FACULTY OF ENGINEERING AND ARCHITECTURE

Reliability-Aware Scheduling on Heterogeneous Multicore Processors

Ajeya Naithani, Stijn Eyerman, and Lieven Eeckhout

GHENT UNIVERSITY

intel

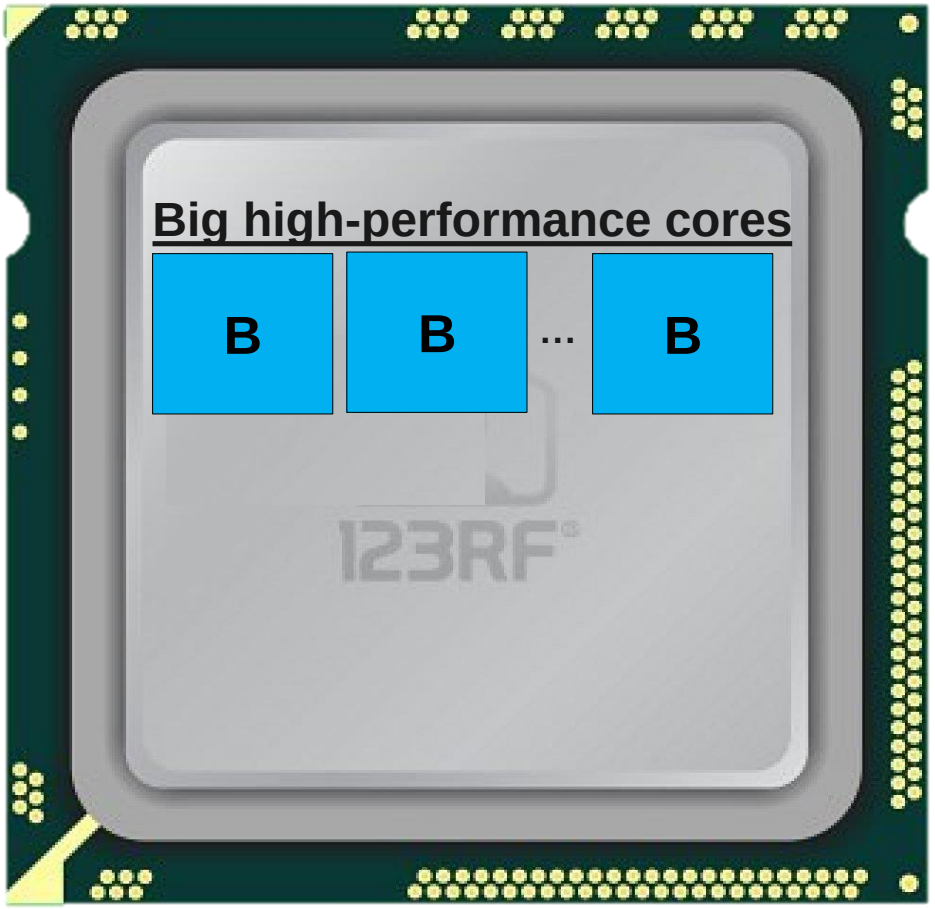
Benefits of Heterogeneous Multicores

Benefits of Heterogeneous Multicores

1. Multiple core types

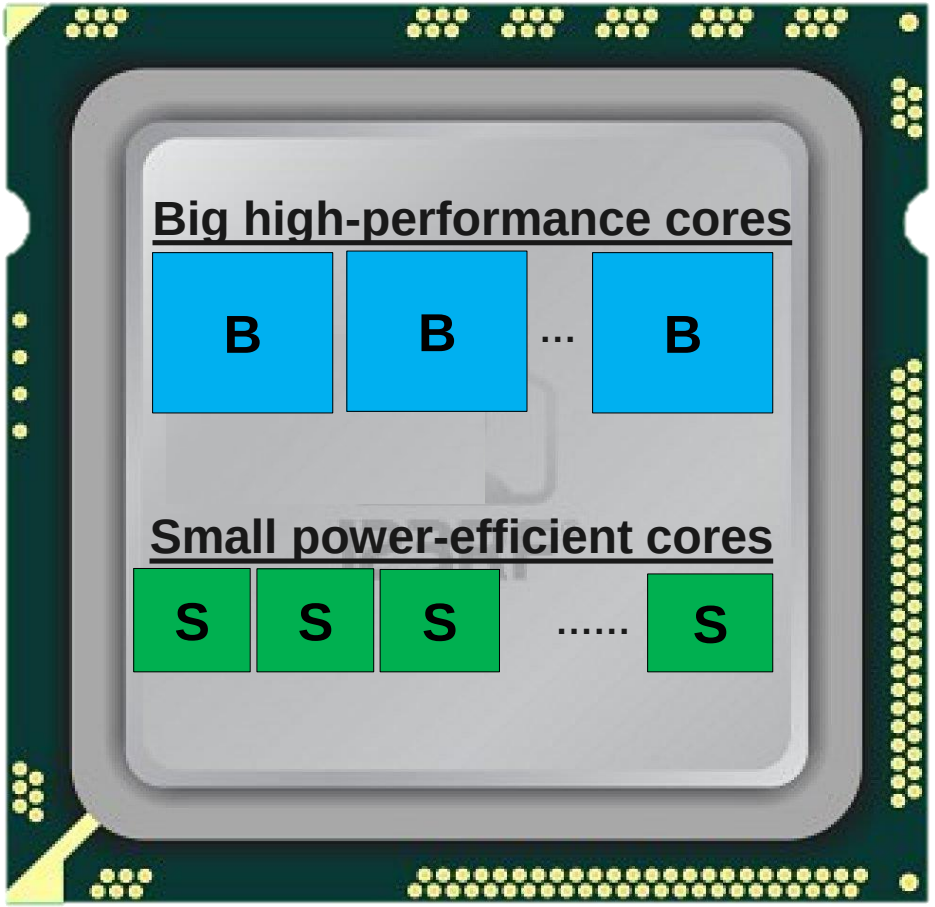
Benefits of Heterogeneous Multicores

1. Multiple core types



Benefits of Heterogeneous Multicores

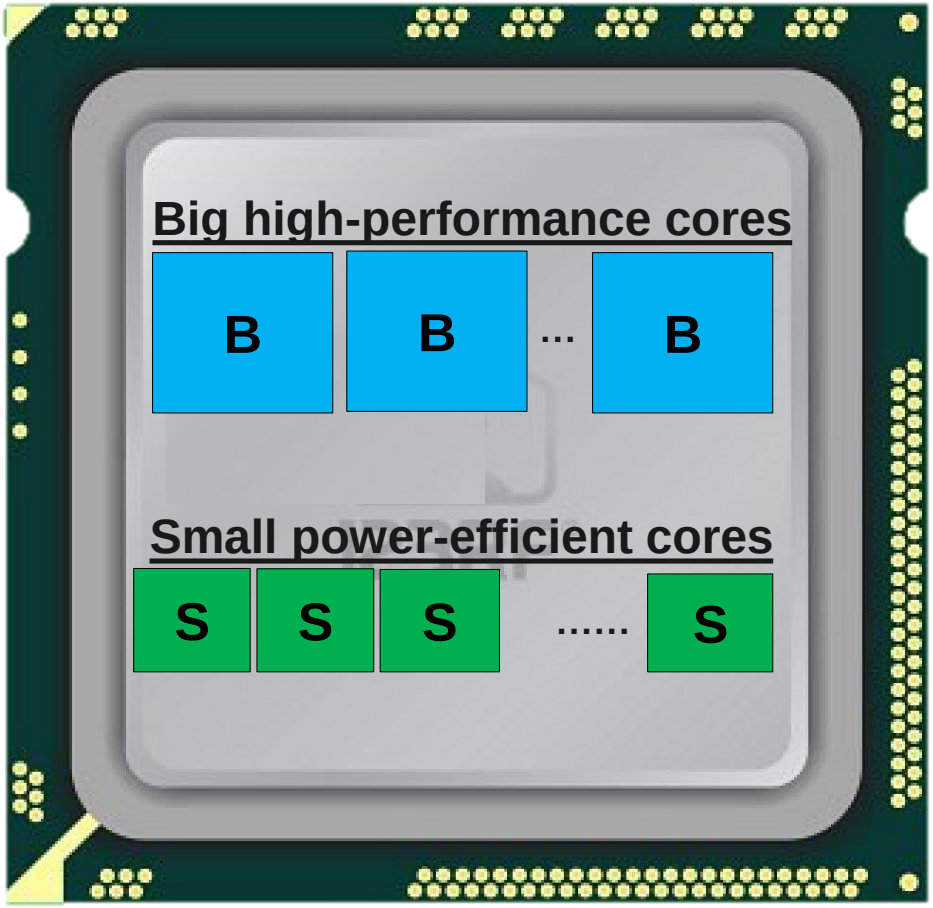
1. Multiple core types



Benefits of Heterogeneous Multicores

1. Multiple core types

2. Well established power benefits*

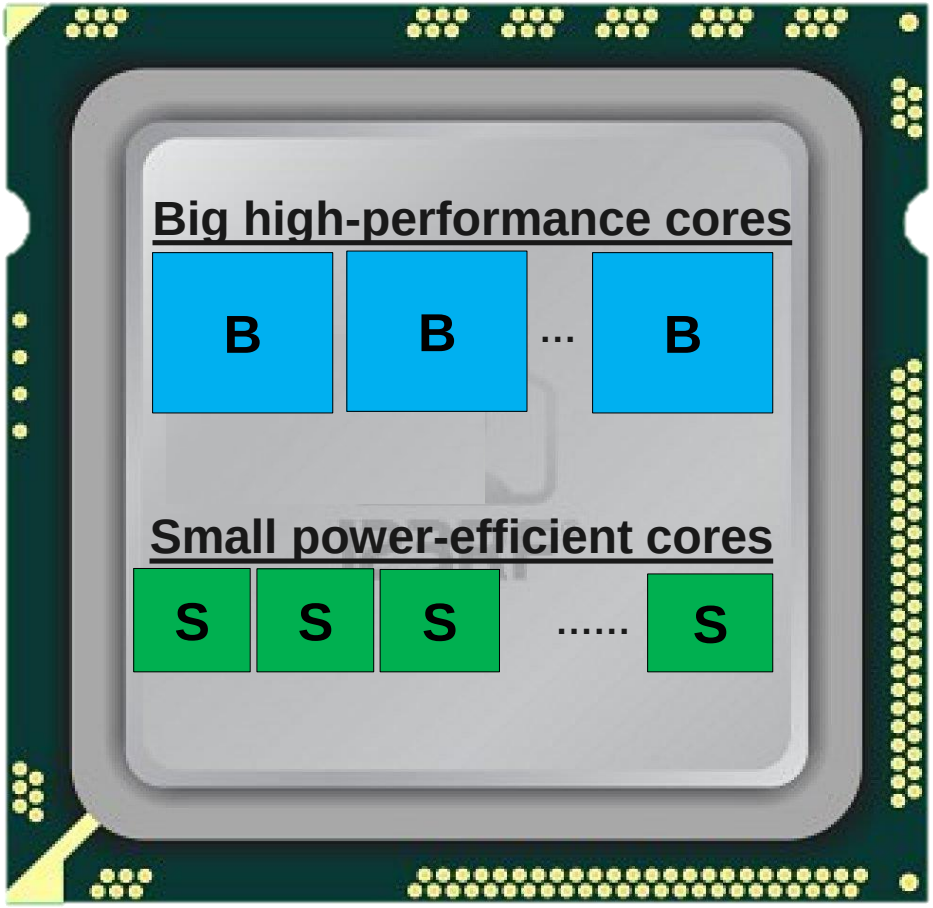


*[Kumar et al. MICRO'03, ISCA'04]

Benefits of Heterogeneous Multicores

1. Multiple core types
2. Well established power benefits*
3. Well established scheduling techniques**

e.g. Big.LITTLE and Kal-El

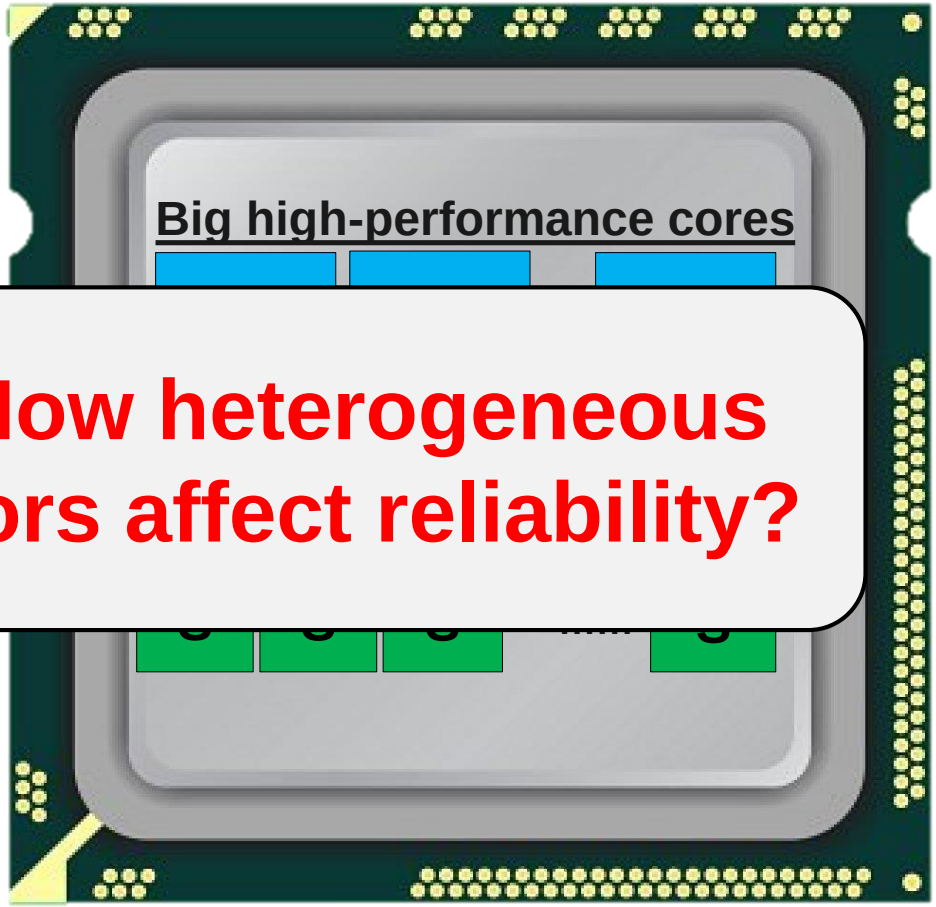


*[Kumar et al. MICRO'03, ISCA'04]

**[Van Craeynest et al. ISCA'12]

Benefits of Heterogeneous Multicores

1. Multiple core types
2. Well established power benefits*
3.



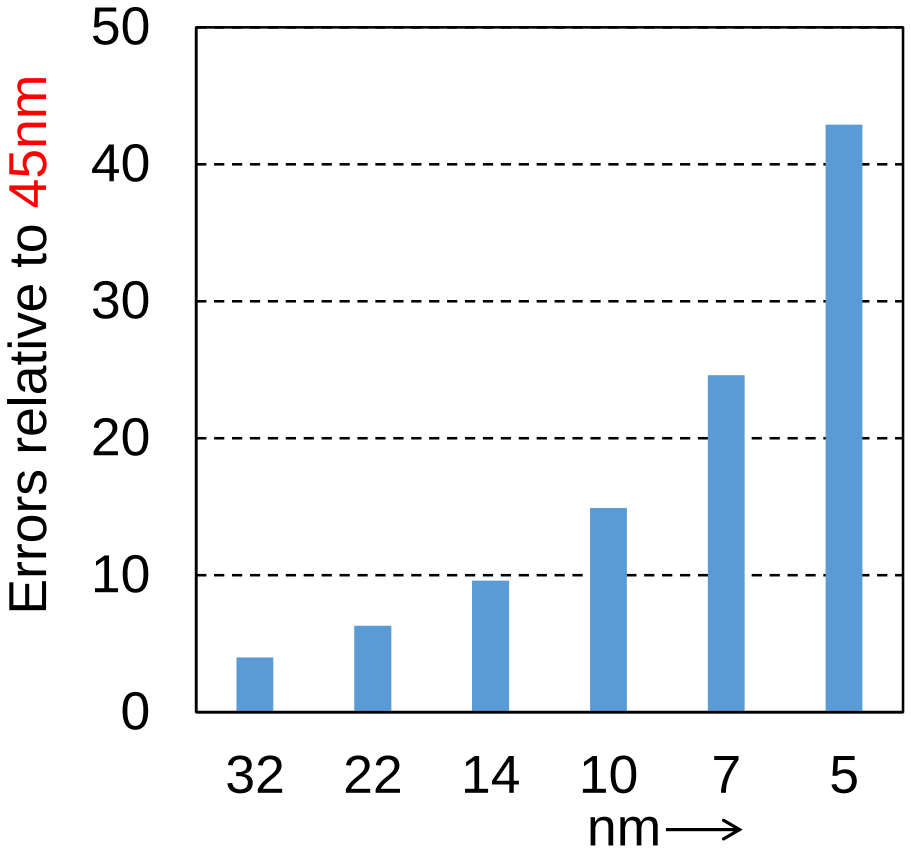
No prior work → How heterogeneous chip-multiprocessors affect reliability?

e.g. Big.LITTLE and Kal-El

*[Kumar et al. MICRO'03, ISCA'04]

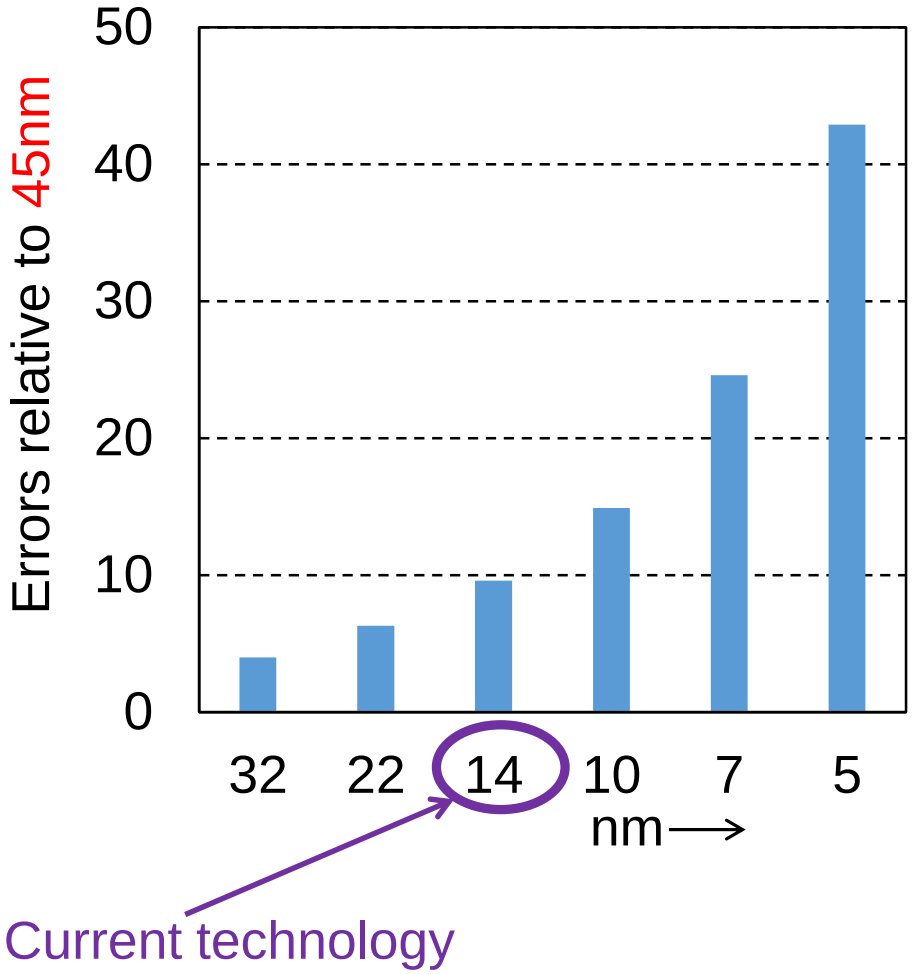
**[Van Craeynest et al. ISCA'12]

Errors vs. Technology



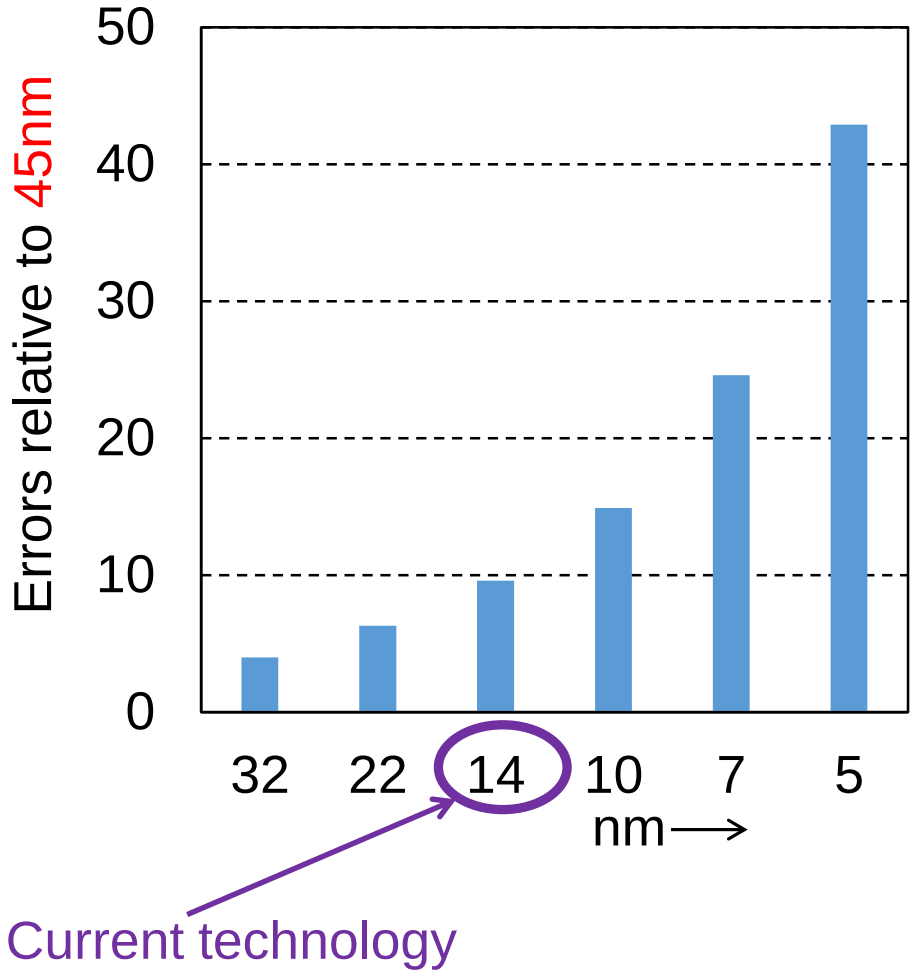
[Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, 2012]

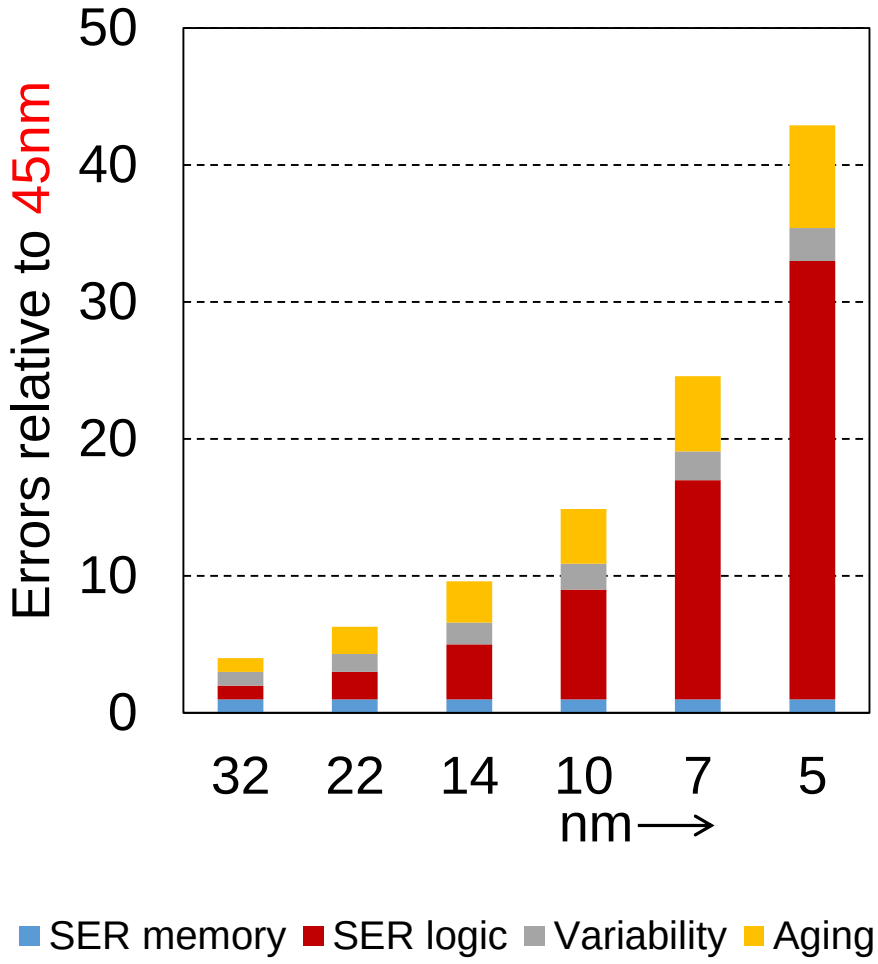
Errors vs. Technology



[Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, 2012]

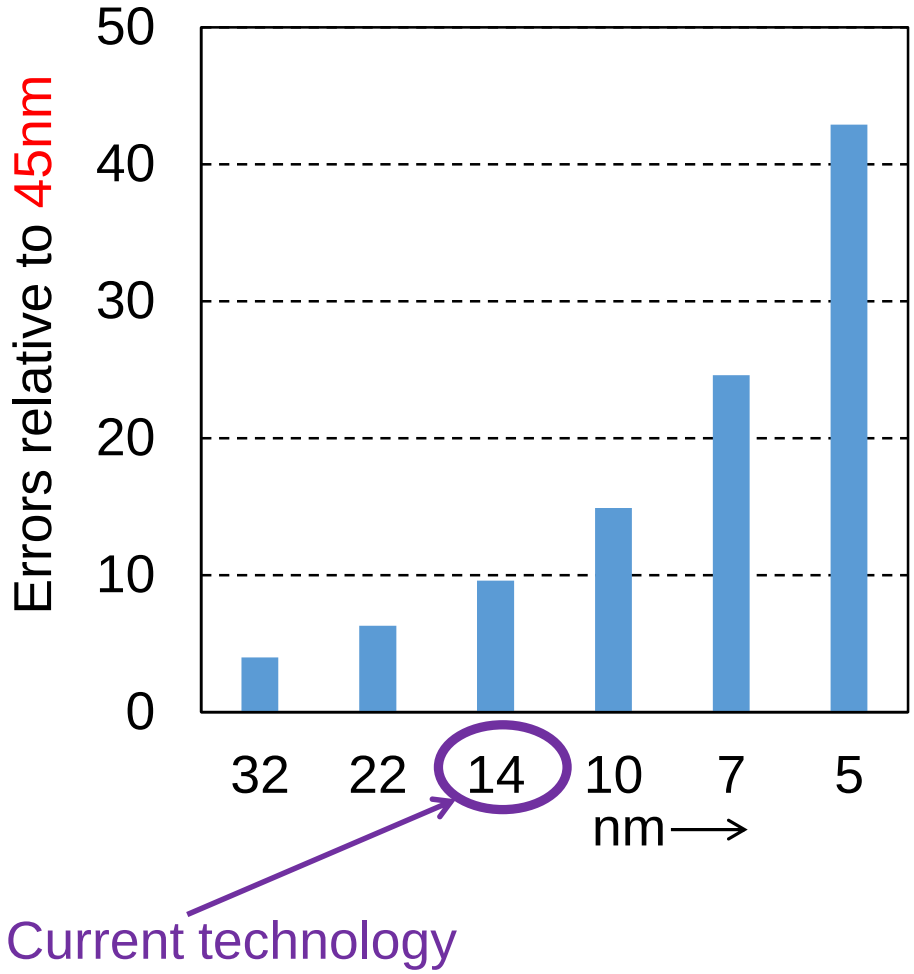
Errors vs. Technology

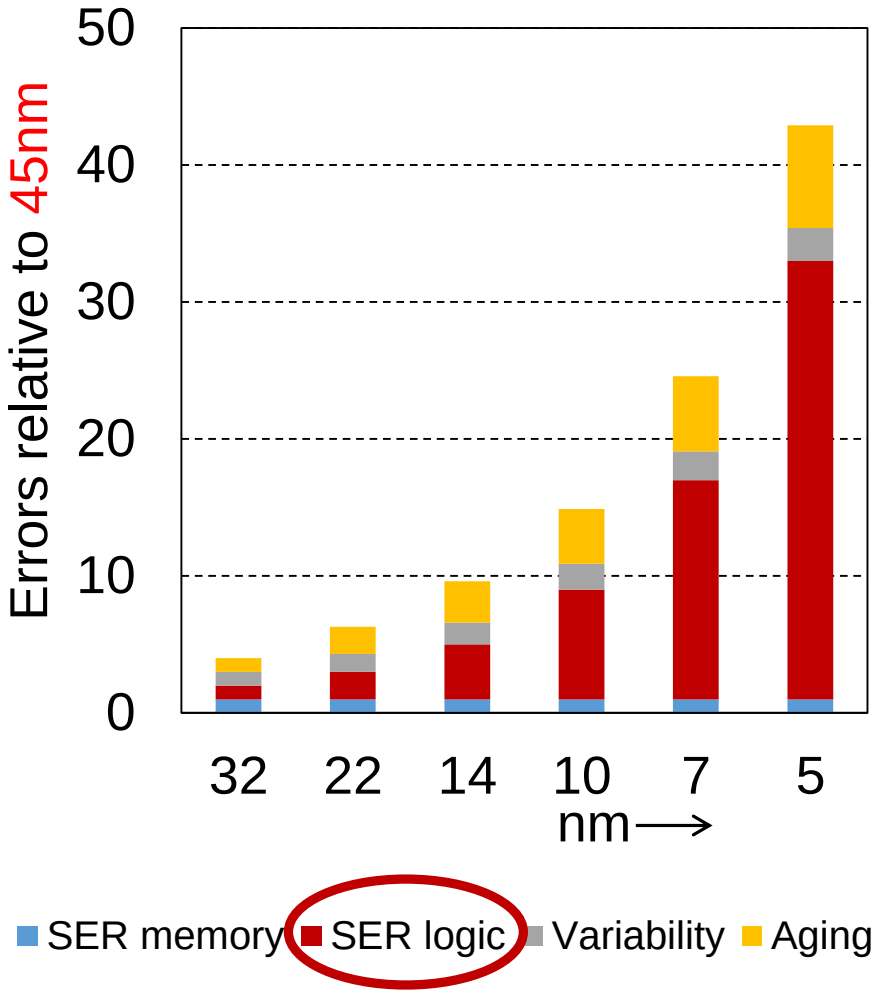




[Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, 2012]

