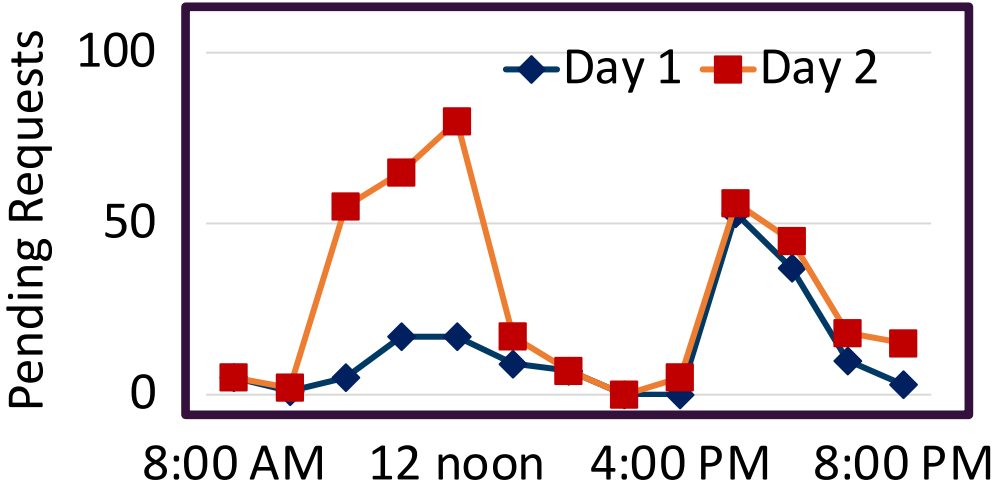
Demand for Limited NISQ Resources

High Demand

Increasing Gap



Expensive and Limited

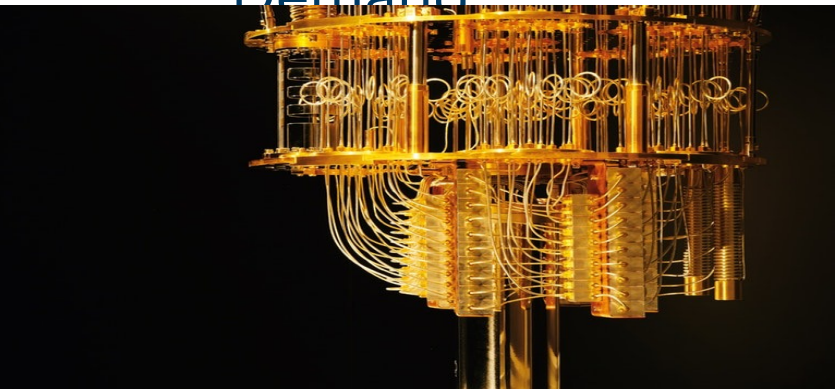


Pending Requests for access to IBM Q16 on different days

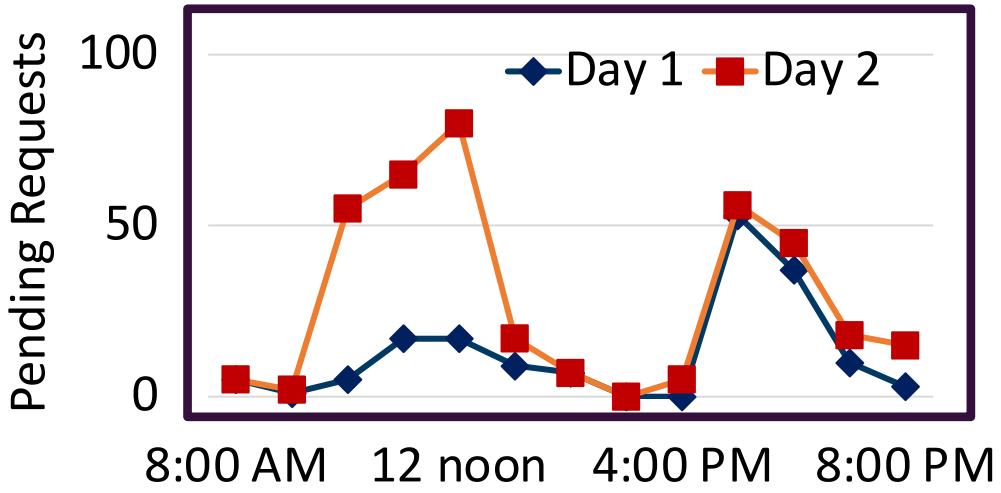
Demand for Limited NISQ Resources

High Demand

Increasing Gap



Expensive and Limited



Pending Requests for access to IBM Q16 on different days

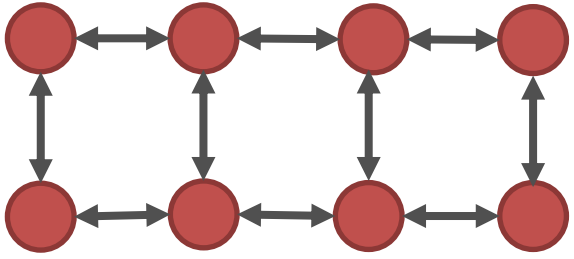
Limited NISQ resources must scale to a large number of users

Our Proposal: Multi-Program Quantum Computers

Multi-programming can improve throughput and utilization

Our Proposal: Multi-Program Quantum Computers

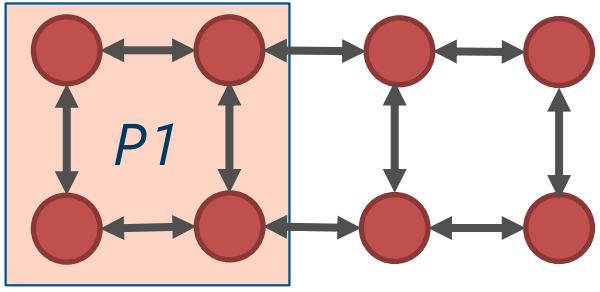
Multi-programming can improve throughput and utilization



Our Proposal: Multi-Program Quantum Computers

Multi-programming can improve throughput and utilization

time = T1

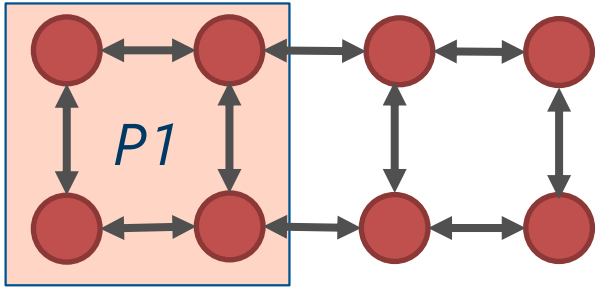


Current Approach

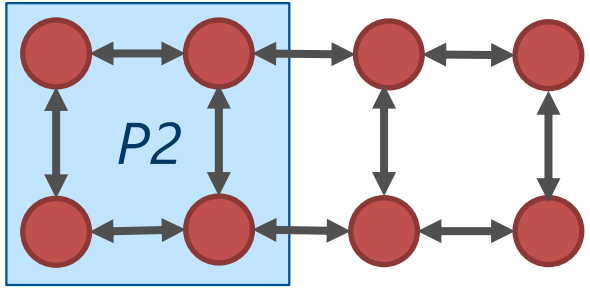
Our Proposal: Multi-Program Quantum Computers

Multi-programming can improve throughput and utilization

time = T1



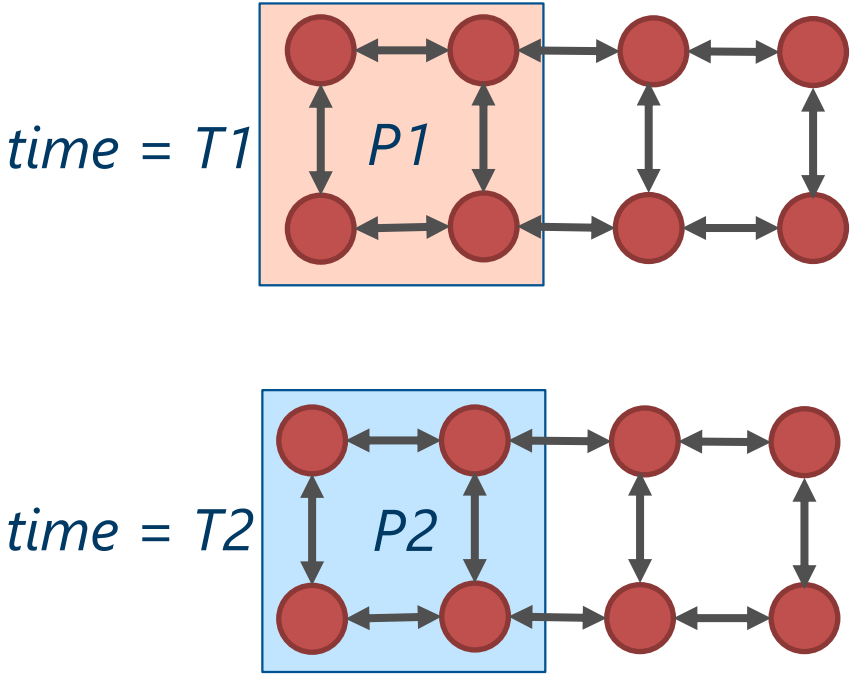
time = T2



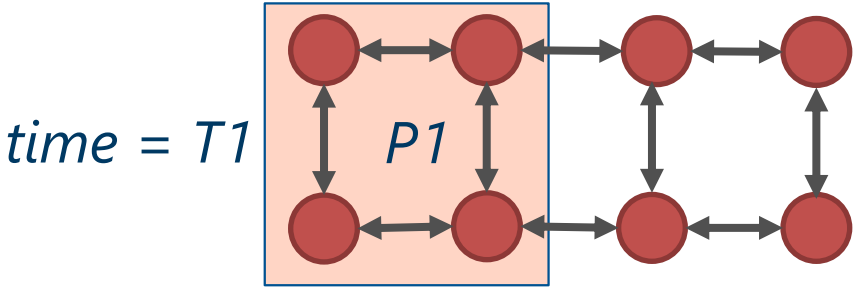
Current Approach

Our Proposal: Multi-Program Quantum Computers

Multi-programming can improve throughput and utilization



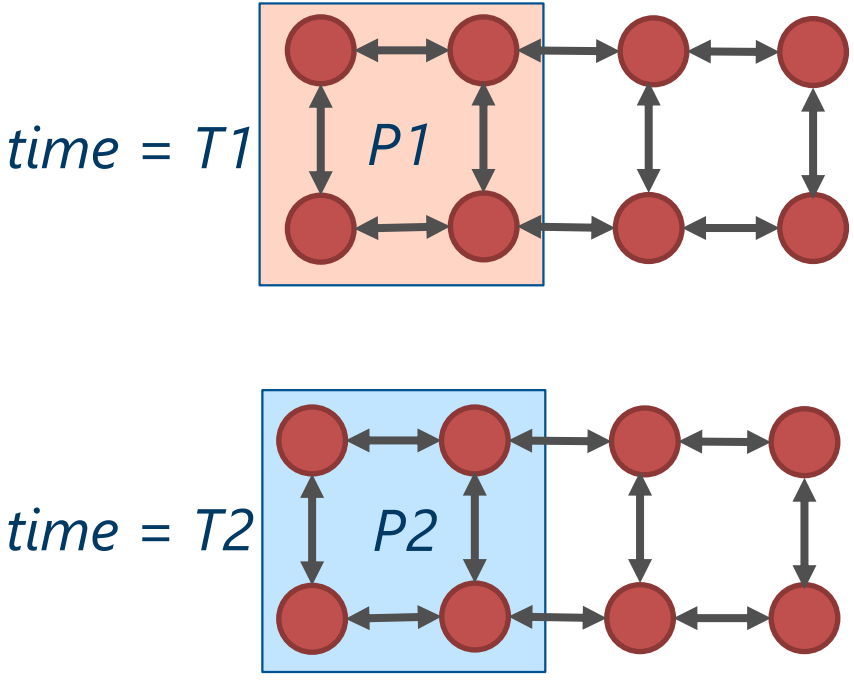
Current Approach



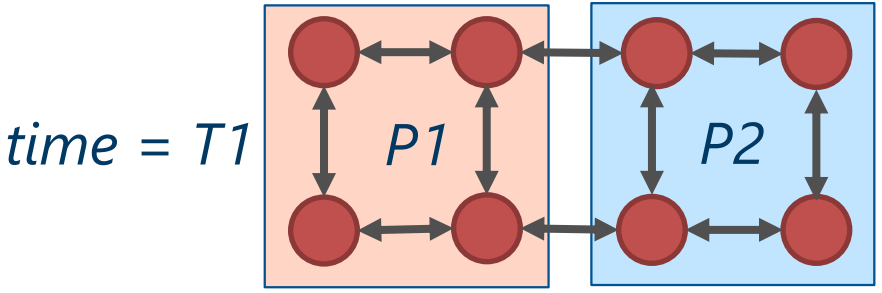
Proposed Approach

Our Proposal: Multi-Program Quantum Computers

Multi-programming can improve throughput and utilization

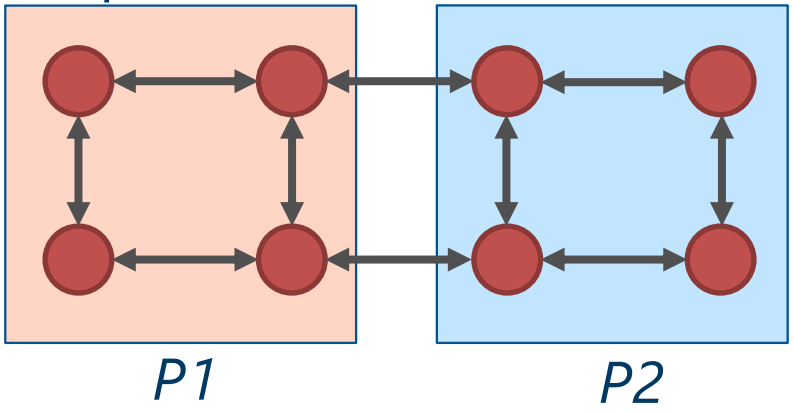


Current Approach

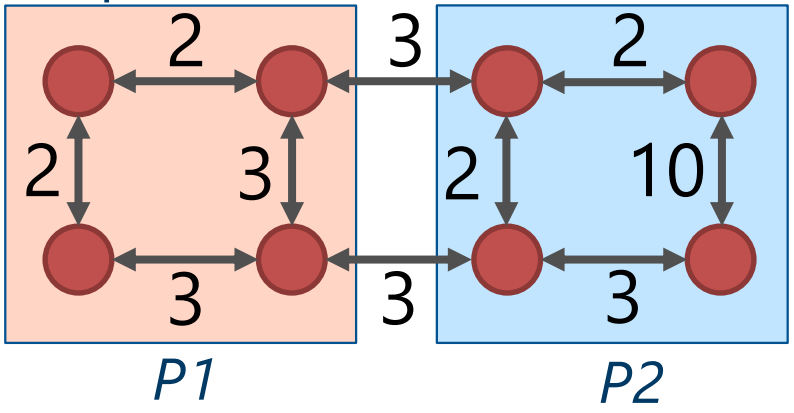


Proposed Approach

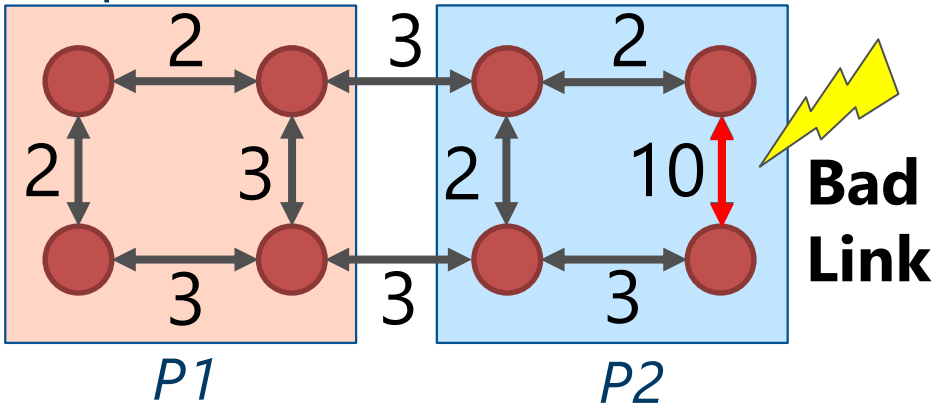
Challenges in Multi-Programming NISQ Computers



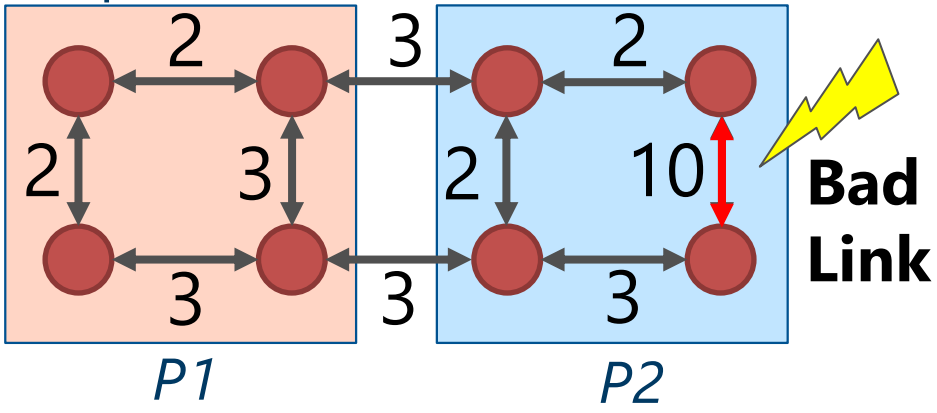
Challenges in Multi-Programming NISQ Computers



Challenges in Multi-Programming NISQ Computers

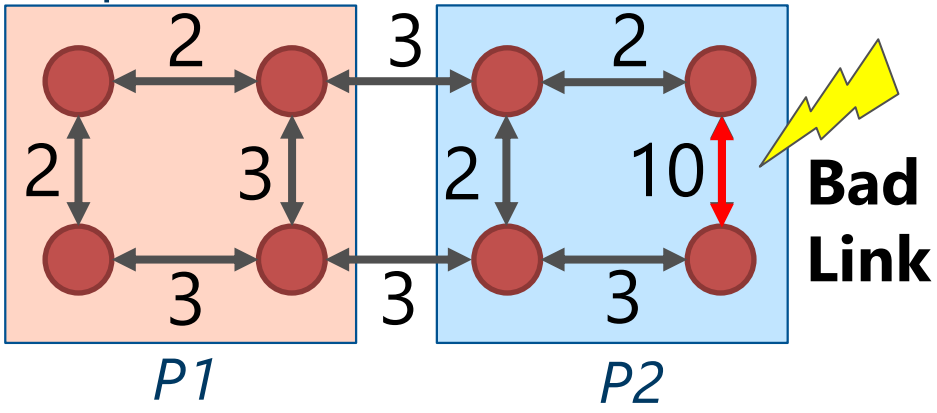


Challenges in Multi-Programming NISQ Computers



Correctness and Reliability Issue!

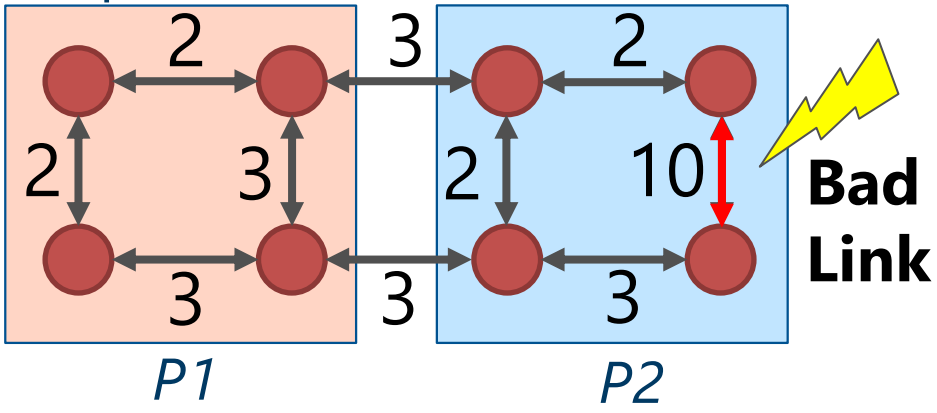
Challenges in Multi-Programming NISQ Computers



Correctness and Reliability Issue!

- Fairness in resource allocation

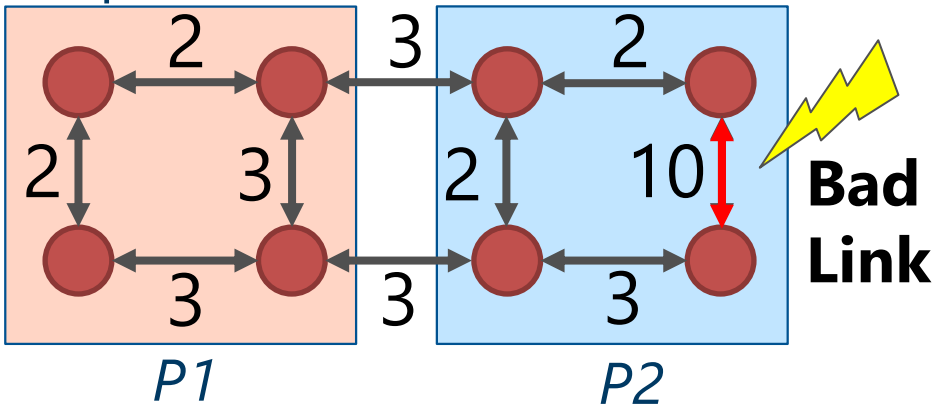
Challenges in Multi-Programming NISQ Computers



Correctness and Reliability Issue!

- Fairness in resource allocation
- Reduce interference

Challenges in Multi-Programming NISQ Computers



Correctness and Reliability Issue!