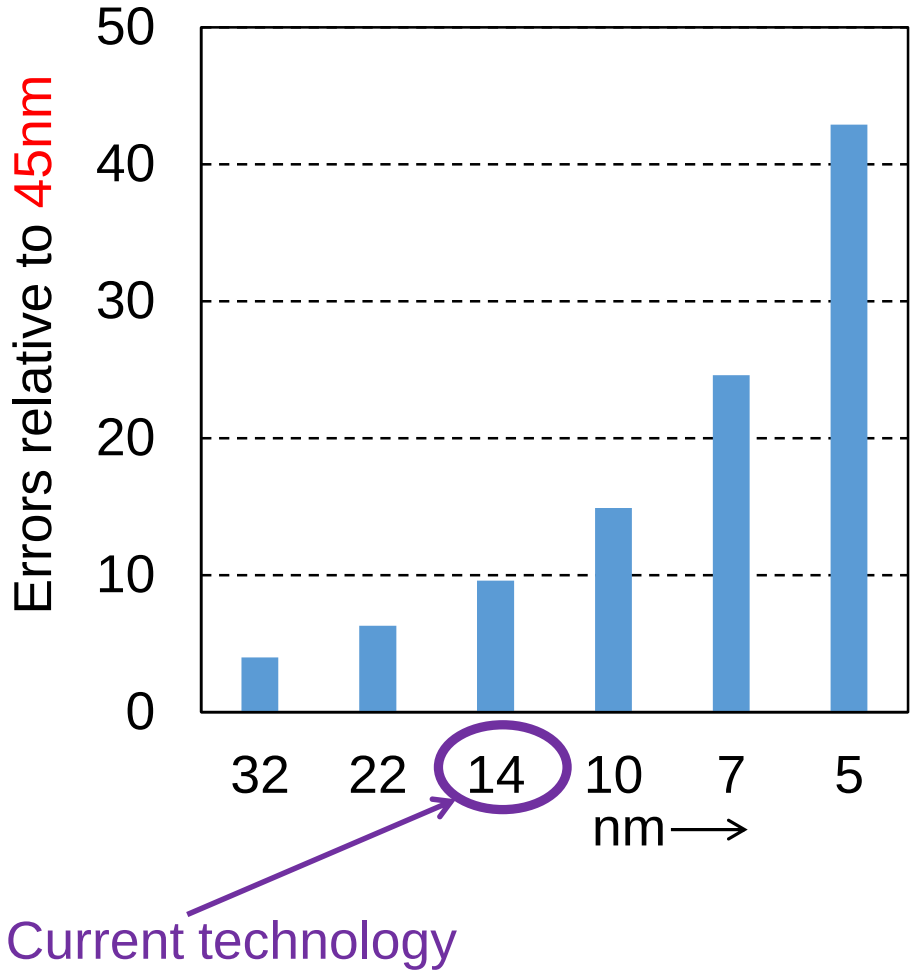
Errors vs. Technology

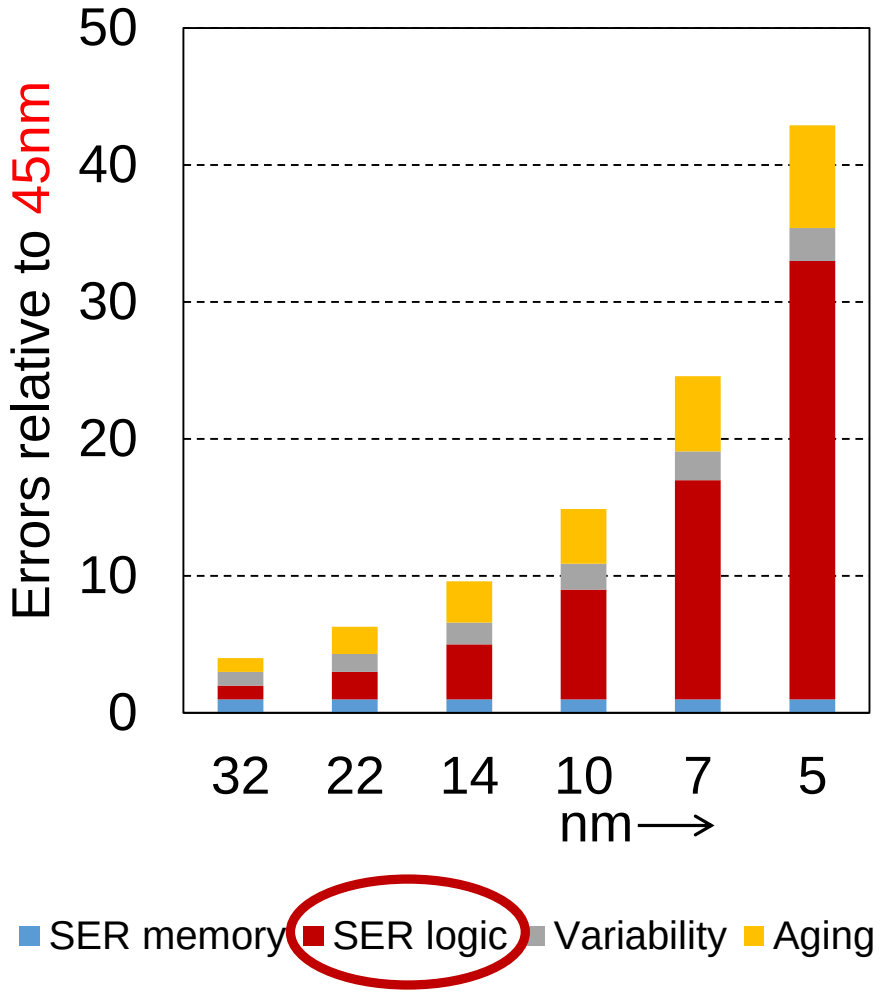




[Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, 2012]

Errors vs. Technology

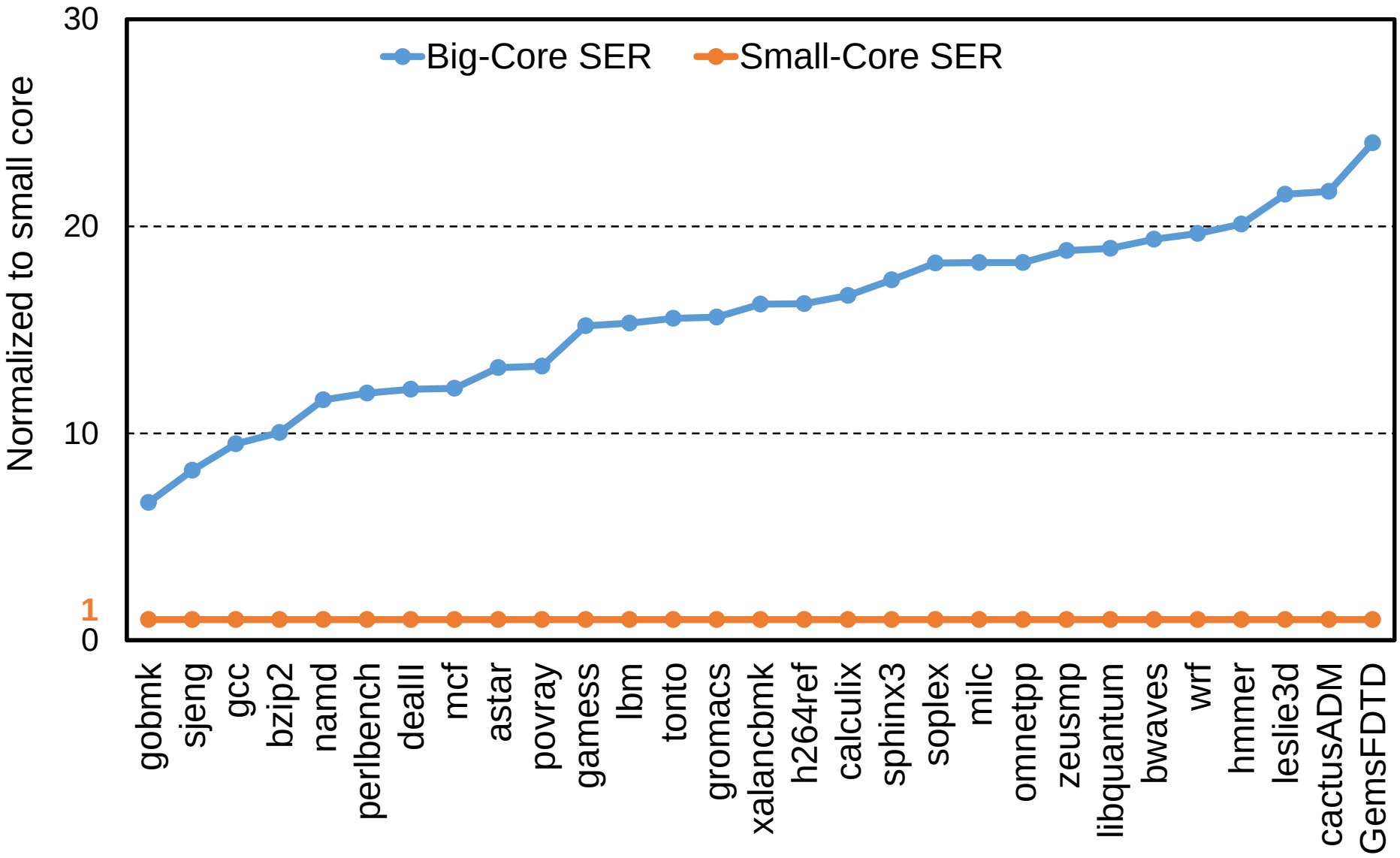




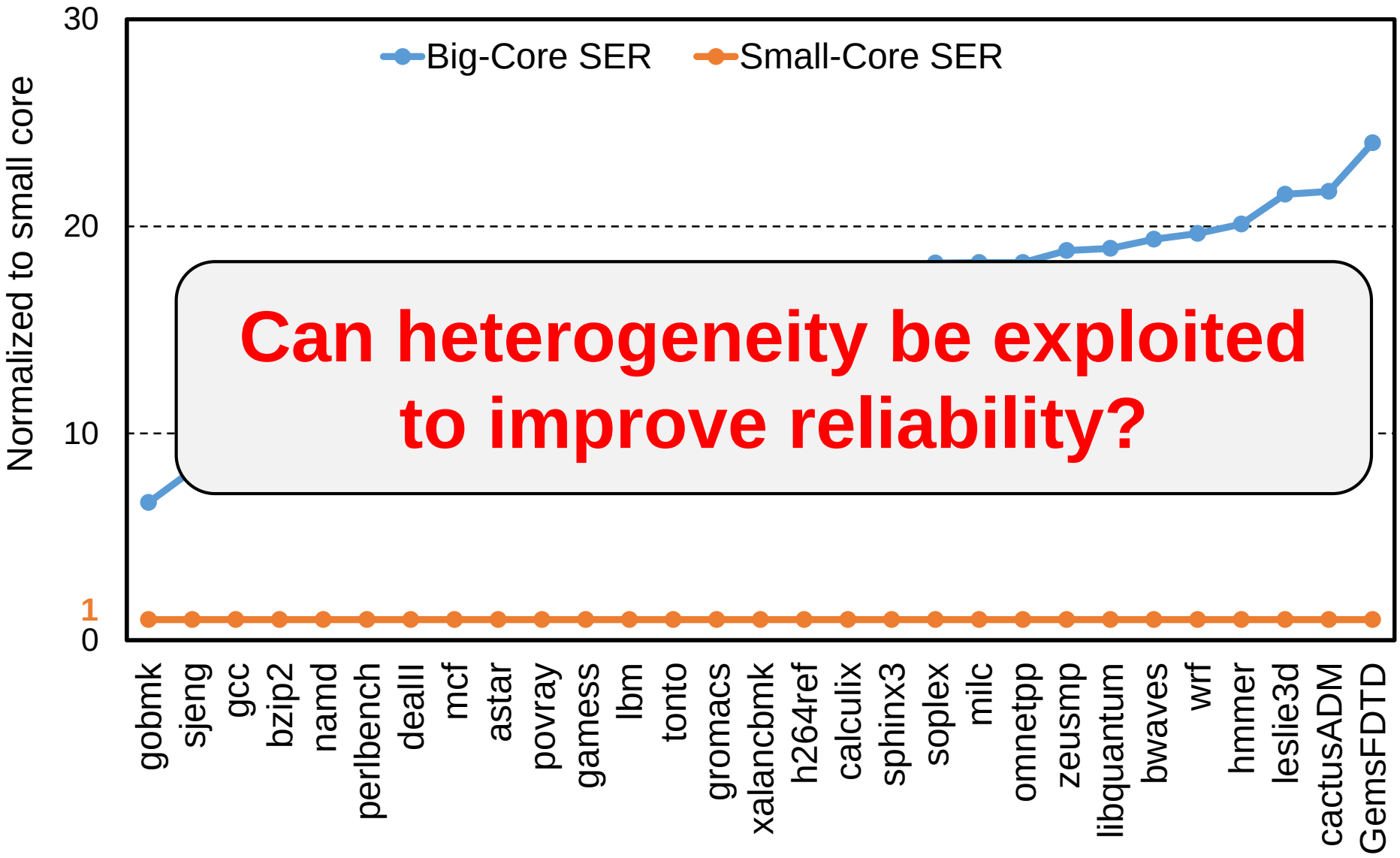
Soft errors are the major source of errors!

[Source: Inter-Agency Workshop on HPC Resilience at Extreme Scale, 2012]

Soft Errors vs. Core Type



Soft Errors vs. Core Type



Key Contributions

Key Contributions

1. Analyze the difference in reliability characteristics between big and small cores

Key Contributions

1. Analyze the difference in reliability characteristics between big and small cores

2. Show the potential for improving reliability through scheduling

Key Contributions

1. Analyze the difference in reliability characteristics between big and small cores

2. Show the potential for improving reliability through scheduling

3. Introduce a novel metric called System-level Soft Error Rate (SSER) for multiprogram workloads on (heterogeneous) multicore processors

Key Contributions

1. Analyze the difference in reliability characteristics between big and small cores

2. Show the potential for improving reliability through scheduling

3. Introduce a novel metric called System-level Soft Error Rate (SSER) for multiprogram workloads on (heterogeneous) multicore processors

4. Propose a dynamic scheduler to optimize reliability
   -- Scheduler improves reliability by 25.4%

Talk Outline

Talk Outline

1. Background and Terminology

Talk Outline

1. Background and Terminology
2. Analyzing Reliability on Heterogeneous Multicores

Talk Outline

1. Background and Terminology
2. Analyzing Reliability on Heterogeneous Multicores
3. Reliability Metric for Multiprogram Workloads

Talk Outline

1. Background and Terminology
2. Analyzing Reliability on Heterogeneous Multicores
3. Reliability Metric for Multiprogram Workloads
4. Reliability-Aware Scheduler

Talk Outline

1. Background and Terminology
2. Analyzing Reliability on Heterogeneous Multicores
3. Reliability Metric for Multiprogram Workloads
4. Reliability-Aware Scheduler
5. Evaluation

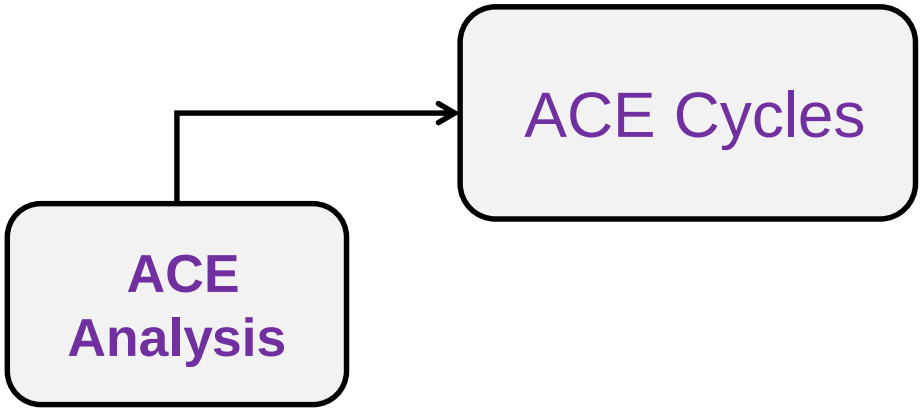
Talk Outline

1. Background and Terminology
2. Analyzing Reliability on Heterogeneous Multicores
3. Reliability Metric for Multiprogram Workloads
4. Reliability-Aware Scheduler
5. Evaluation
6. Conclusions

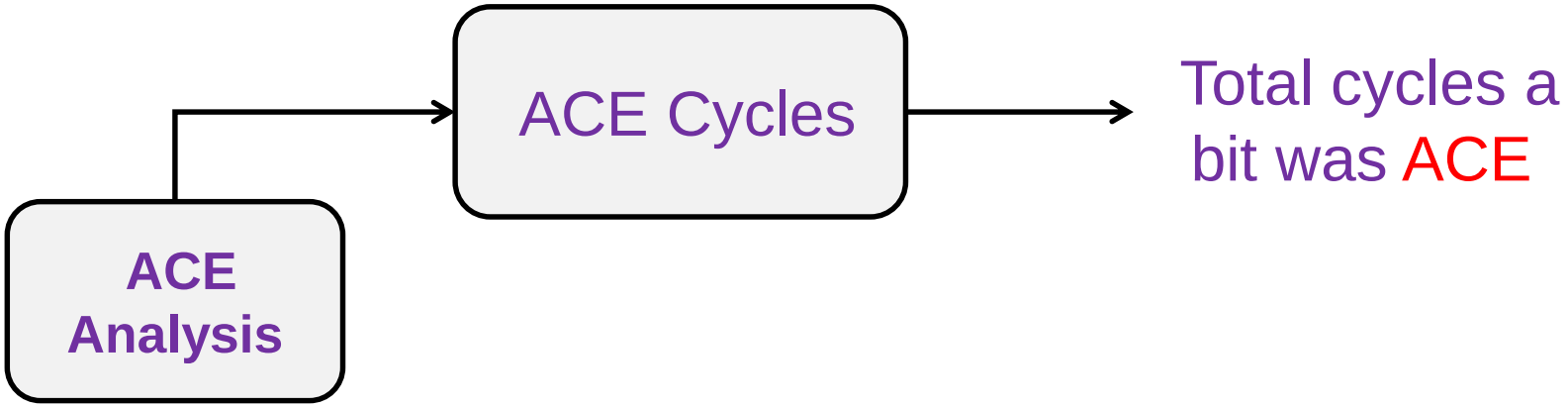
Estimating Soft Errors

ACE Analysis

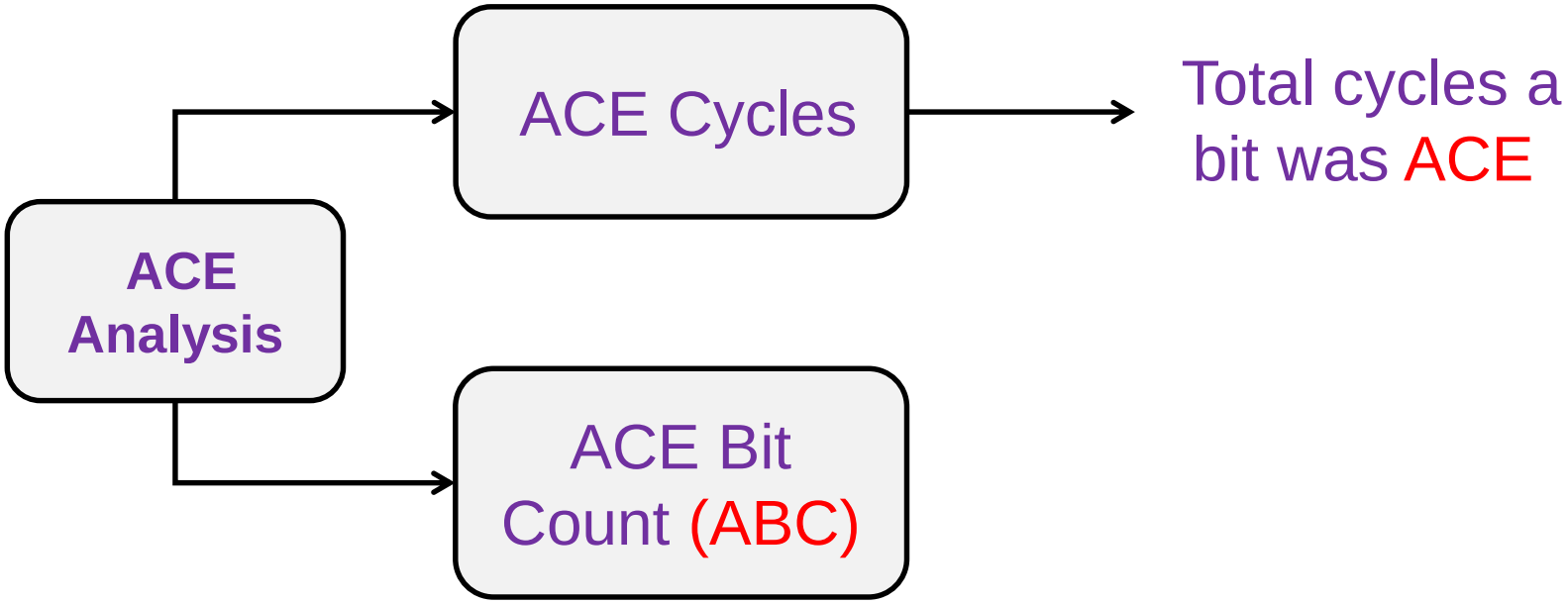
Estimating Soft Errors



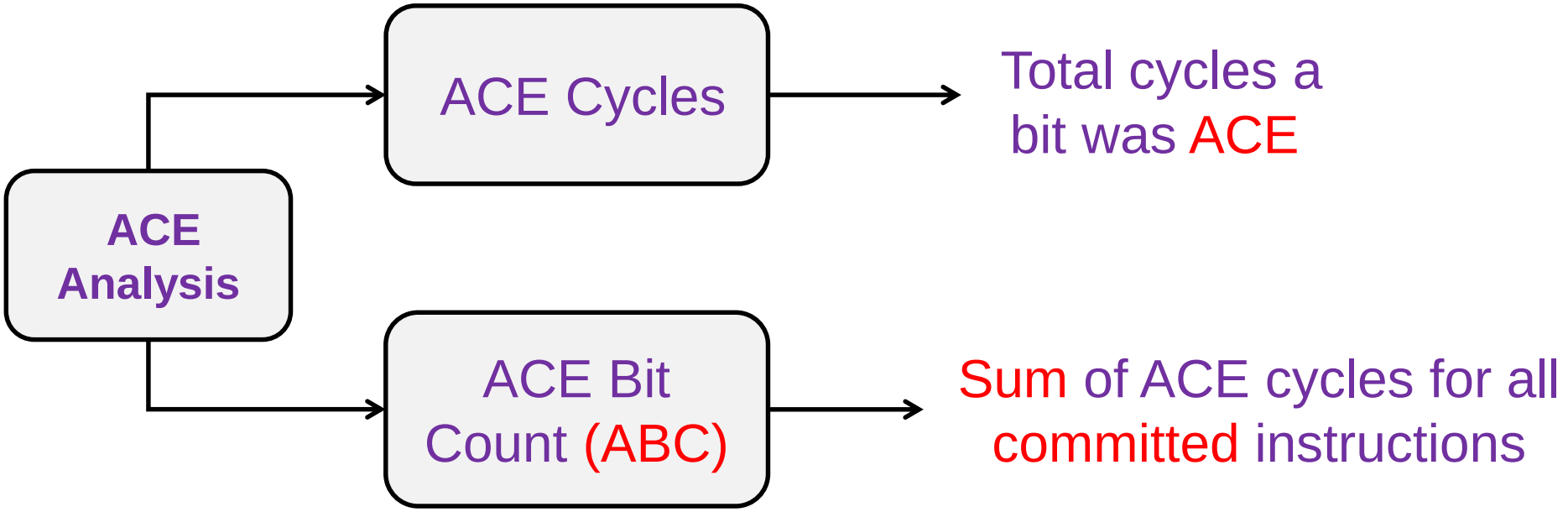
Estimating Soft Errors



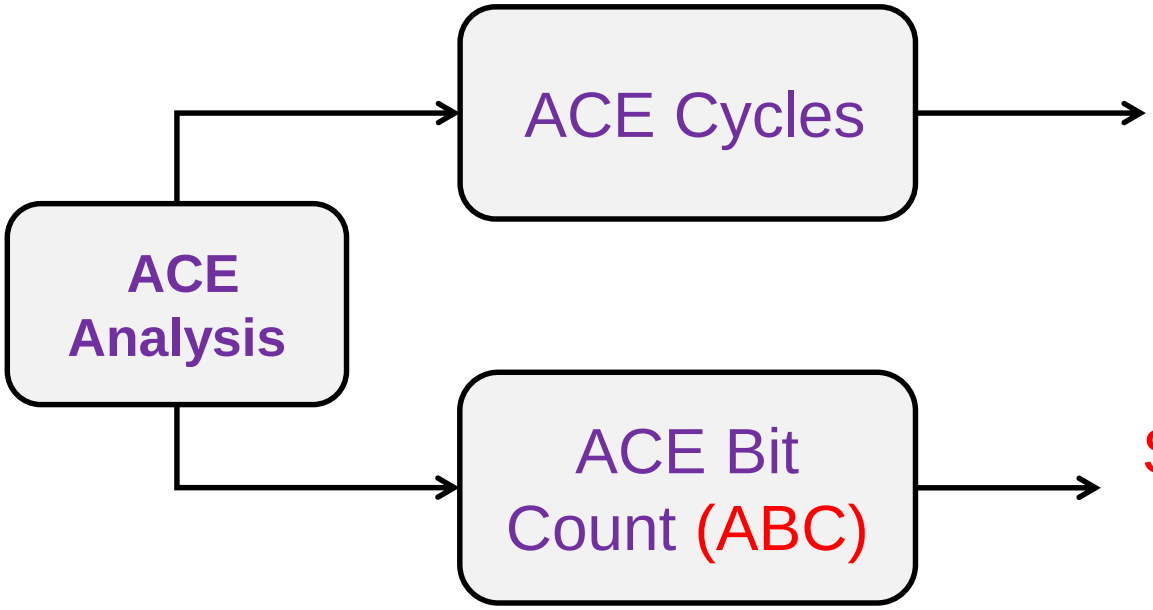
Estimating Soft Errors



Estimating Soft Errors



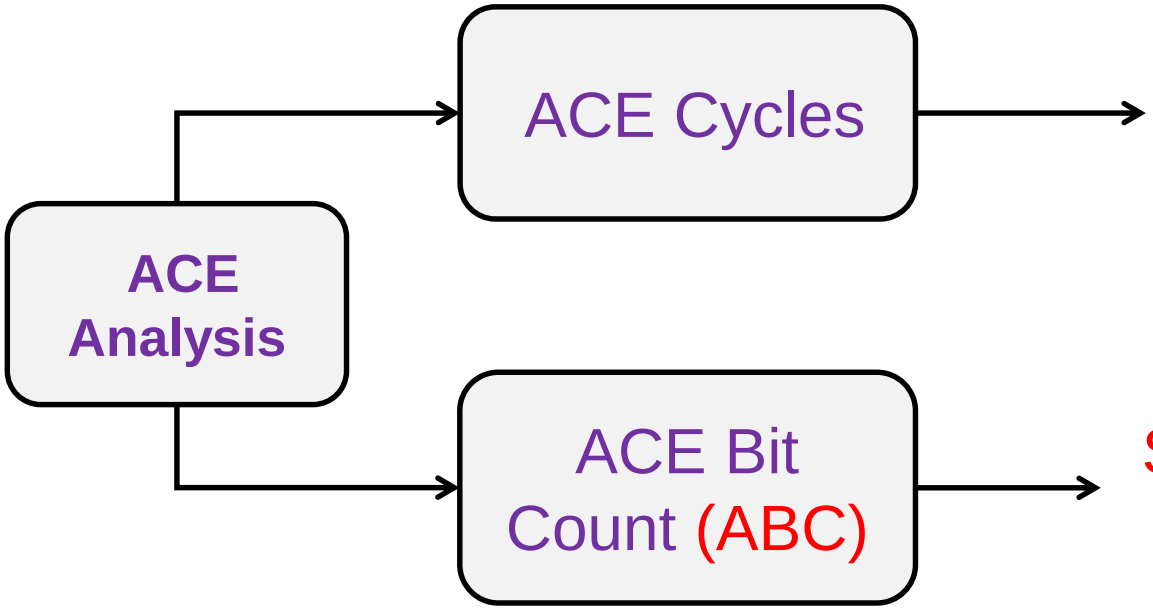
Estimating Soft Errors



$$\text{Soft Error Rate (SER)} = \frac{\text{ABC}}{\text{Total cycles}} \times \text{IFR}$$

IFR → Probability of a soft error per second

Estimating Soft Errors

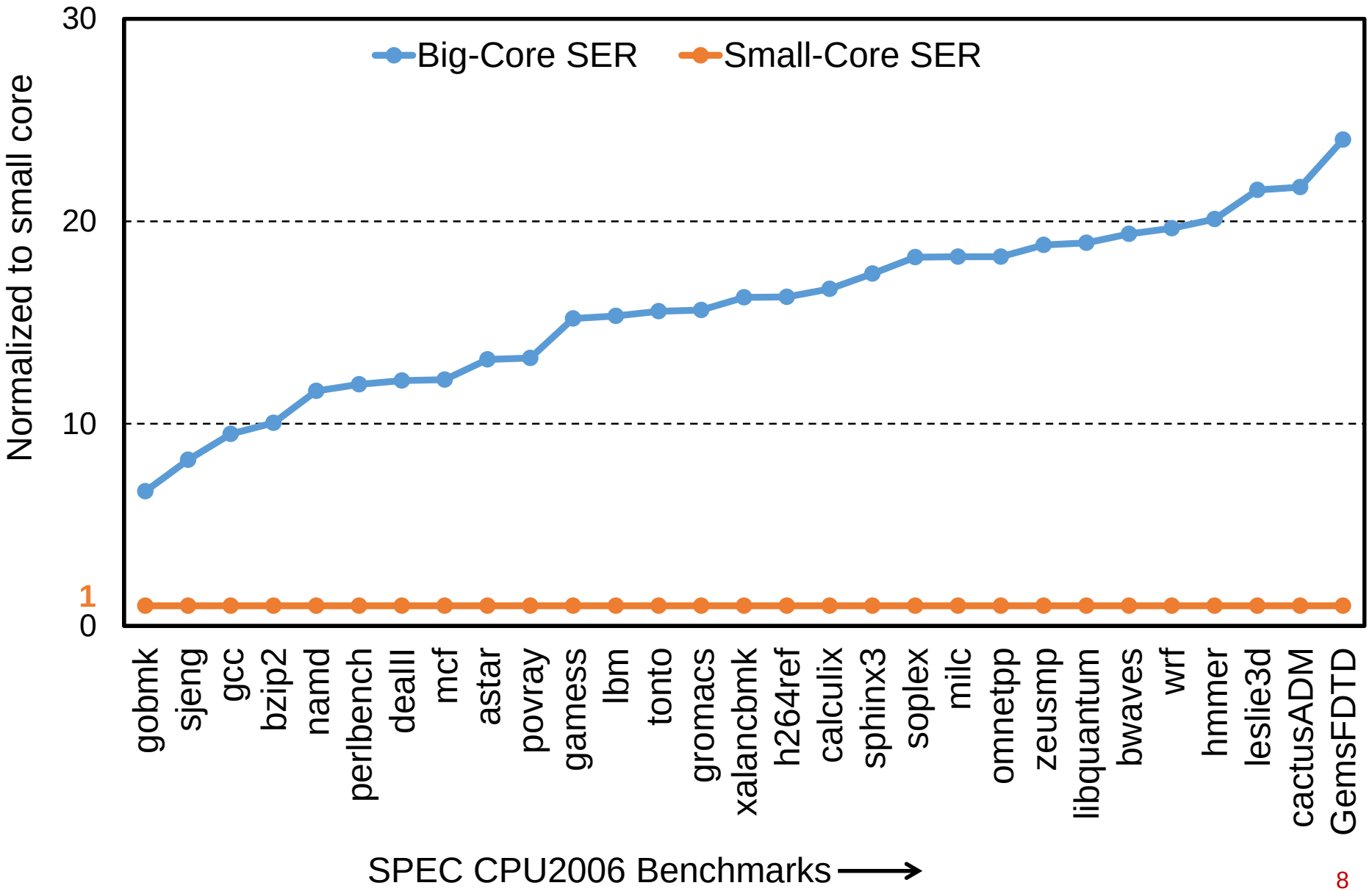


$$\text{Soft Error Rate (SER)} = \frac{\text{ABC}}{\text{Total cycles}} \times \text{IFR}$$

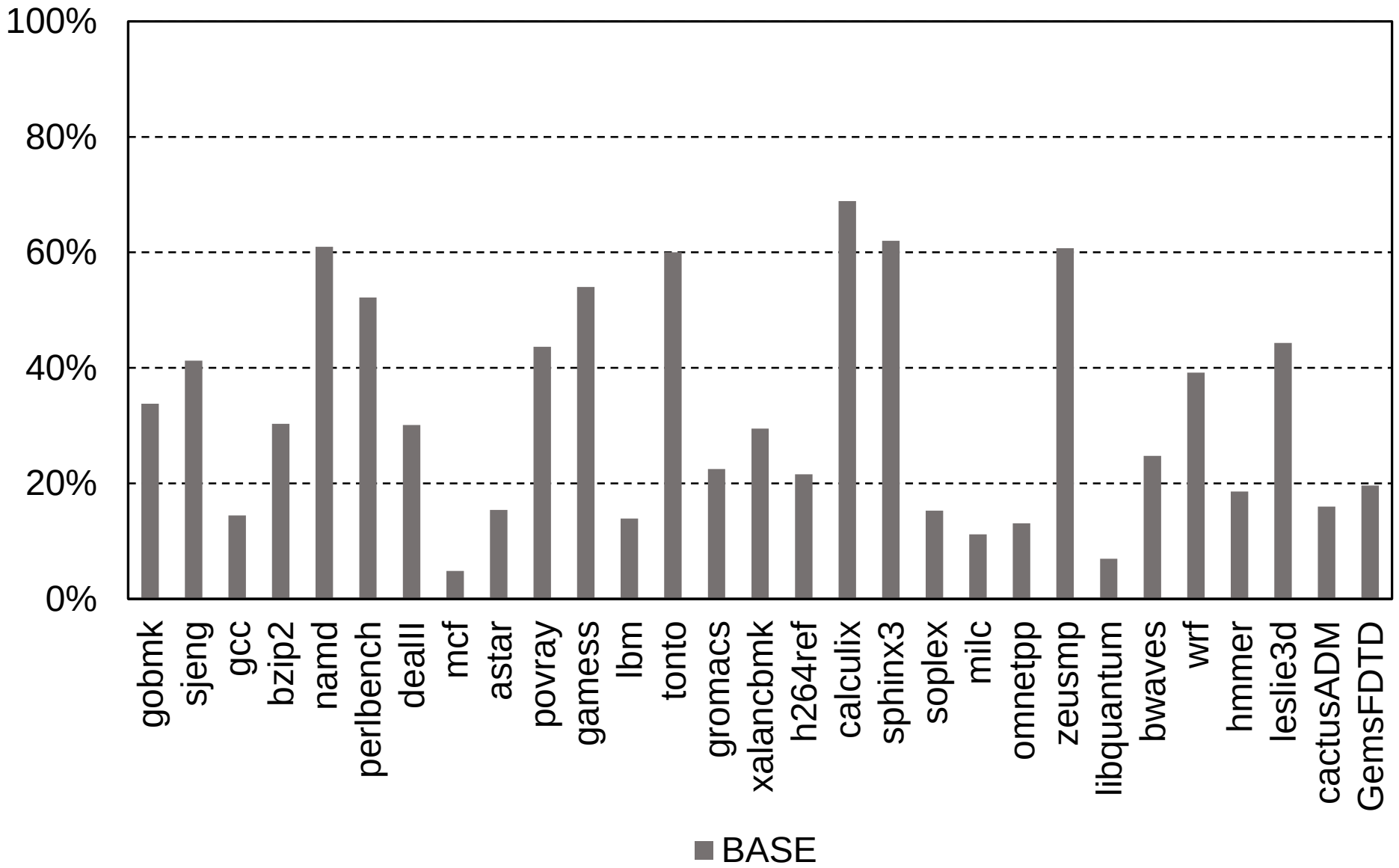
IFR → Probability of a soft error per second

$$\text{Architectural Vulnerability Factor (AVF)} = \frac{\text{ABC}}{\text{Total cycles} \times \text{Core size}}$$