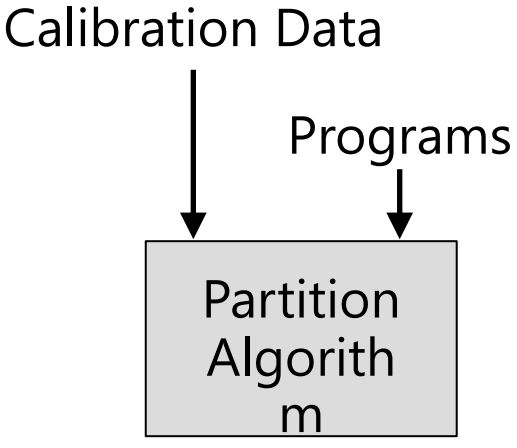
- Fairness in resource allocation
- Reduce interference

Our goal is to enable multi-programming to improve the throughput and utilization of quantum computers while minimizing the impact on reliability

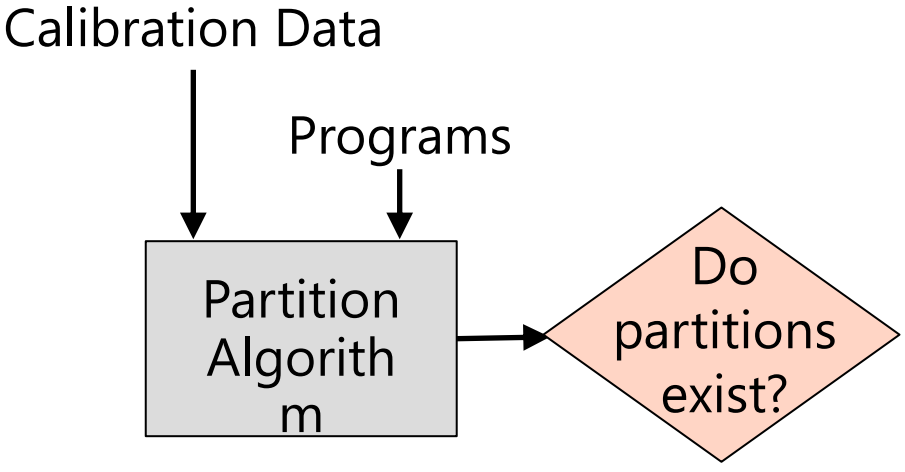
Outline

- Introduction
- Background and Motivation
- Policies for Multi-Programming
- Evaluation Methodology
- Adaptive Multi-Programming Design
- Results and Conclusion

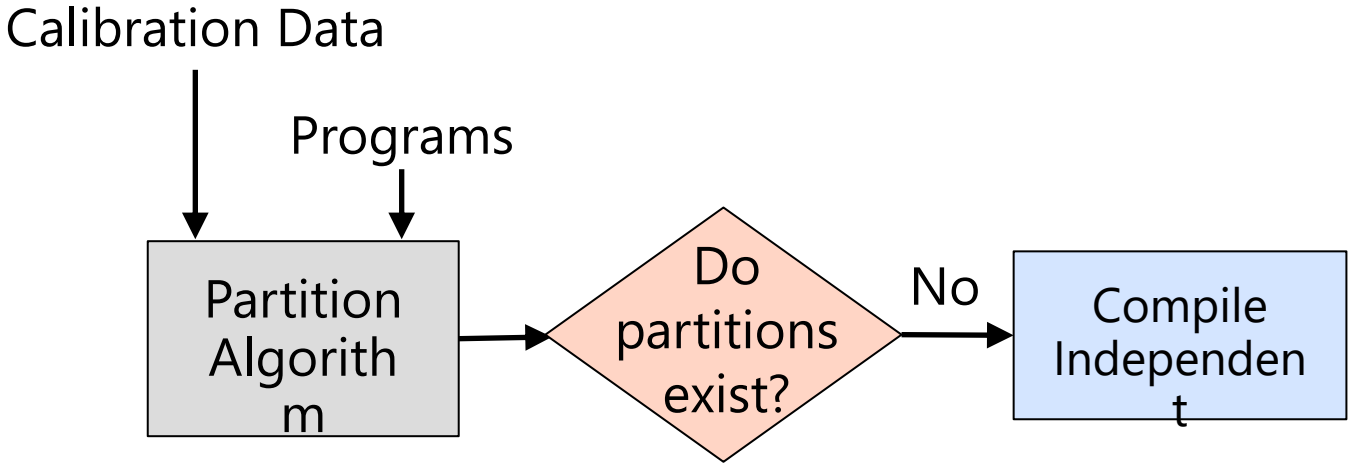
Overview Of Our Proposals



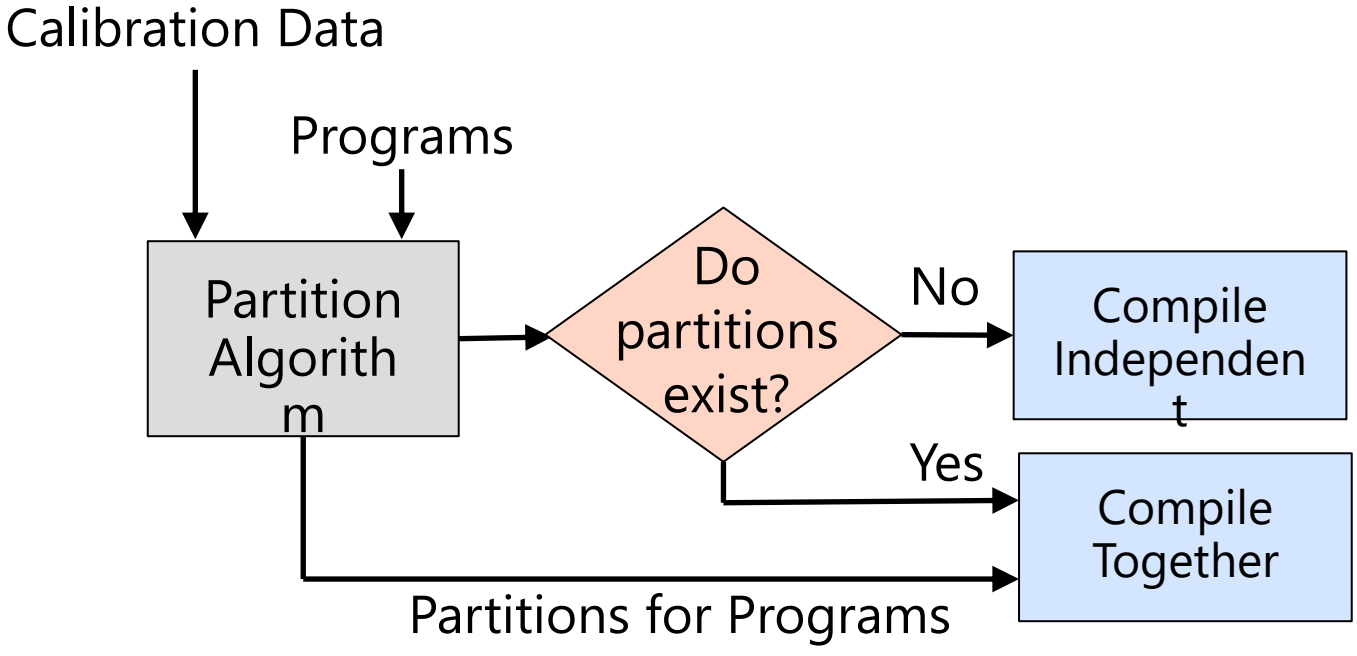
Overview Of Our Proposals



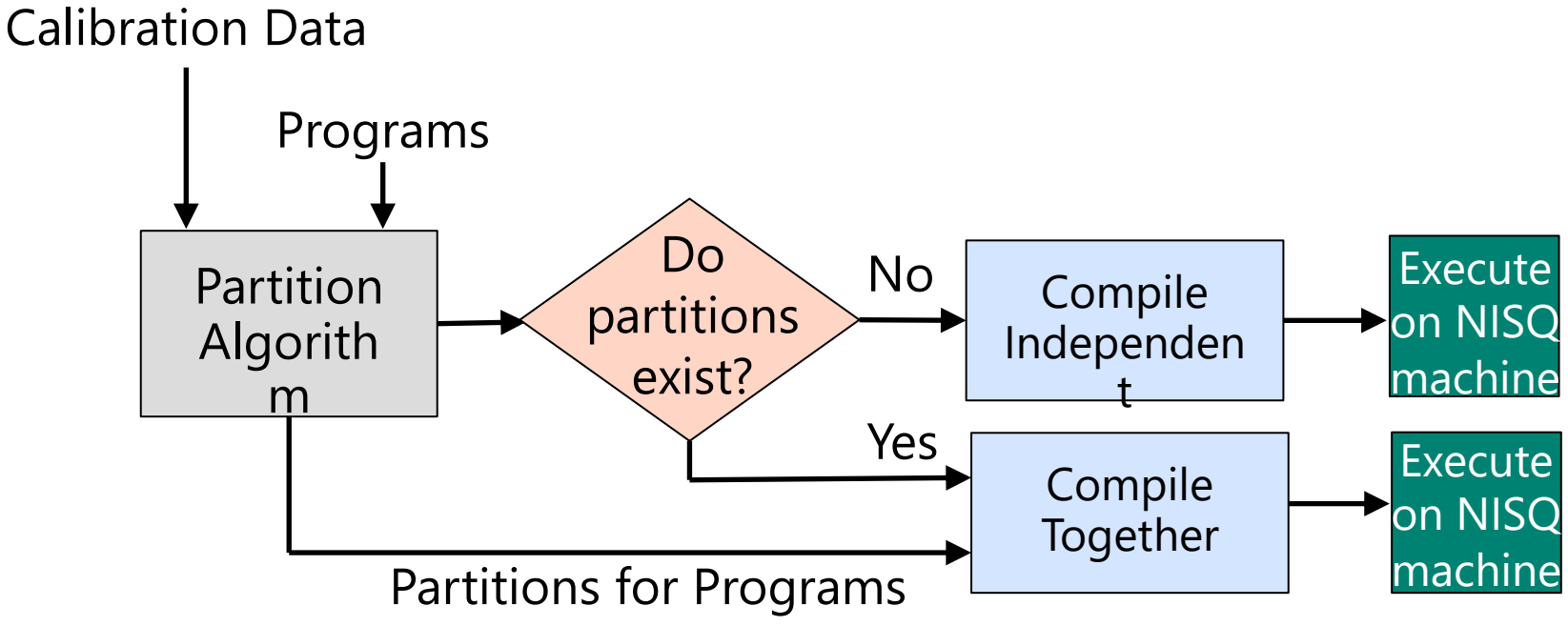
Overview Of Our Proposals



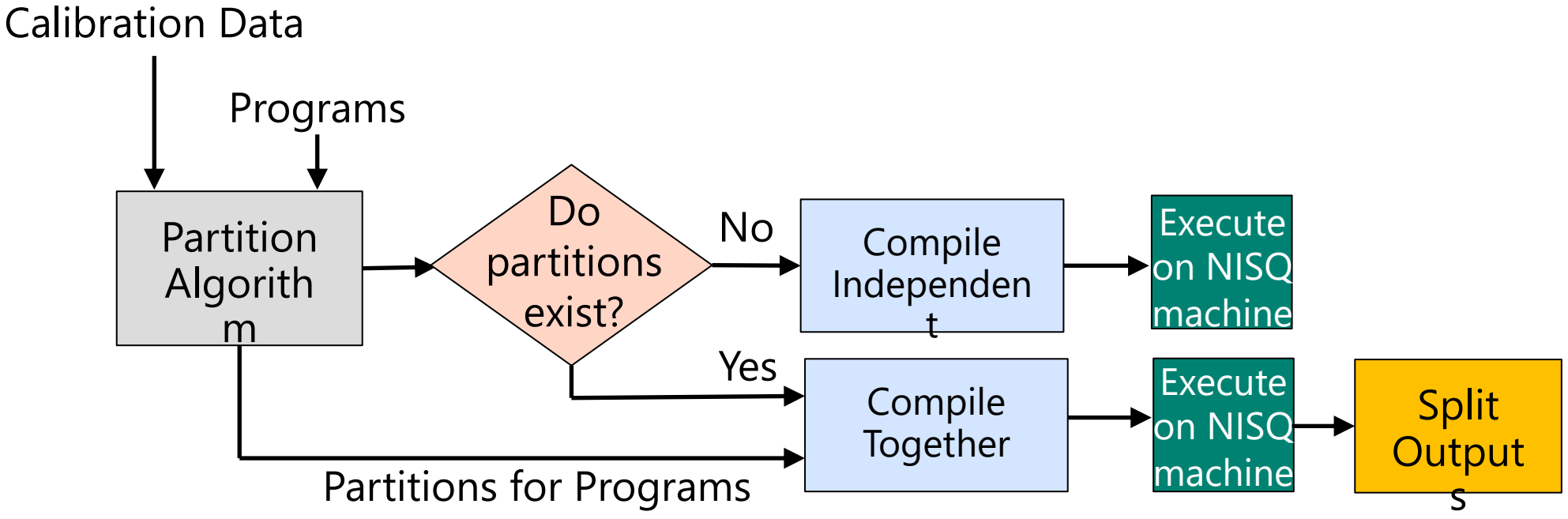
Overview Of Our Proposals



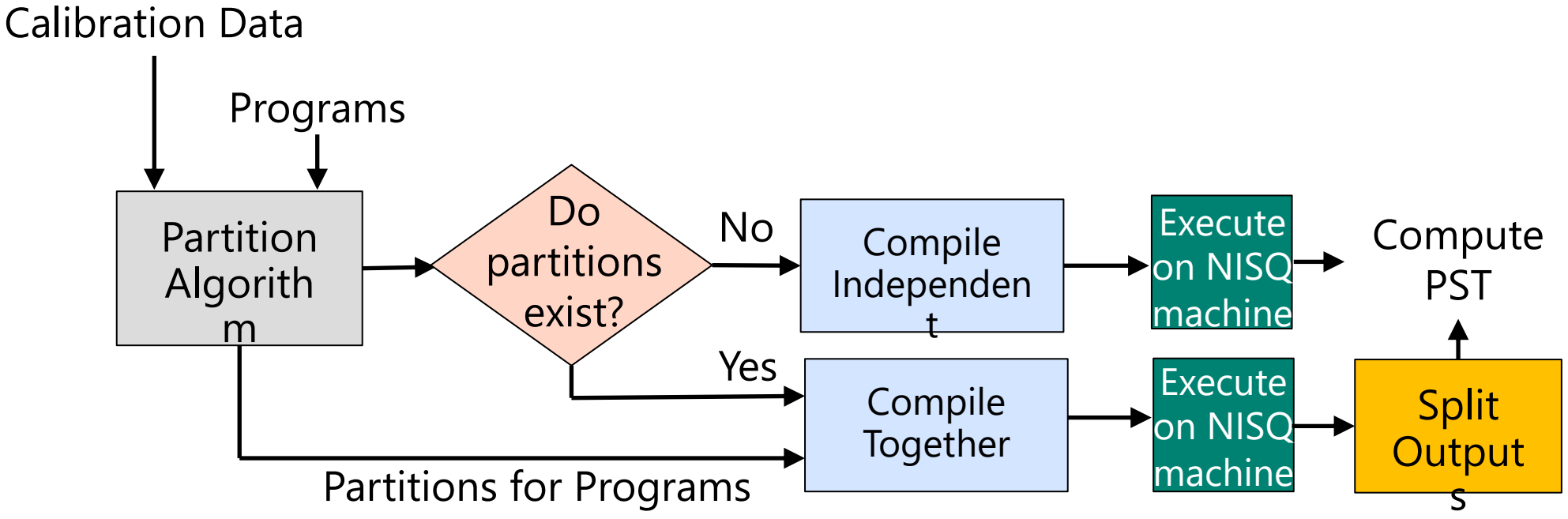
Overview Of Our Proposals



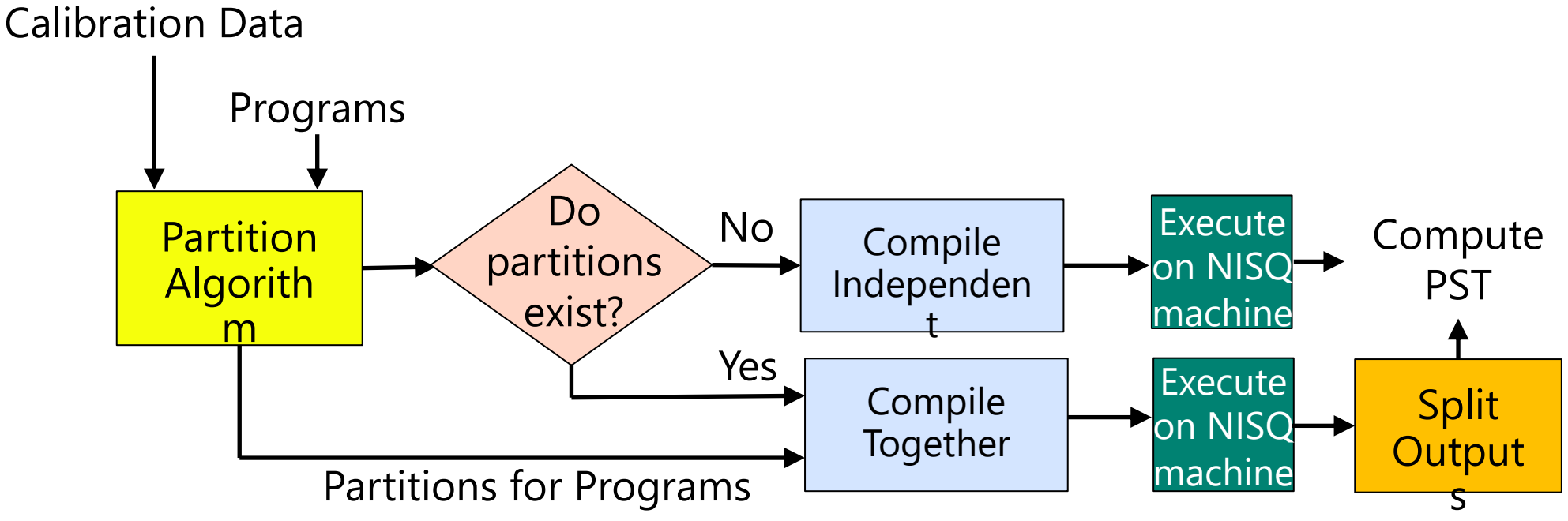
Overview Of Our Proposals



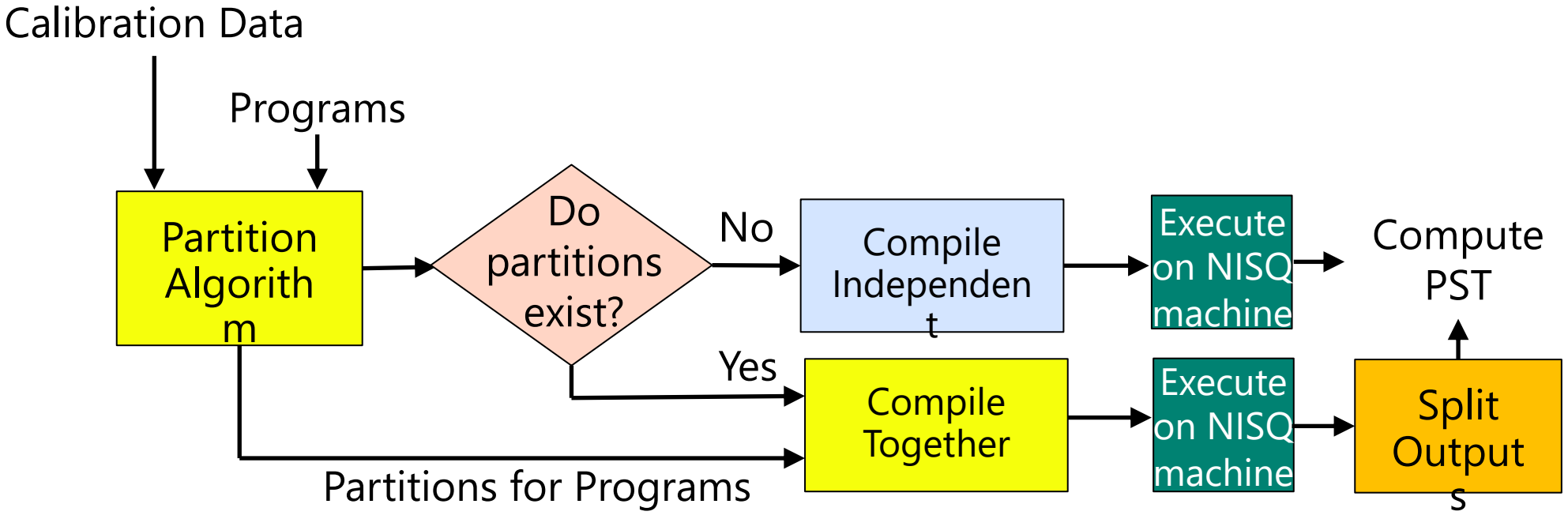
Overview Of Our Proposals



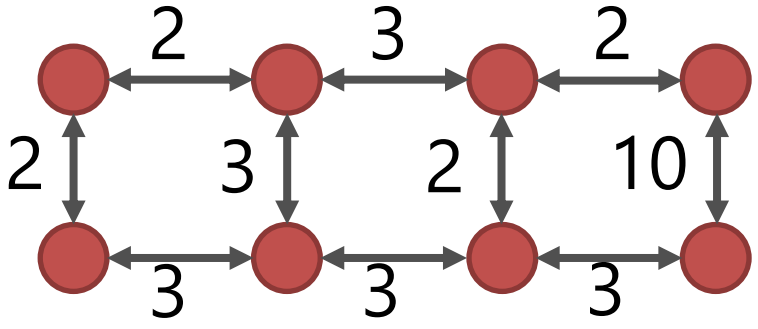
Overview Of Our Proposals



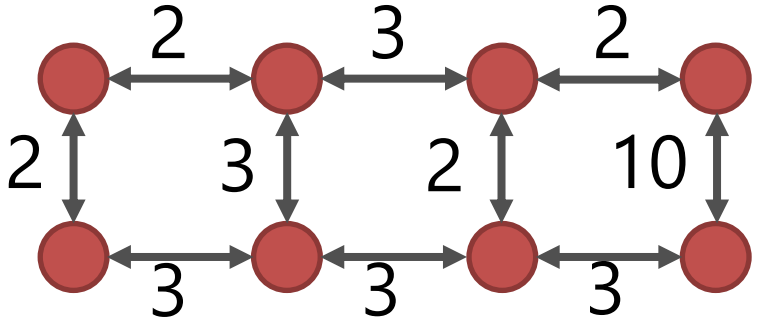
Overview Of Our Proposals



Fair And Reliable Partitioning *(FRP)* Algorithm

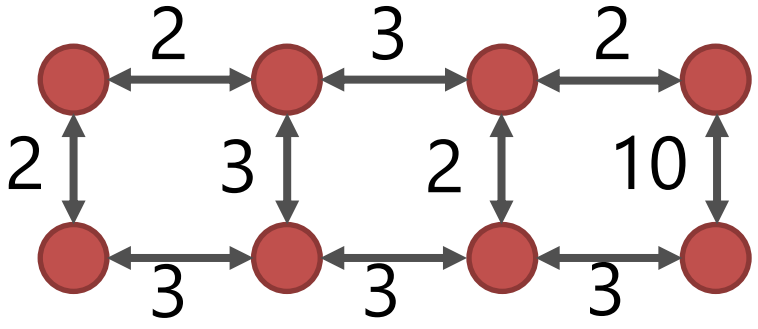


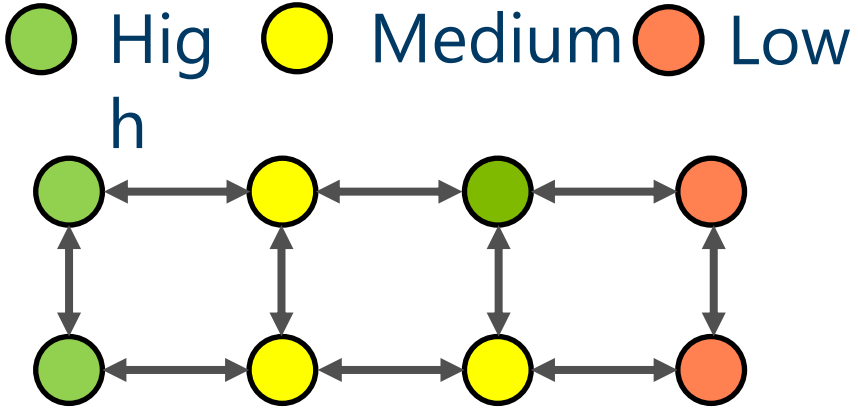
Fair And Reliable Partitioning *(FRP)* Algorithm



$$Utility = \frac{Number\ of\ links}{\sum Link\ error\ rates}$$

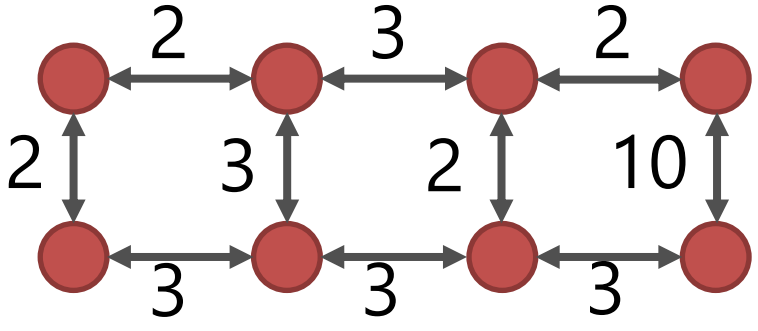
Fair And Reliable Partitioning *(FRP)* Algorithm