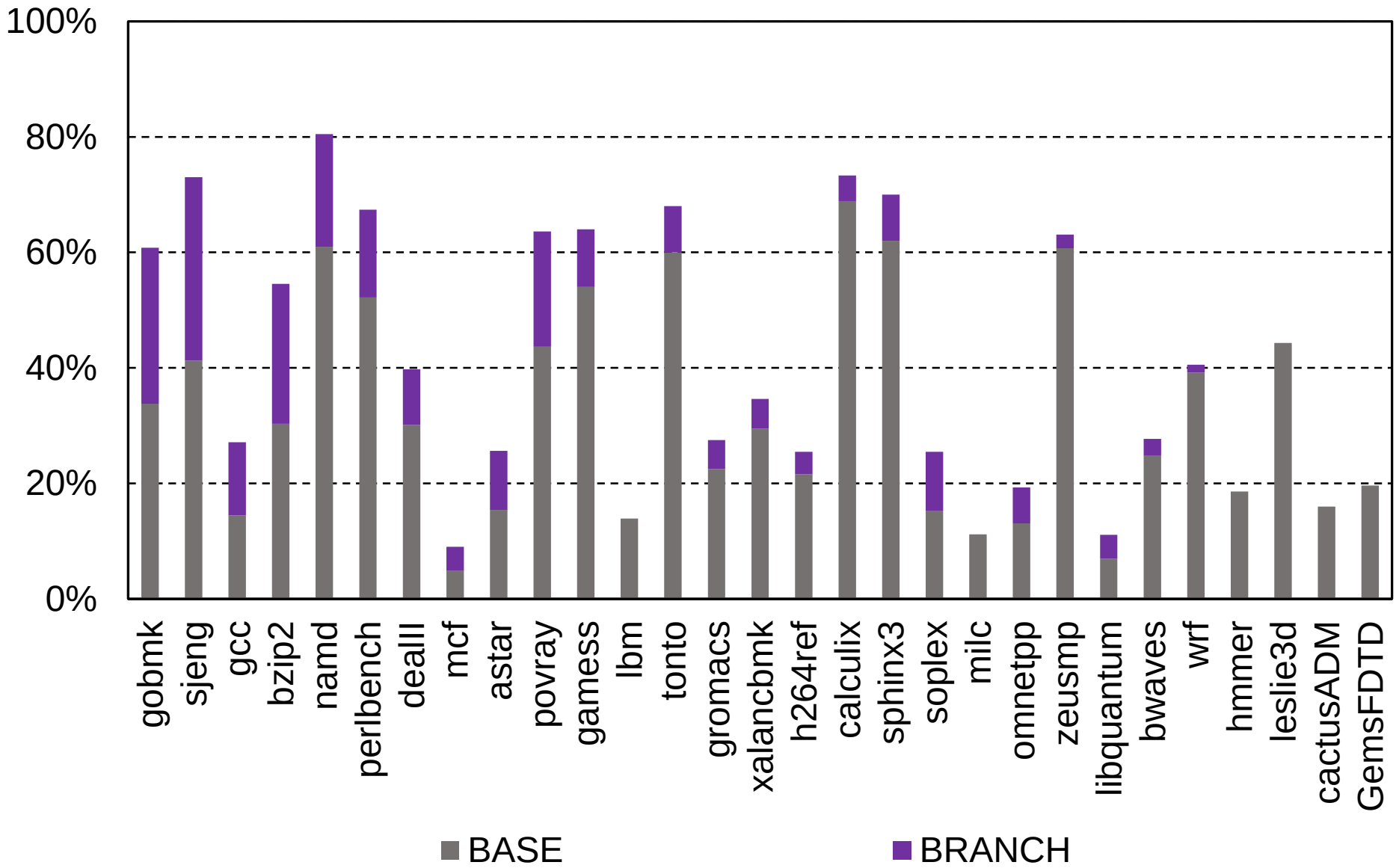
Vulnerability vs. Core Type



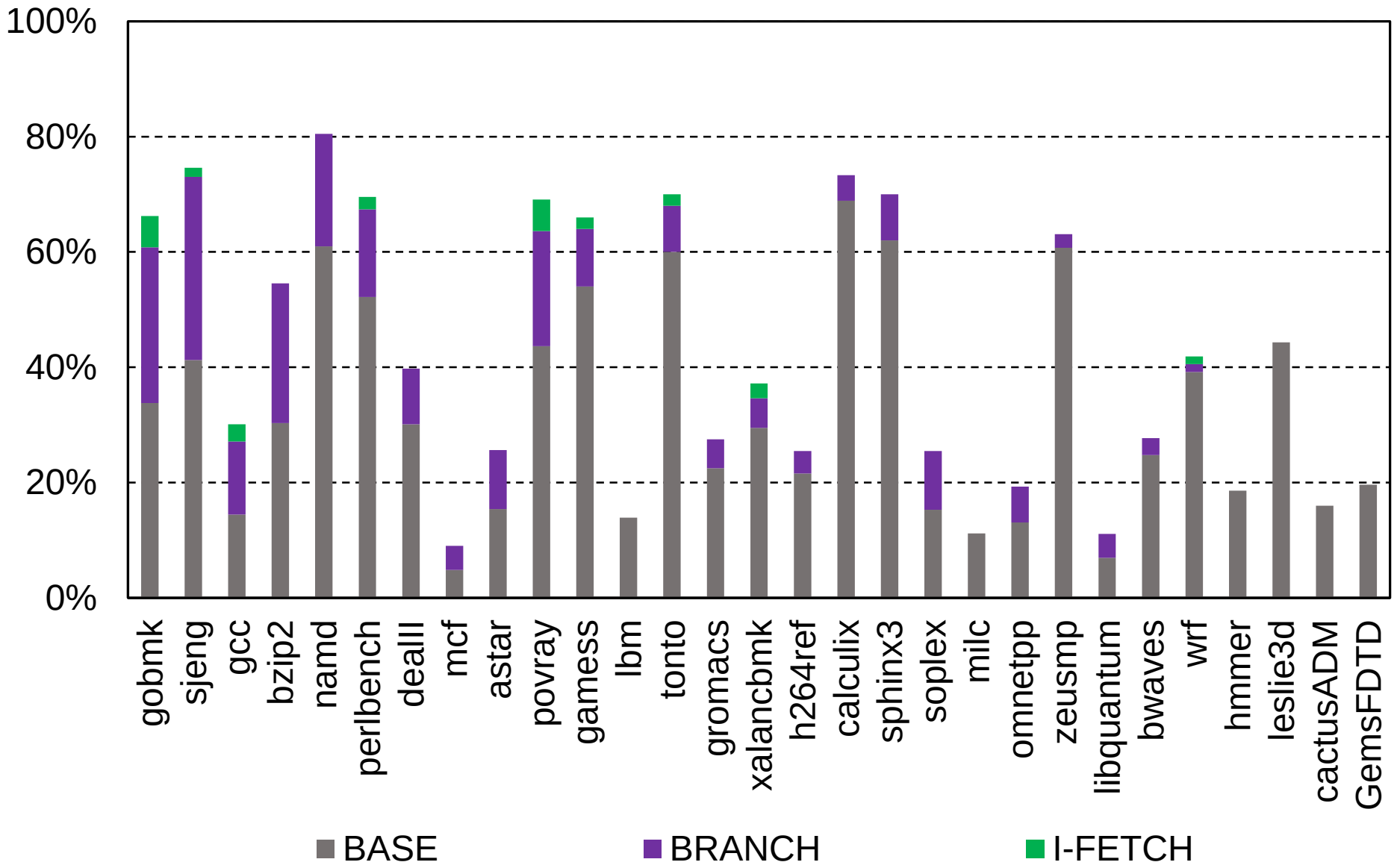
Vulnerability vs. CPI Stacks



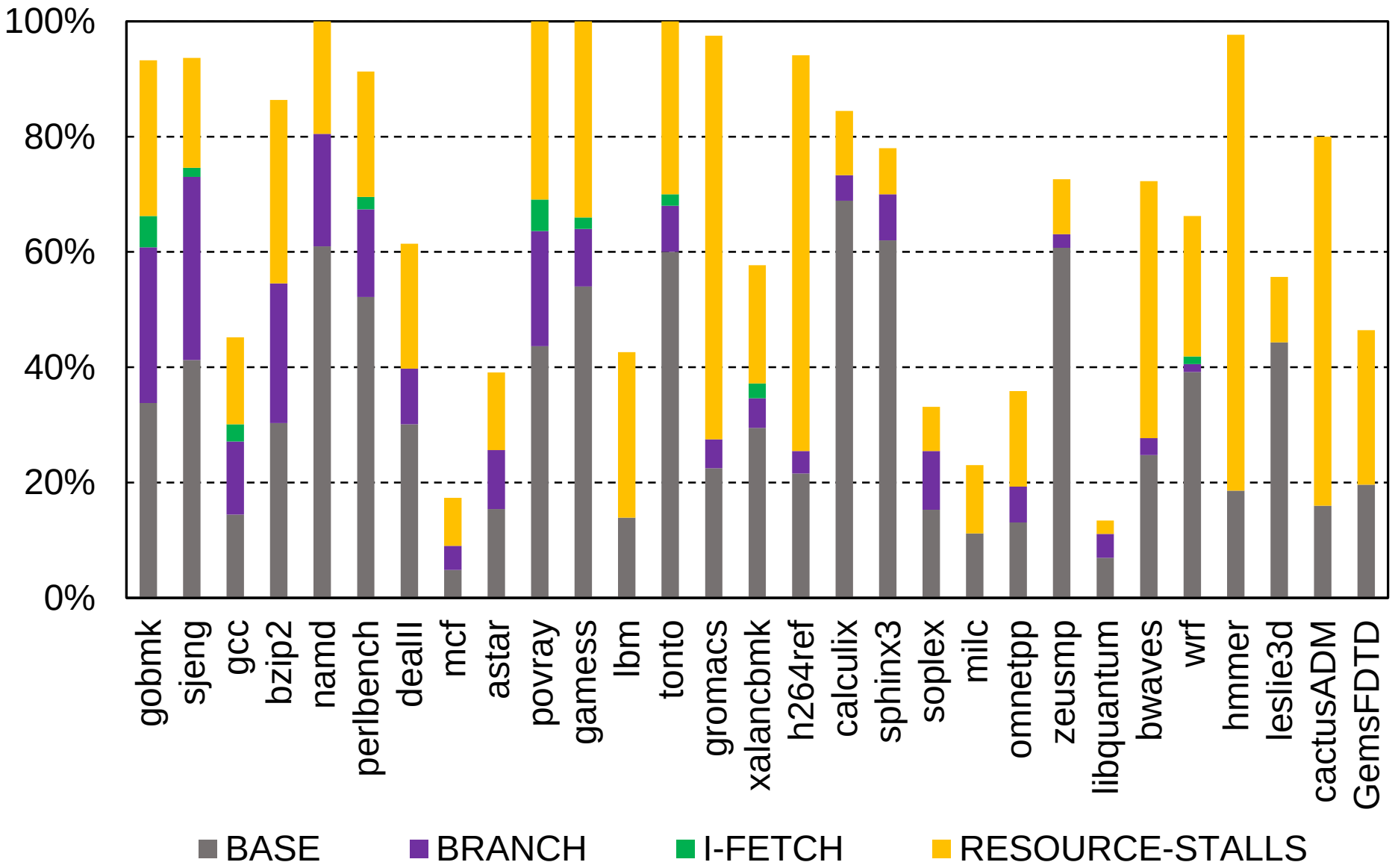
Vulnerability vs. CPI Stacks



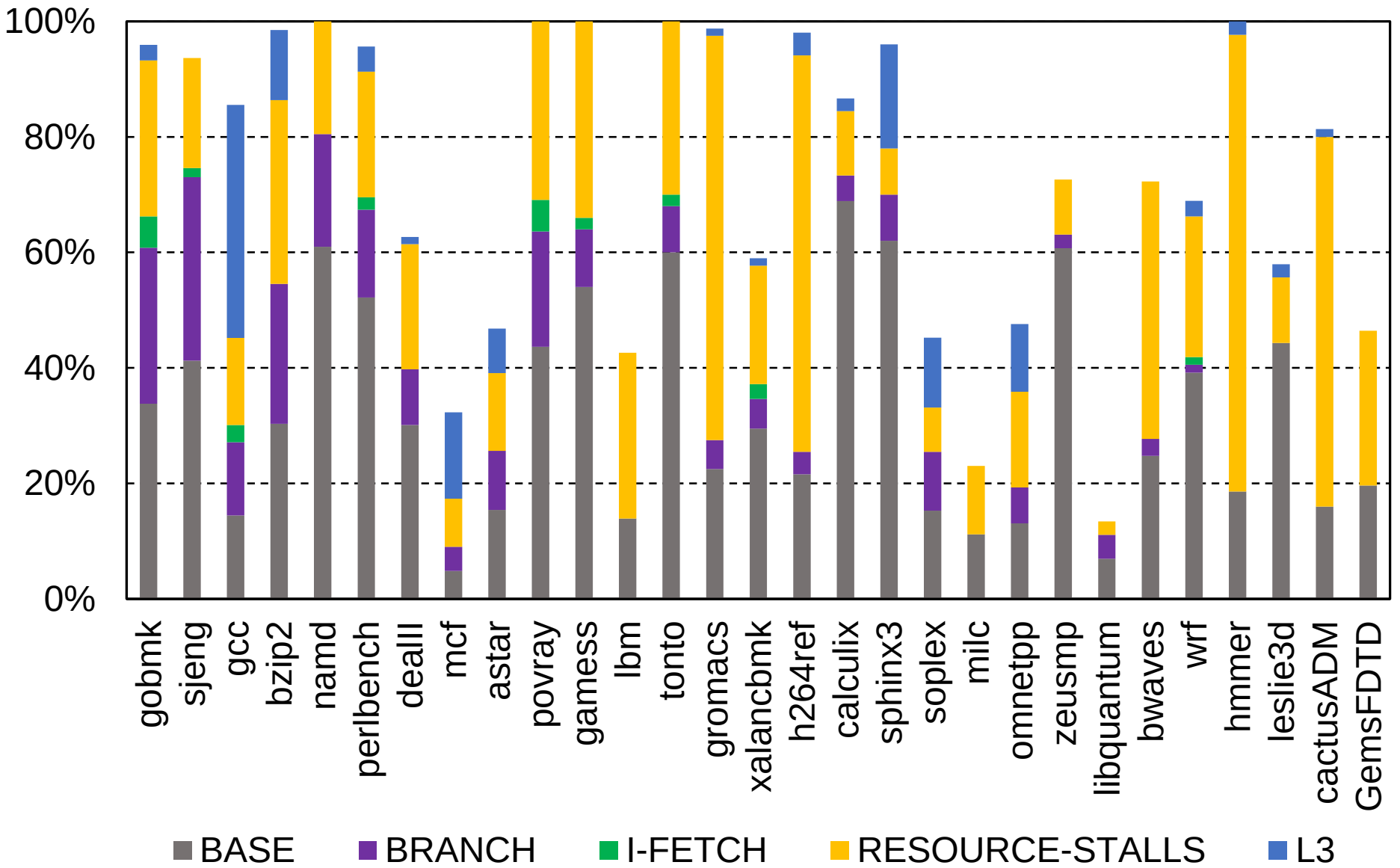
Vulnerability vs. CPI Stacks



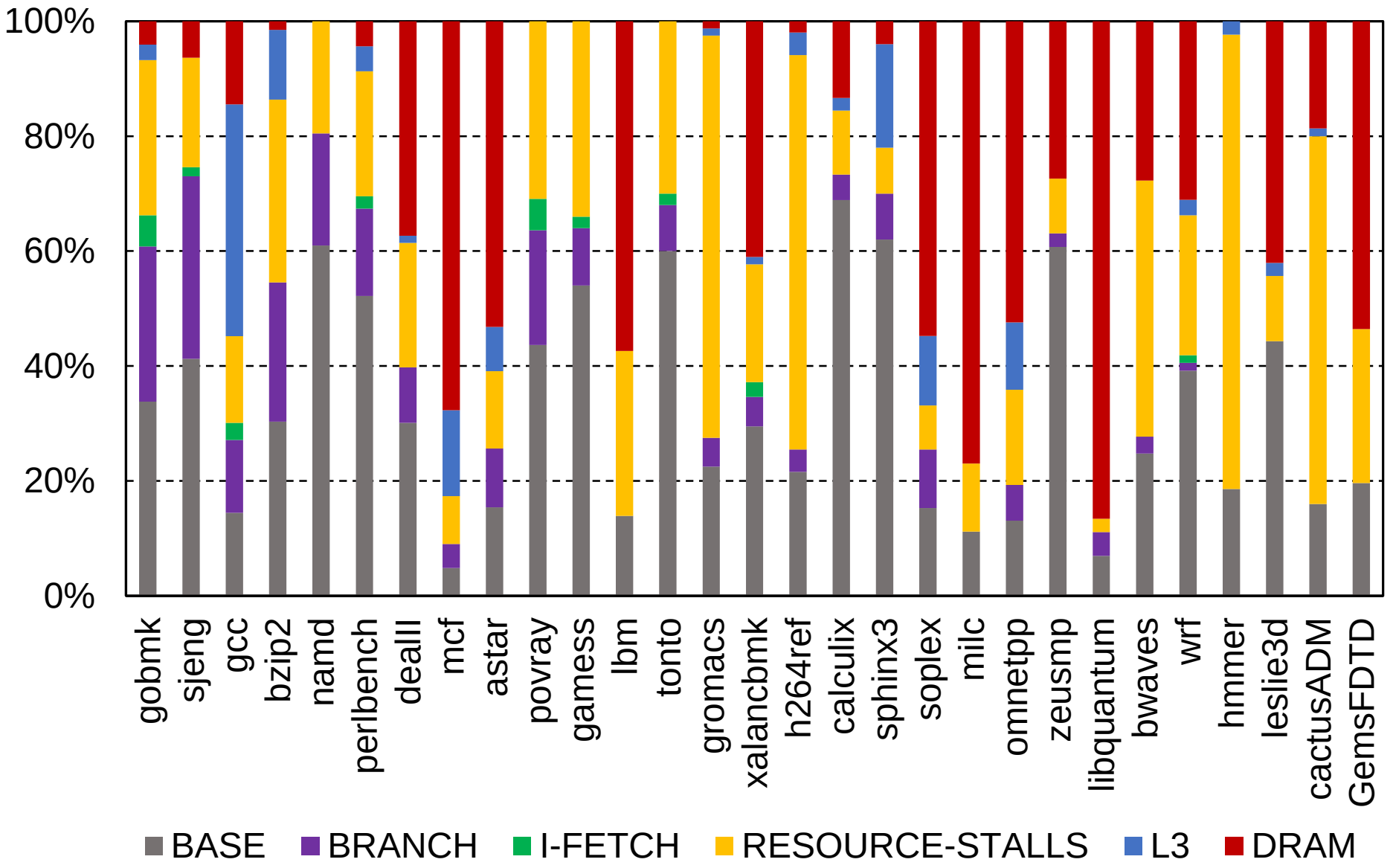
Vulnerability vs. CPI Stacks



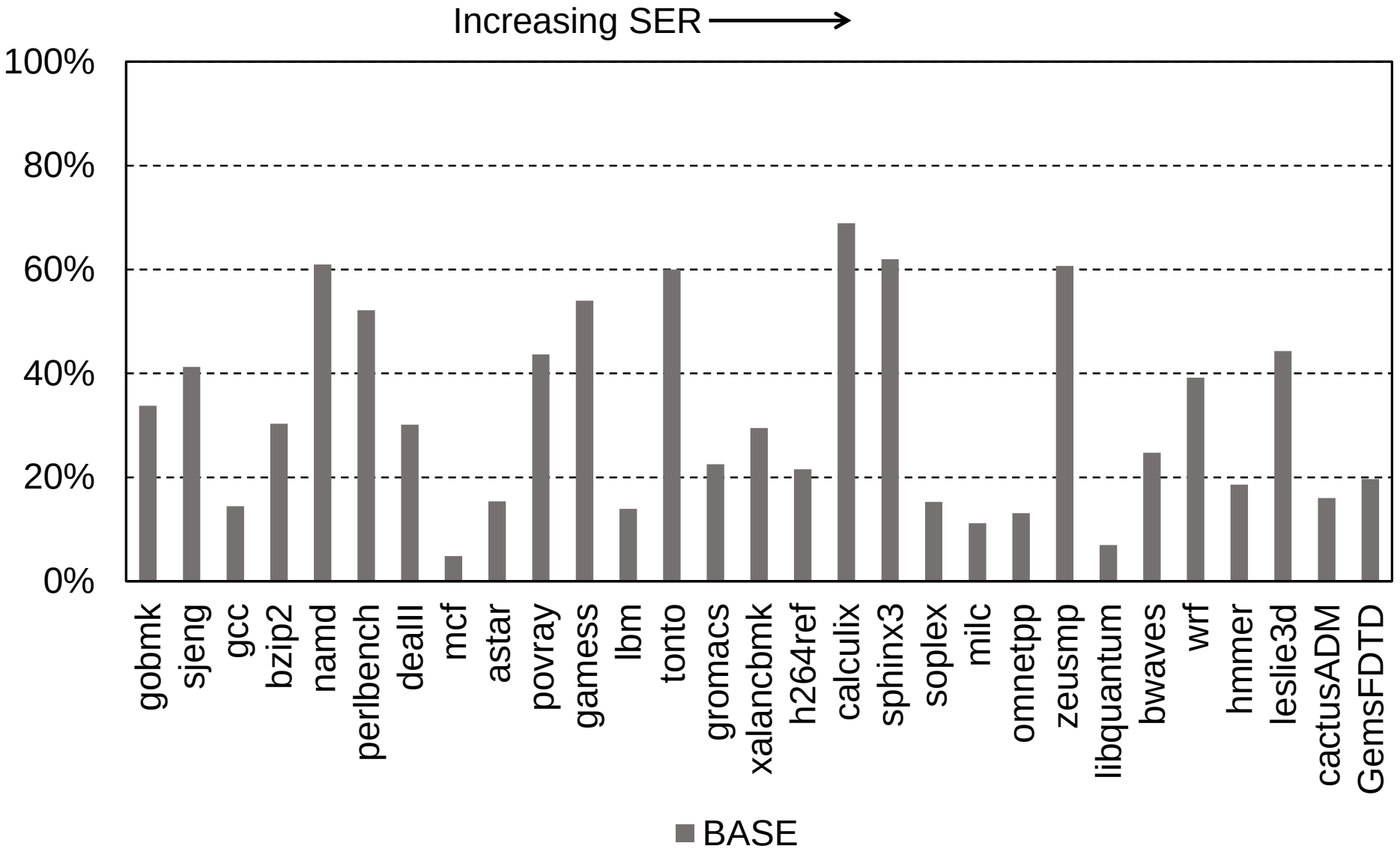
Vulnerability vs. CPI Stacks



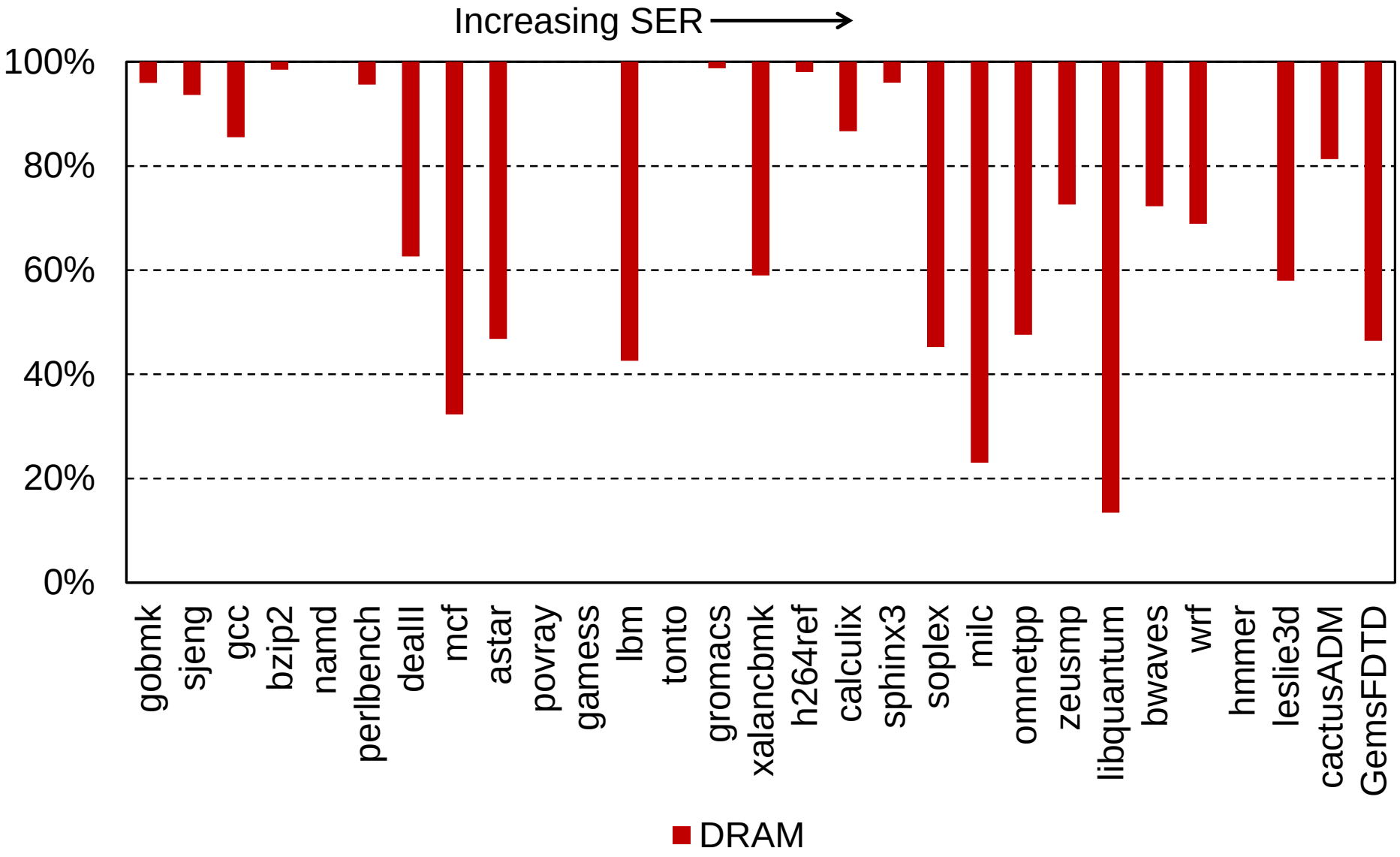
Vulnerability vs. CPI Stacks



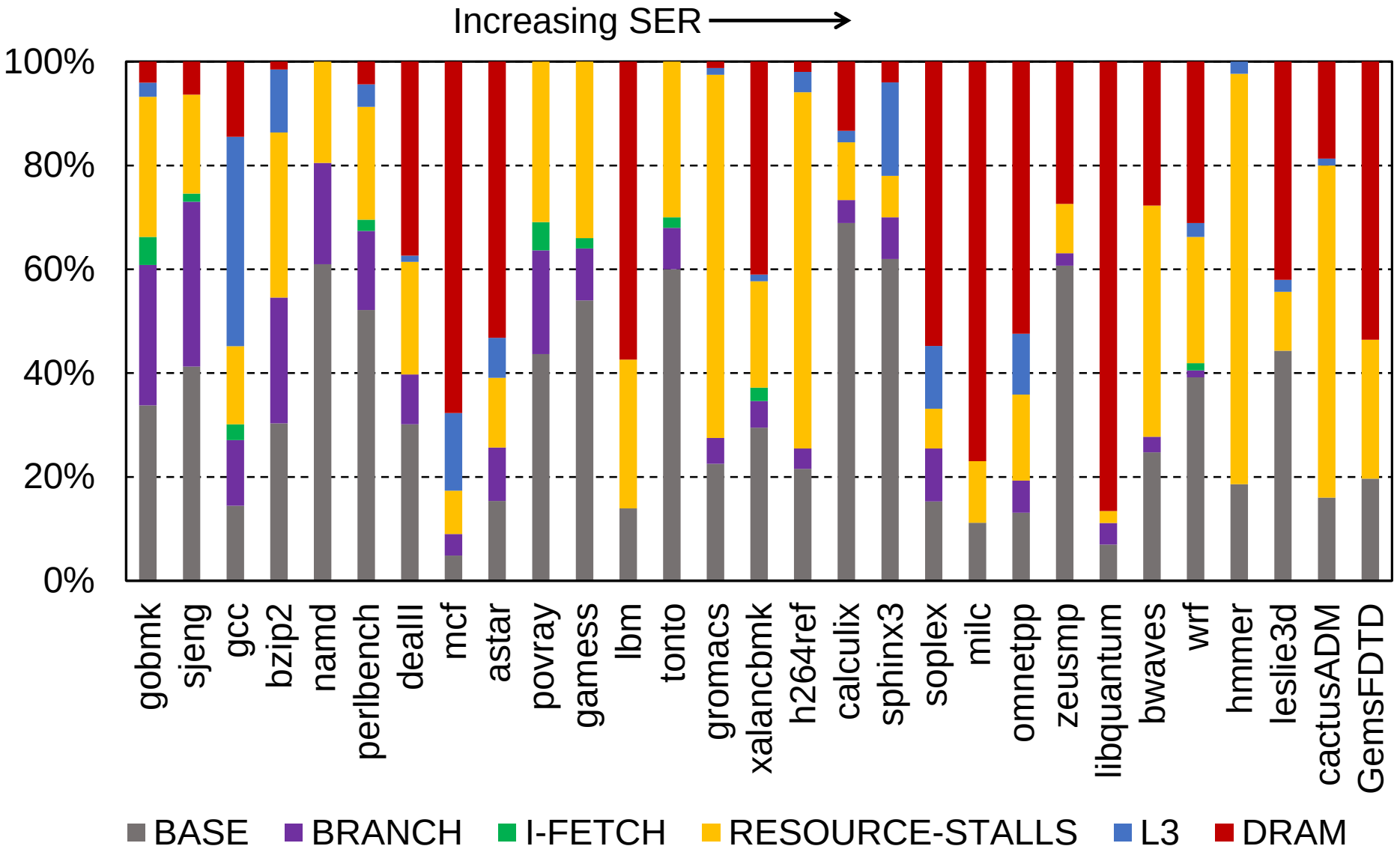
Vulnerability vs. Compute Intensity



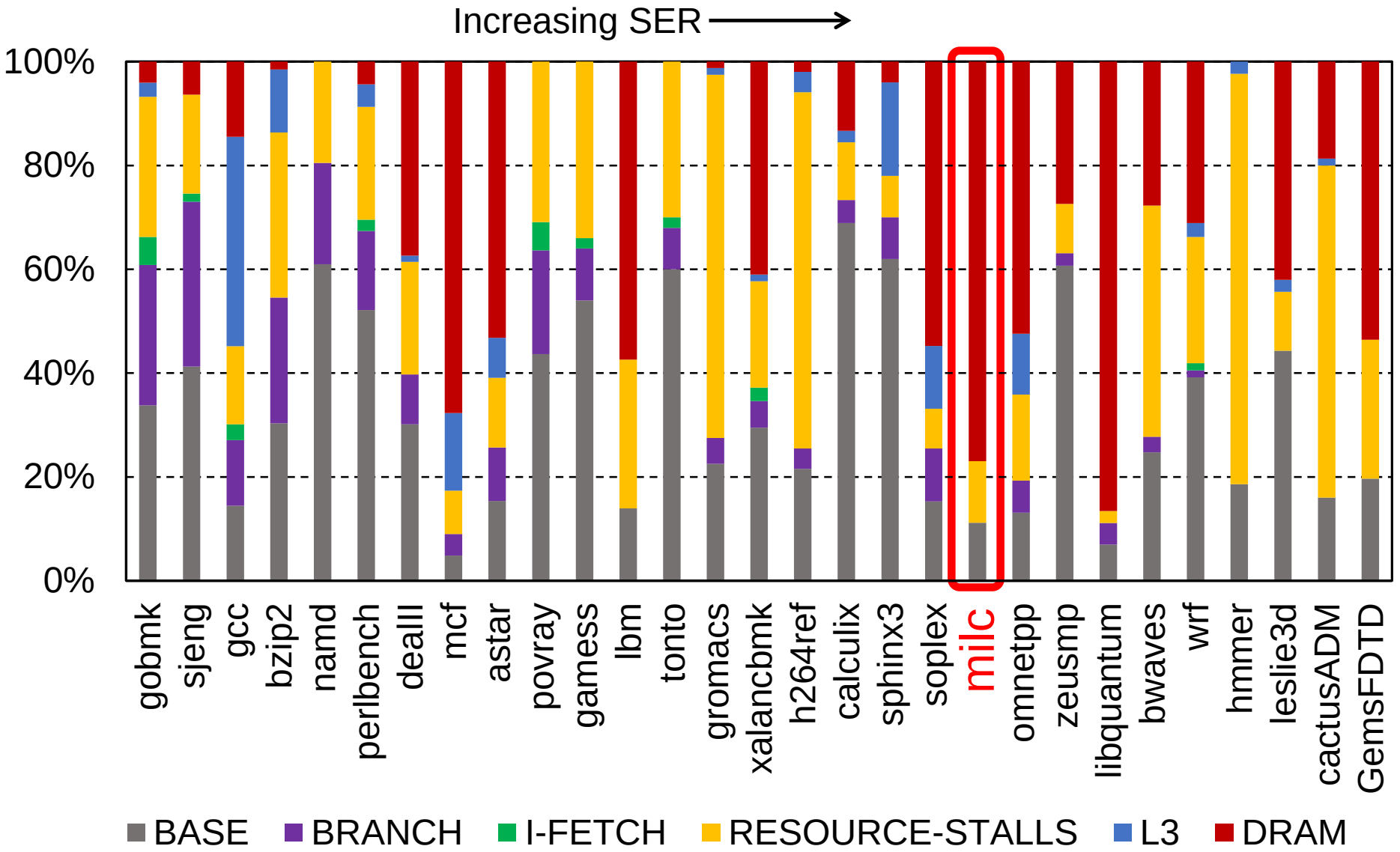
Vulnerability vs. Memory Intensity



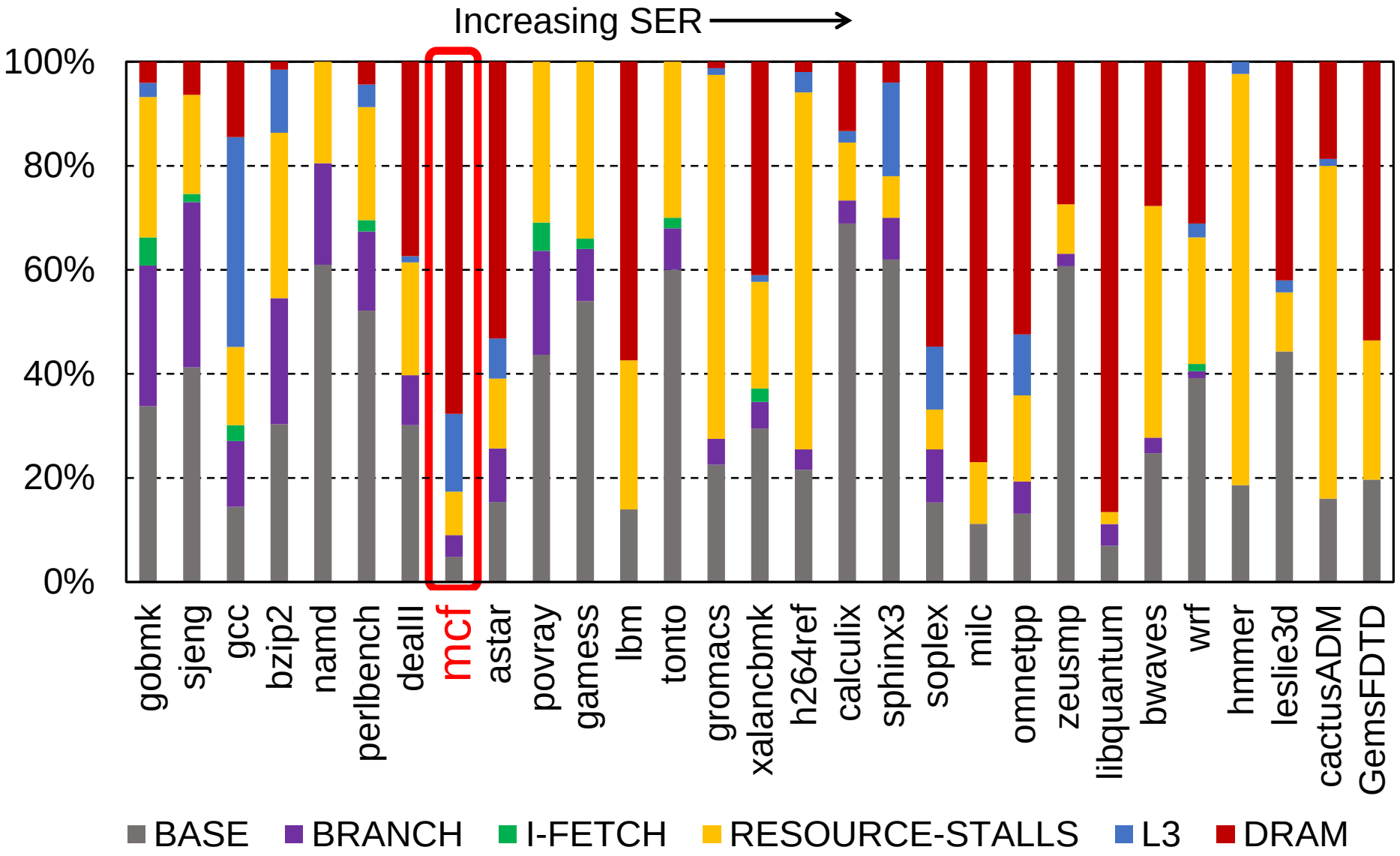
Vulnerability vs. CPI Stacks



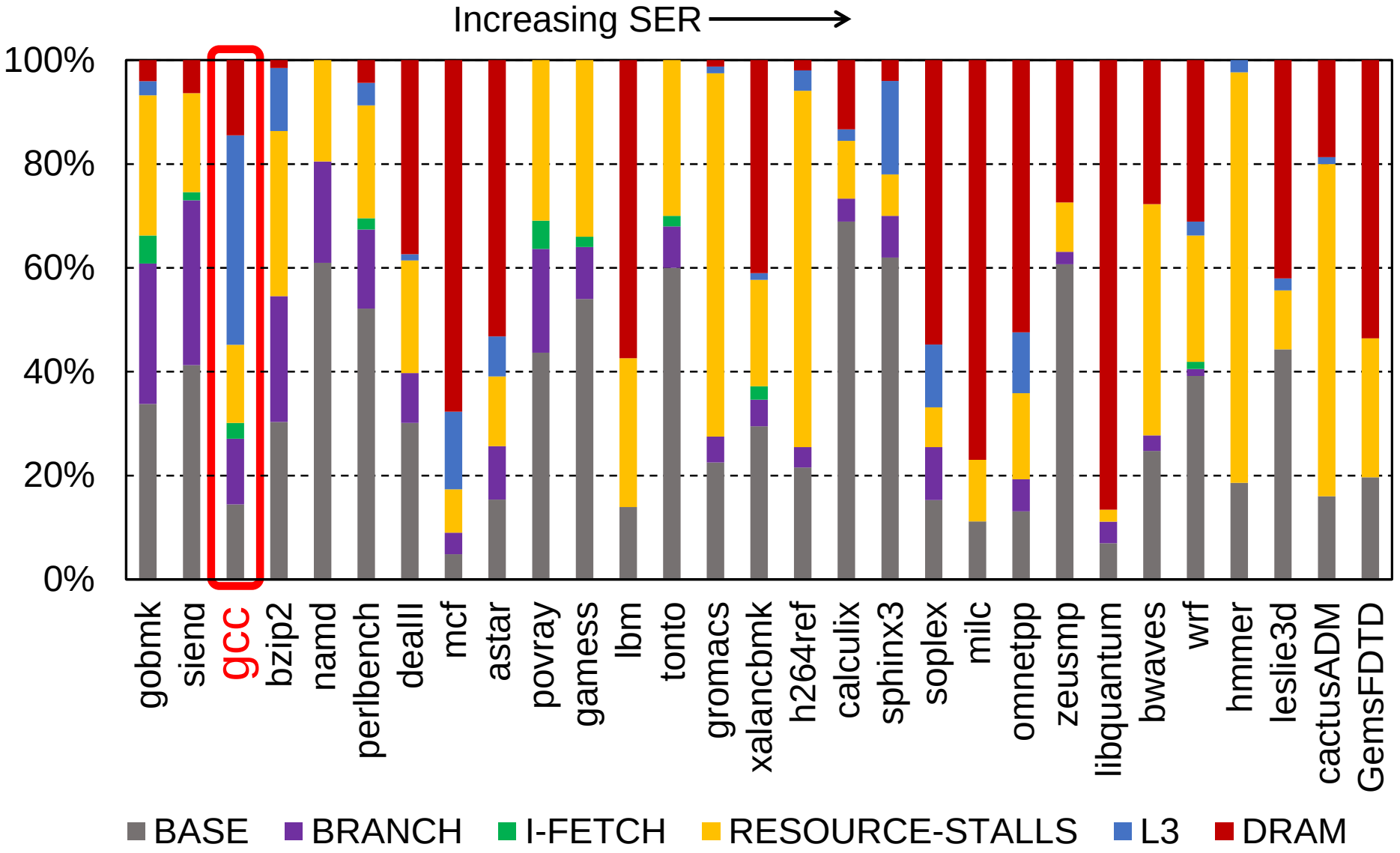
Vulnerability vs. CPI Stacks



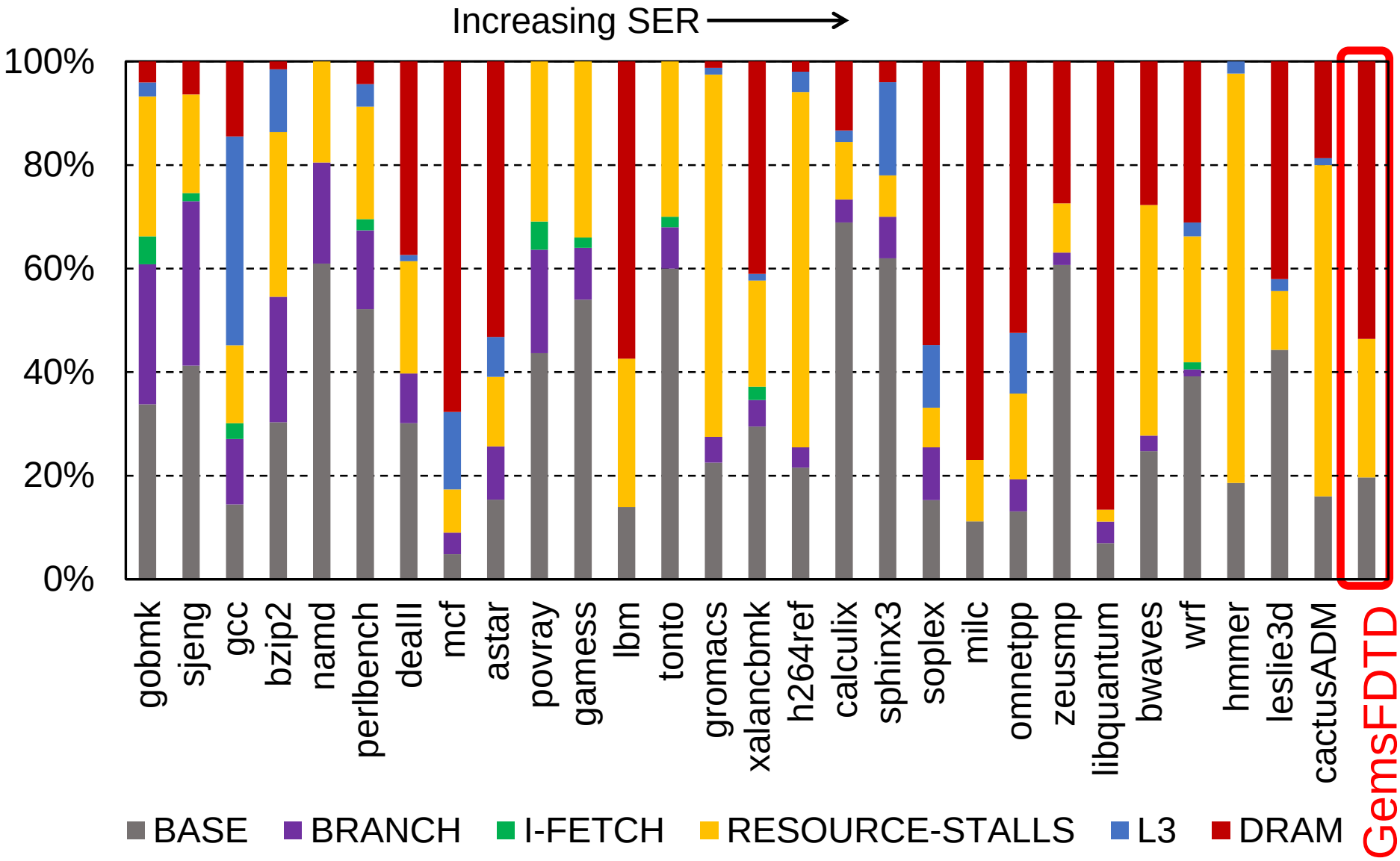
Vulnerability vs. CPI Stacks



Vulnerability vs. CPI Stacks



Vulnerability vs. CPI Stacks



Reliability vs. Application Type

Reliability vs. Application Type

1. No simple workload characteristic
   - For example, memory-intensity or compute-intensity

Reliability vs. Application Type

1. No simple workload characteristic
   - For example, memory-intensity or compute-intensity

2. Predicting SER is not straightforward*
   - Complex interaction of various workload characteristics

*[Nair et al. ISCA'12]

Reliability vs. Application Type

1. No simple workload characteristic
   - For example, memory-intensity or compute-intensity

2. Predicting SER is not straightforward*
   - Complex interaction of various workload characteristics

3. A dynamic mechanism required
   - To monitor reliability on either core type

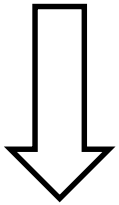
*[Nair et al. ISCA'12]

Reliability vs. Application Type

1. No simple workload characteristic
   - For example, memory-intensity or compute-intensity

2. Predicting SER is not straightforward*
   - Complex interaction of various workload characteristics

3. A dynamic mechanism required
   - To monitor reliability on either core type



Reliability-Aware Scheduler

*[Nair et al. ISCA'12]

Reliability Metric for Multiprogram Workloads

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

A 

time = 1

SER = 4

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

A 

time = 1 SER = 4

B 

time = 1 SER = 1

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

As a 2-program workload on an CMP with two big cores:

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

A  time = 1 SER = 4

B  time = 1 SER = 1

As a 2-program workload on an CMP with two big cores:

A  time = 1 SER = 4

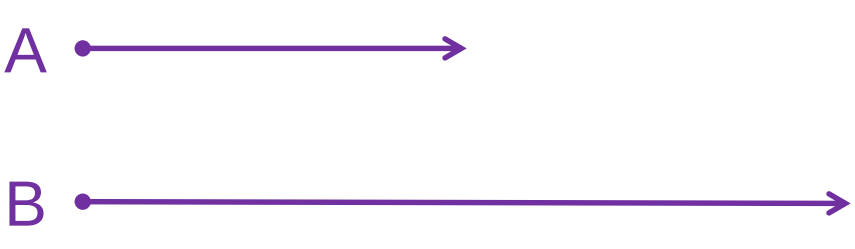
Reliability Metric for Multiprogram Workloads

In isolation on one big core:

time = 1 SER = 4

time = 1 SER = 1

As a 2-program workload on an CMP with two big cores:



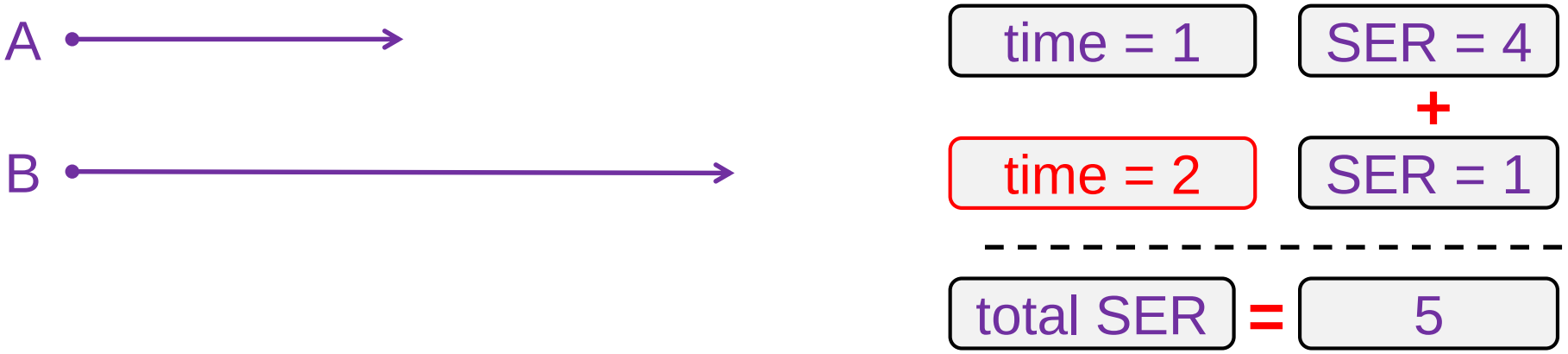
time = 1 SER = 4

time = 2 SER = 1

Reliability Metric for Multiprogram Workloads

In isolation on one big core:

As a 2-program workload on an CMP with two big cores:

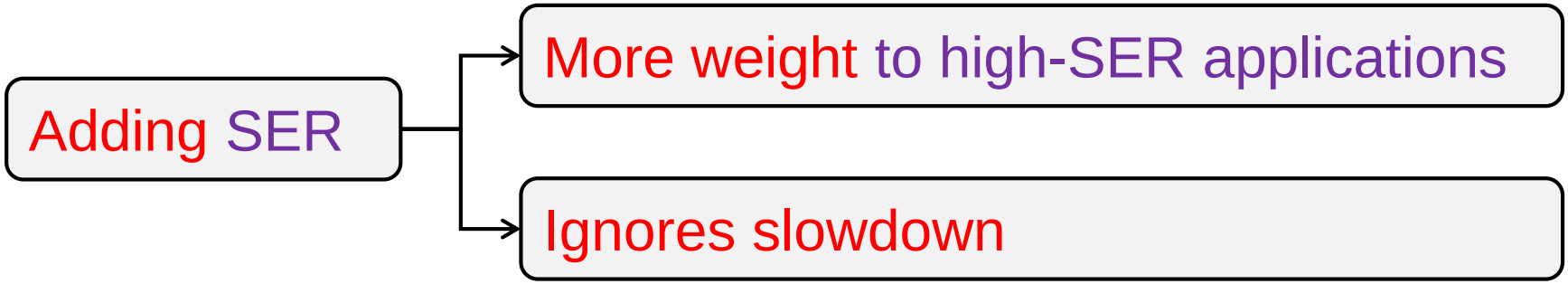


Reliability Metric for Multiprogram Workloads

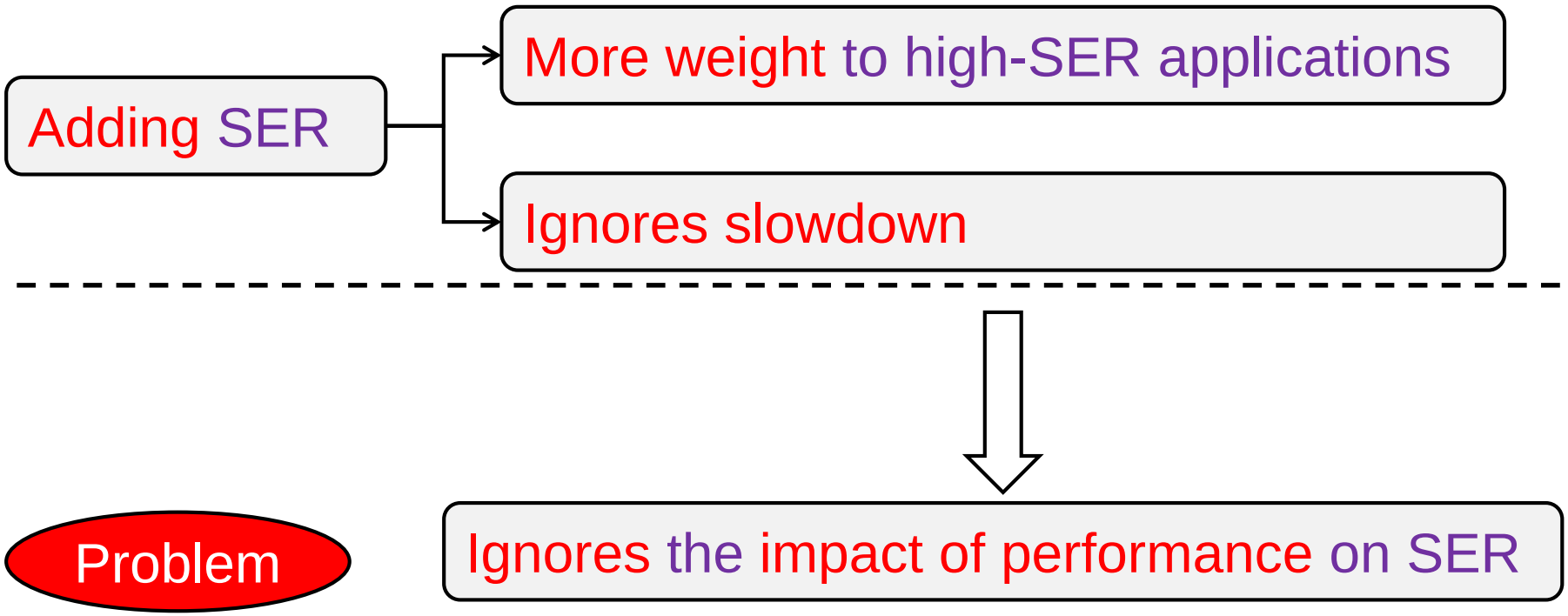
Adding SER

Reliability Metric for Multiprogram Workloads

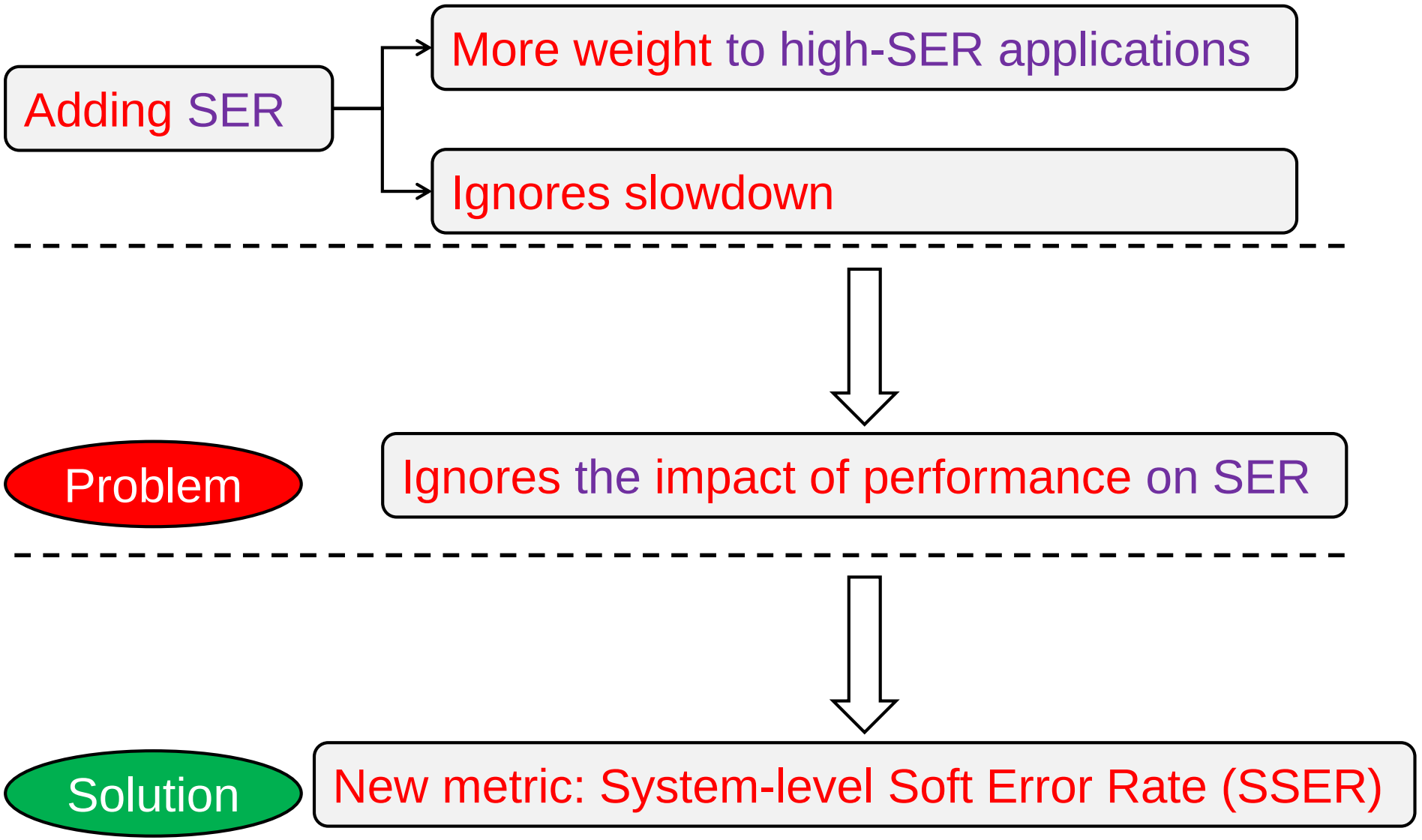
Reliability Metric for Multiprogram Workloads