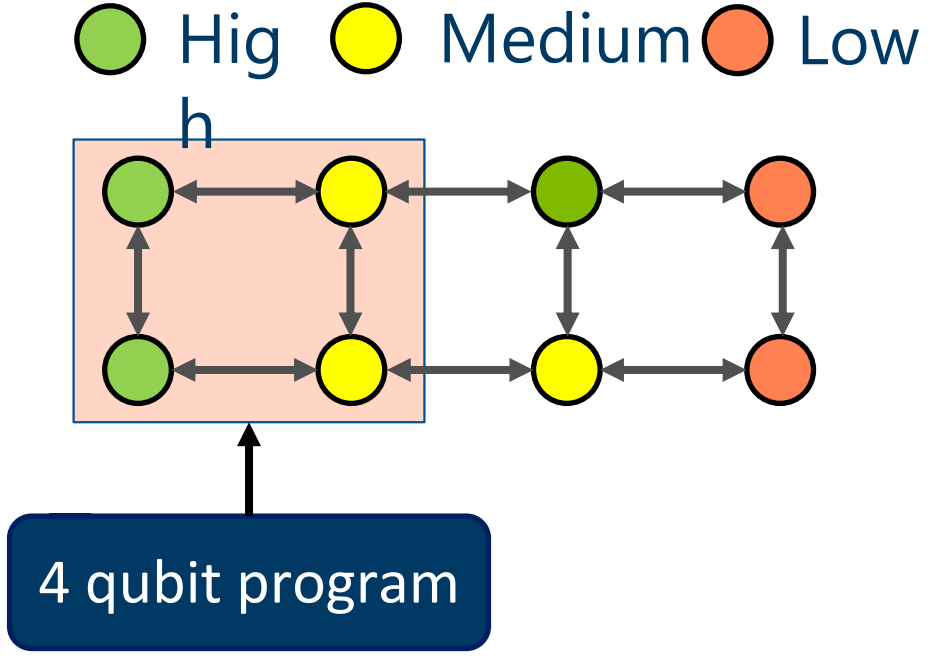


$$Utility = \frac{Number\ of\ links}{\sum Link\ error\ rates}$$

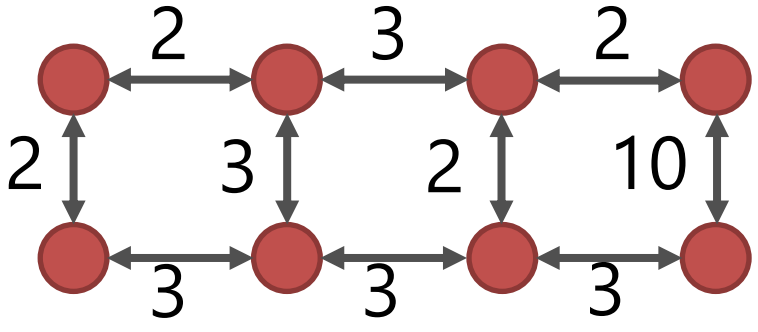
Fair And Reliable Partitioning *(FRP)* Algorithm



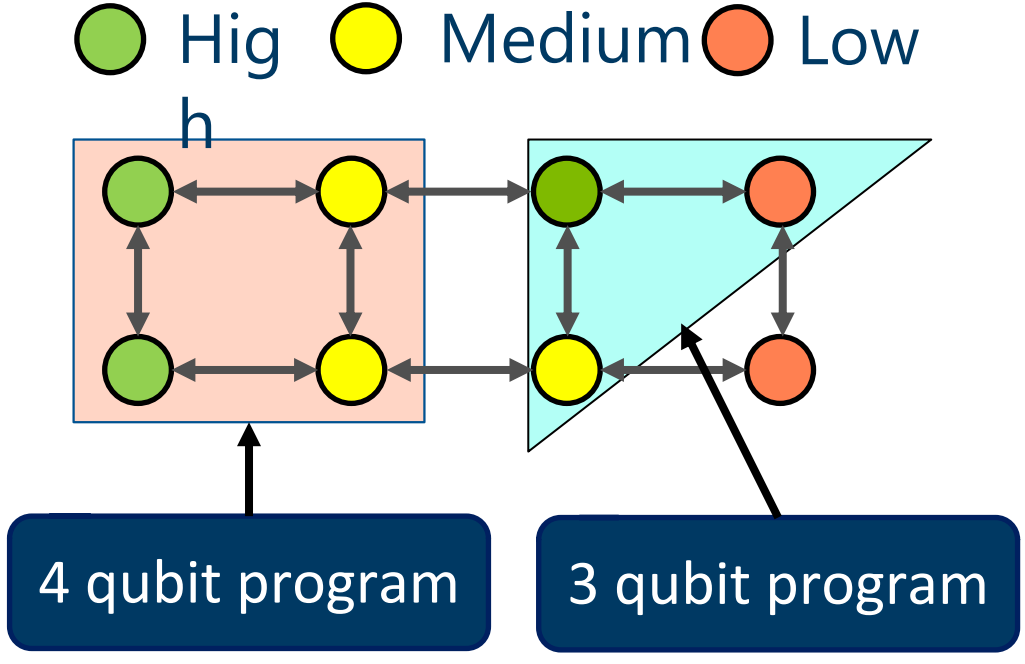
$$Utility = \frac{Number\ of\ links}{\sum Link\ error\ rates}$$



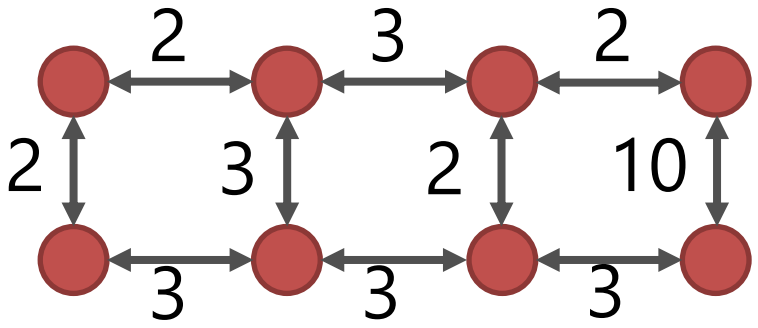
Fair And Reliable Partitioning *(FRP)* Algorithm



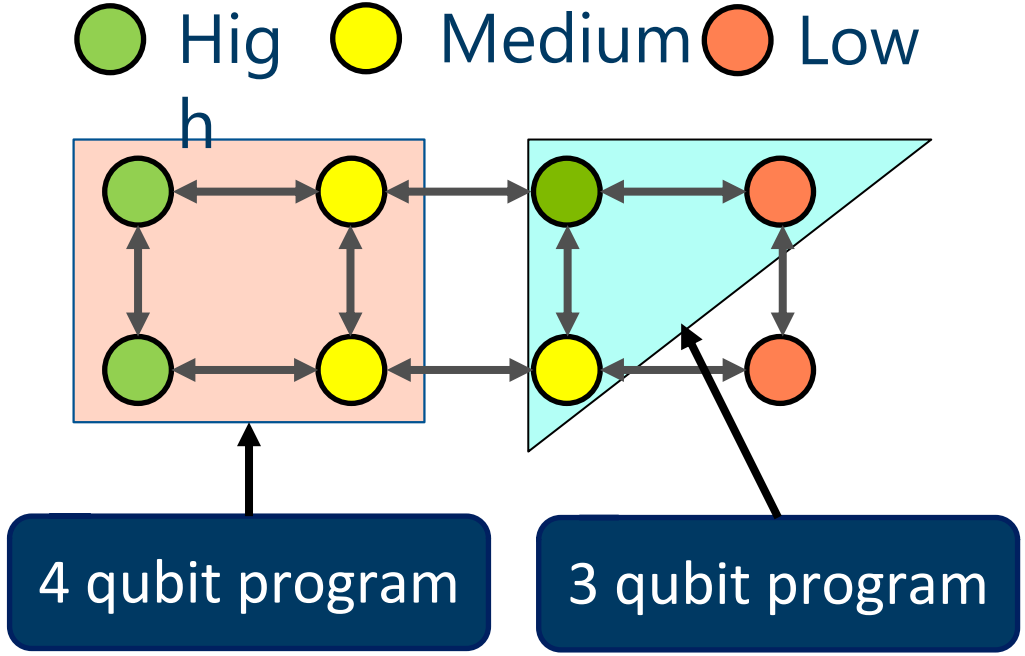
$$Utility = \frac{Number\ of\ links}{\sum Link\ error\ rates}$$



Fair And Reliable Partitioning *(FRP)* Algorithm



$$Utility = \frac{Number\ of\ links}{\sum Link\ error\ rates}$$



Algorithm ensures each program is allocated reliable qubits

Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?

Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?

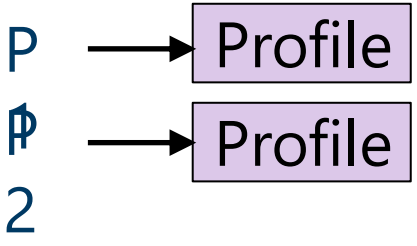
P

₱

2

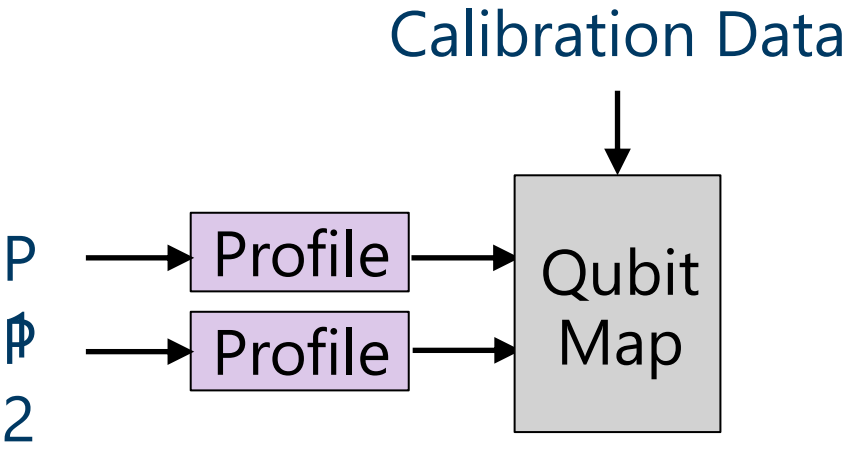
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



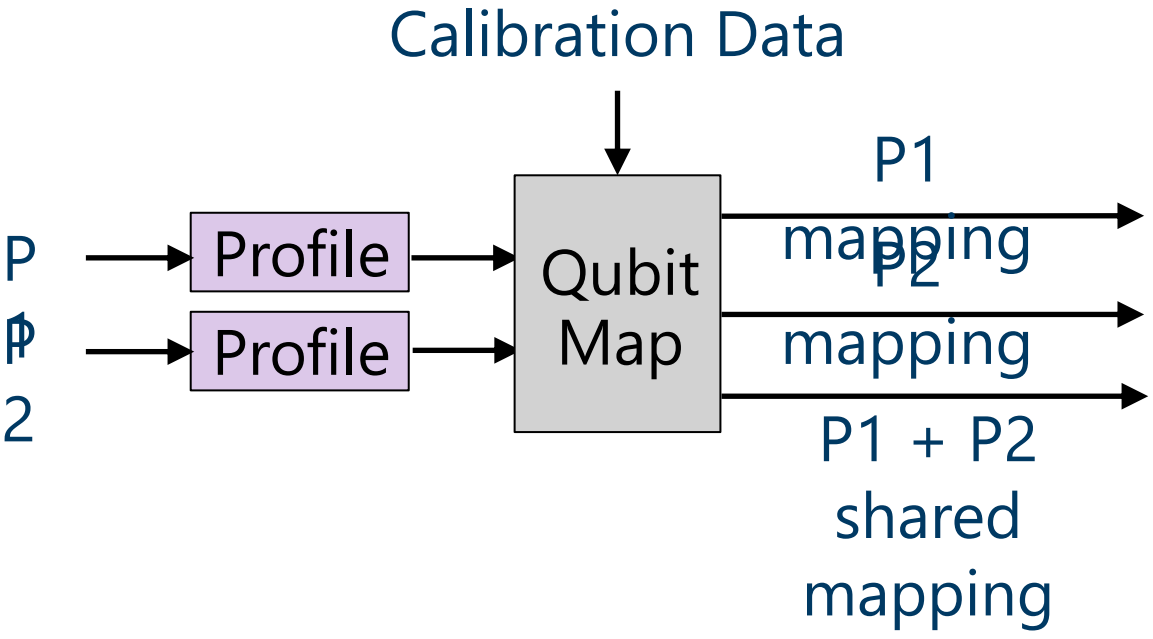
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



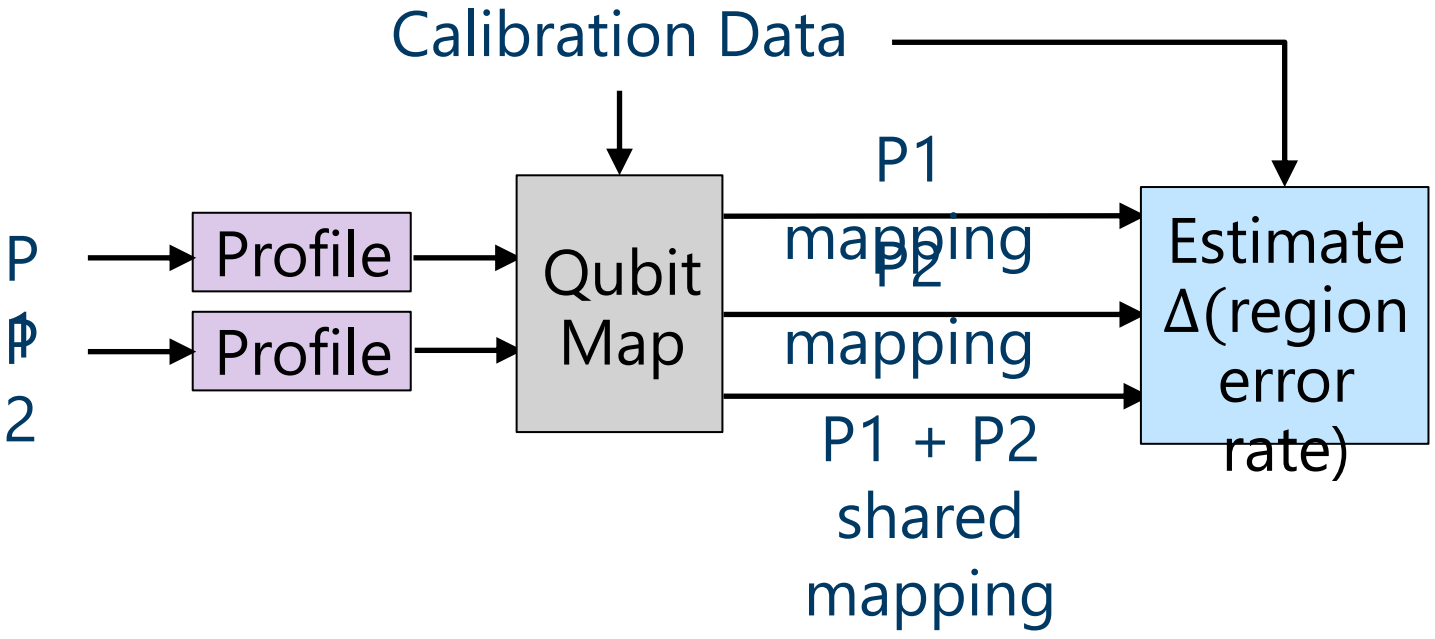
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



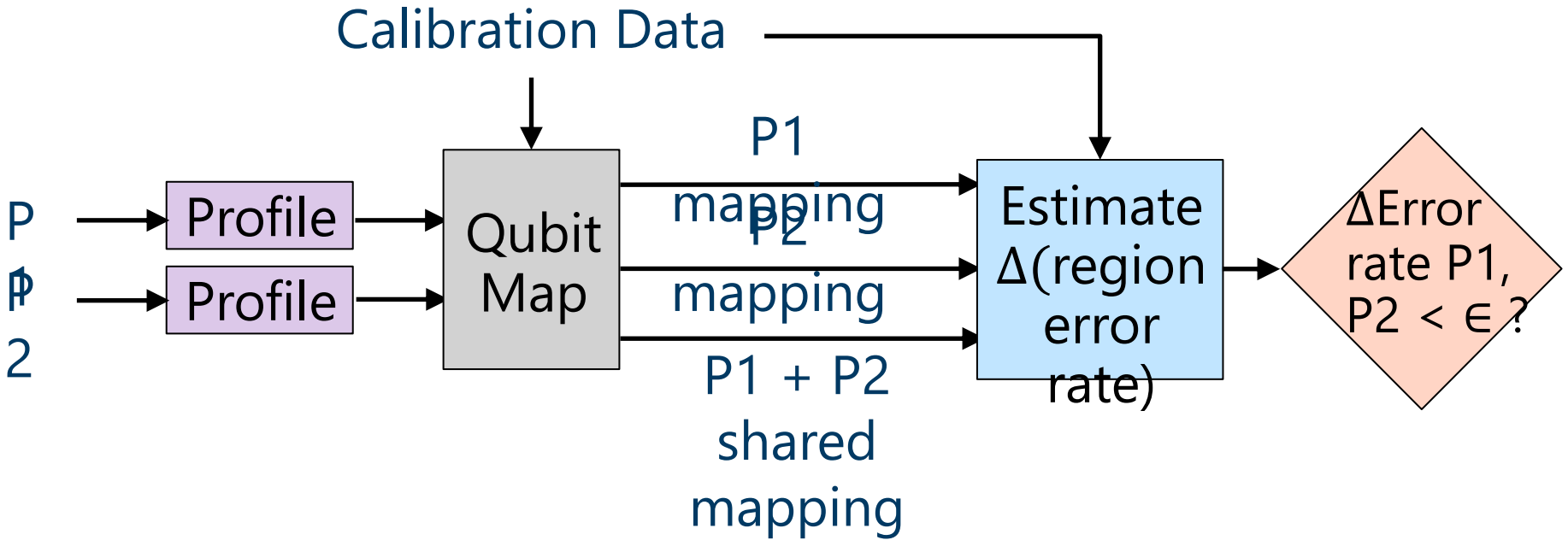
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



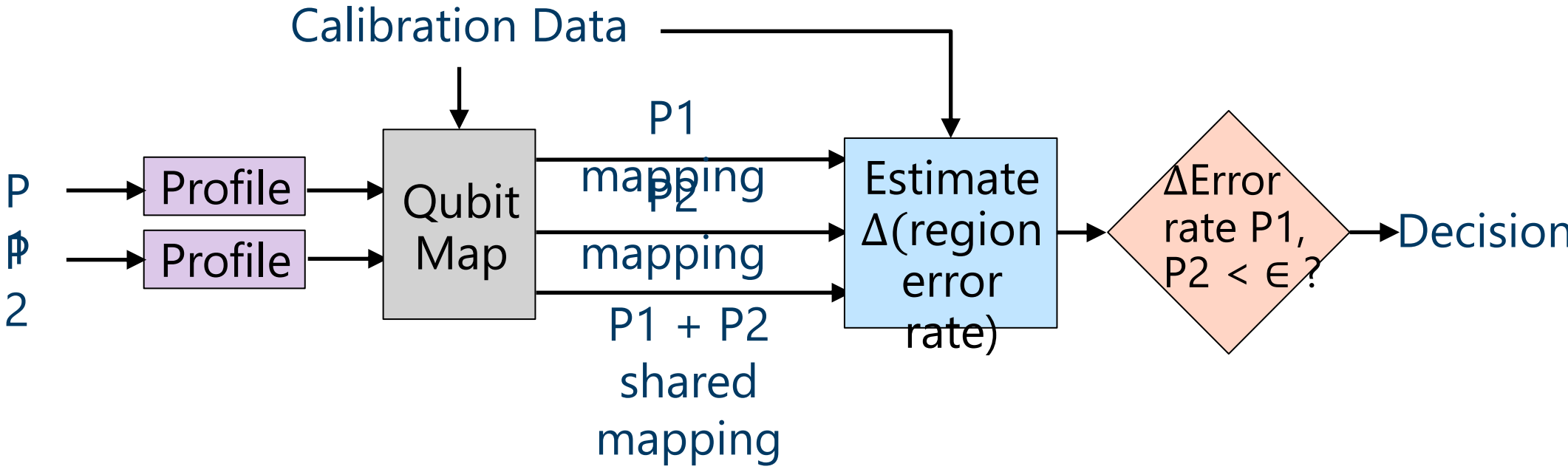
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



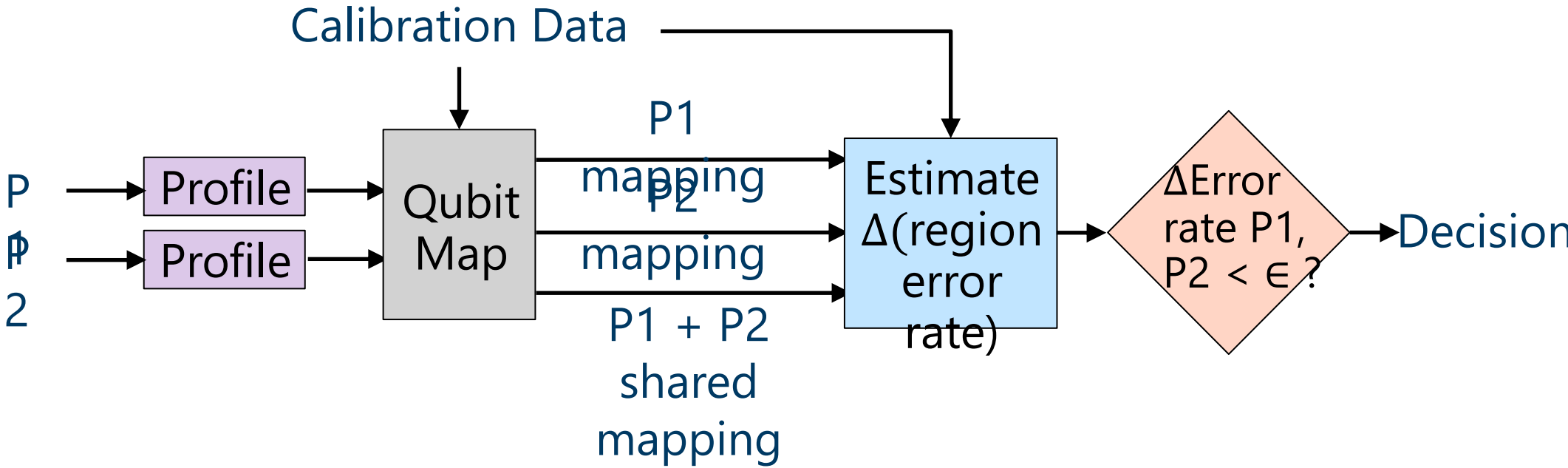
Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



Fair And Reliable Partitioning *(FRP)* Algorithm

Is a program's resource allocation in shared environment fair?



Algorithm ensures fairness while sharing resources between programs

Instruction Scheduling

How to schedule parallel programs?

Instruction Scheduling

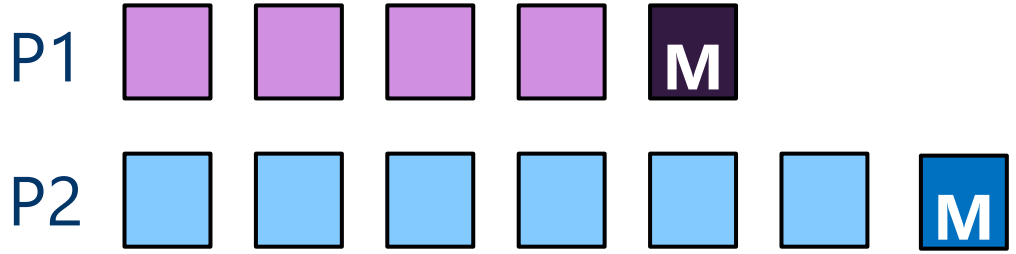
How to schedule parallel programs?

Programmers' View

Instruction Scheduling

How to schedule parallel programs?

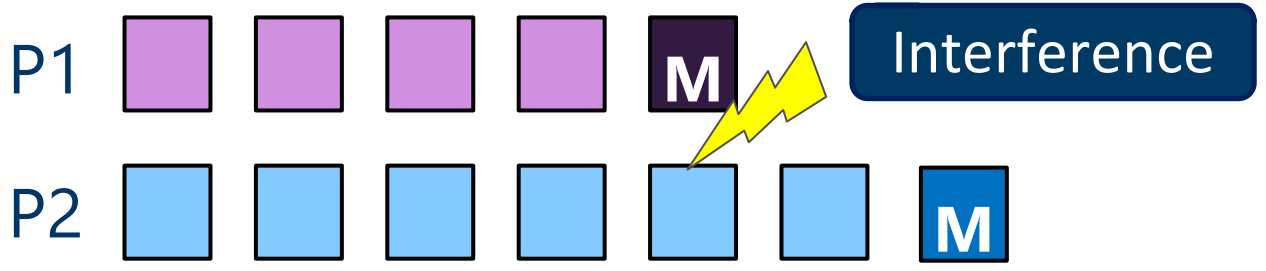
Programmers' View



Instruction Scheduling

How to schedule parallel programs?

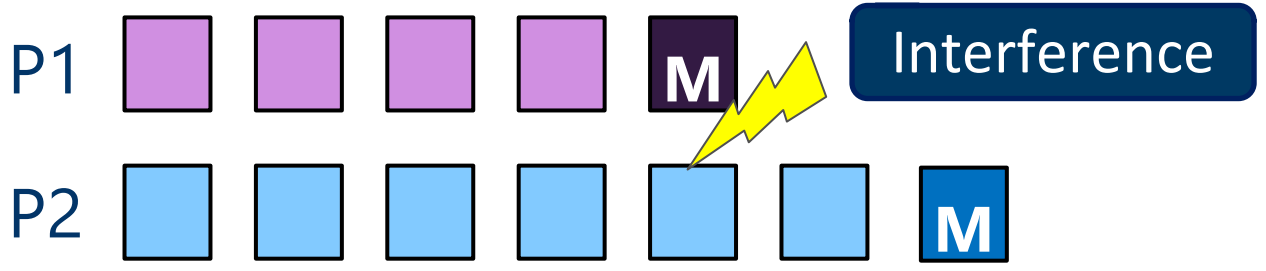
Programmers' View



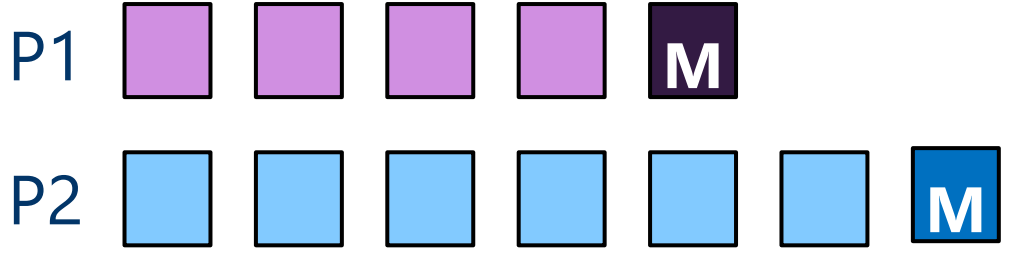
Instruction Scheduling

How to schedule parallel programs?

Programmers' View



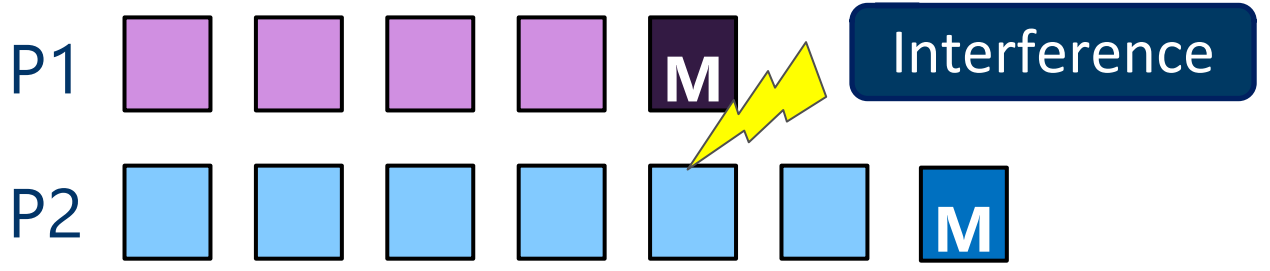
Existing approach



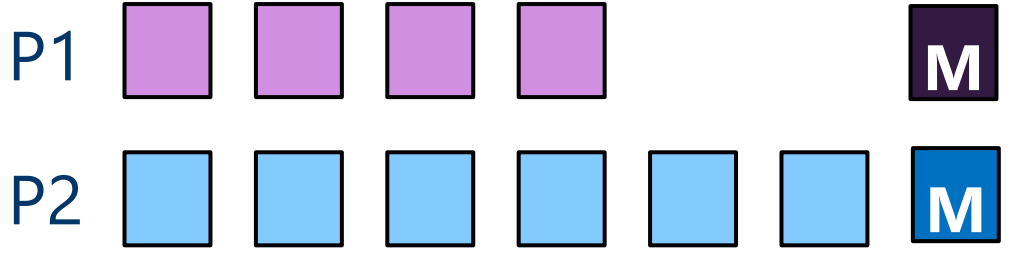
Instruction Scheduling

How to schedule parallel programs?

Programmers' View



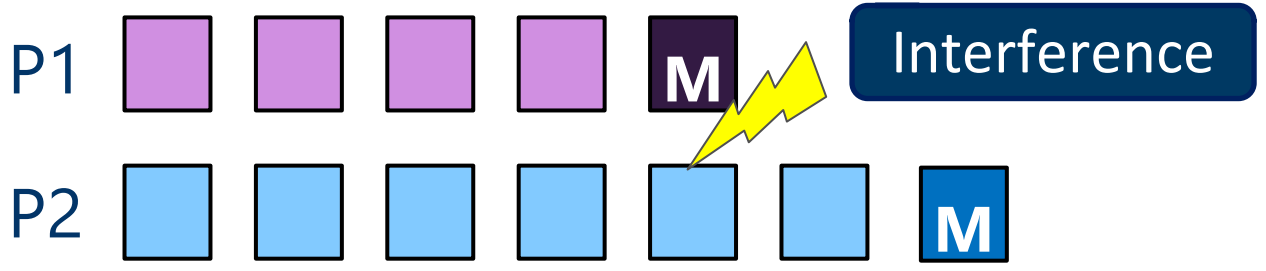
Existing approach



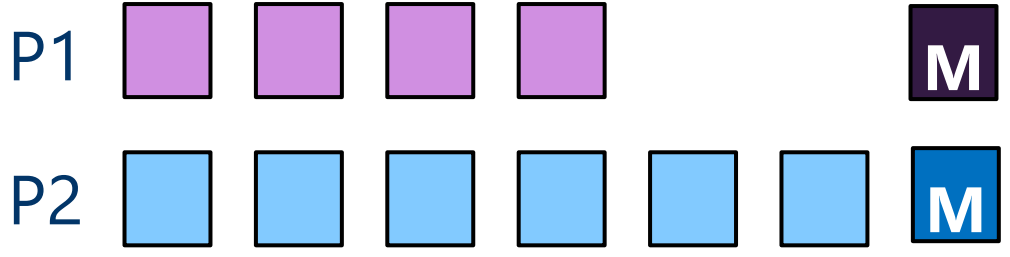
Instruction Scheduling

How to schedule parallel programs?

Programmers' View



Existing approach

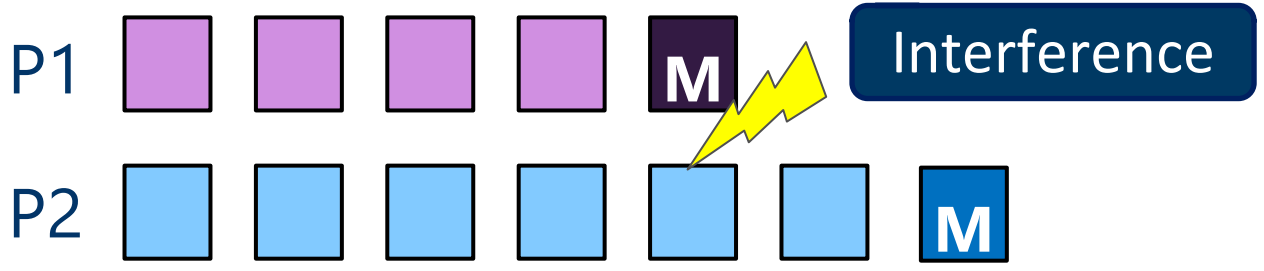


IBM's Compiler schedules measurements after all gates

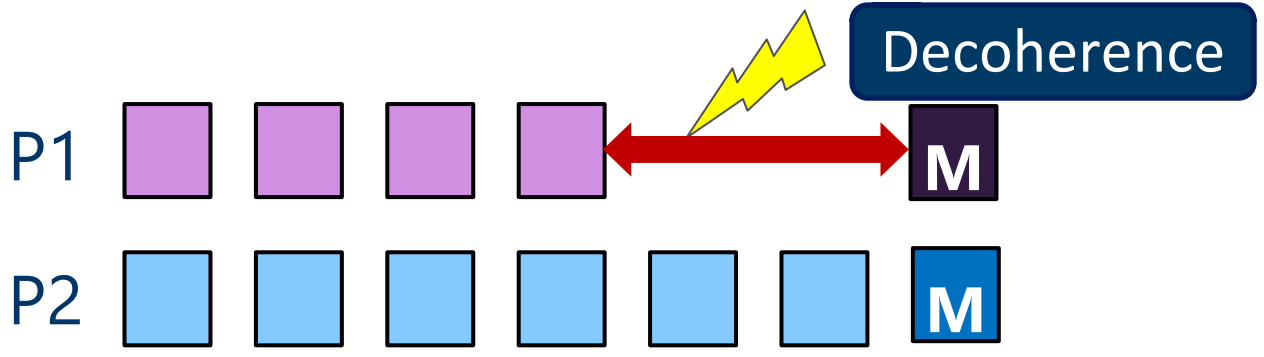
Instruction Scheduling

How to schedule parallel programs?

Programmers' View



Existing approach

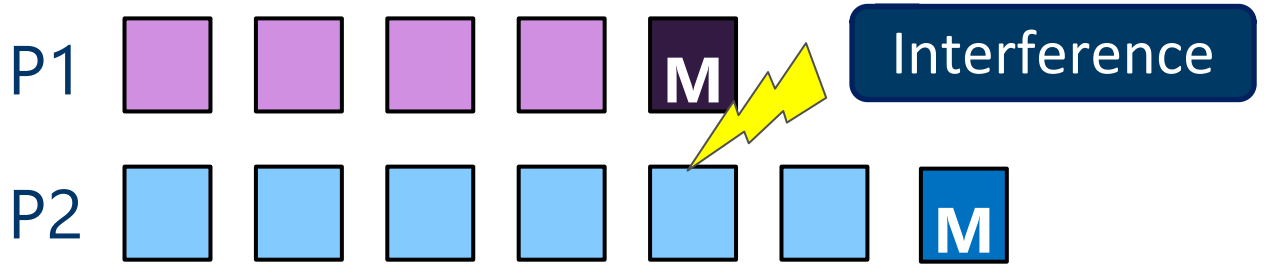


IBM's Compiler schedules measurements after all gates

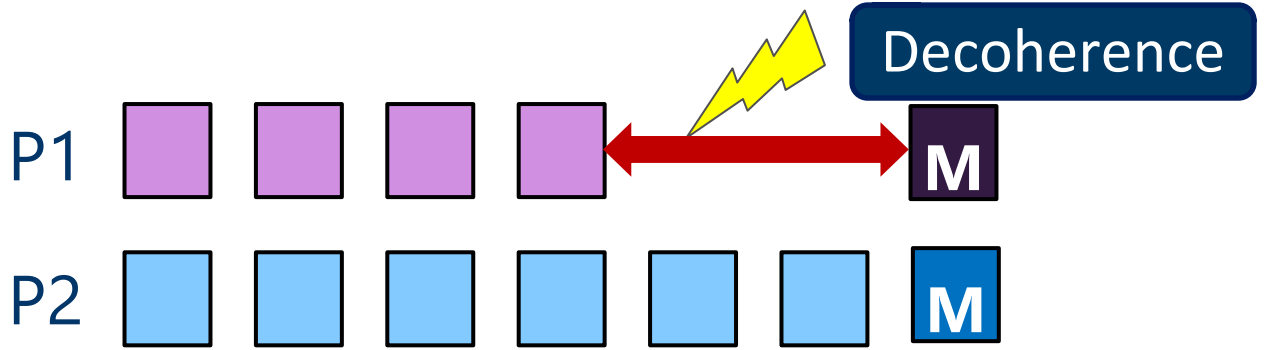
Instruction Scheduling

How to schedule parallel programs?

Programmers' View



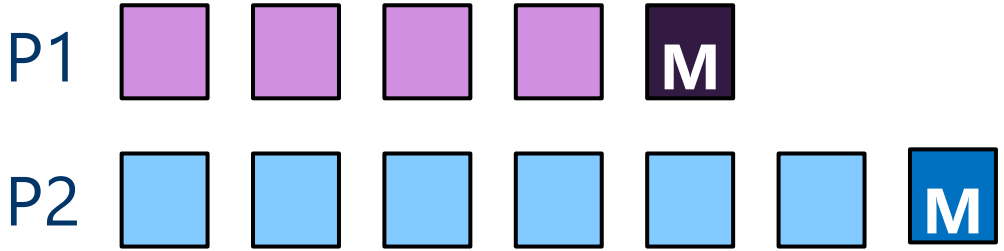
Existing approach



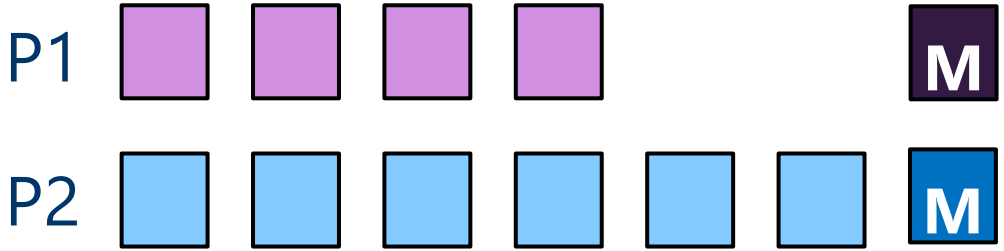
IBM's Compiler schedules measurements after all

Two irregular sized programs can suffer from interference and decoherence

Delayed Instruction Scheduling *(DIS)* Policy

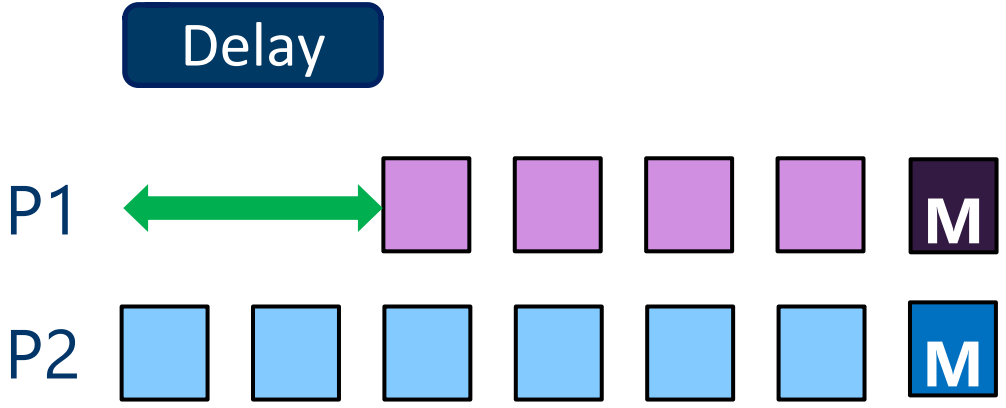


Delayed Instruction Scheduling *(DIS)* Policy



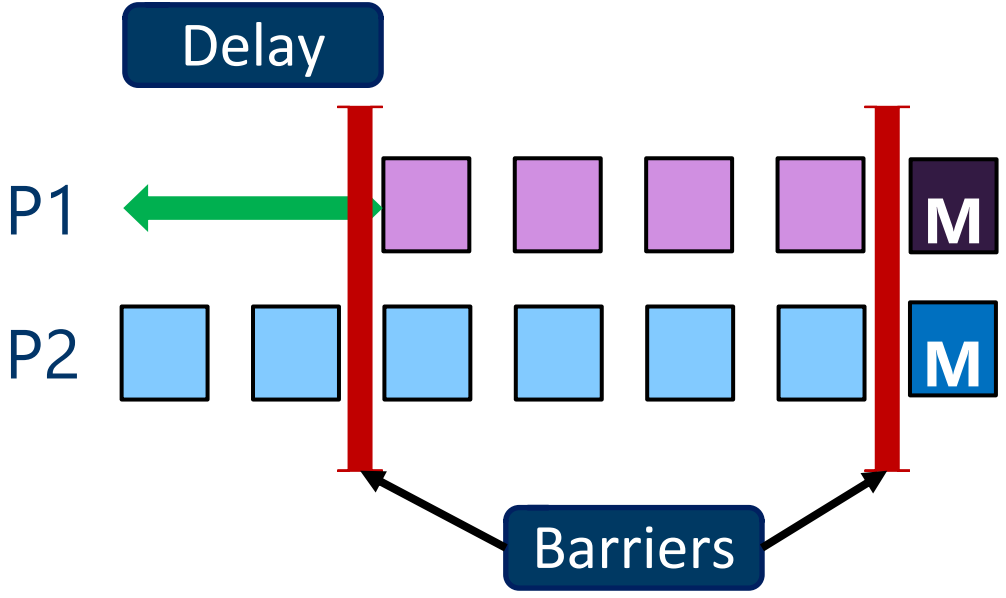
- Measurements at the end
  - Reduces interference

Delayed Instruction Scheduling *(DIS)* Policy



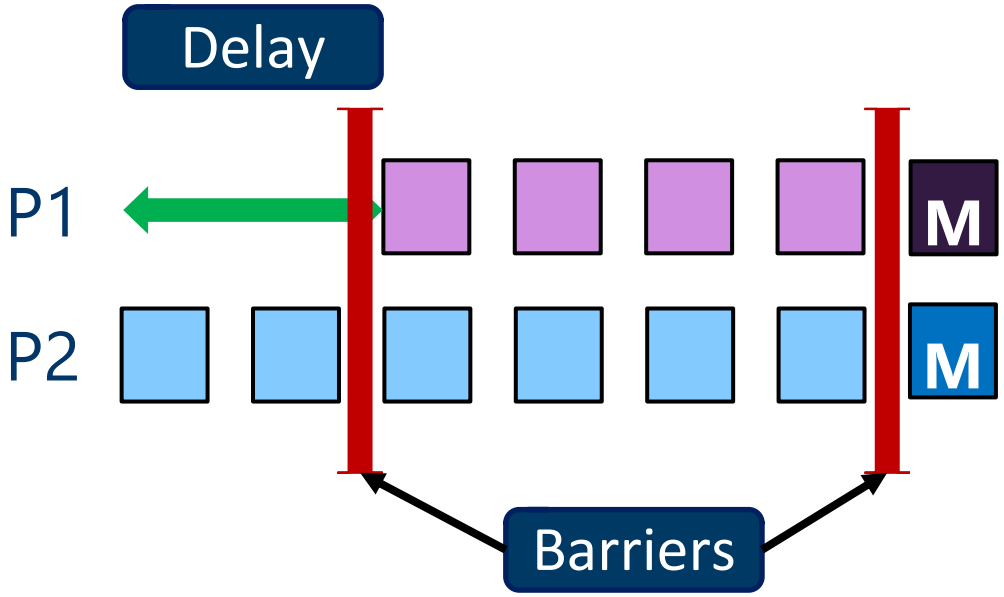
- Measurements at the end
  - Reduces interference
- Delay shorter program
  - Reduces decoherence

Delayed Instruction Scheduling *(DIS)* Policy



- Measurements at the end
  - Reduces interference
- Delay shorter program
  - Reduces decoherence
- Barriers

Delayed Instruction Scheduling *(DIS)* Policy



- Measurements at the end
  - Reduces interference
- Delay shorter program
  - Reduces decoherence
- Barriers

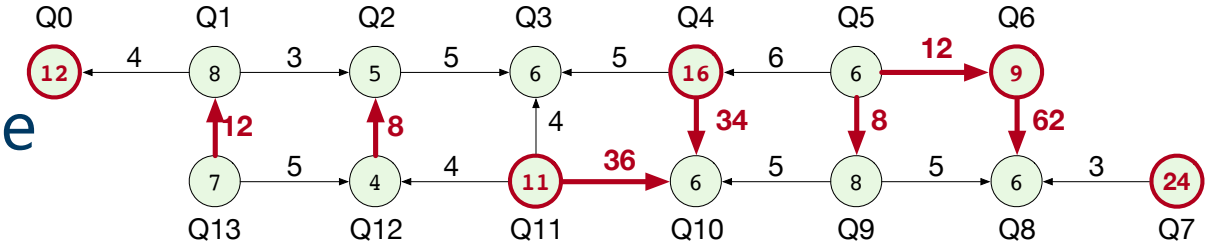
Our proposed *DIS* policy reduces interference and decoherence and is scalable

Outline

- Introduction
- Background and Motivation
- Policies for Multi-Programming
- Evaluation Methodology
- Adaptive Multi-Programming Design
- Results and Conclusion