Reliability Metric for Multiprogram Workloads



Reliability Metric for Multiprogram Workloads



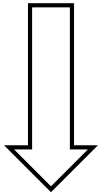
System-level Soft Error Rate (SSER)

System-level Soft Error Rate (SSER)

Weight per-application SER by its slowdown

System-level Soft Error Rate (SSER)

Weight per-application SER by its slowdown

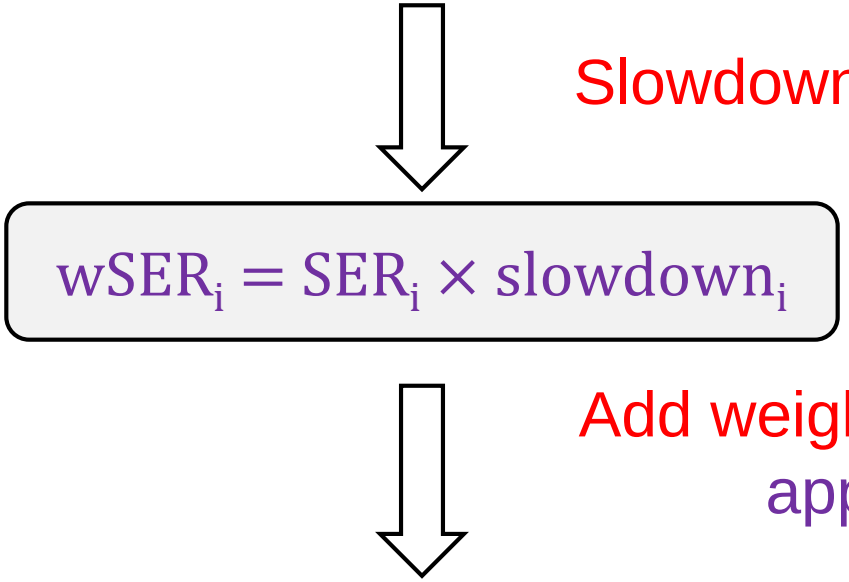


Slowdown → More errors

$$wSER_i = SER_i \times slowdown_i$$

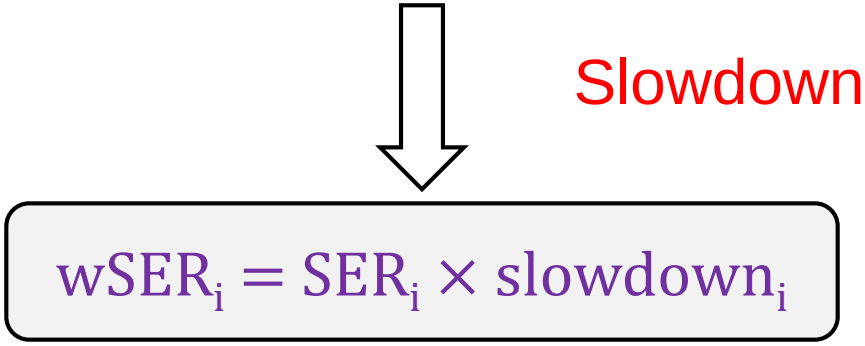
System-level Soft Error Rate (SSER)

Weight per-application SER by its slowdown



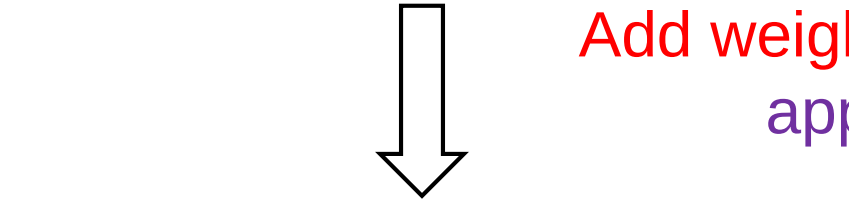
System-level Soft Error Rate (SSER)

Weight per-application SER by its slowdown



Slowdown → More errors

$$wSER_i = SER_i \times slowdown_i$$



Add weighted SER for all applications

$$SSER = \sum_{i=1}^{n} wSER_i$$

System-level Soft Error Rate (SSER)

Weight per-application SER by its slowdown

Slowdown → More errors

$$wSER_i = SER_i \times slowdown_i$$

Add weighted SER for all applications

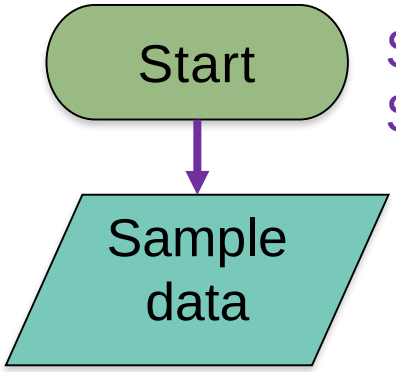
$$SSER = \sum_{i=1}^{n} wSER_i$$

A novel metric to assess soft error vulnerability

Scheduling Algorithm

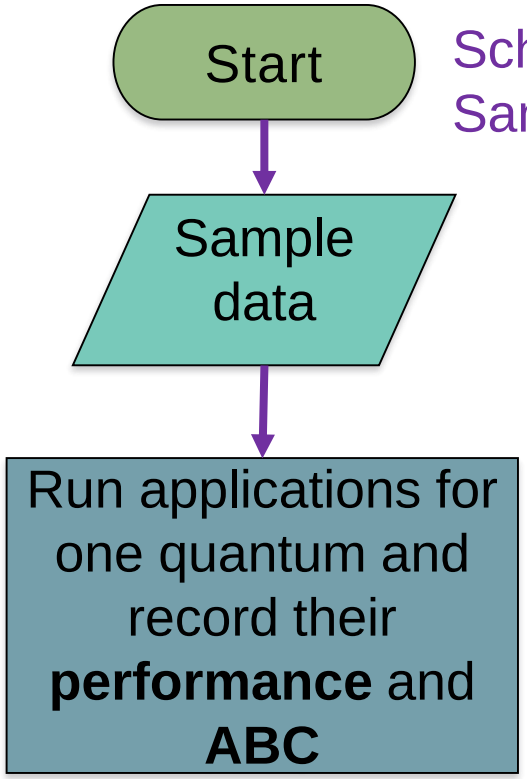
Scheduling Algorithm

Scheduling Algorithm



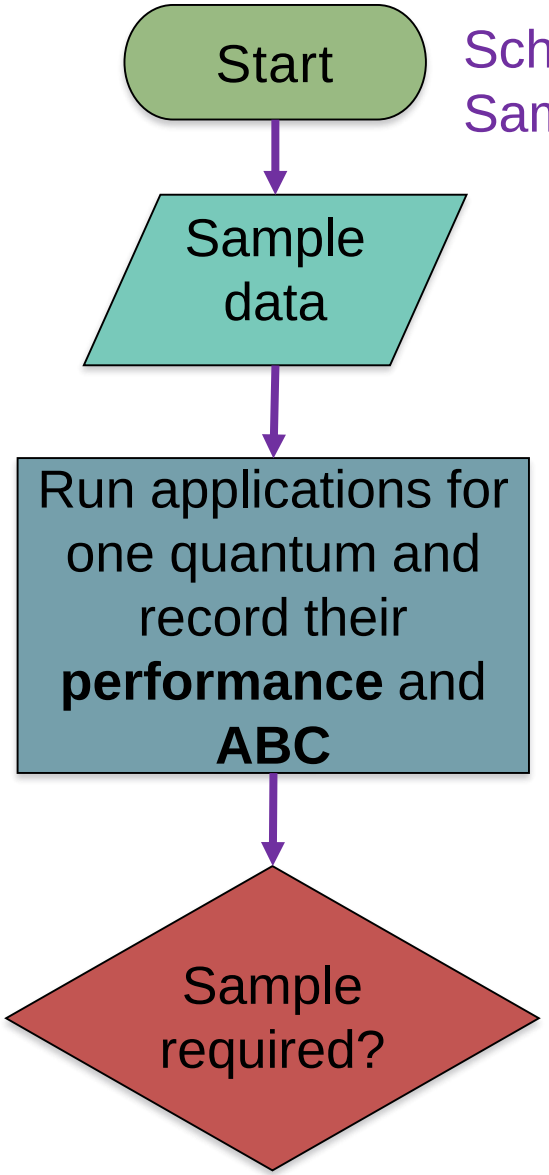
Scheduler quantum →1ms
Sampling for 0.1ms

Scheduling Algorithm



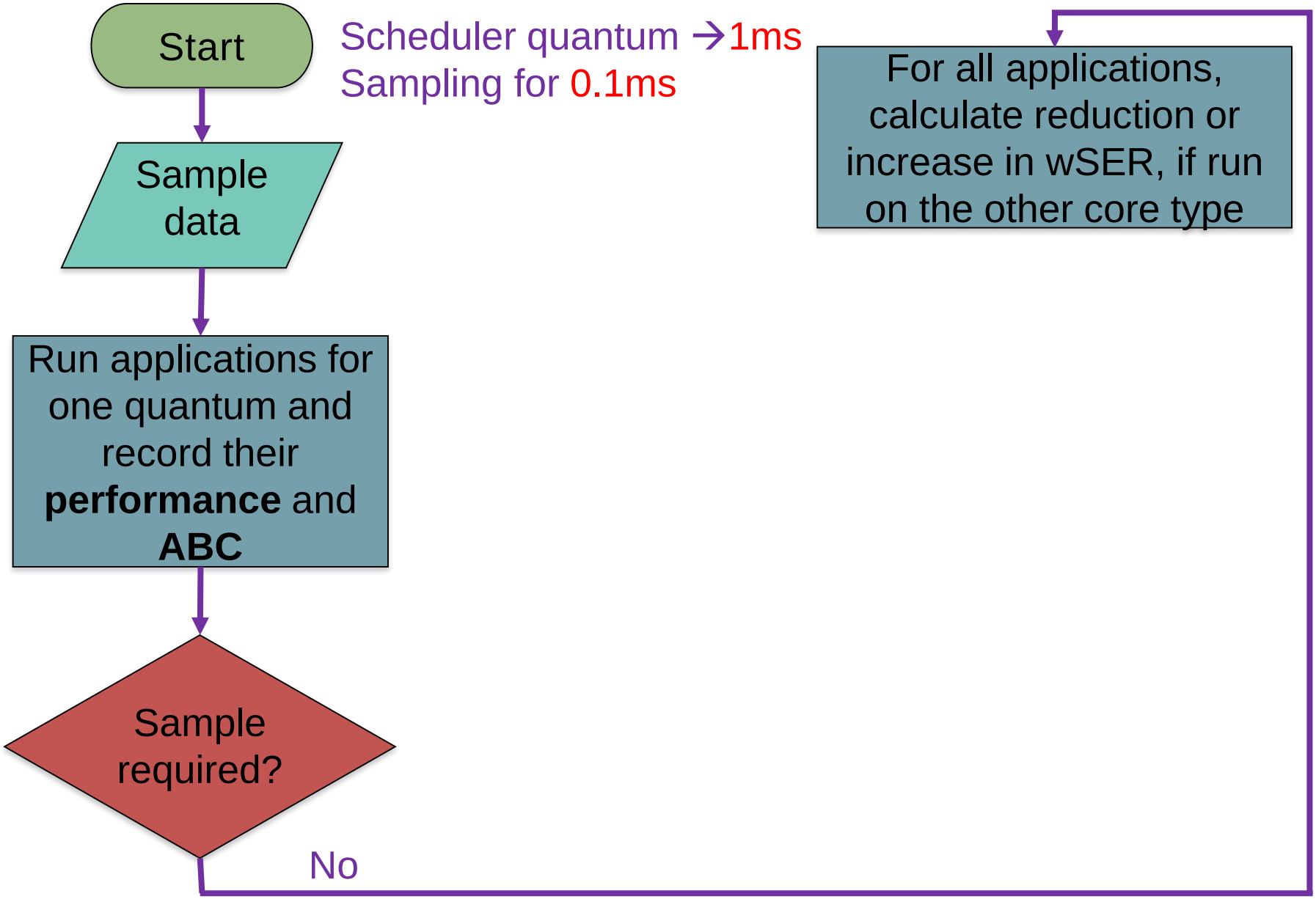
Scheduler quantum →1ms
Sampling for 0.1ms

Scheduling Algorithm

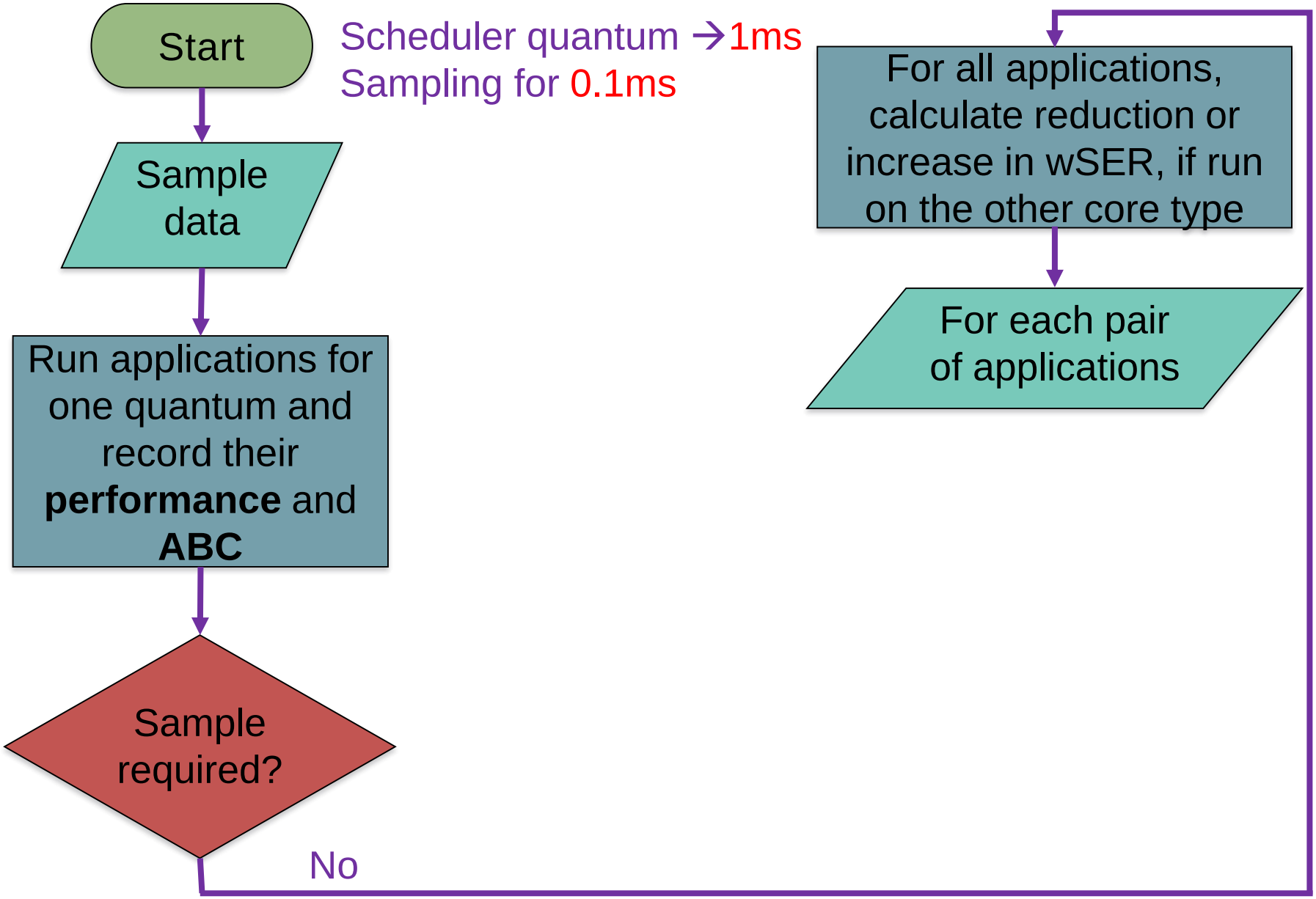


Scheduler quantum →1ms
Sampling for 0.1ms

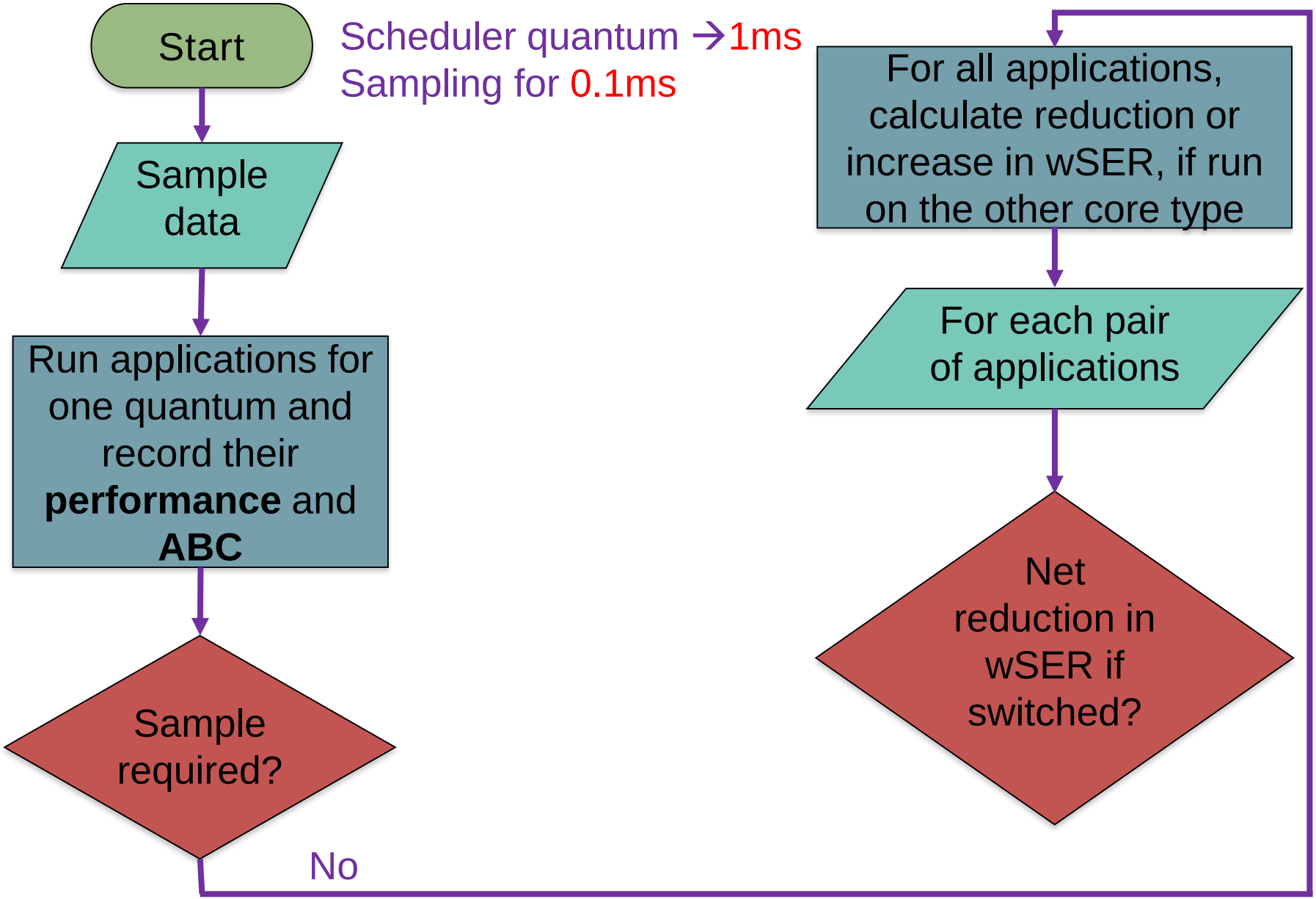
Scheduling Algorithm



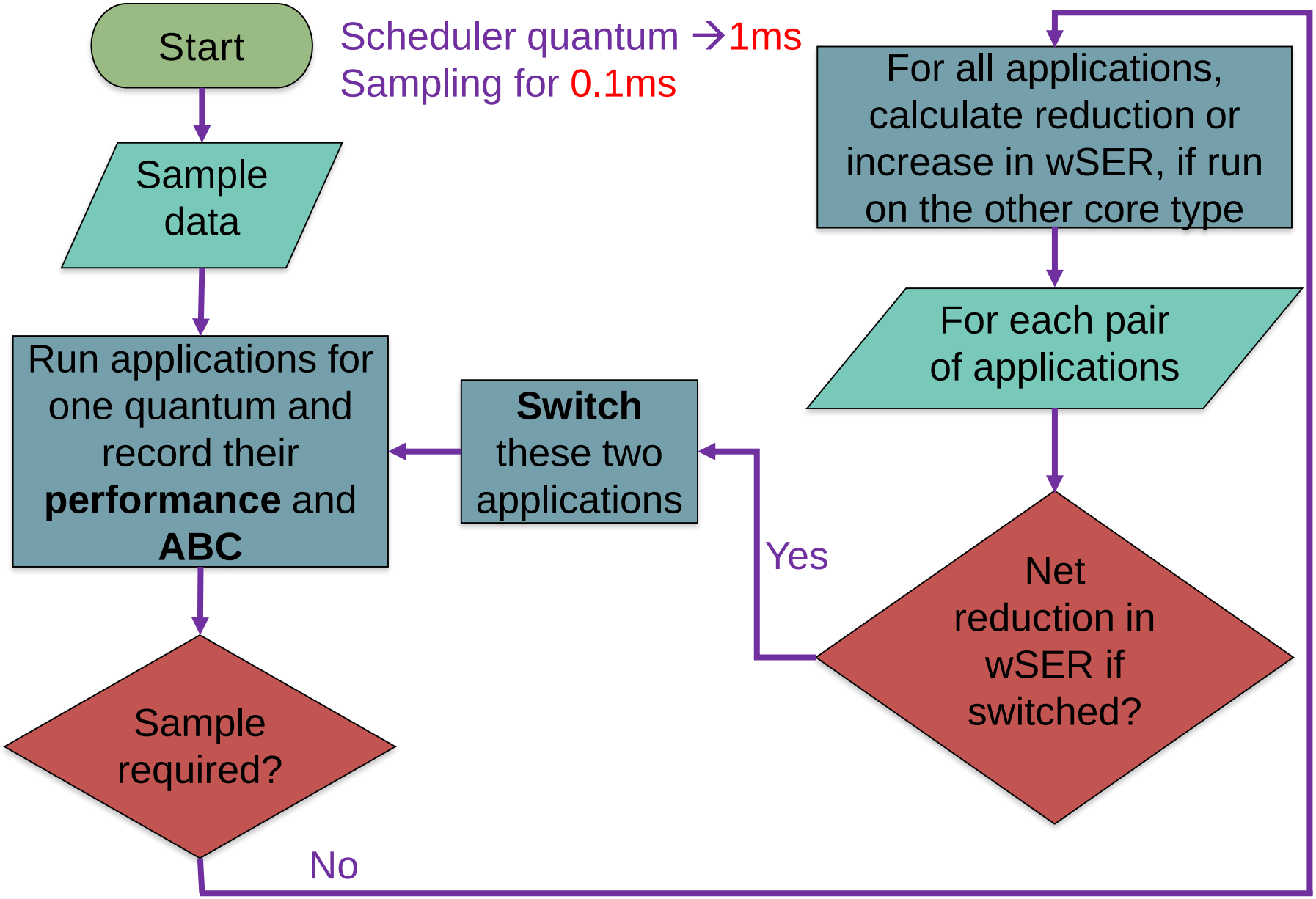
Scheduling Algorithm



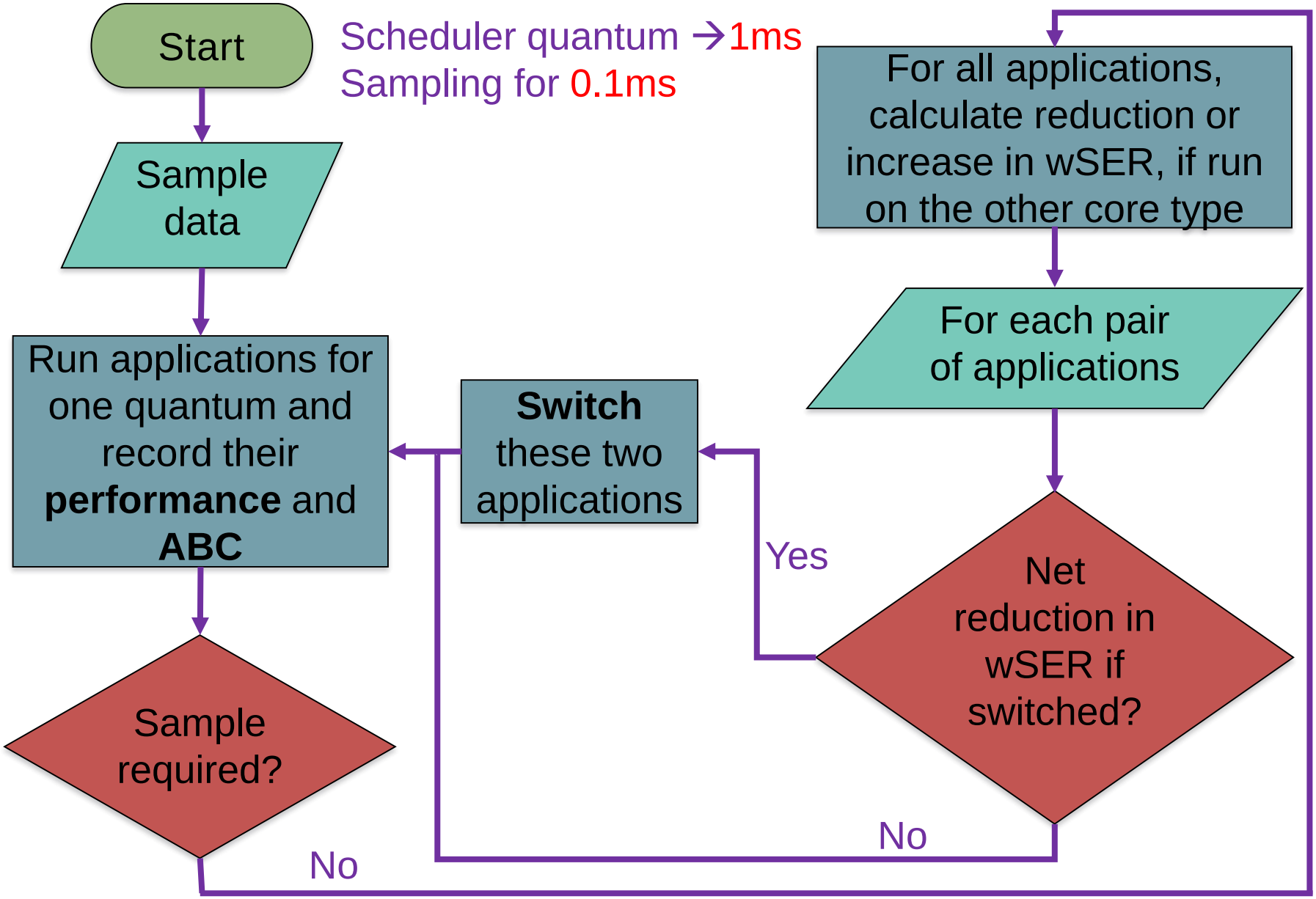
Scheduling Algorithm



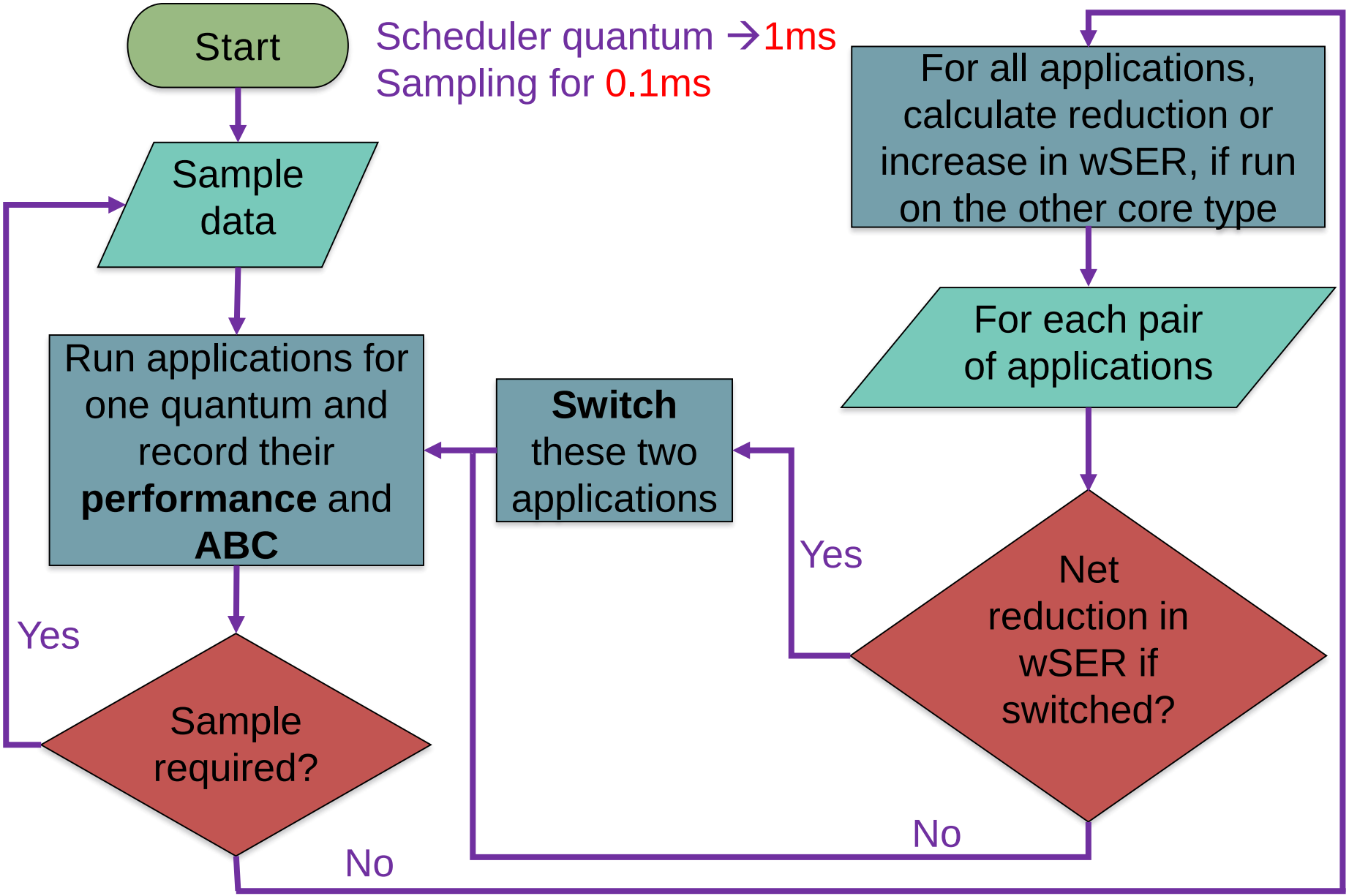
Scheduling Algorithm



Scheduling Algorithm



Scheduling Algorithm



Hardware Overhead

Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times

Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times

Hardware Overhead

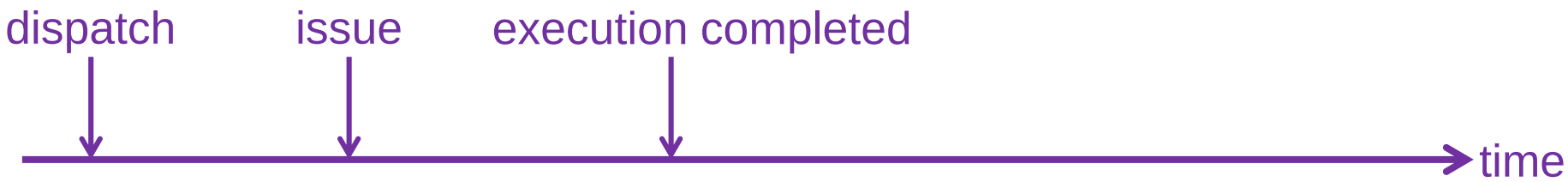
Two 12-bit counters per ROB entry to record dispatch and issue times

Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times

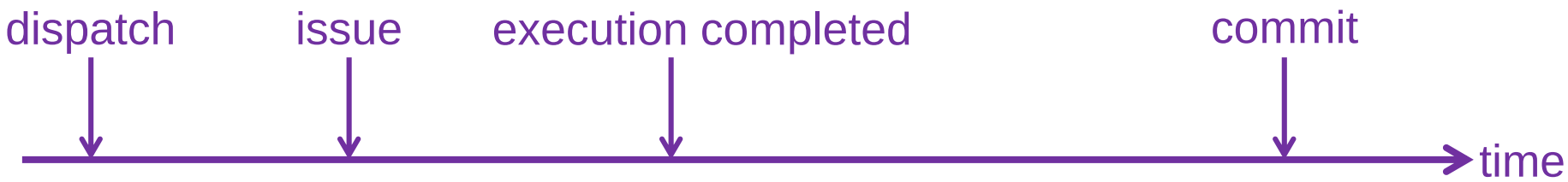
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



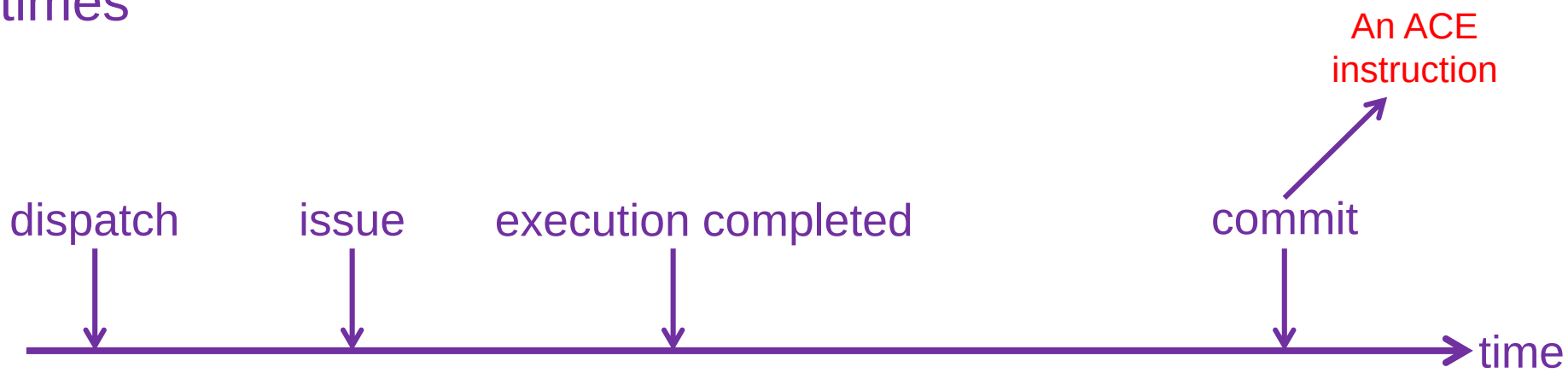
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



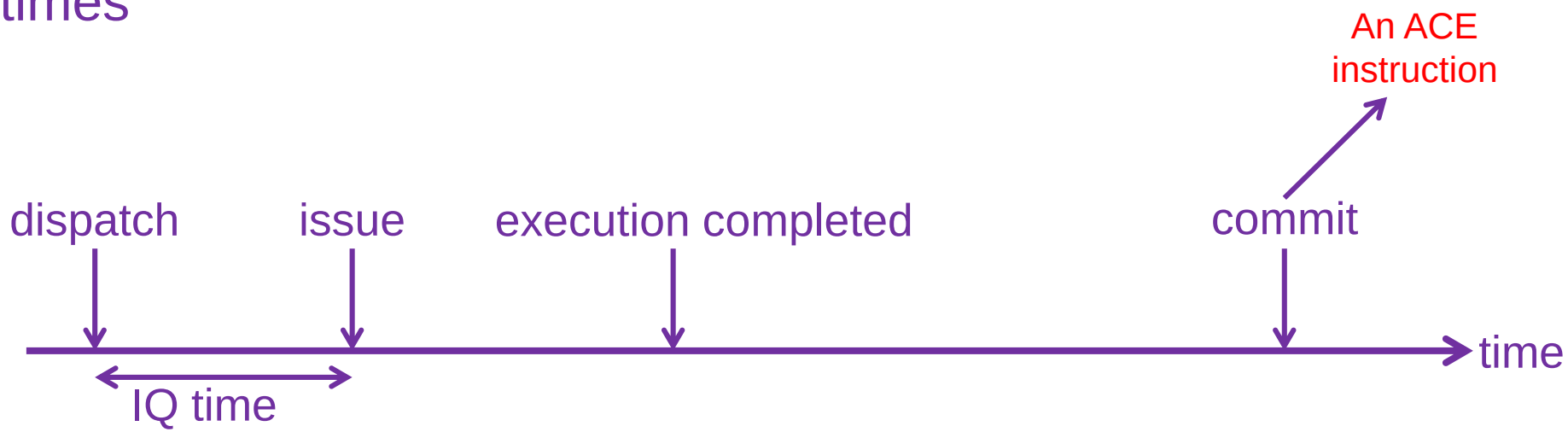
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



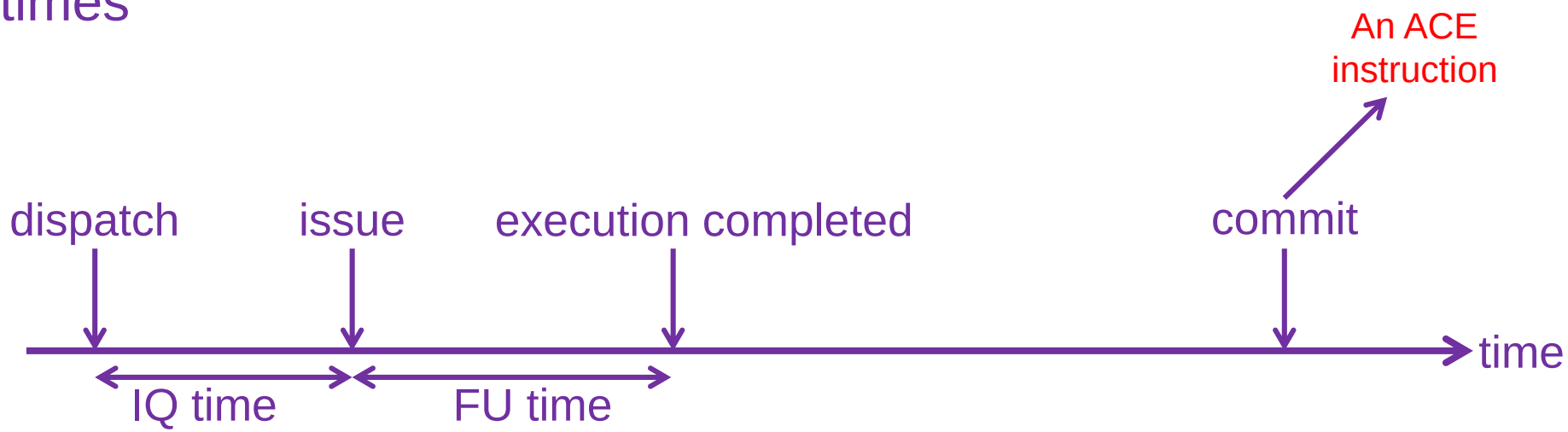
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



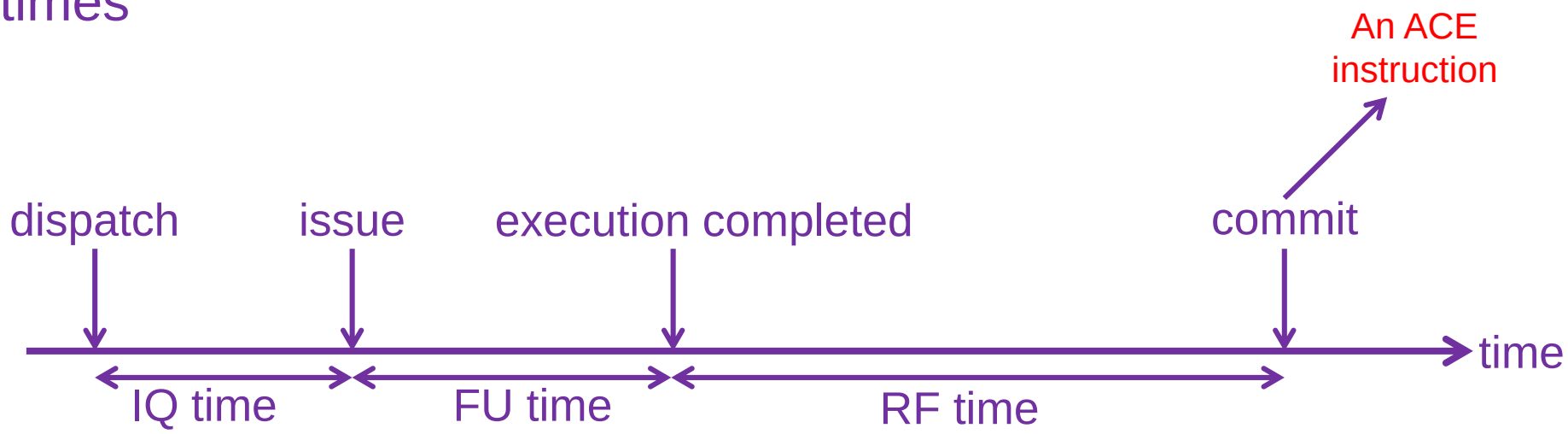
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



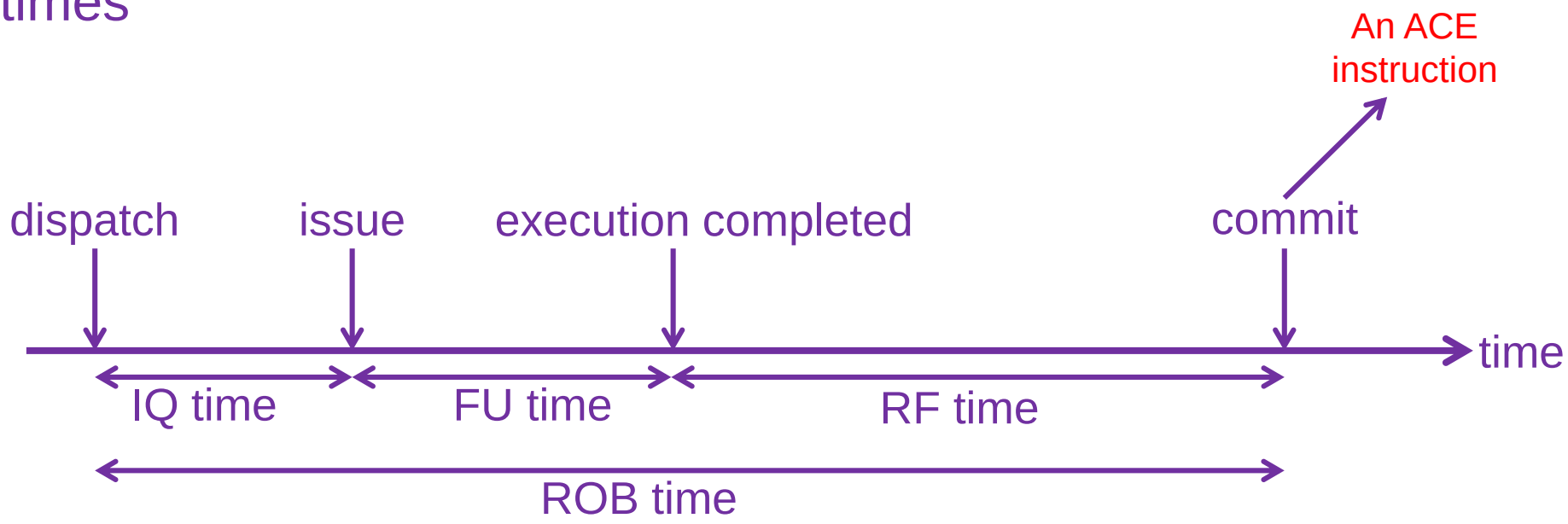
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



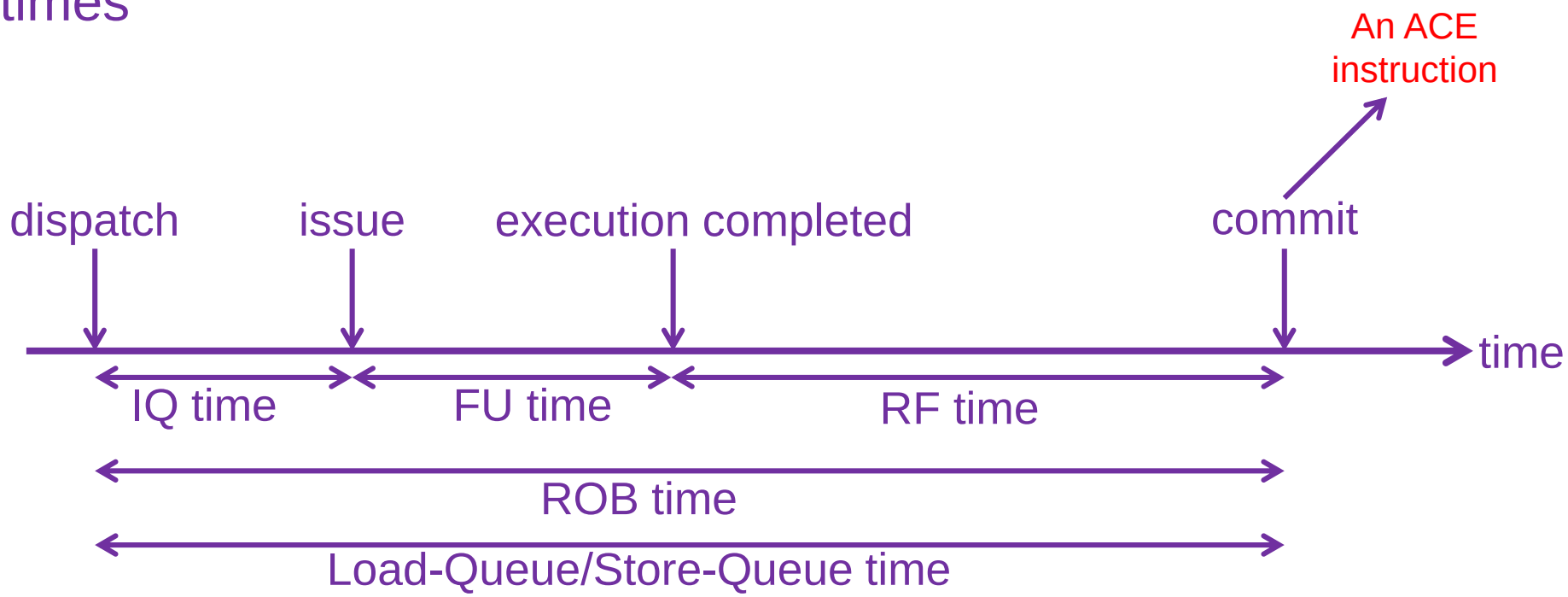
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



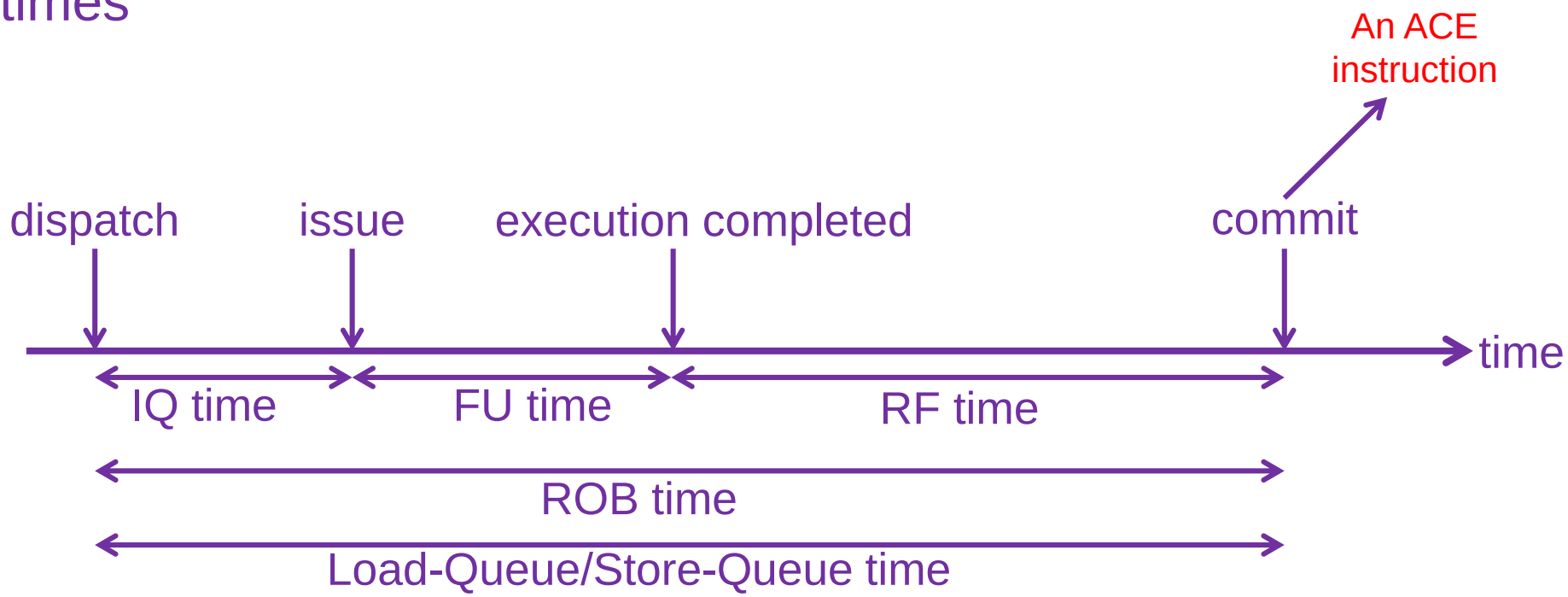
Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



Hardware Overhead

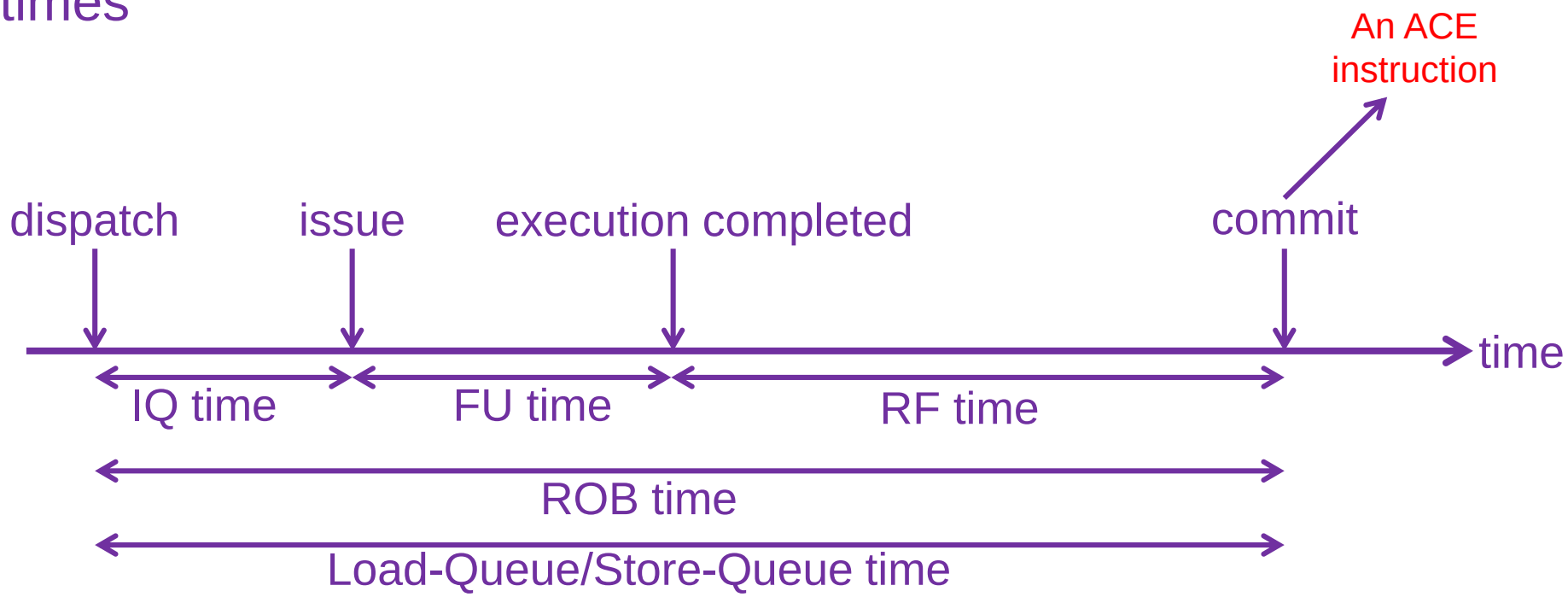
Two 12-bit counters per ROB entry to record dispatch and issue times



Total hardware overhead → 904 bytes per core

Hardware Overhead

Two 12-bit counters per ROB entry to record dispatch and issue times



Total hardware overhead → 904 bytes per core

Hardware-optimized version → 296 bytes per core

Experimental Setup

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

✓ L→ Low AVF

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

4-program workloads

- 6 categories → HHHH, HHMM, HHLL, MMMM, MMLL, LLLL

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

4-program workloads

- 6 categories → HHHH, HHMM, HHLL, MMMM, MMLL, LLLL
- No duplication

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

4-program workloads

- 6 categories → HHHH, HHMM, HHLL, MMMM, MMLL, LLLL
- No duplication
- 6 mixes in each category → 36 workloads

Experimental Setup

Sniper 6.0:

- Augmented with ACE Bit Counters
- Migration overhead between cores → 20 microseconds
- Scheduler quantum → 1ms, sampling for 0.1ms

SPEC CPU2006:

- 1B instruction SimPoints
- Based on AVF-intensity

✓ L→ Low AVF

✓ M → Medium AVF

✓ H → High AVF

4-program workloads

- 6 categories → HHHH, HHMM, HHLL, MMMM, MMLL, LLLL
- No duplication
- 6 mixes in each category → 36 workloads

HCMP → two big and two small cores (2B2S)

Evaluation

Reliability-Optimized Scheduler

- Optimizes SSER using our scheduling algorithm

Evaluation

Reliability-Optimized Scheduler

- Optimizes SSER using our scheduling algorithm

Performance-Optimized Scheduler

- Optimizes System Throughput (STP)

Evaluation

Reliability-Optimized Scheduler

- Optimizes SSER using our scheduling algorithm

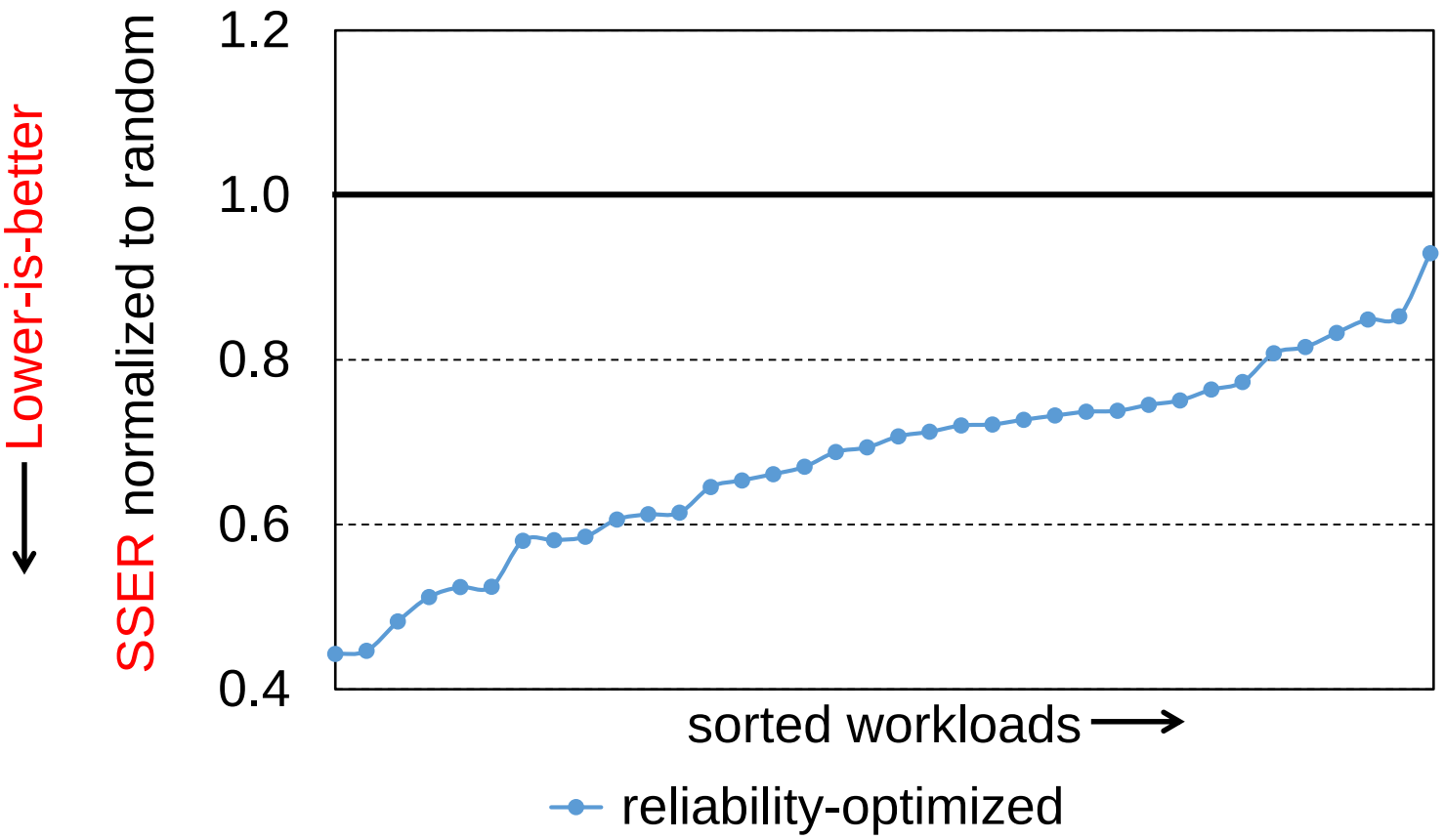
Performance-Optimized Scheduler

- Optimizes System Throughput (STP)

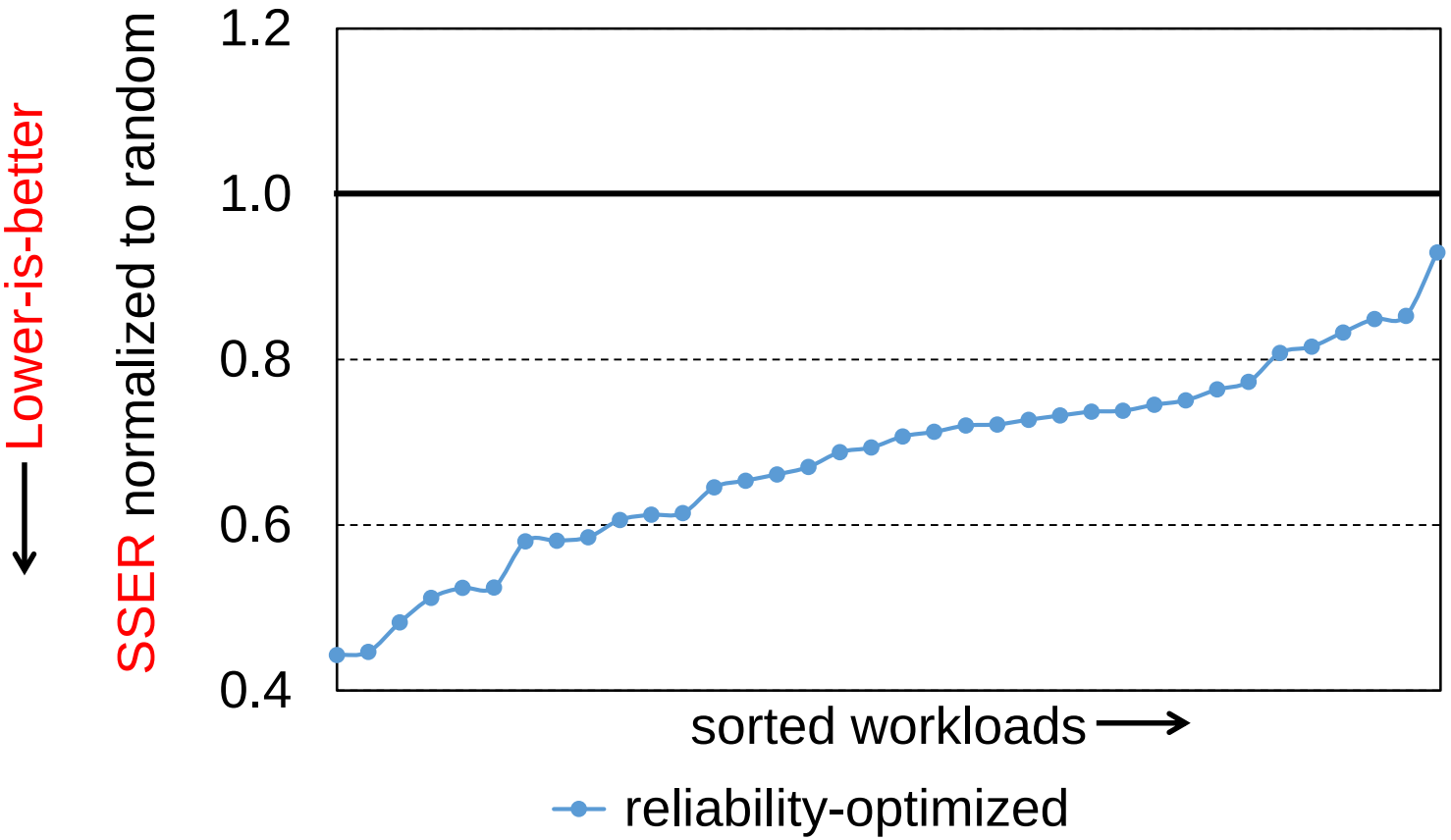
Random Scheduler

- Randomly selects applications to run on the big core(s)