Evaluation Methodology

- IBM Q16
  - 14 qubit public machine

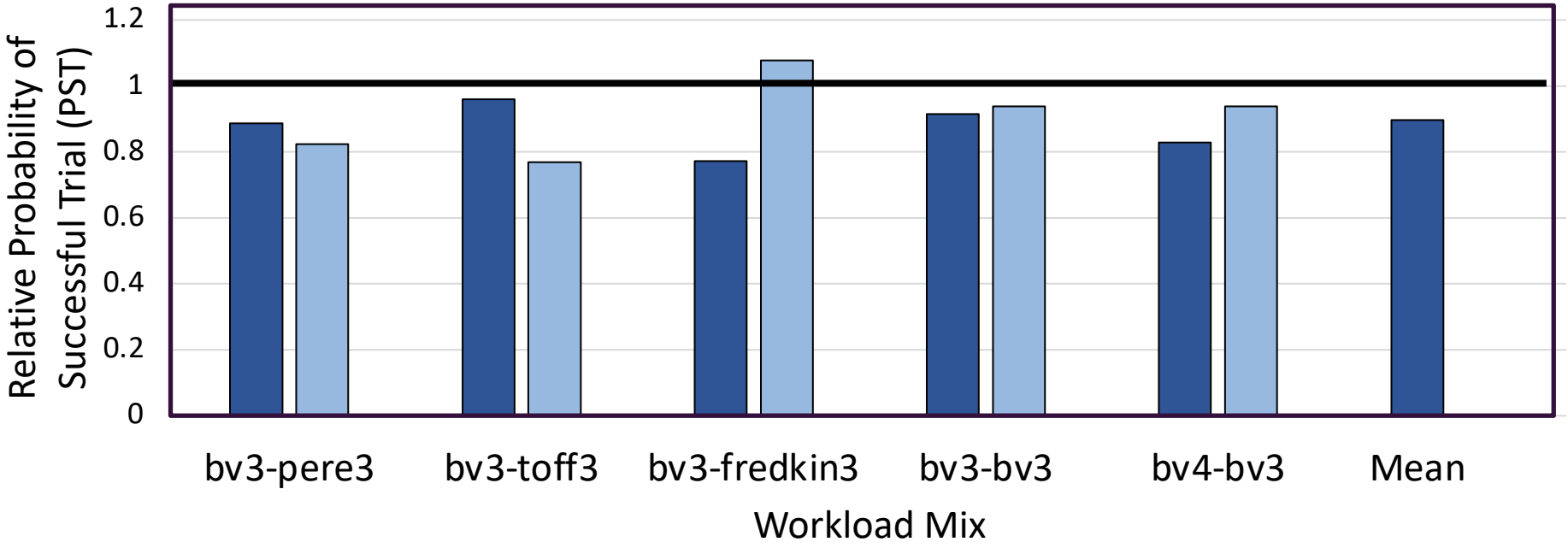


- Benchmarks
  - From prior works

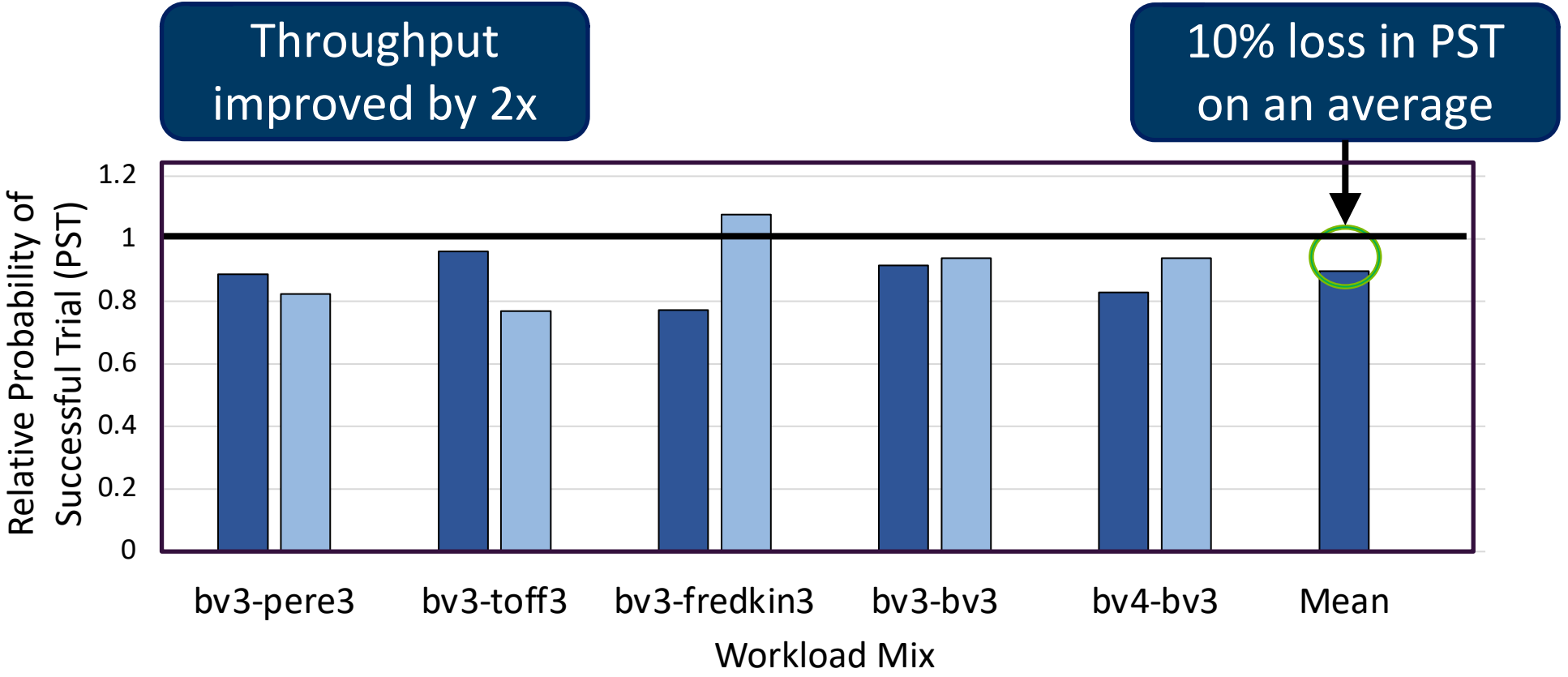
Benchmark	Description	#Insts	#CNOT
bv_n3	Bernstein Vazirani	8	2
bv_n4	Bernstein Vazirani	11	3
Toffoli_n3	Toffoli gate	15	6
Fredkin_n3	Fredkin gate	16	8
Peres_n3	Peres gate	16	7

- Baseline: Isolated execution using best qubit mapping

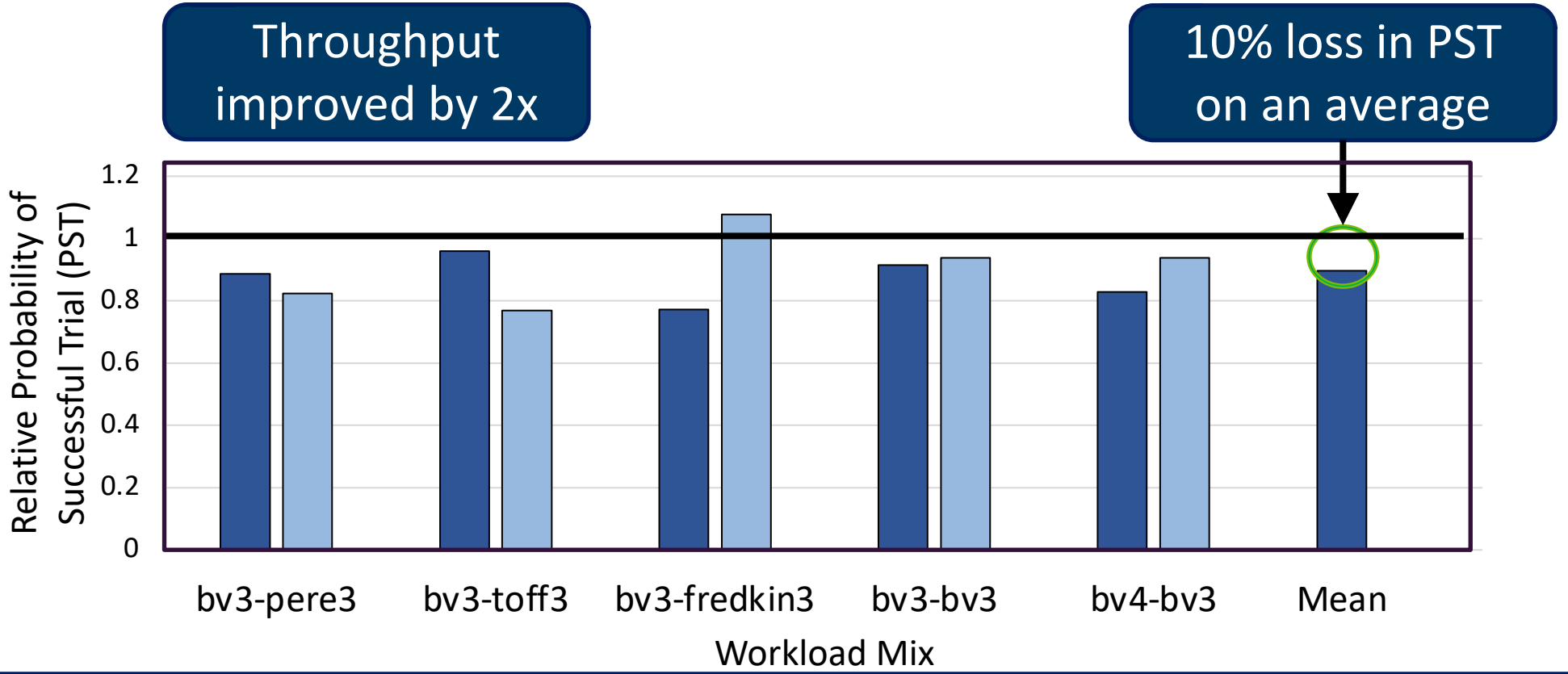
Results of Multi-Programming



Results of Multi-Programming

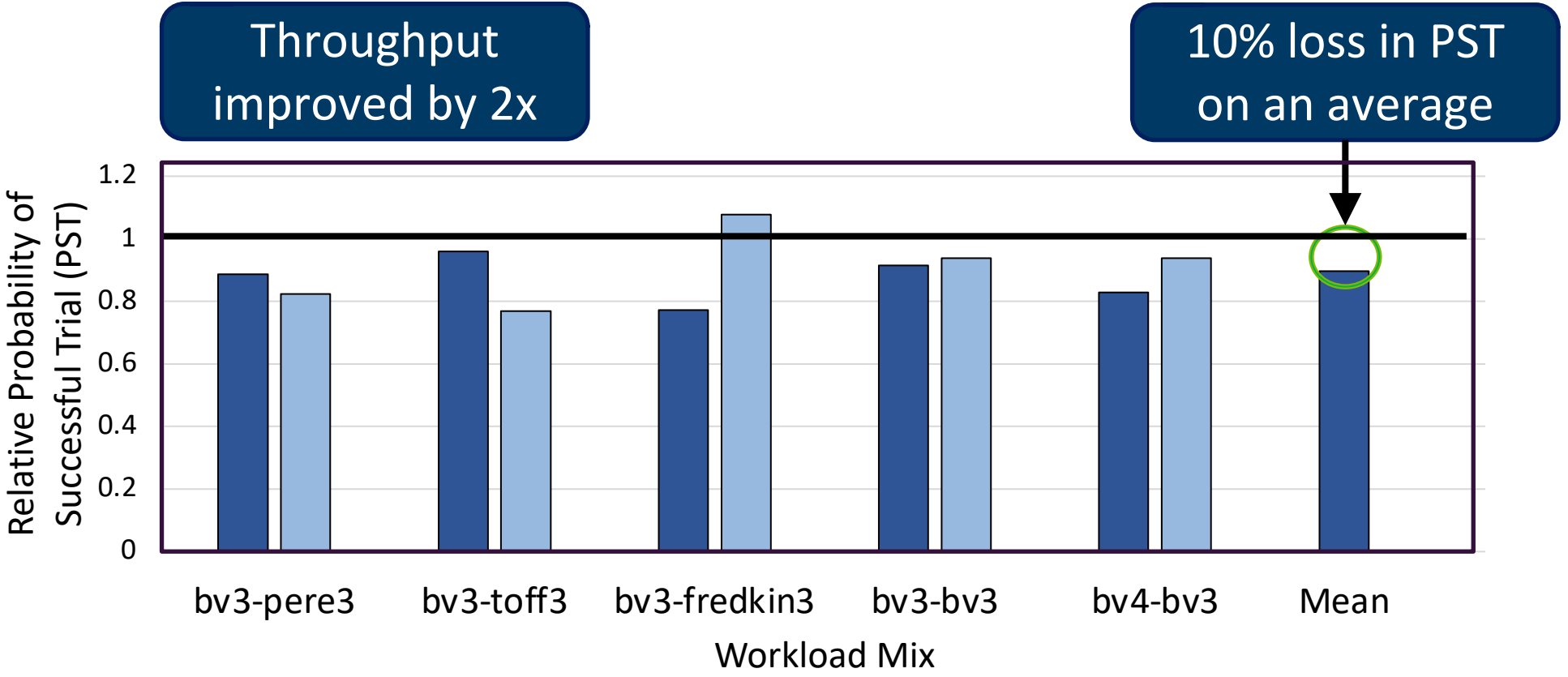


Results of Multi-Programming

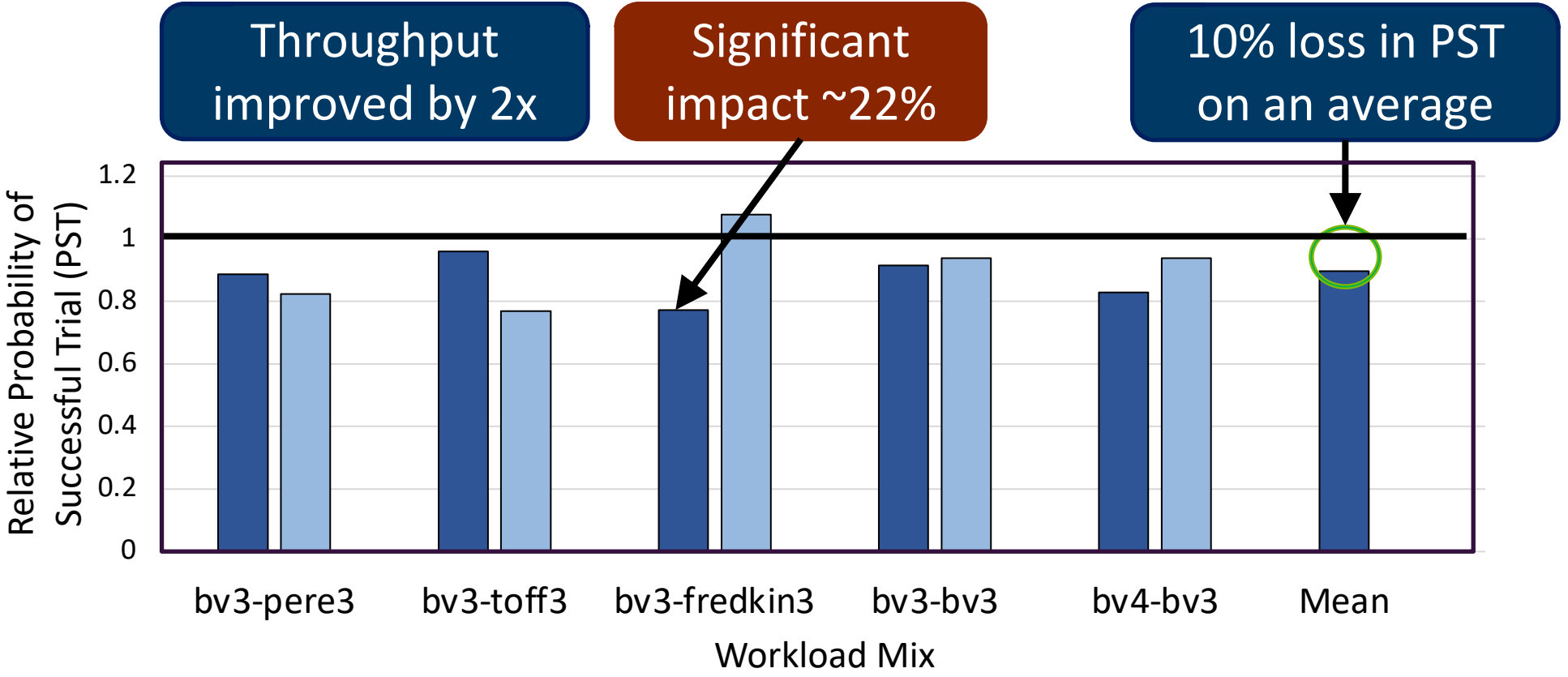


Multi-programming can improve throughput with slight impact on PST

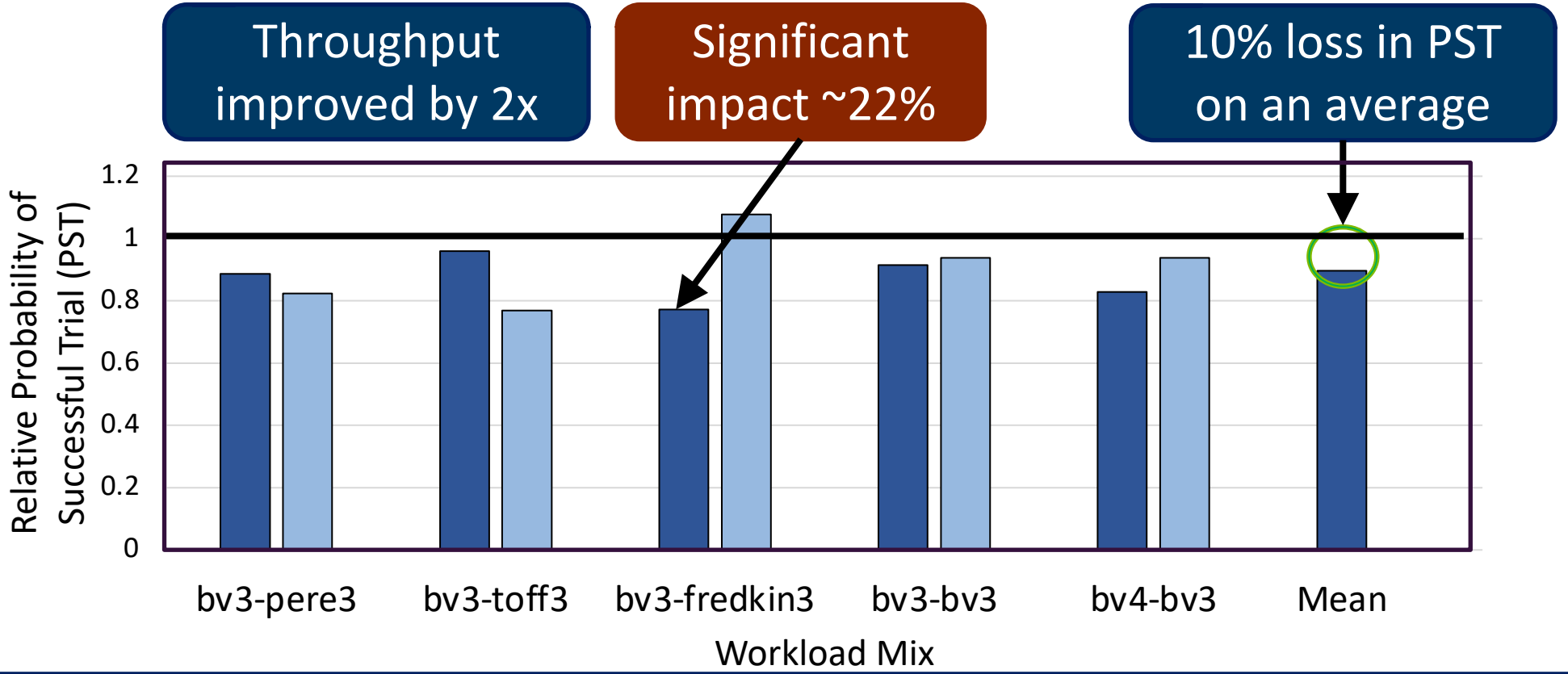
Limiting the Reliability Impact of Multi-Prog



Limiting the Reliability Impact of Multi-Prog



Limiting the Reliability Impact of Multi-Prog



Multi-programming must adapt to minimize significant impact on reliability

Outline

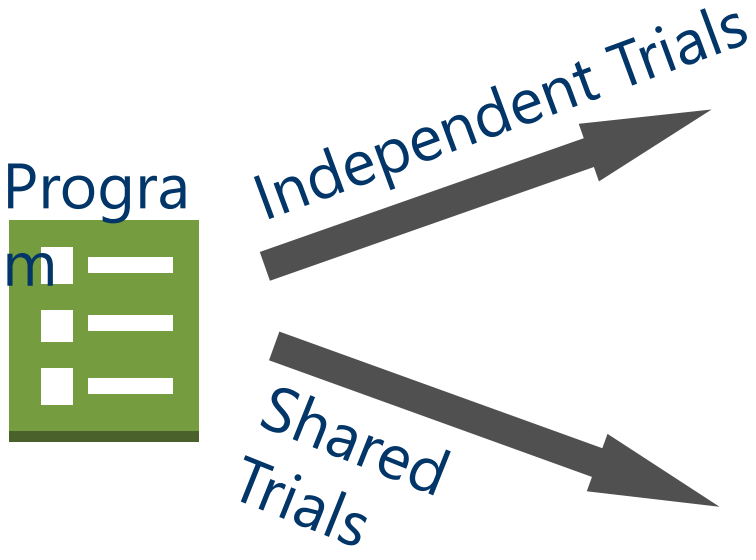
- Introduction
- Background and Motivation
- Policies for Multi-Programming
- Evaluation Methodology
- Adaptive Multi-Programming Design
- Results and Conclusion

Runtime Reliability Monitoring

Can we detect impact on reliability at runtime?

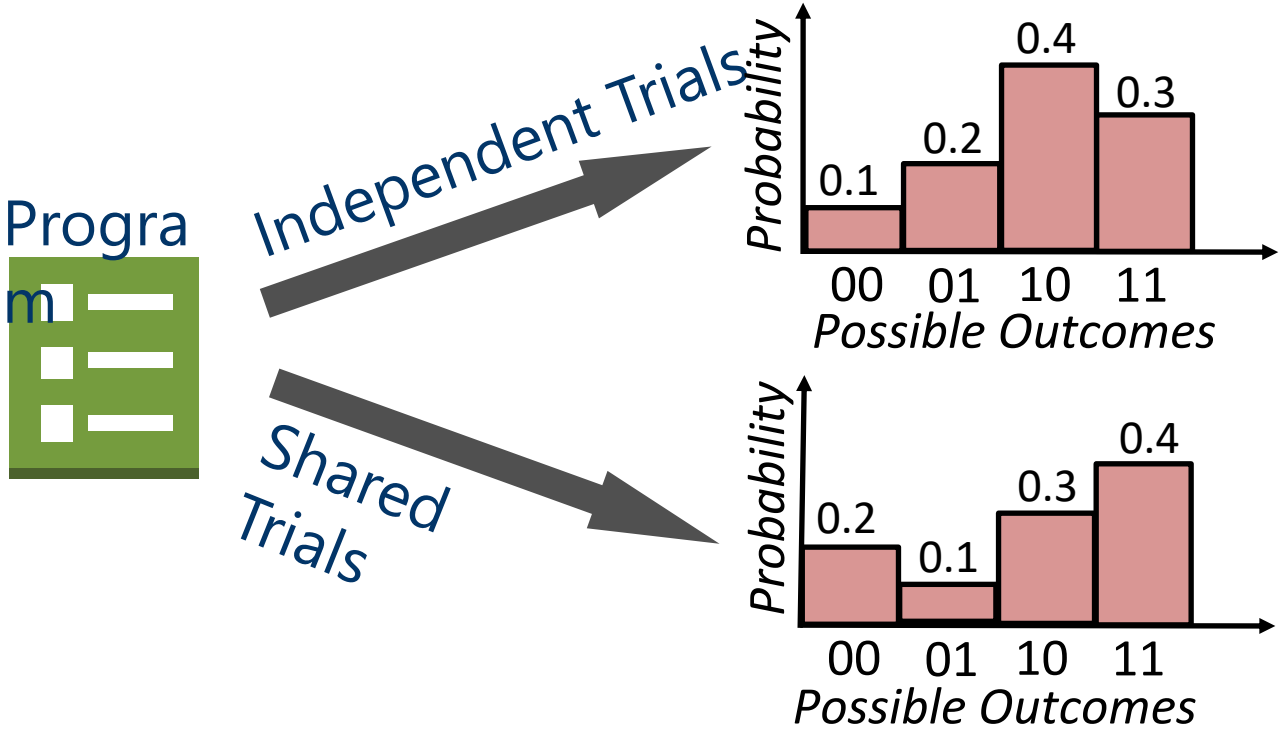
Runtime Reliability Monitoring

Can we detect impact on reliability at runtime?