2B2S Results – Reliability

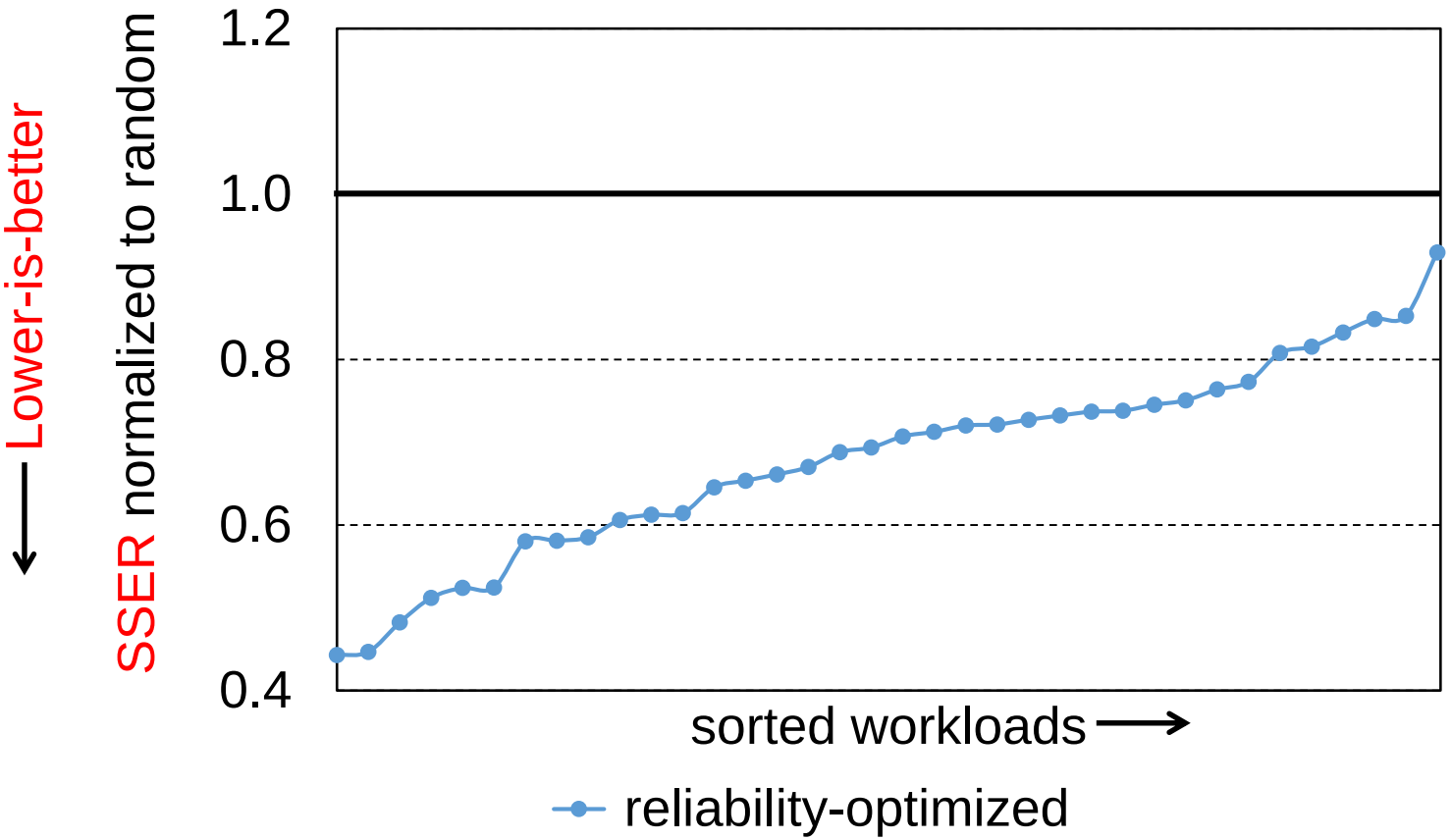


2B2S Results – Reliability



Compared to the random scheduler:

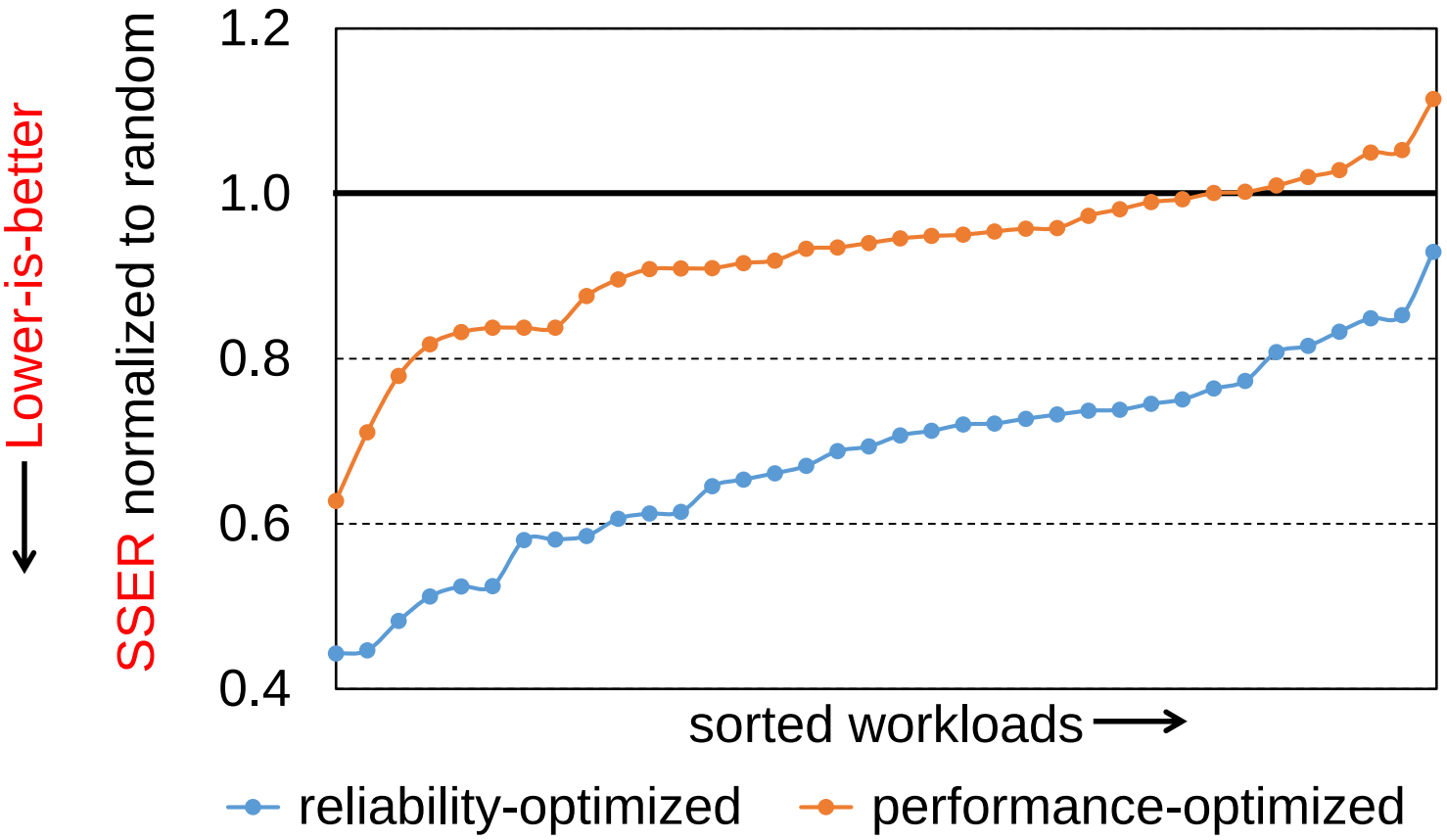
2B2S Results – Reliability



Compared to the random scheduler:

Reliability-optimized scheduler improves SSER by 32% on average and up to 55.6%

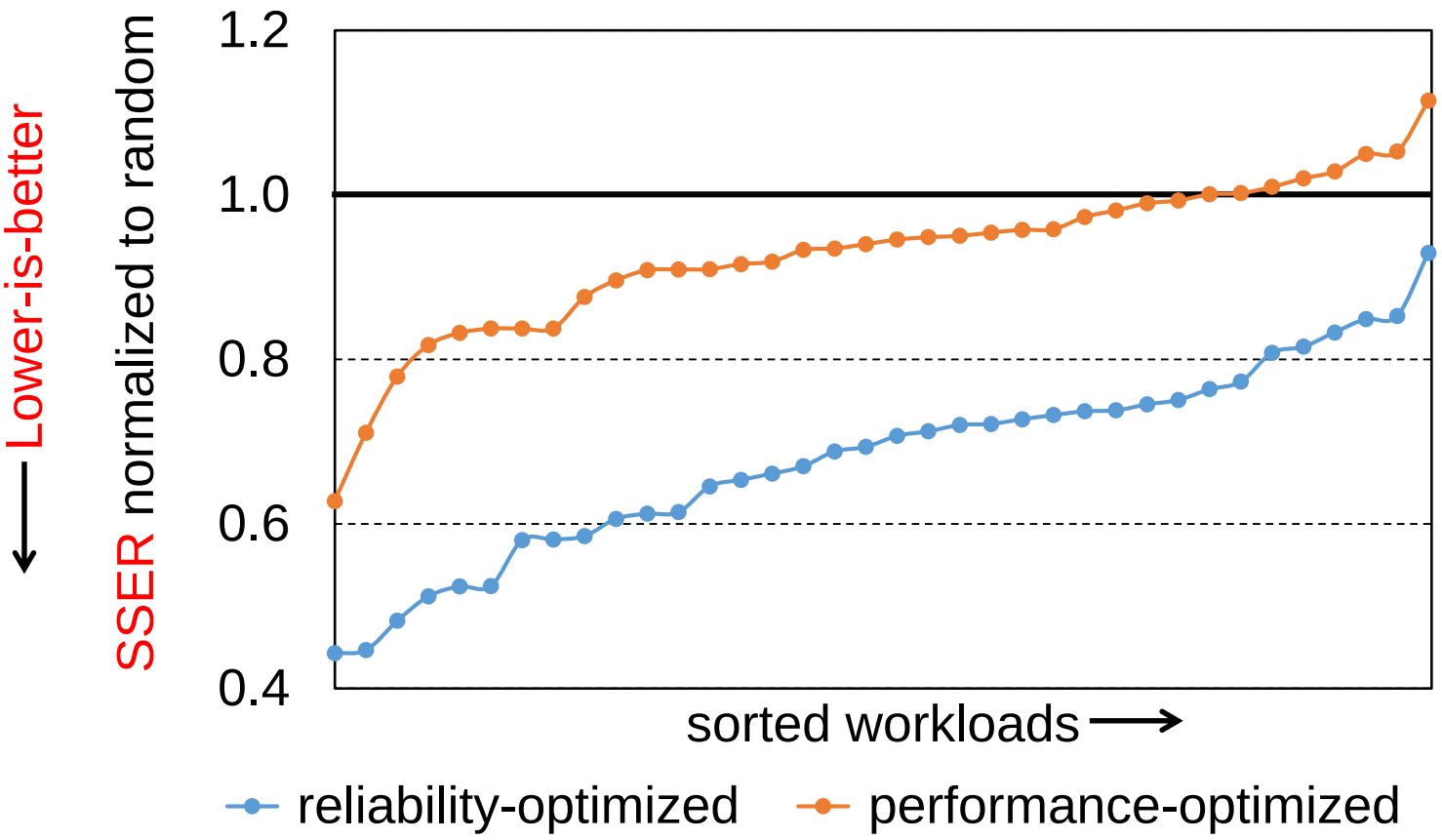
2B2S Results – Reliability



Compared to the random scheduler:

Reliability-optimized scheduler improves SSER by 32% on average and up to 55.6%

2B2S Results – Reliability

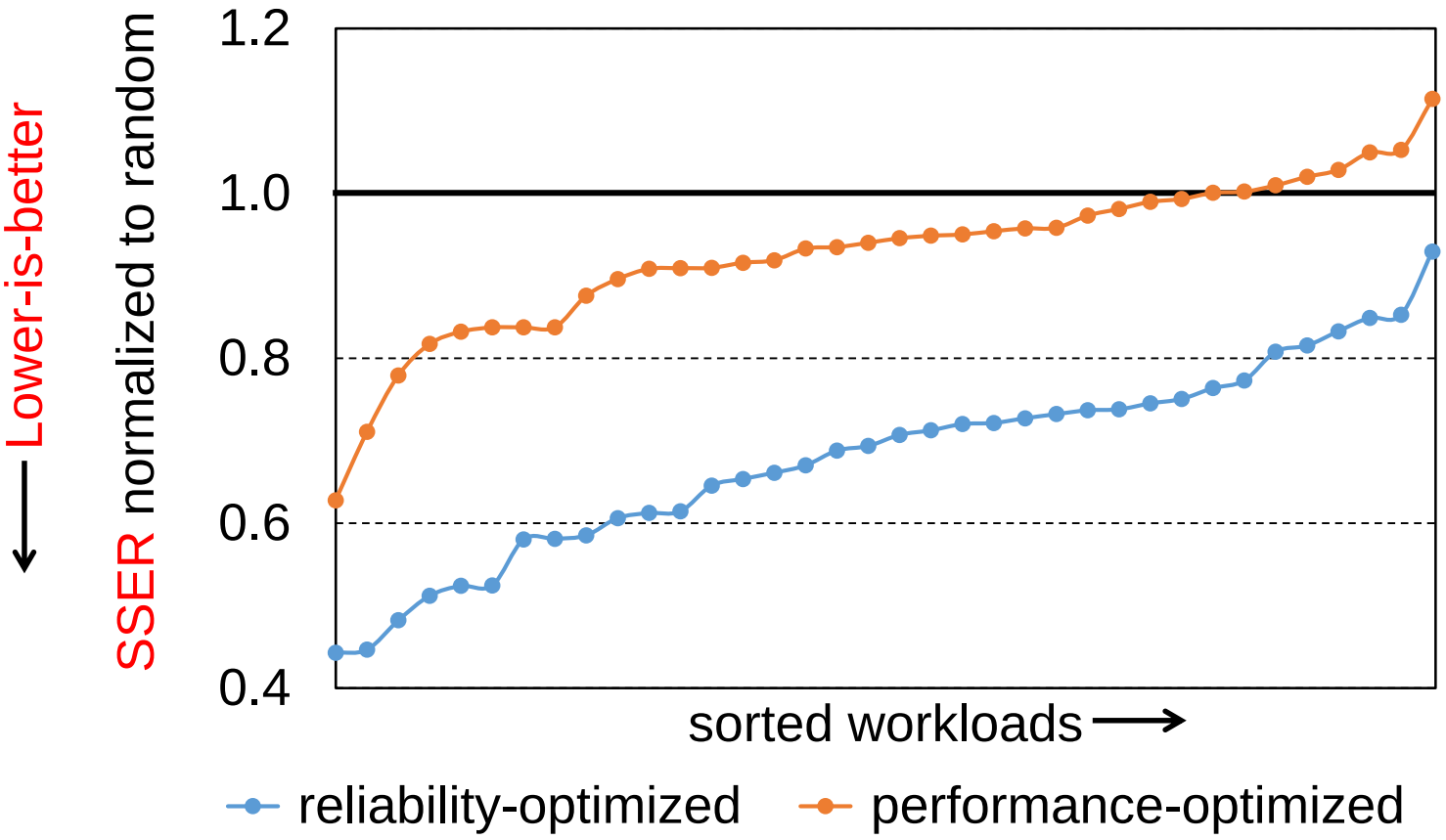


Compared to the random scheduler:

Reliability-optimized scheduler improves SSER by 32% on average and up to 55.6%

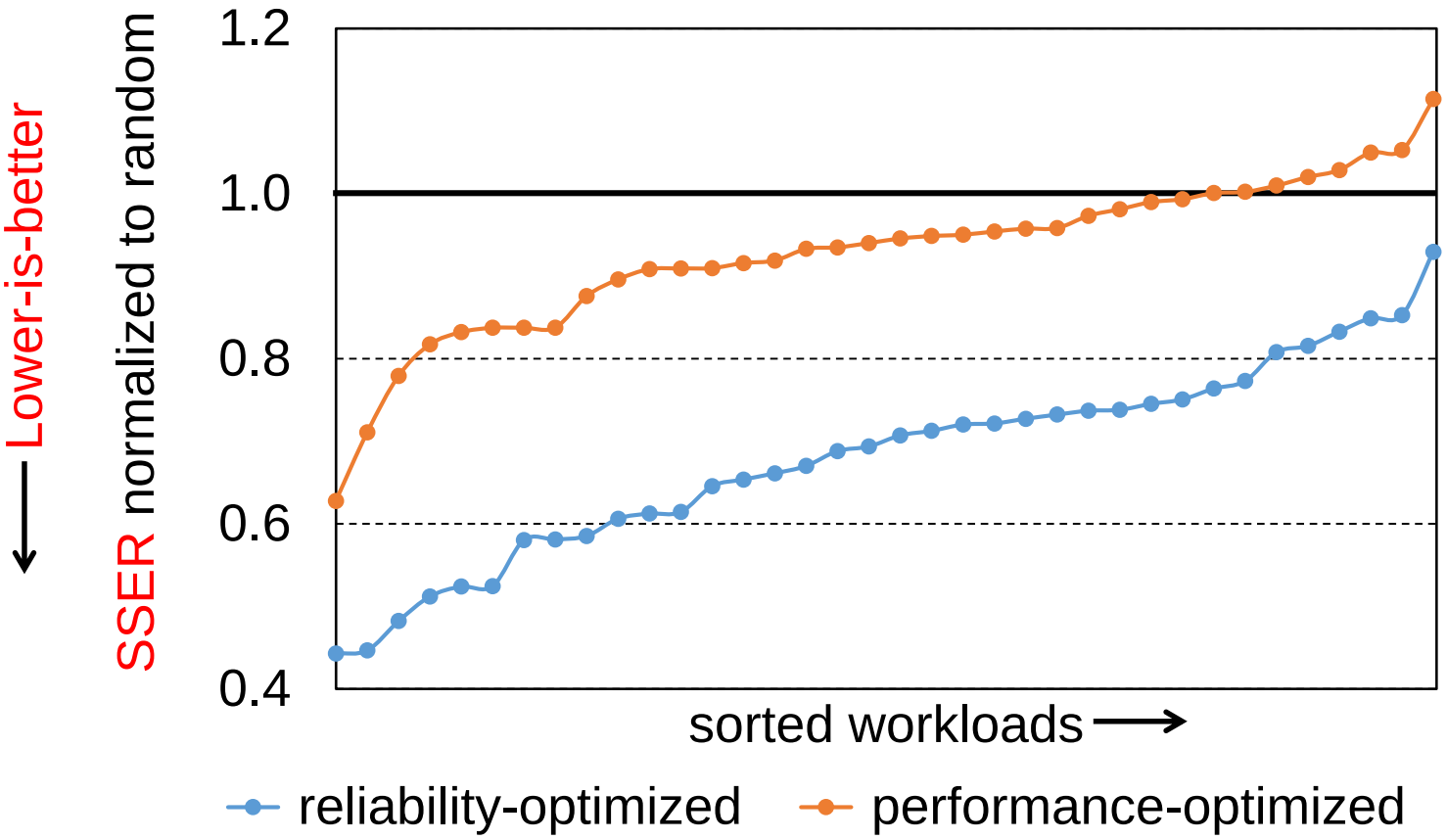
Performance-optimized scheduler also improves SSER by 7.3% on average

2B2S Results – Reliability



Compared to the performance-optimized scheduler:

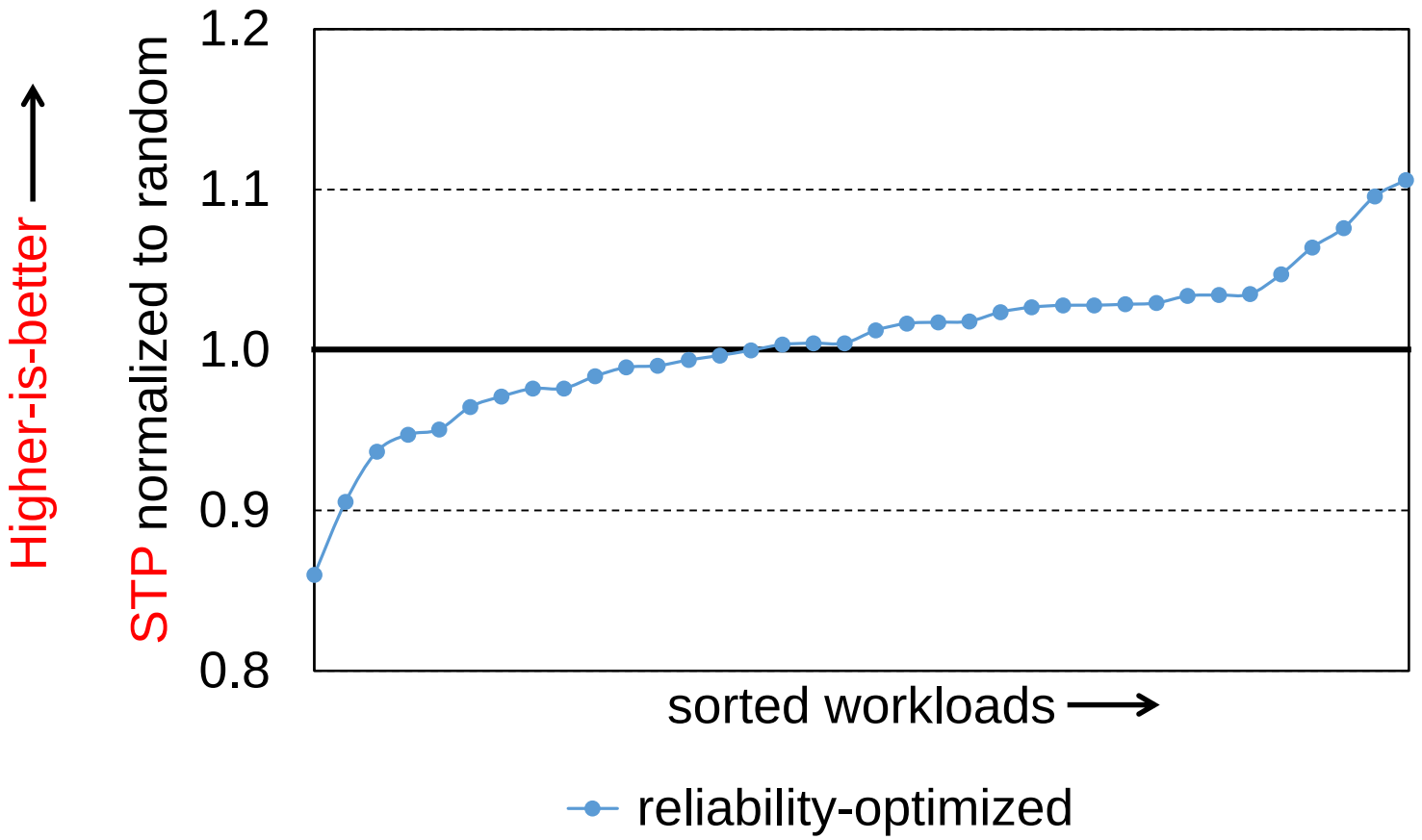
2B2S Results – Reliability



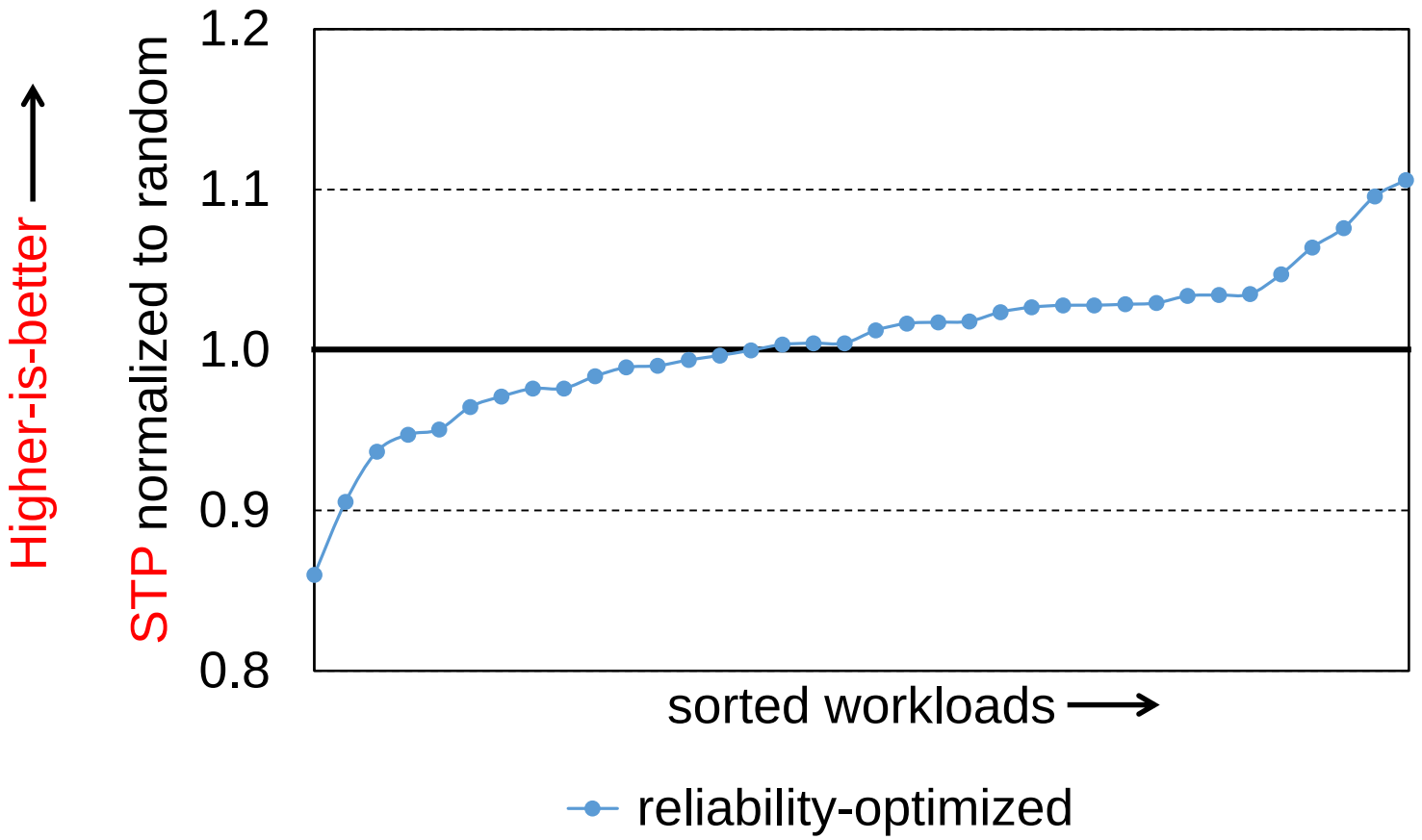
Compared to the performance-optimized scheduler:

Reliability-optimized scheduler improves SSER by 25.4% on average and up to 60.2%

2B2S Results – Performance

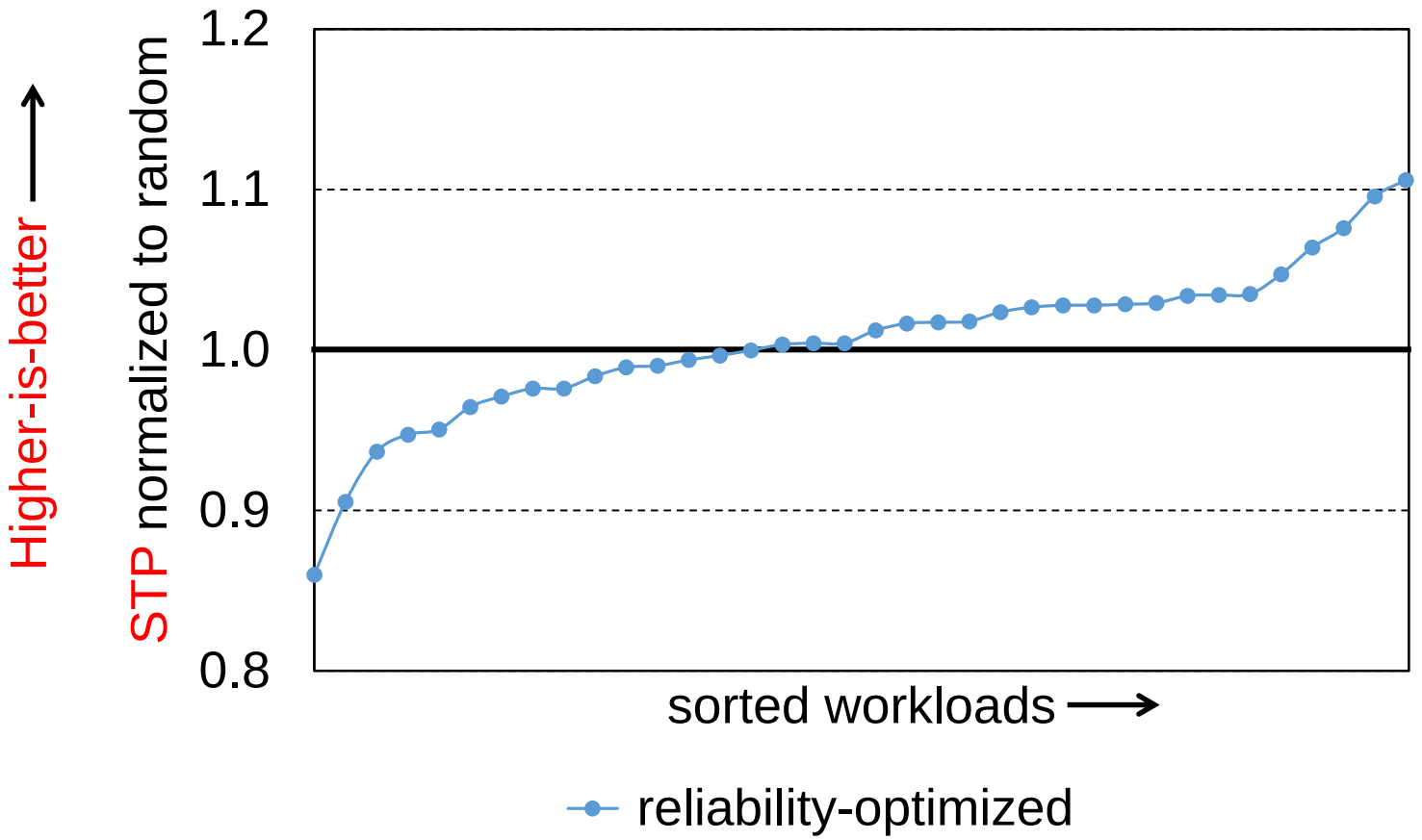


2B2S Results – Performance



Compared to the random scheduler:

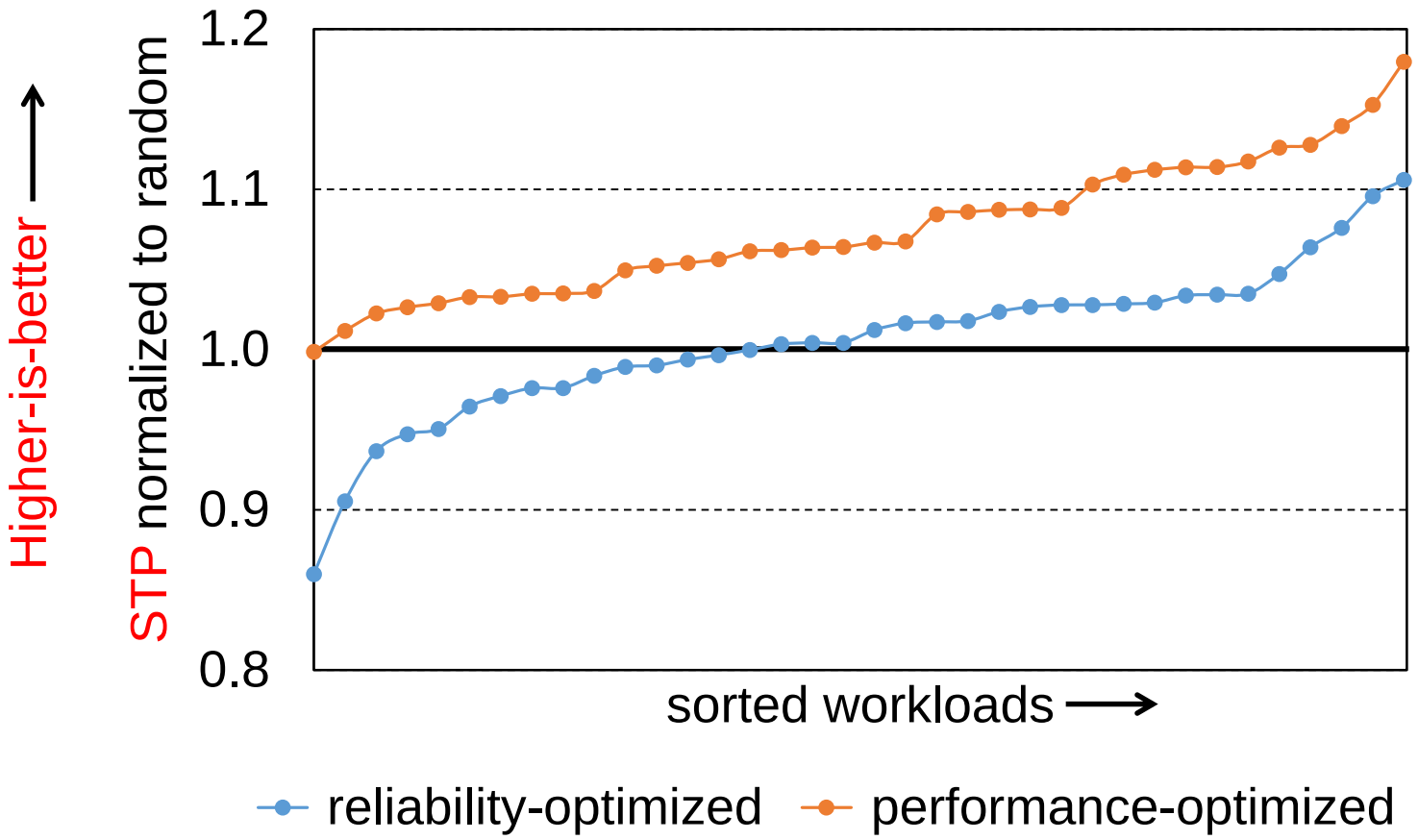
2B2S Results – Performance



Compared to the random scheduler:

Reliability-optimized scheduler does not degrade STP on average

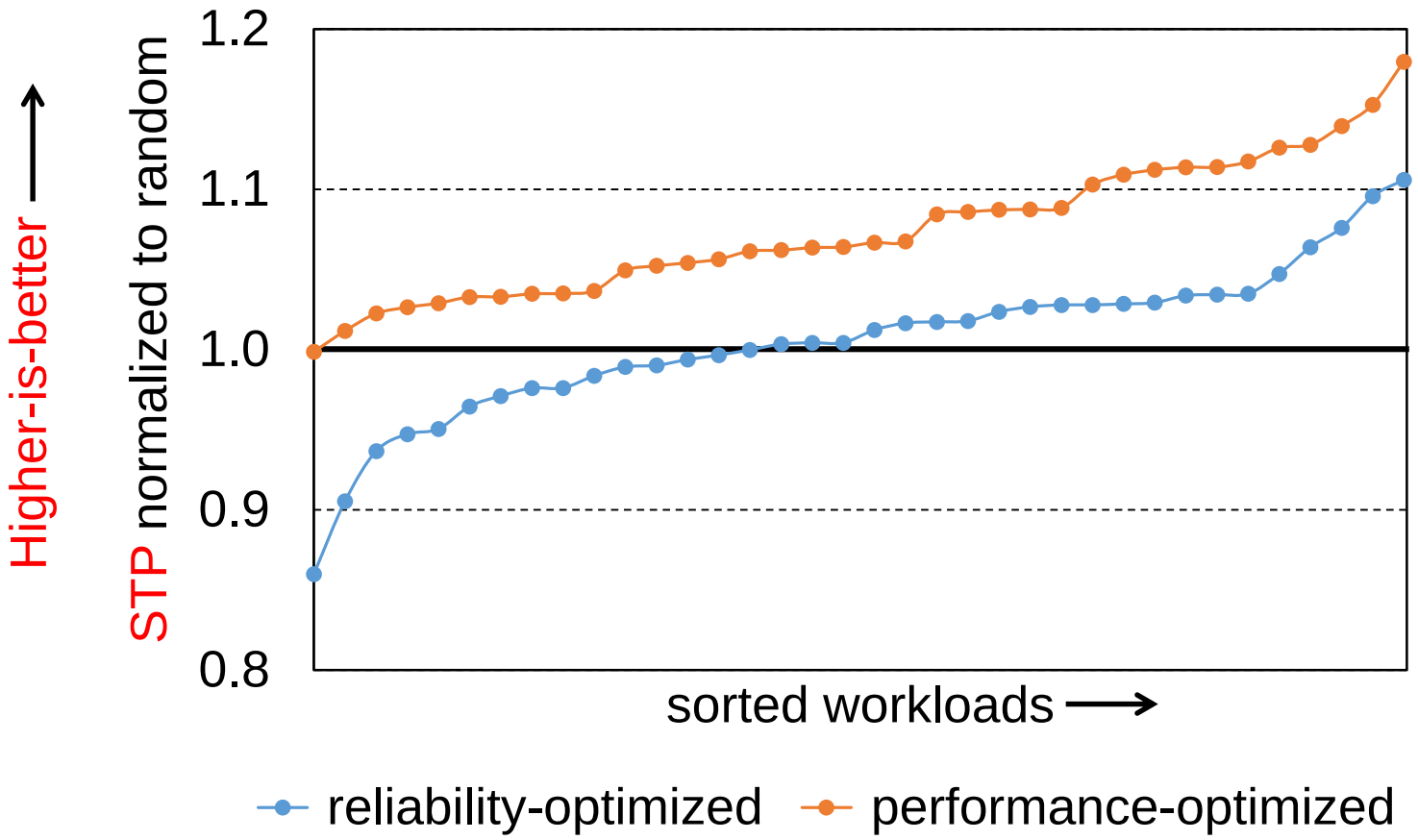
2B2S Results – Performance



Compared to the random scheduler:

Reliability-optimized scheduler does not degrade STP on average

2B2S Results – Performance

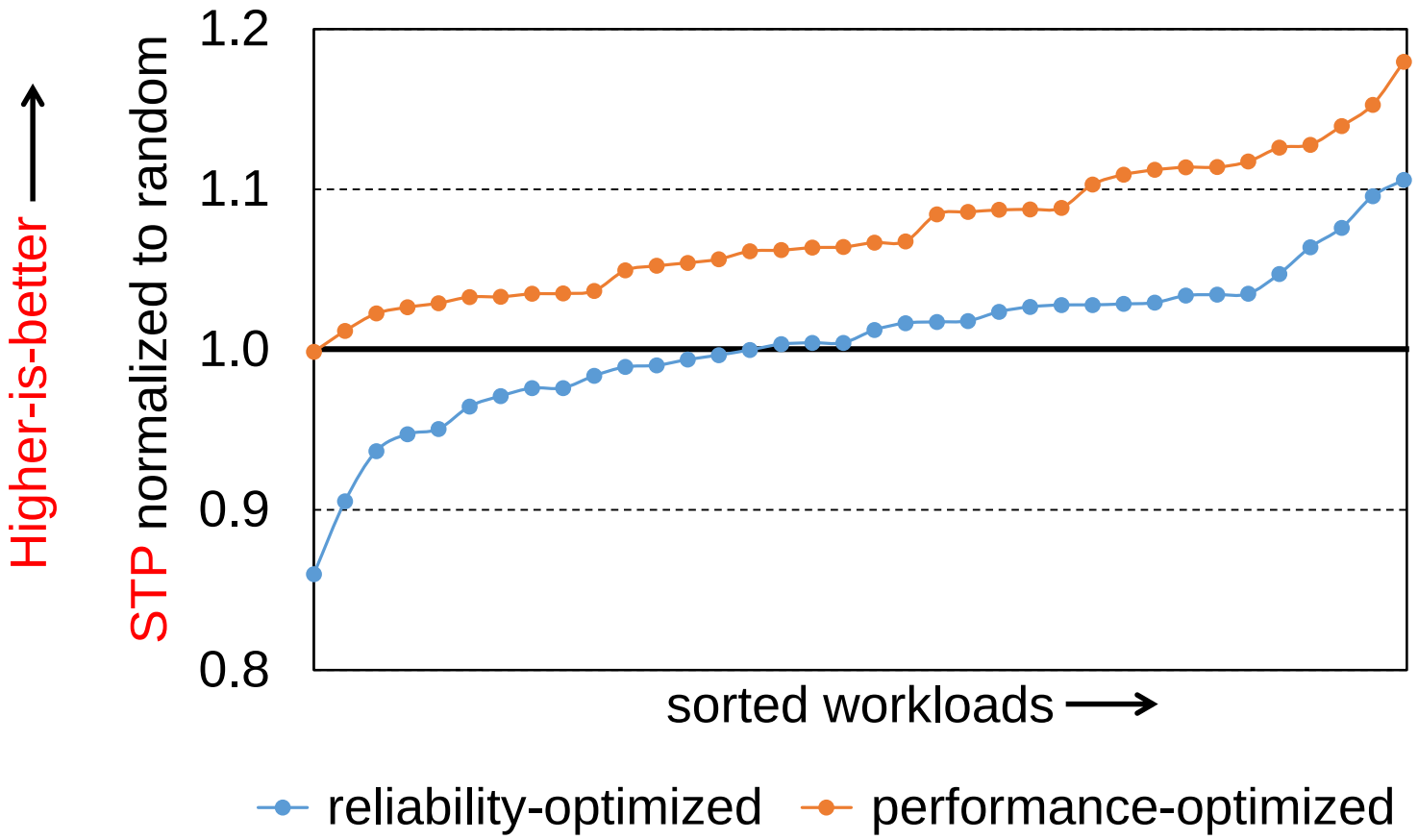


Compared to the random scheduler:

Reliability-optimized scheduler does not degrade STP on average

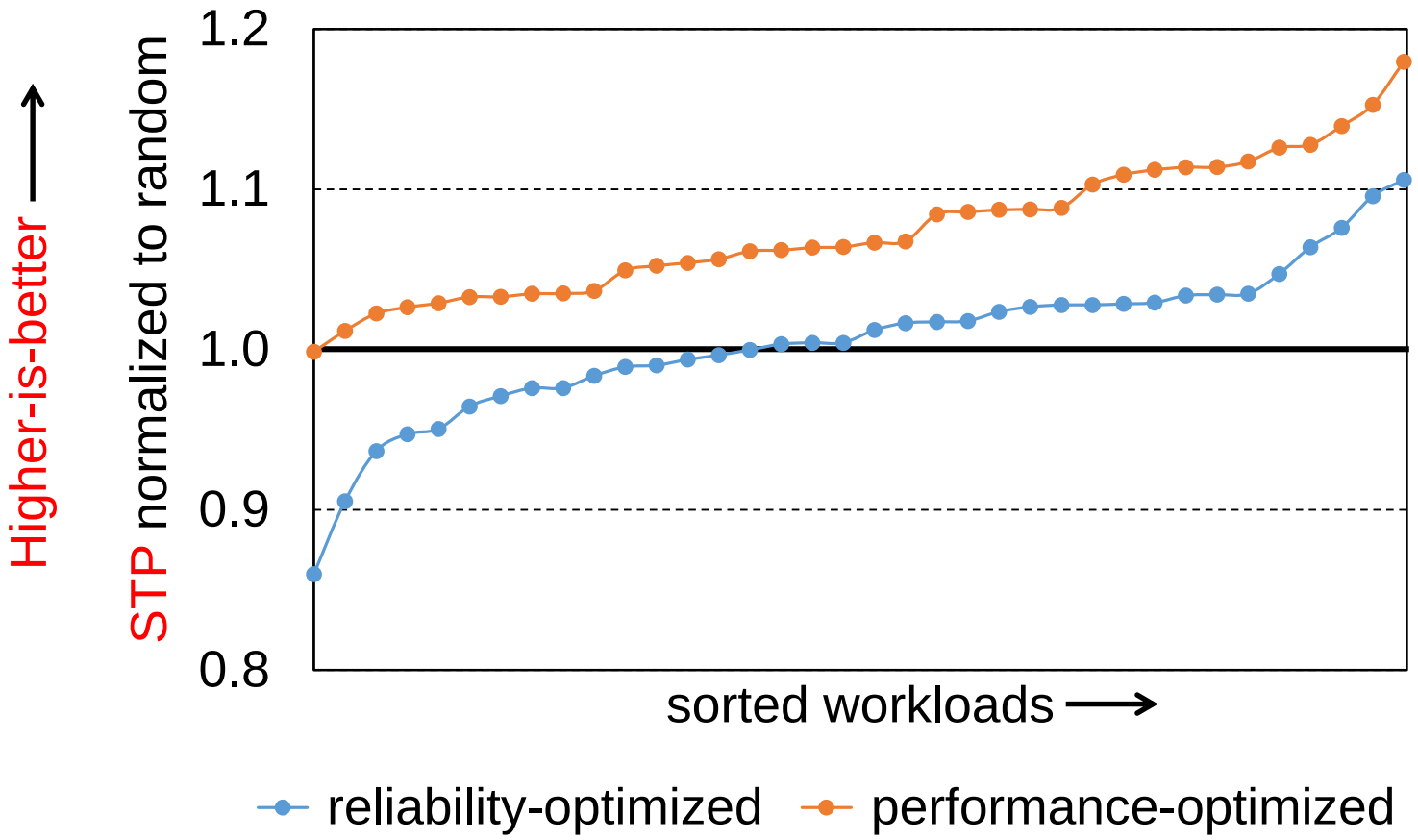
Performance-optimized scheduler improves STP by 6.3% on average and up to 18.7%

2B2S Results – Performance



Compared to the performance-optimized scheduler:

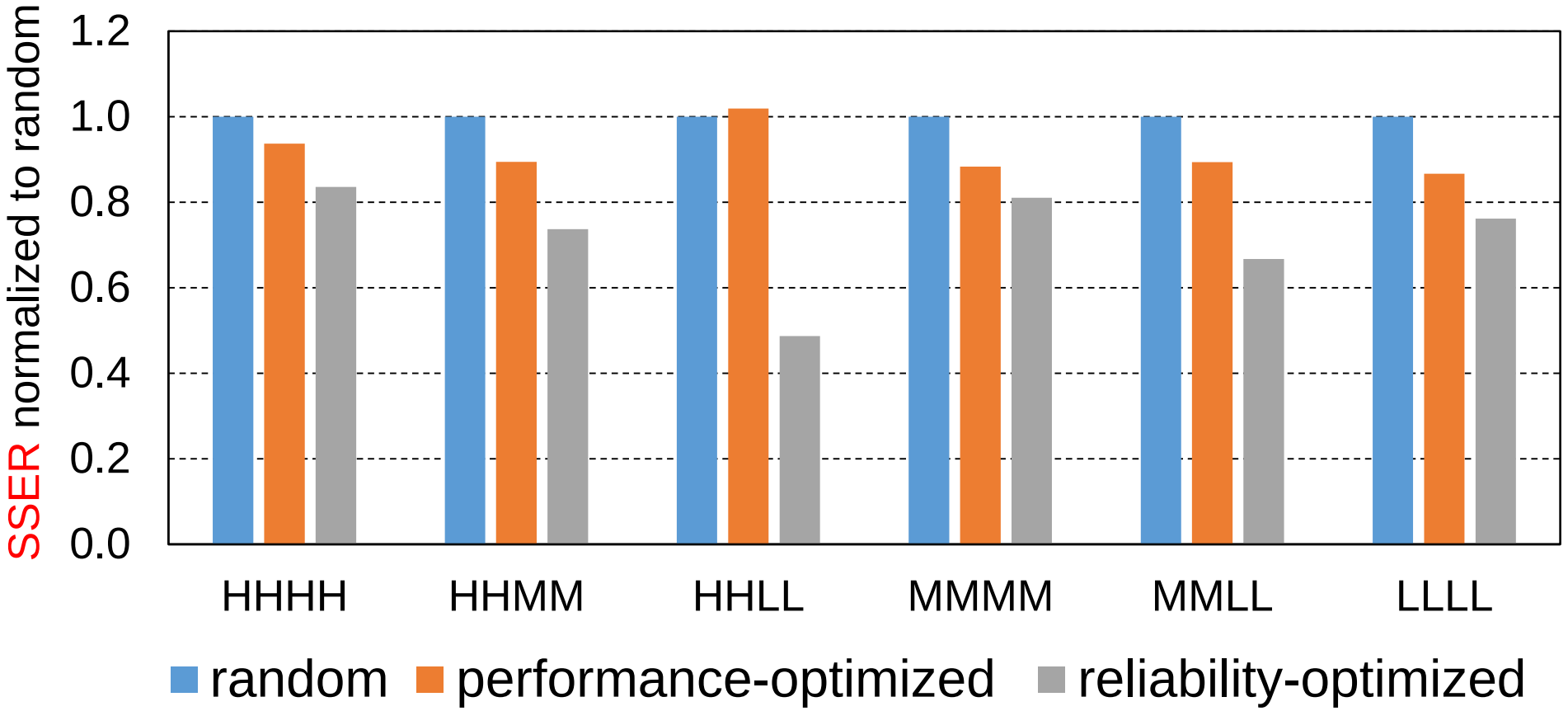
2B2S Results – Performance



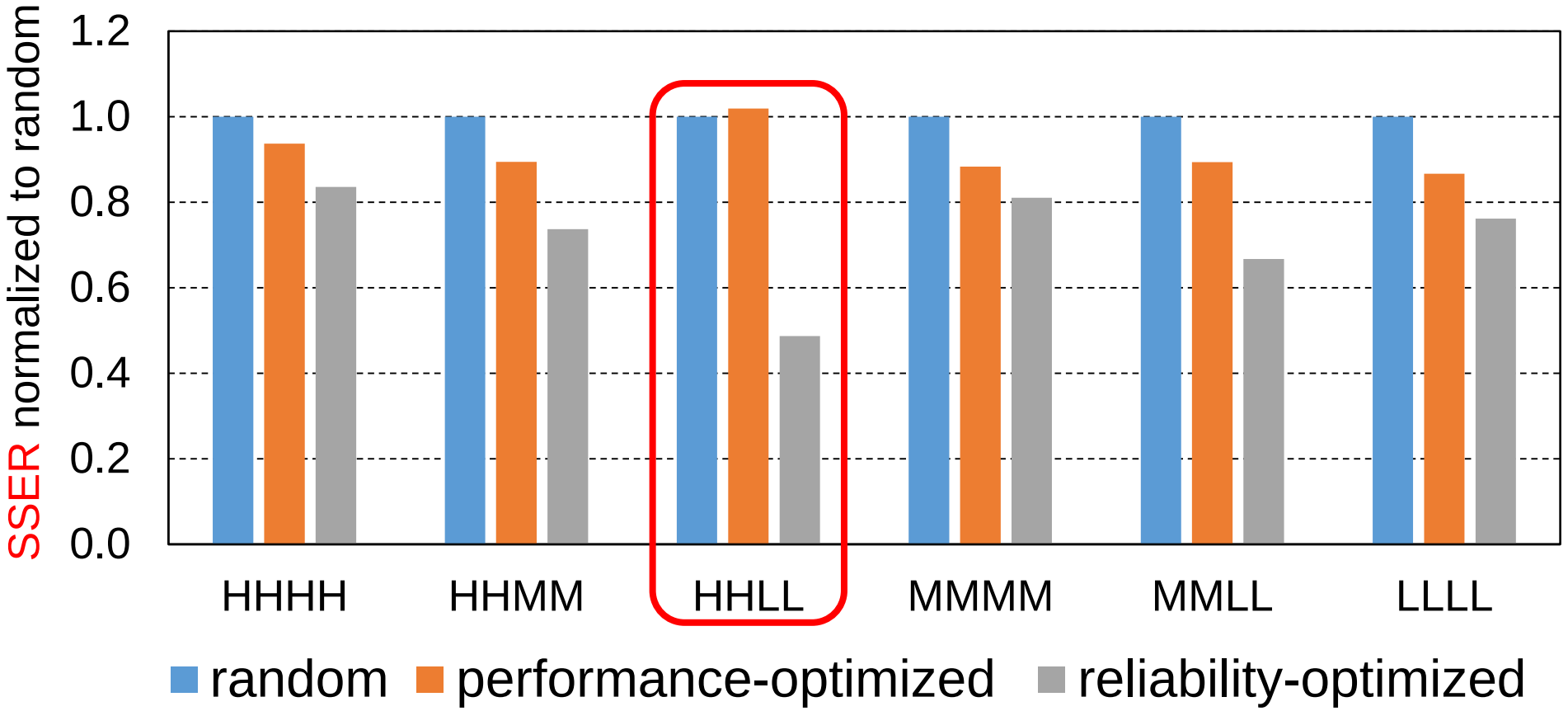
Compared to the performance-optimized scheduler:

Reliability-optimized scheduler degrades STP by 6.3% on average and up to 18.7%

2B2S Results – by Workload Category

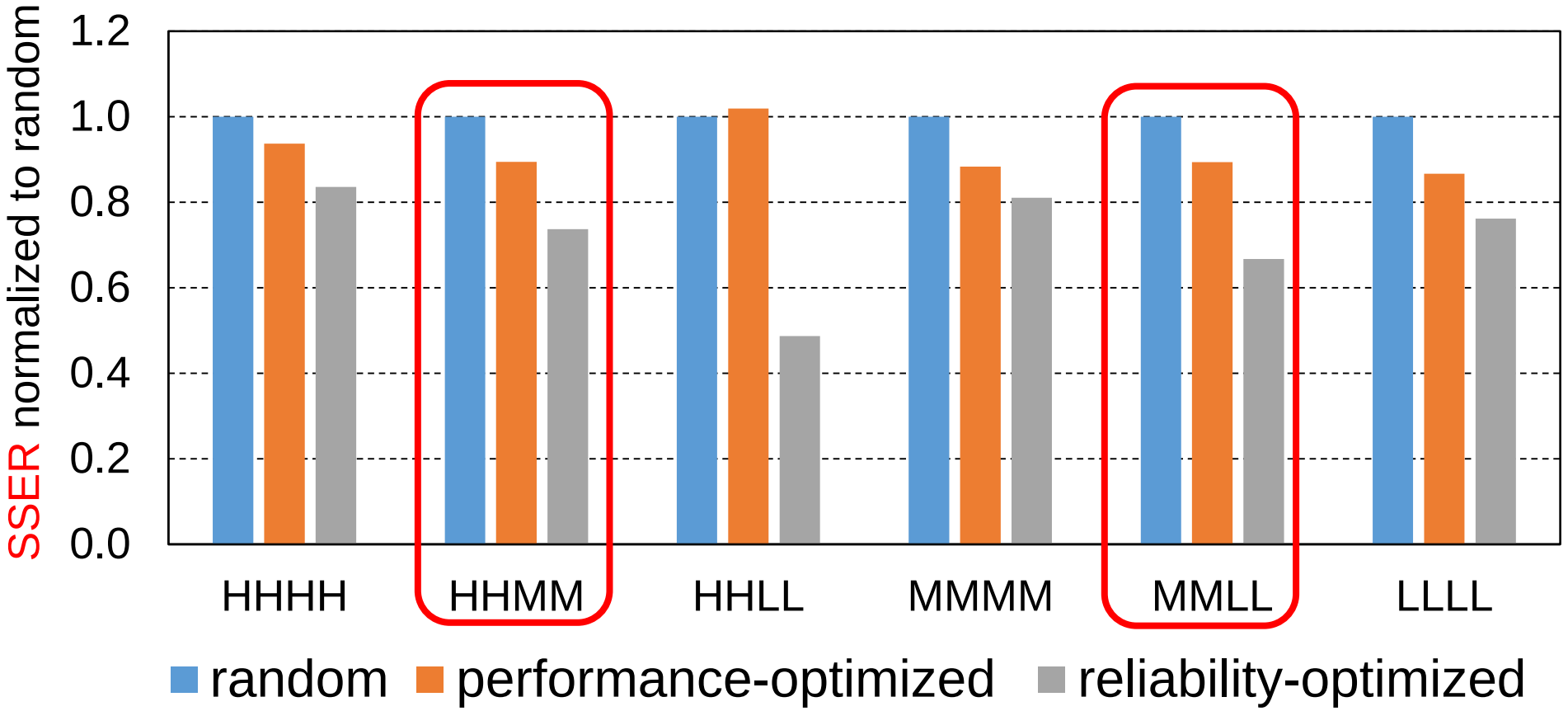


2B2S Results – by Workload Category



Highest improvement in reliability for HHLL (53.2%)

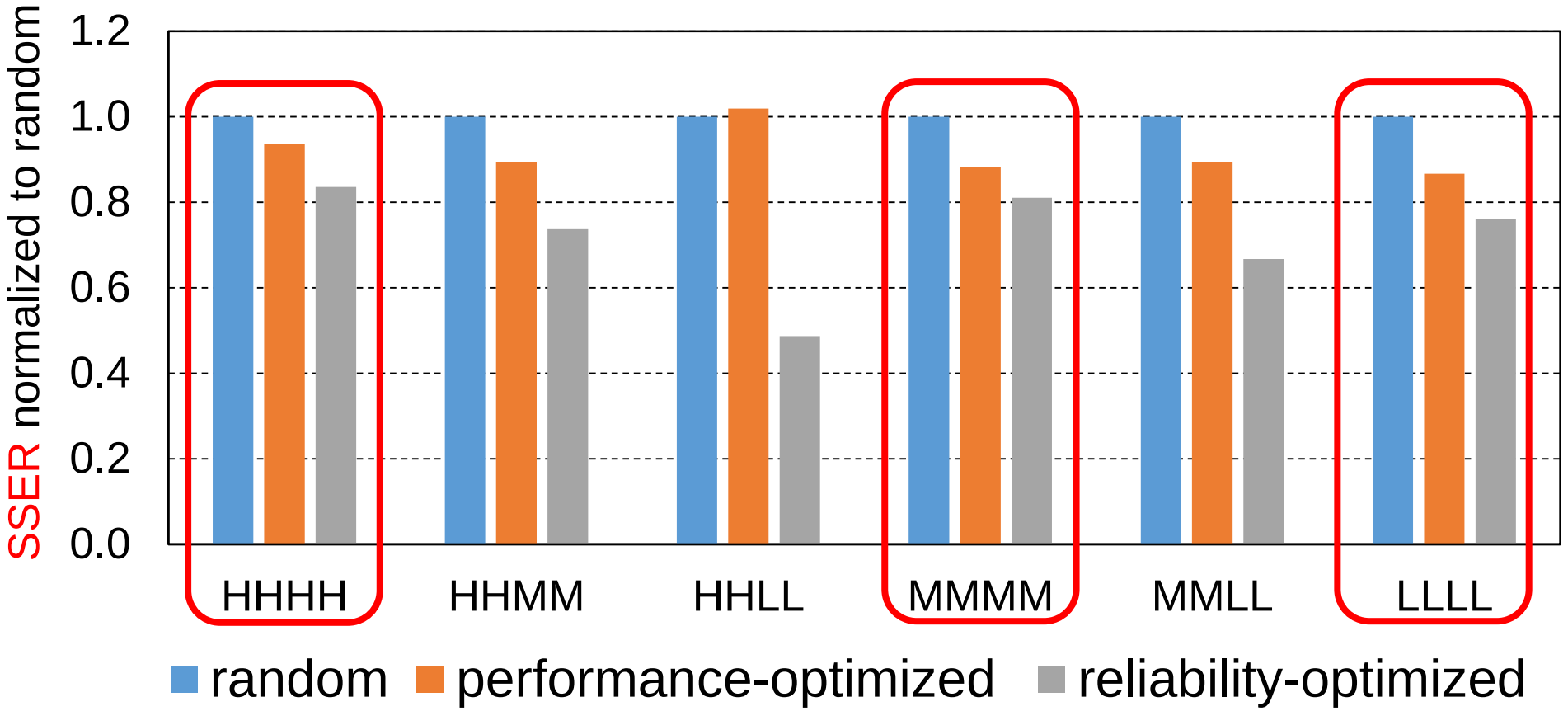
2B2S Results – by Workload Category



Highest improvement in reliability for HHLL (53.2%)

- Next for HHMM (15.8%) and MMLL (22.7%)

2B2S Results – by Workload Category



Highest improvement in reliability for HHLL (53.2%)

- Next for HHMM (15.8%) and MMLL (22.7%)
- Significant improvement for HHHH (10.2%), MMMM (7.3%) and LLLL (10.6%)

Also in the Paper

1. Detailed hardware overhead analysis
2. Asymmetric HCMPs → 1B3S and 3B1S
3. Changing core count → 1B1S, 2B2S and 4B4S
4. Lowering small core frequency from 2.66 GHz → 1.33 GHz
5. Impact on power consumption → Improves by 6.2%
6. Impact of changing sampling and scheduler quanta

Conclusions

Conclusions

1. Reliability characteristics of applications vary on big and small cores

Conclusions

1. Reliability characteristics of applications vary on big and small cores

   - Provides an opportunity for improving reliability on heterogeneous (multicore) processors

Conclusions

1. Reliability characteristics of applications vary on big and small cores

   - Provides an opportunity for improving reliability on heterogeneous (multicore) processors

2. Proposed a dynamic reliability-aware scheduler

   - Improved reliability by 25.4% with 6.3% performance degradation

Conclusions

1. Reliability characteristics of applications vary on big and small cores

   - Provides an opportunity for improving reliability on heterogeneous (multicore) processors

2. Proposed a dynamic reliability-aware scheduler

   - Improved reliability by 25.4% with 6.3% performance degradation

3. Introduced System-level Soft Error Rate (SSER), a novel soft error reliability metric for multiprogramed workloads

FACULTY OF ENGINEERING AND ARCHITECTURE

Reliability-Aware Scheduling on Heterogeneous Multicore Processors

Thank you!

Questions?

Ajeya Naithani, Stijn Eyerman, and Lieven Eeckhout

GHENT UNIVERSITY

intel