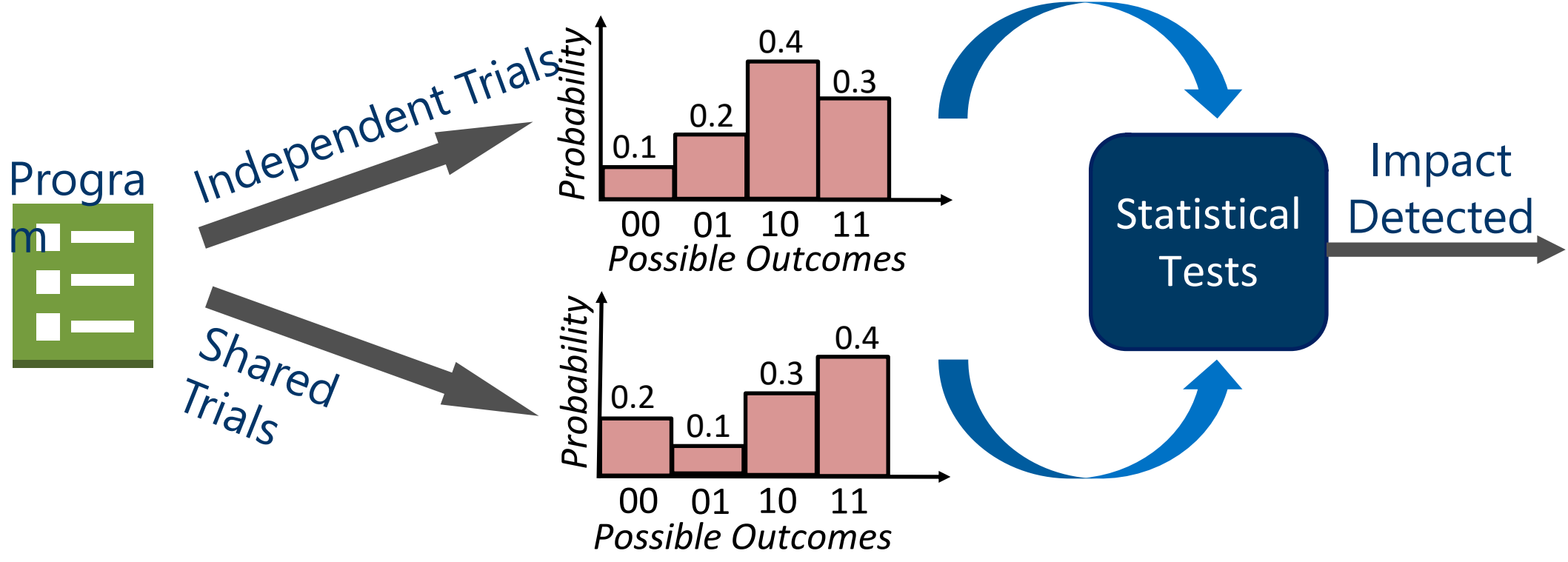
Runtime Reliability Monitoring

Can we detect impact on reliability at runtime?



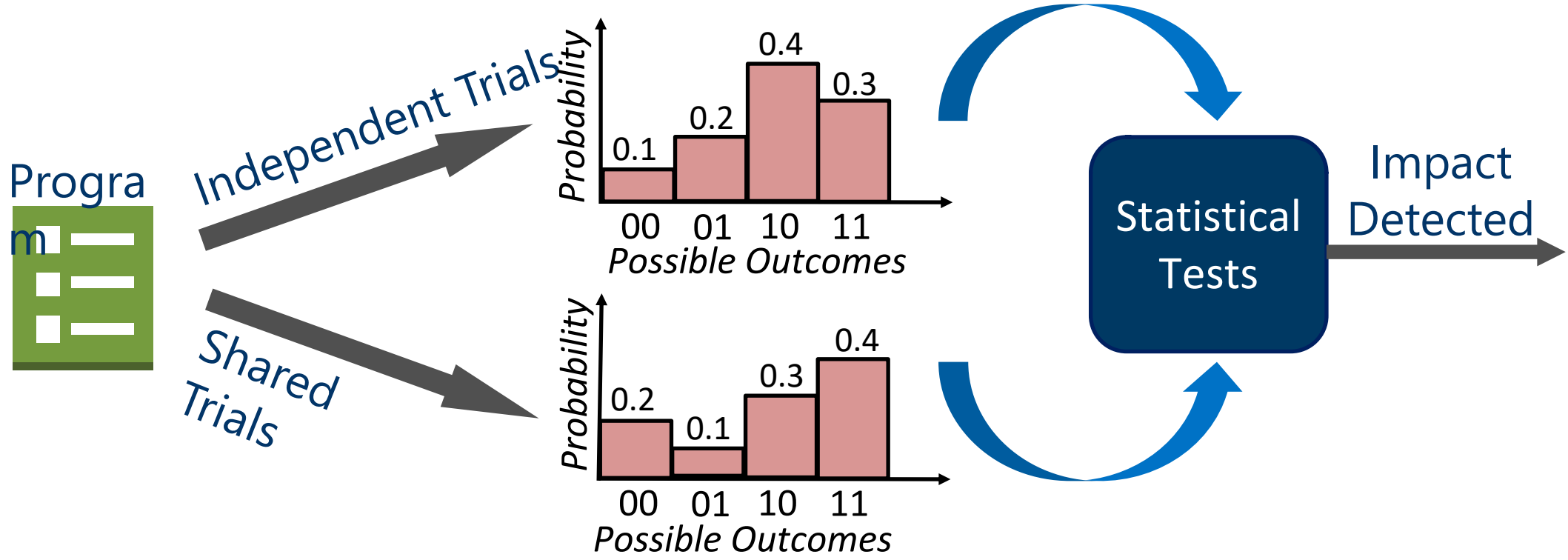
Runtime Reliability Monitoring

Can we detect impact on reliability at runtime?



Runtime Reliability Monitoring

Can we detect impact on reliability at runtime?



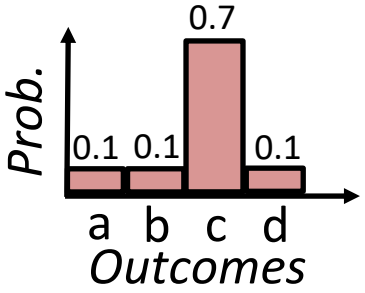
Using statistical tests degradation in program reliability can be captured

Description of the statistical tests

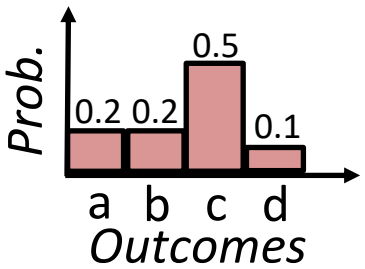
Entropy: measures randomness

Description of the statistical tests

Entropy: measures randomness



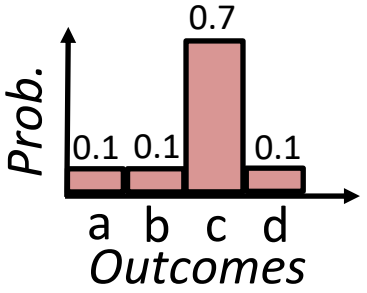
Dist 1



Dist 2

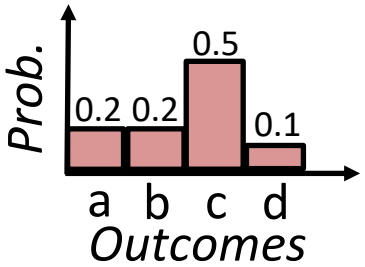
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy

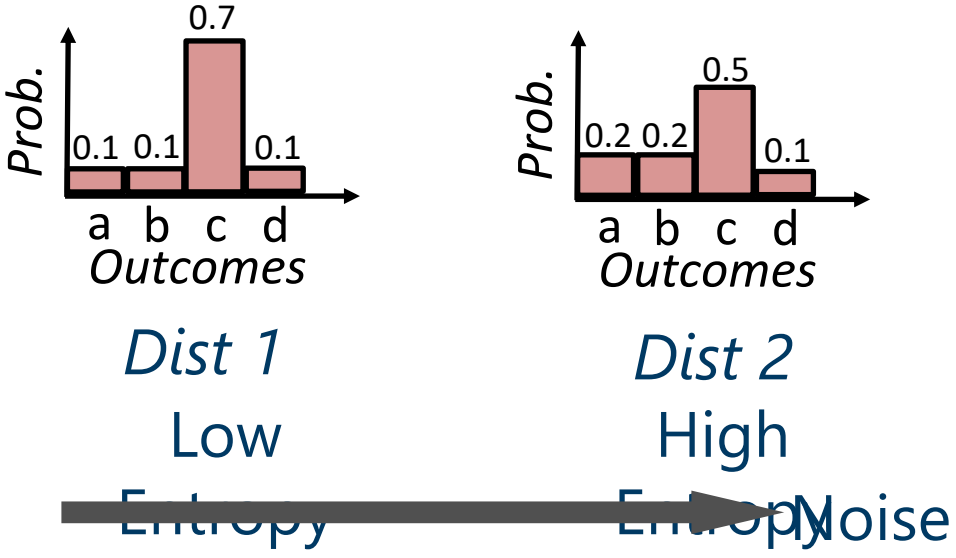


Dist 2

High Entropy

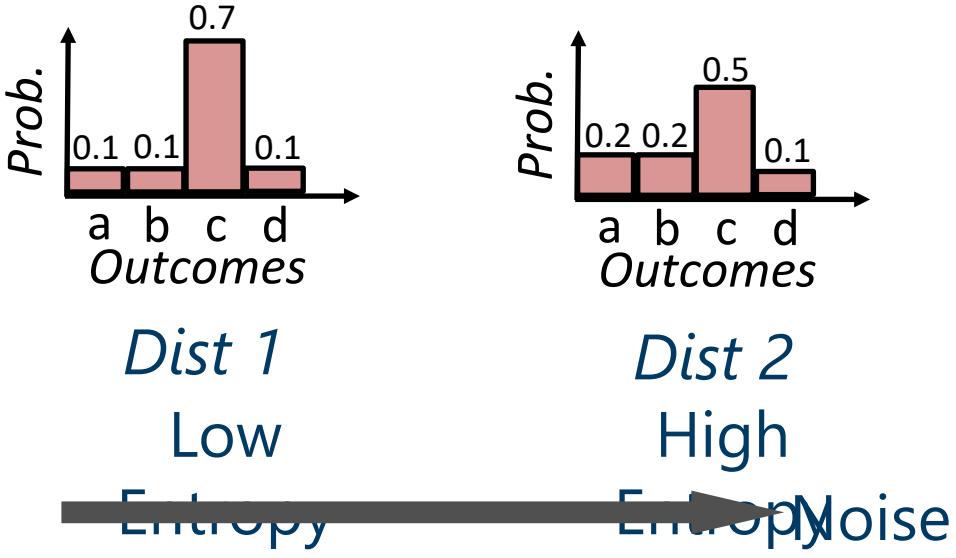
Description of the statistical tests

Entropy: measures randomness



Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy

Dist 2

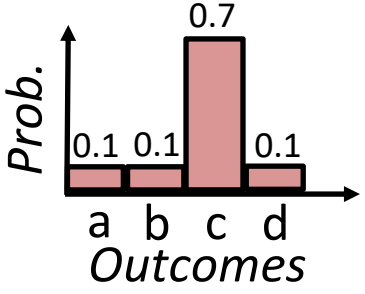
High Entropy

Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

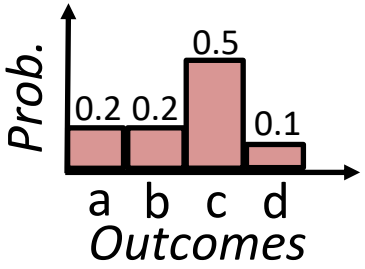
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy



Dist 2

High Entropy

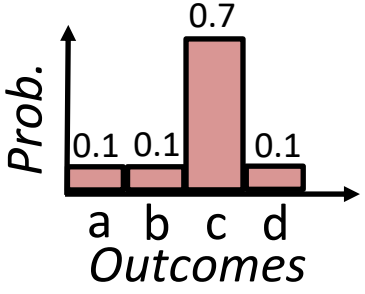
Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

Hellinger Distance: measures correlation between distributions

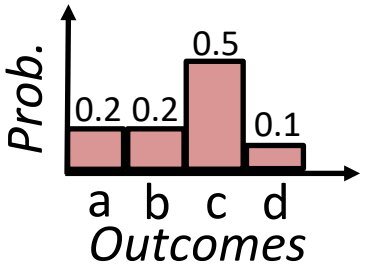
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy



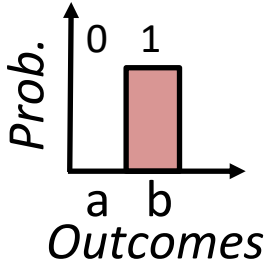
Dist 2

High Entropy

Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

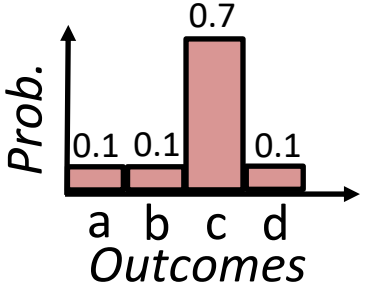
Hellinger Distance: measures correlation between distributions



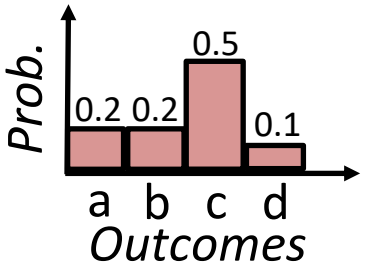
D1

Description of the statistical tests

Entropy: measures randomness



Dist 1



Dist 2

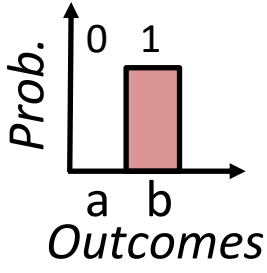
Low Entropy

High Entropy

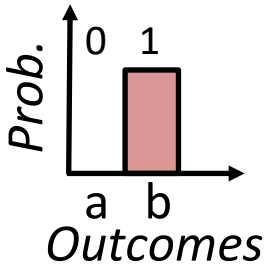
Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

Hellinger Distance: measures correlation between distributions



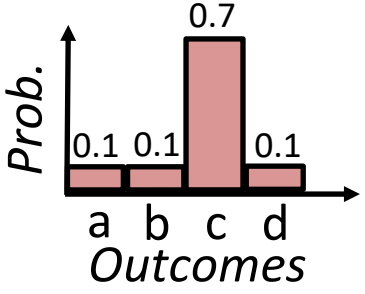
D1



D2

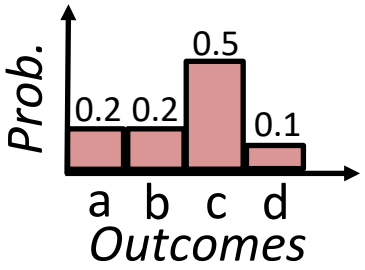
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy



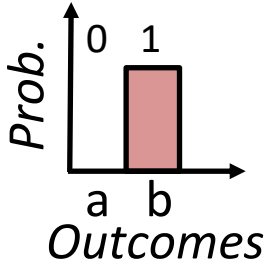
Dist 2

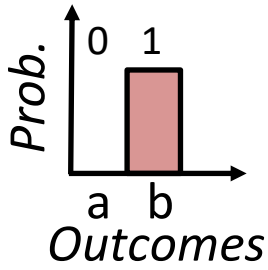
High Entropy

Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

Hellinger Distance: measures correlation between distributions



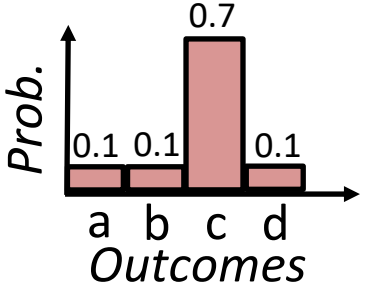


D1 *D2*

HD=0

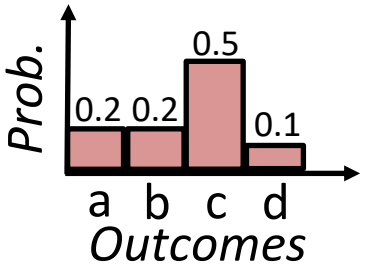
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy



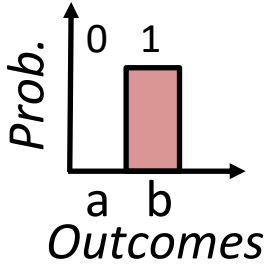
Dist 2

High Entropy

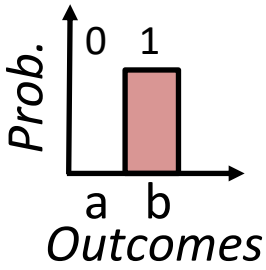
Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

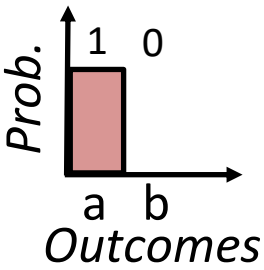
Hellinger Distance: measures correlation between distributions



D1



D2



D3

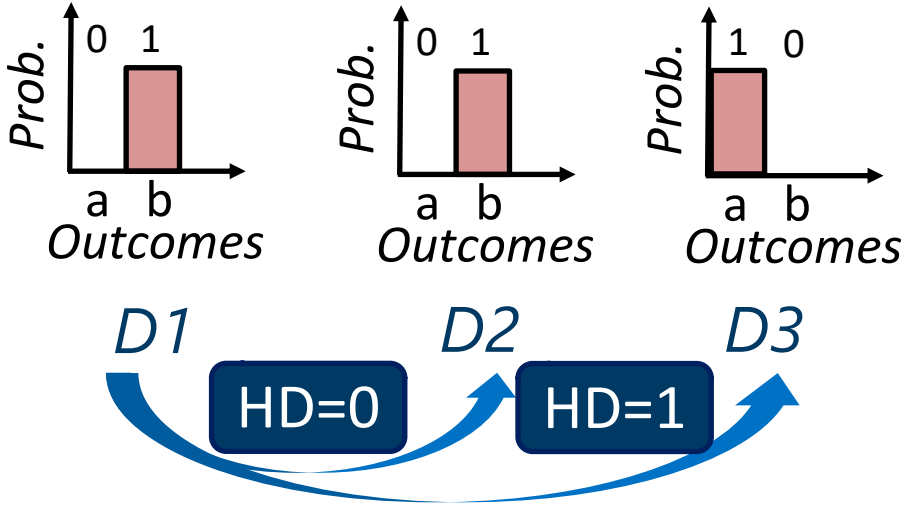
HD=0

Description of the statistical tests

Entropy: measures randomness

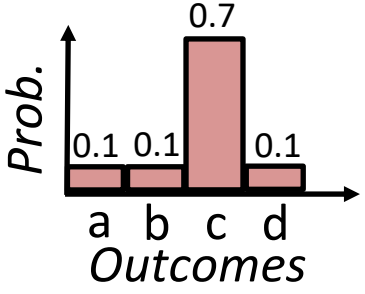
Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

Hellinger Distance: measures correlation between distributions



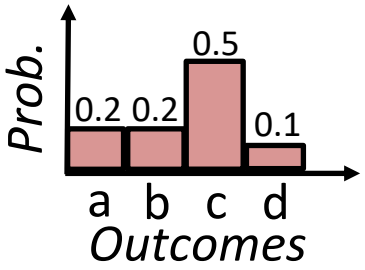
Description of the statistical tests

Entropy: measures randomness



Dist 1

Low Entropy



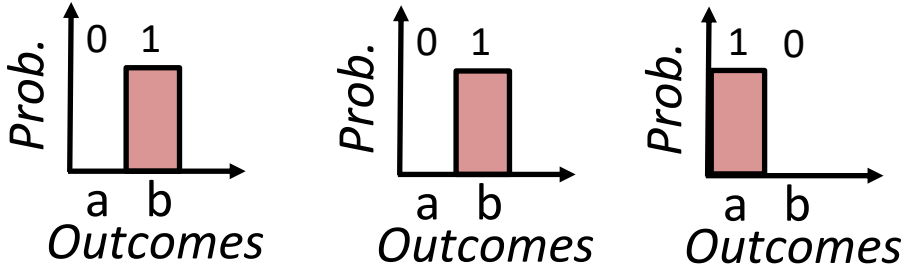
Dist 2

High Entropy

Noise

Compare $Entropy(Shared\ Trials)$ and $Entropy(Independent\ Trials)$

Hellinger Distance: measures correlation between distributions



D1 *D2* *D3*

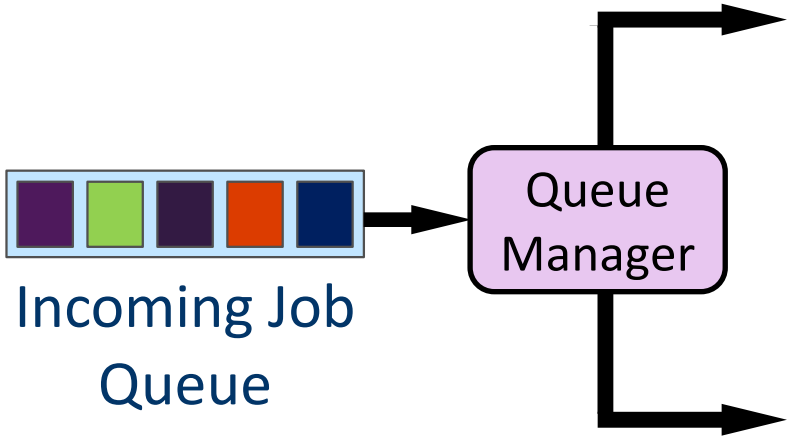
HD=0 HD=1

How similar or dissimilar are the shared and independent trials?

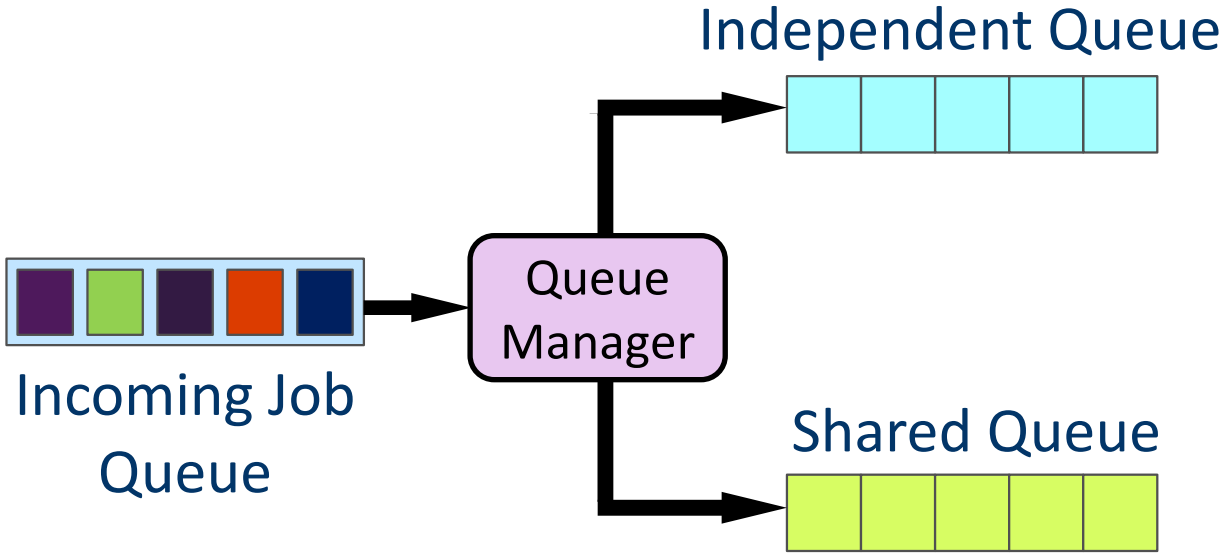
AMP Design Implementation

Incoming Job Queue

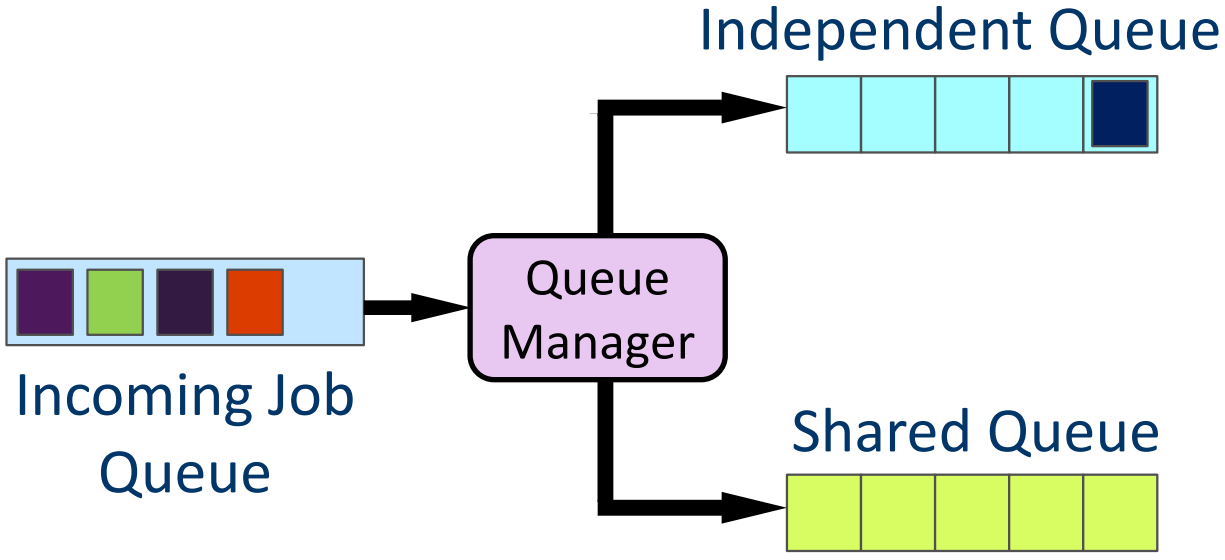
AMP Design Implementation



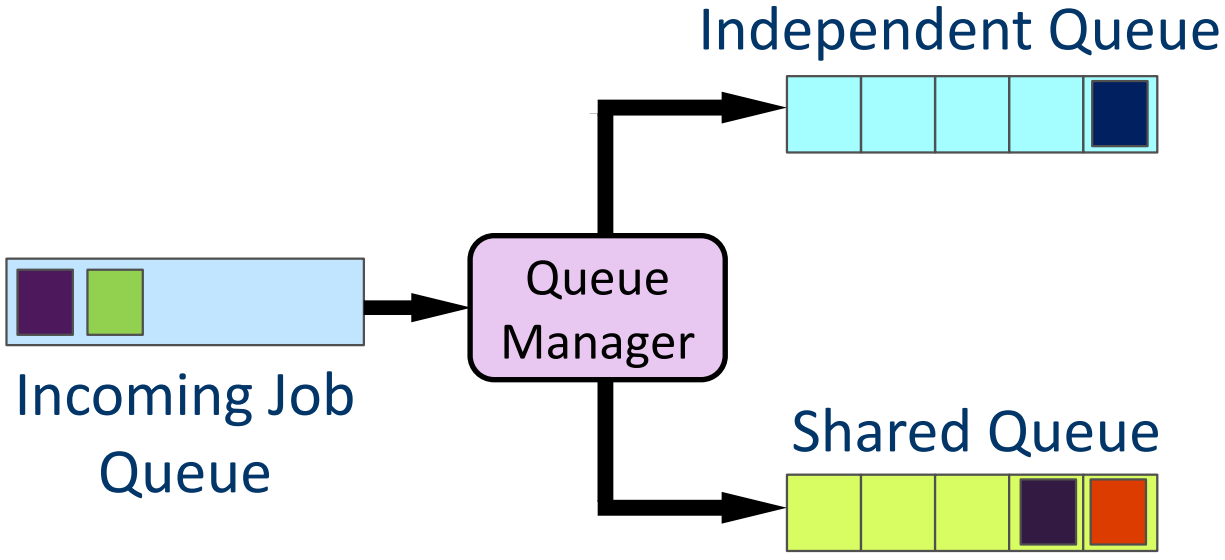
AMP Design Implementation



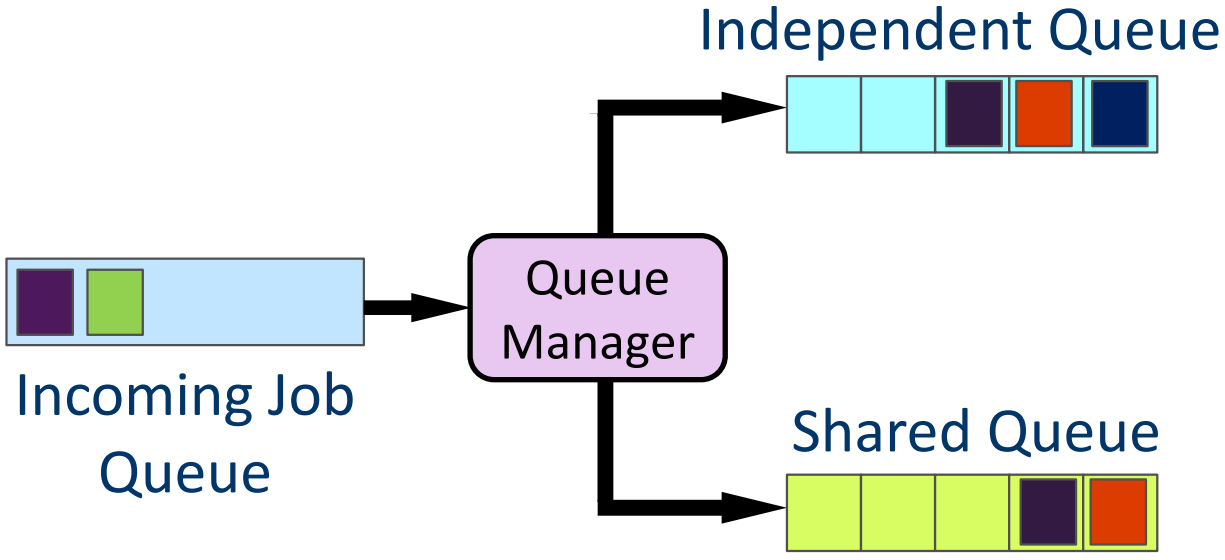
AMP Design Implementation



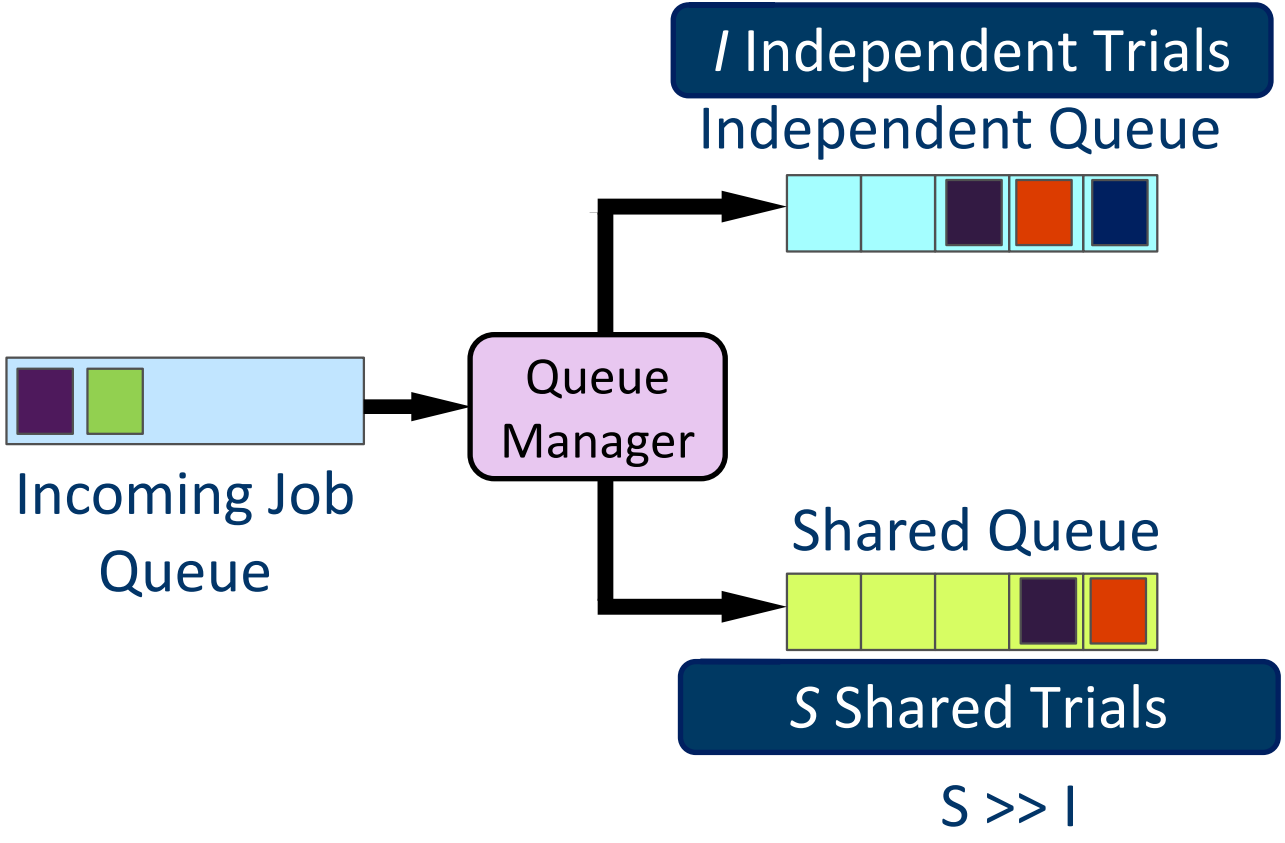
AMP Design Implementation



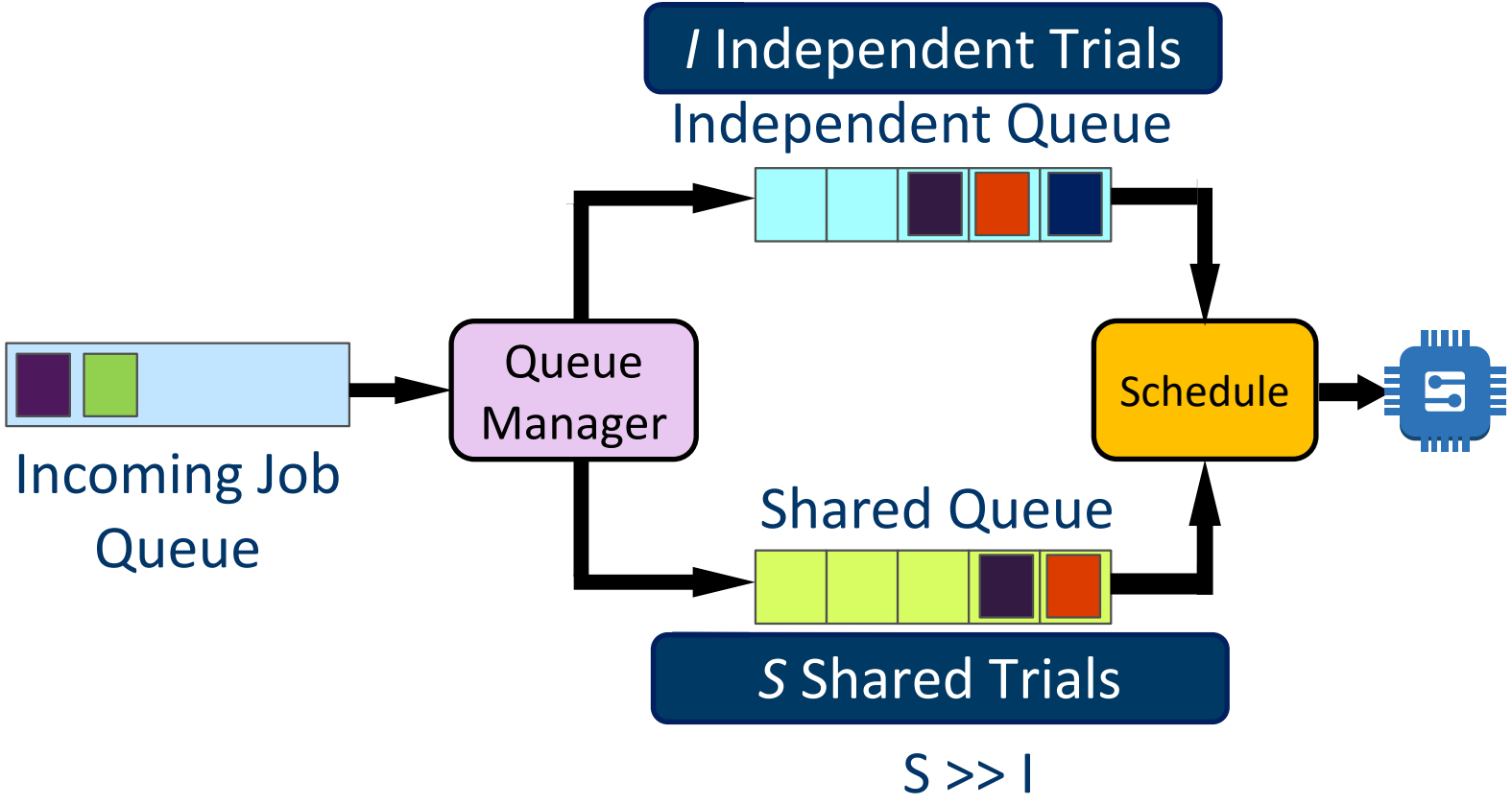
AMP Design Implementation



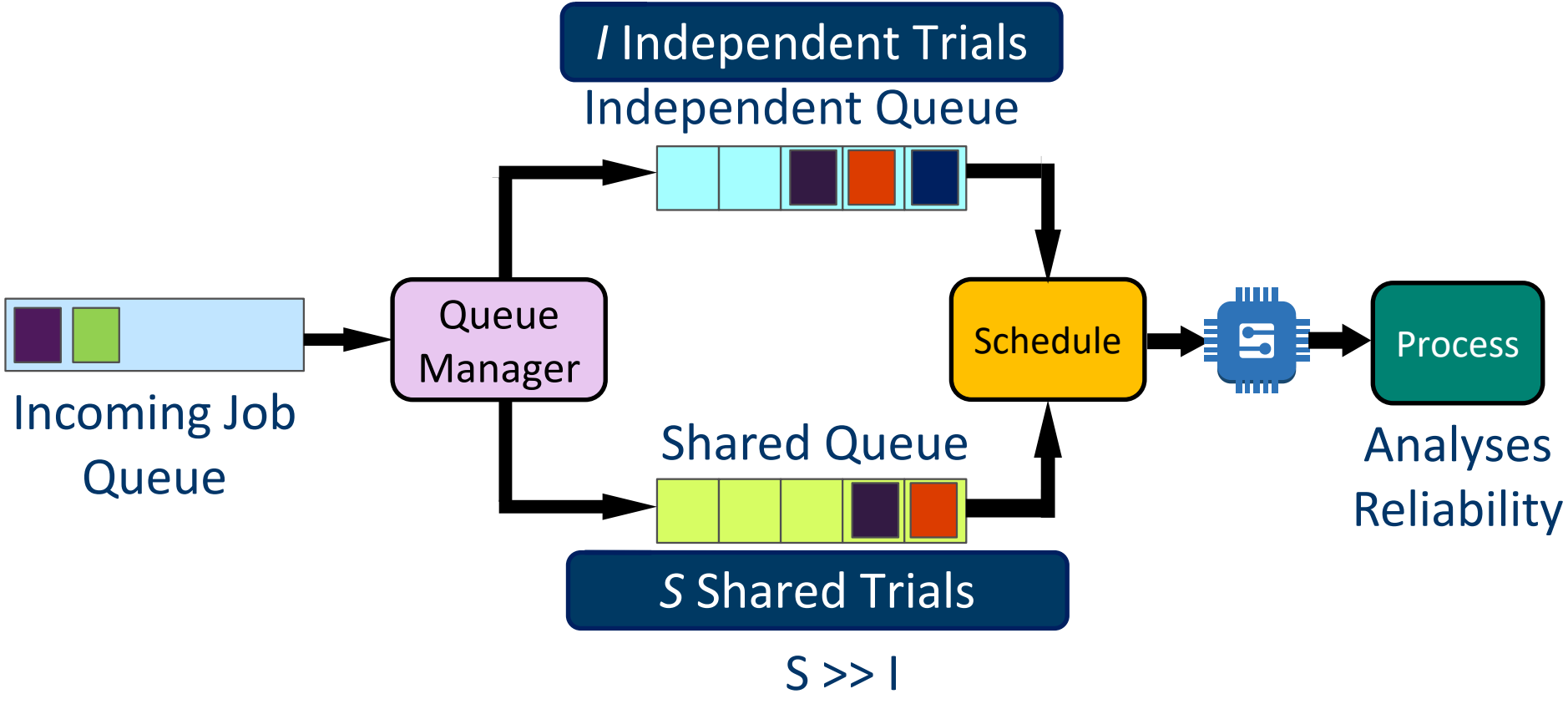
AMP Design Implementation



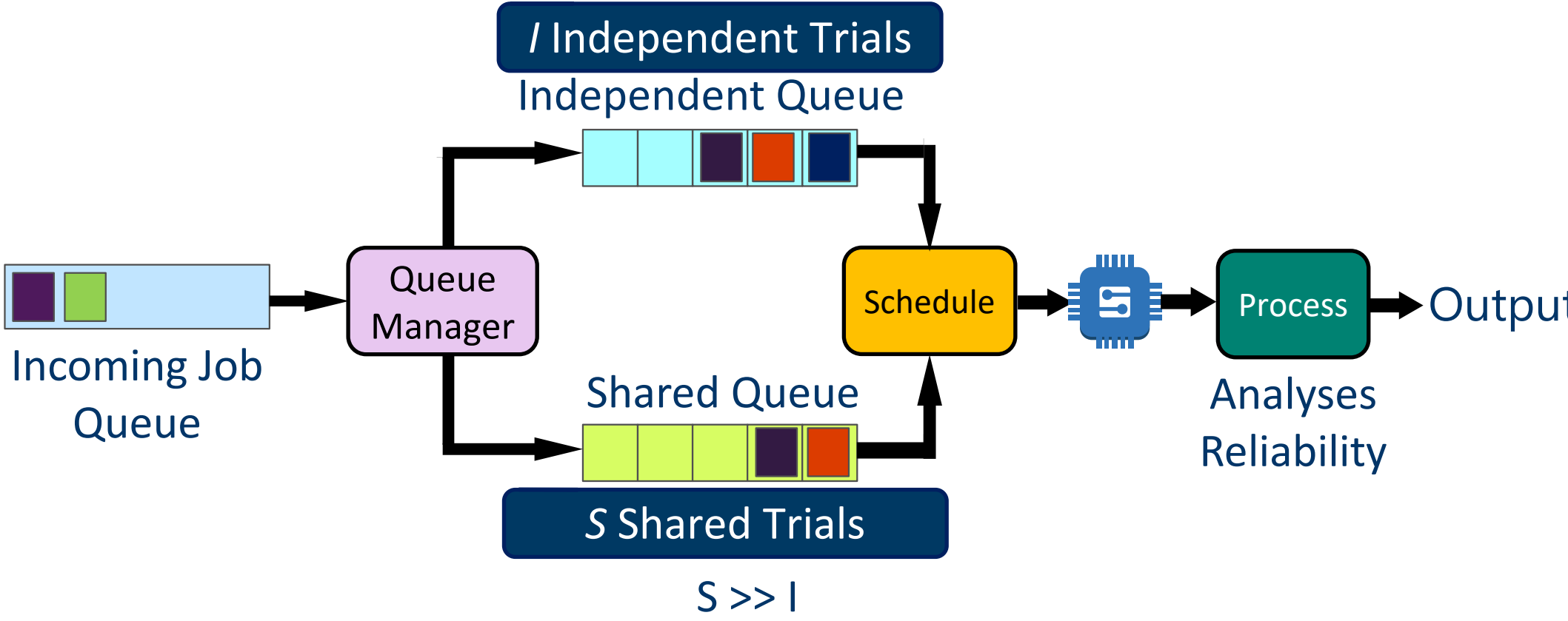
AMP Design Implementation



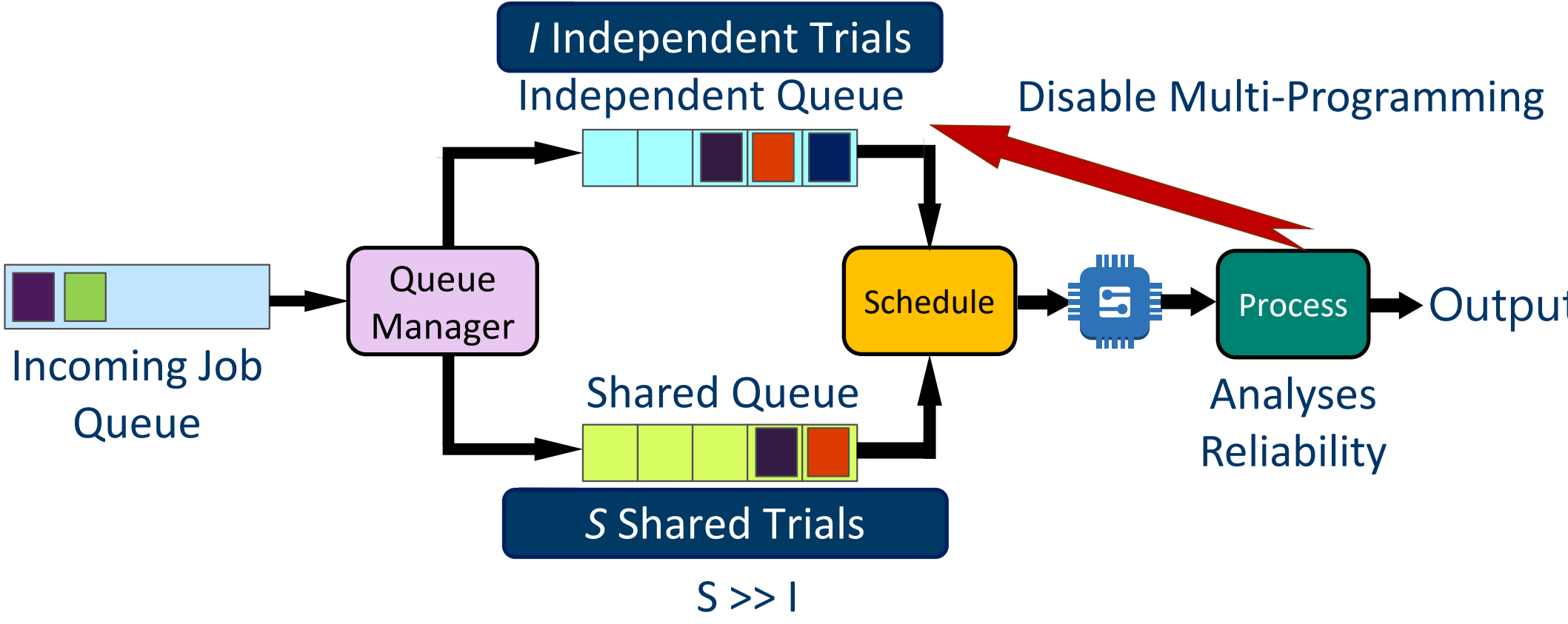
AMP Design Implementation



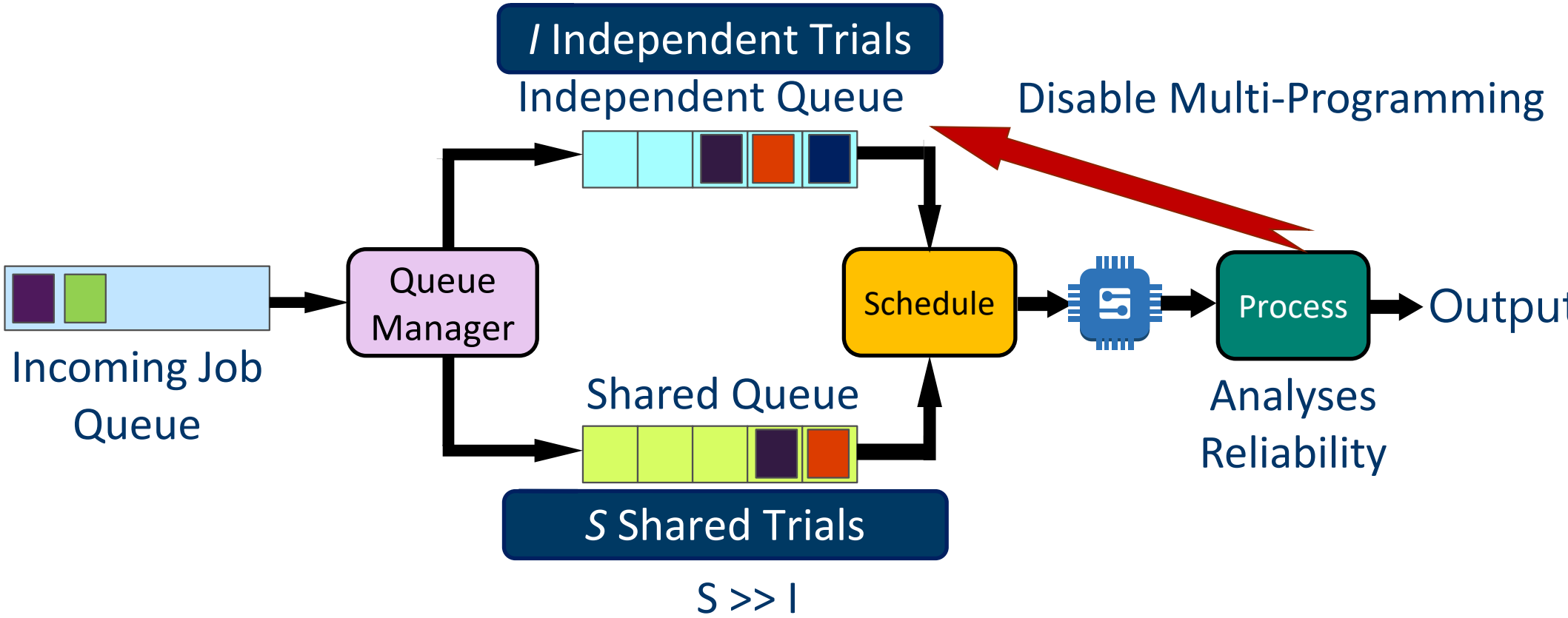
AMP Design Implementation



AMP Design Implementation



AMP Design Implementation

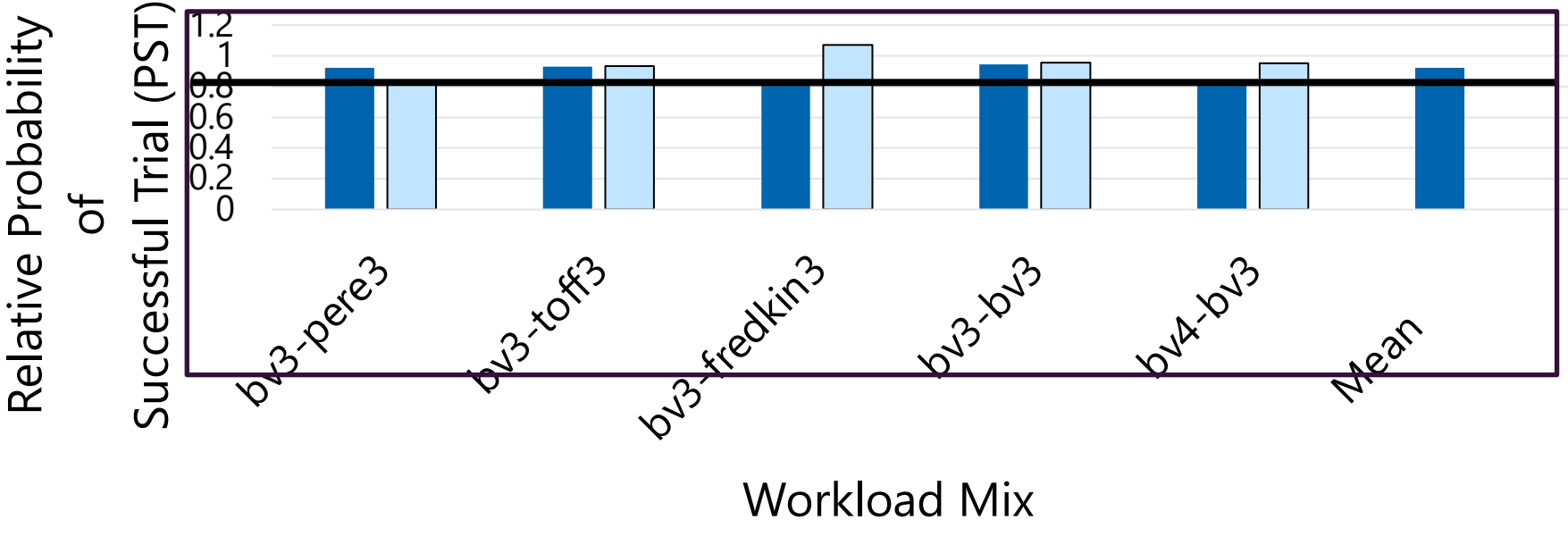


Multi-programming is adaptive to mitigate severe impact on reliability

Outline

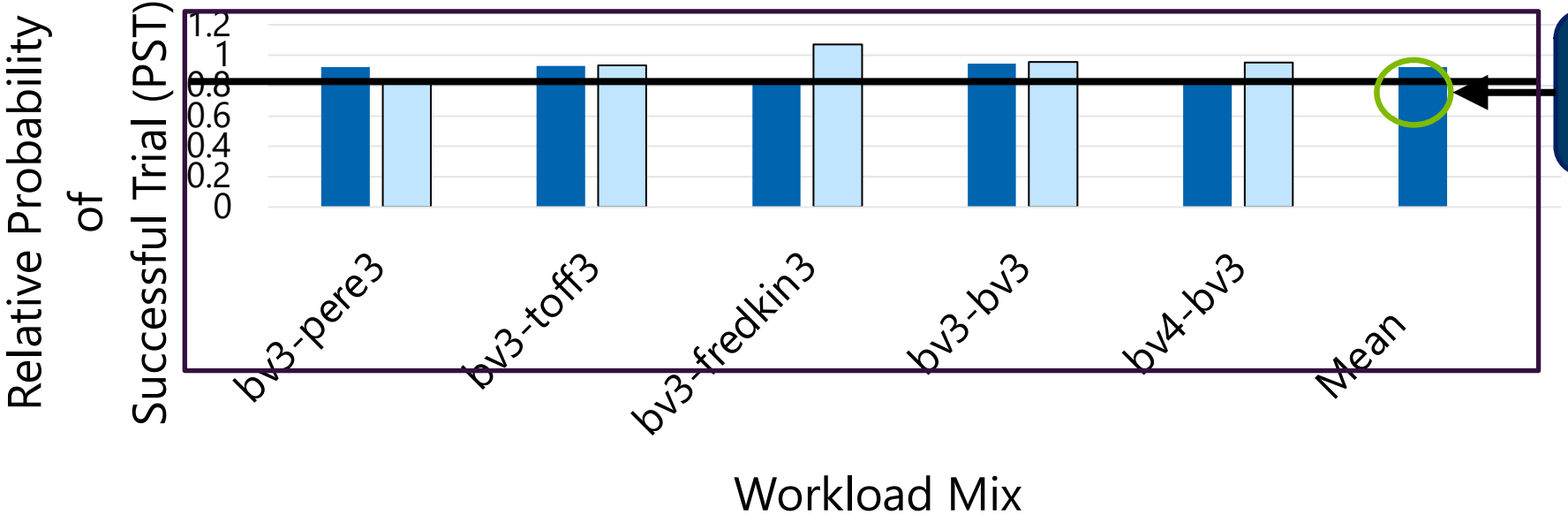
- Introduction
- Background and Motivation
- Policies for Multi-Programming
- Evaluation Methodology
- Adaptive Multi-Programming Design
- Results and Conclusion

Final Results



Mode: S=6K, I=2K

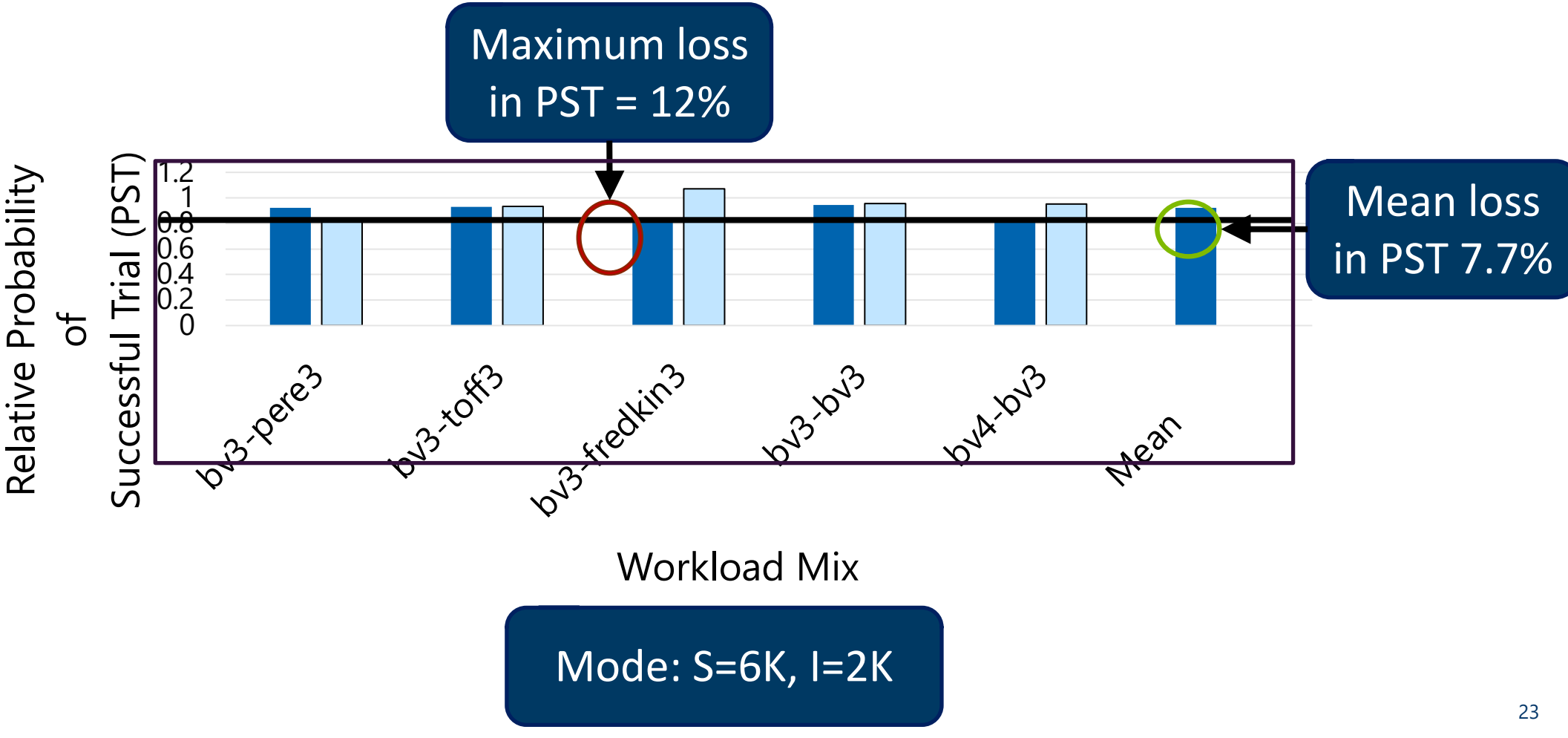
Final Results



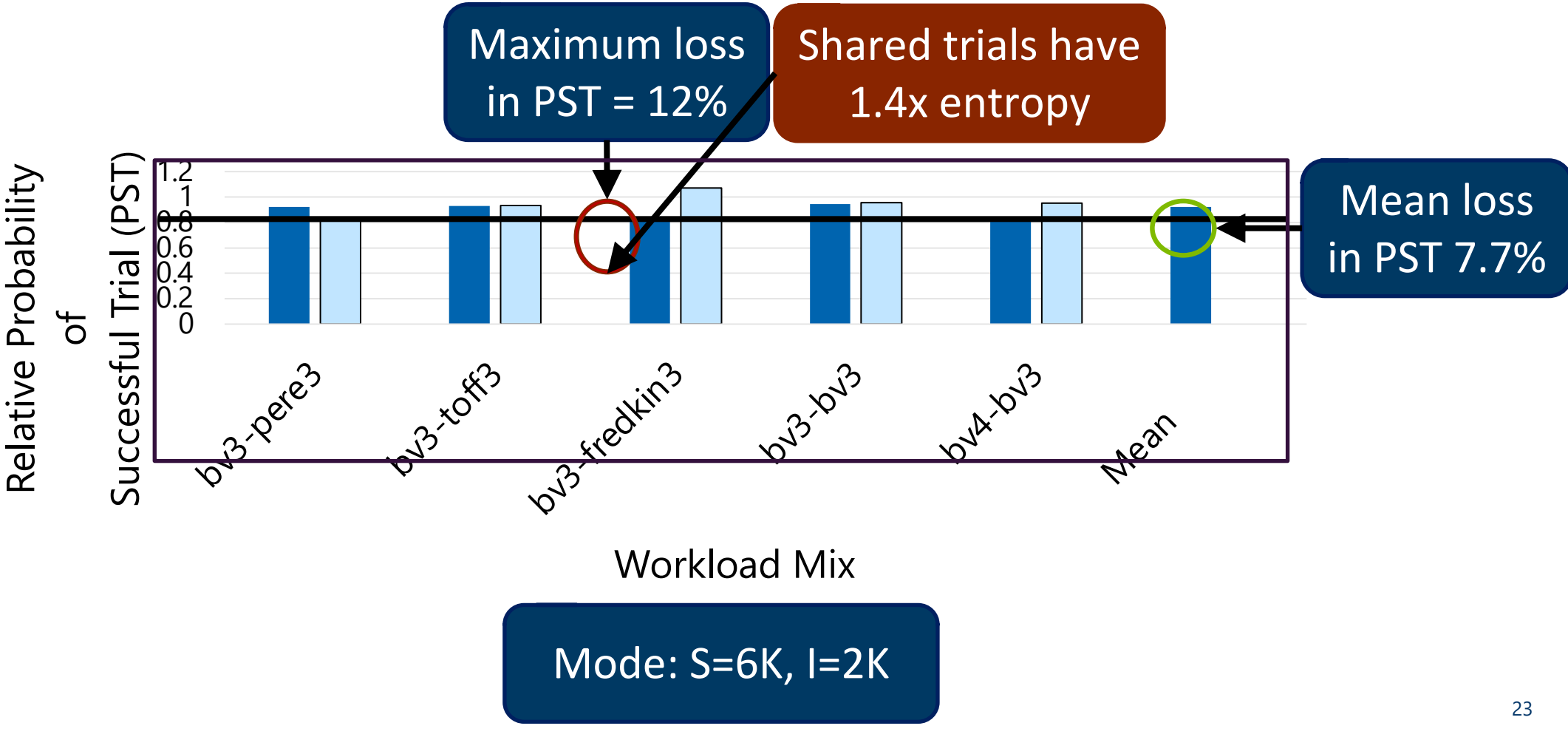
Mean loss in PST 7.7%

Mode: S=6K, I=2K

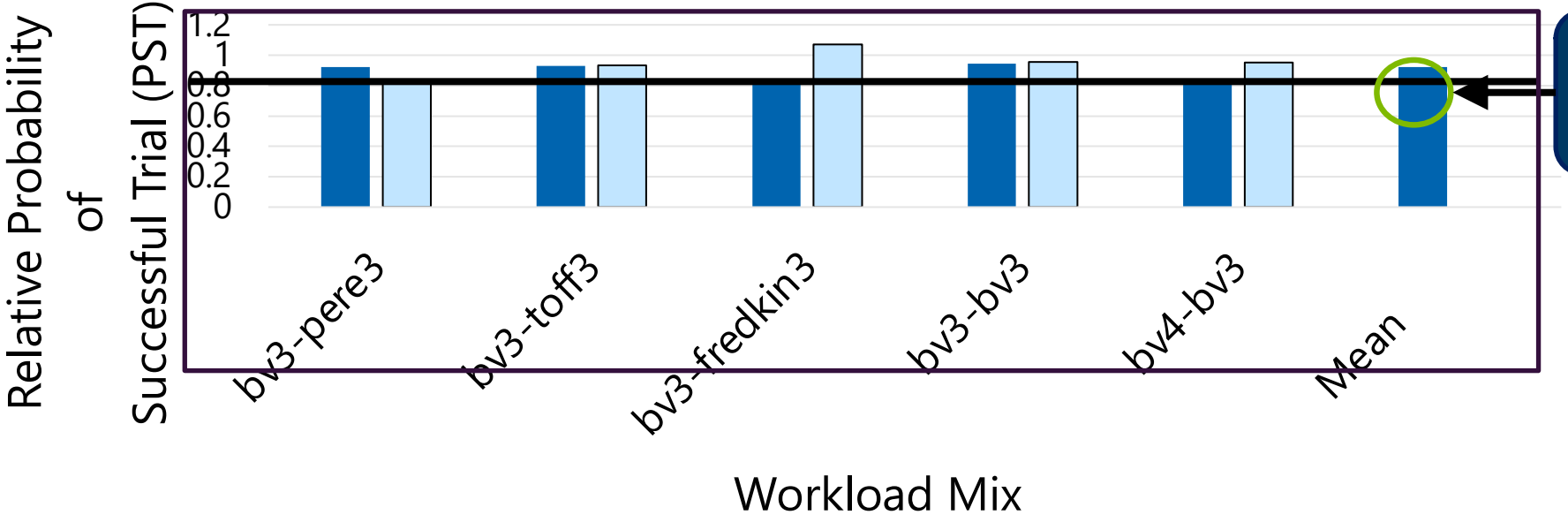
Final Results



Final Results



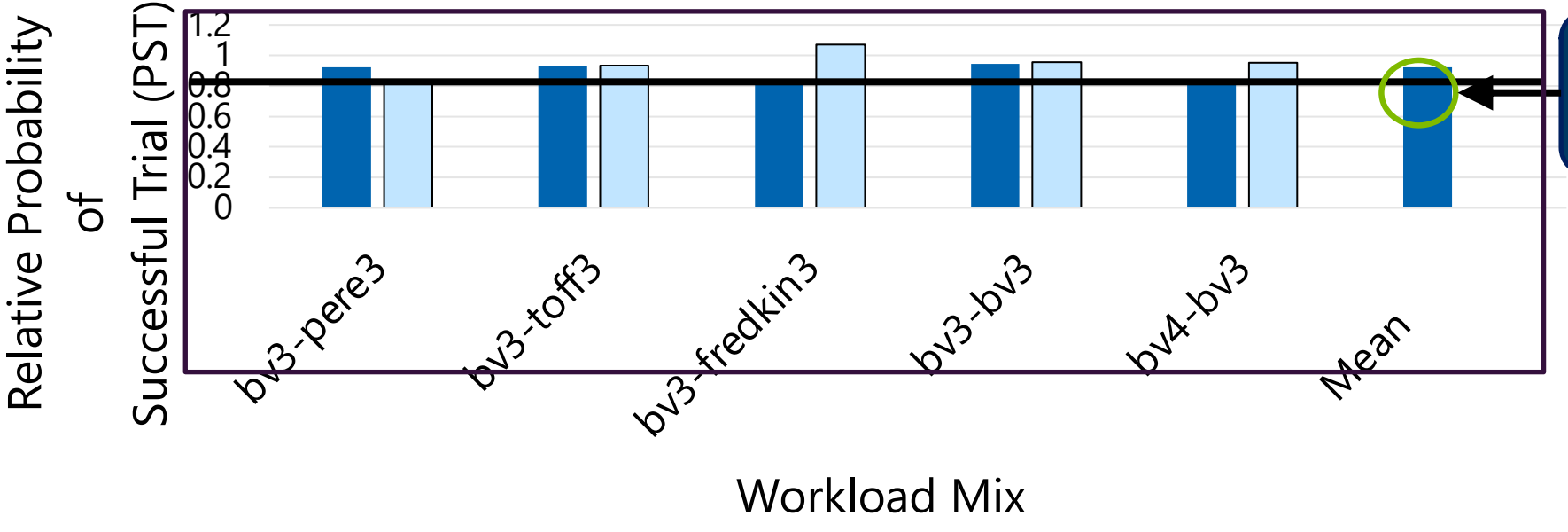
Final Results



Mean loss in PST 7.7%

Mode: S=6K, I=2K

Final Results



Mean loss in PST 7.7%

Mode: S=6K, I=2K

Throughput improved by 1.6x

Conclusion

- We proposed multi-programming to improve NISQ machine throughput and utilization

Conclusion

- We proposed multi-programming to improve NISQ machine throughput and utilization

- We designed scalable policies for fair resource allocation, minimizing interference, and adaptive multi-programming

Conclusion

- We proposed multi-programming to improve NISQ machine throughput and utilization
- We designed scalable policies for fair resource allocation, minimizing interference, and adaptive multi-programming
- Our solutions can be implemented by both user and service provider

Conclusion

- We proposed multi-programming to improve NISQ machine throughput and utilization
- We designed scalable policies for fair resource allocation, minimizing interference, and adaptive multi-programming
- Our solutions can be implemented by both user and service provider
- Machine throughput can be improved up to 2x with minimal loss in PST

Thank You!