# Improving Soft Error Reliability in Modern Processors

**Ajeya Naithani**

**Advisor: Prof. Lieven Eeckhout**

**Doctoral Thesis Defense**

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

# Reliability in Computer Science

The term reliability refers to the ability of a computer hardware or software component to consistently perform according to its specifications.

# What if the Reliability is Compromised?

# What if the Reliability is Compromised?

```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: kbdhid.sys

MANUALLY_INITIATED_CRASH

If this is the first time you've seen this stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use safe mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical Information:

*** STOP: 0x000000e2 (0x00000000, 0x00000000, 0x00000000, 0x00000000)

*** kbdhid.sys - Address 0x94efd1aa base at 0x94efb000 DateStamp 0x4a5bc705
```

# What if the Reliability is Compromised?

SARAH SCOLES SCIENCE 06.04.2018 07:00 AM

## Cosmic Ray Showers Crash Supercomputers. Here's What to Do About It

What happens when a national laboratory's supercomputers start glitching?

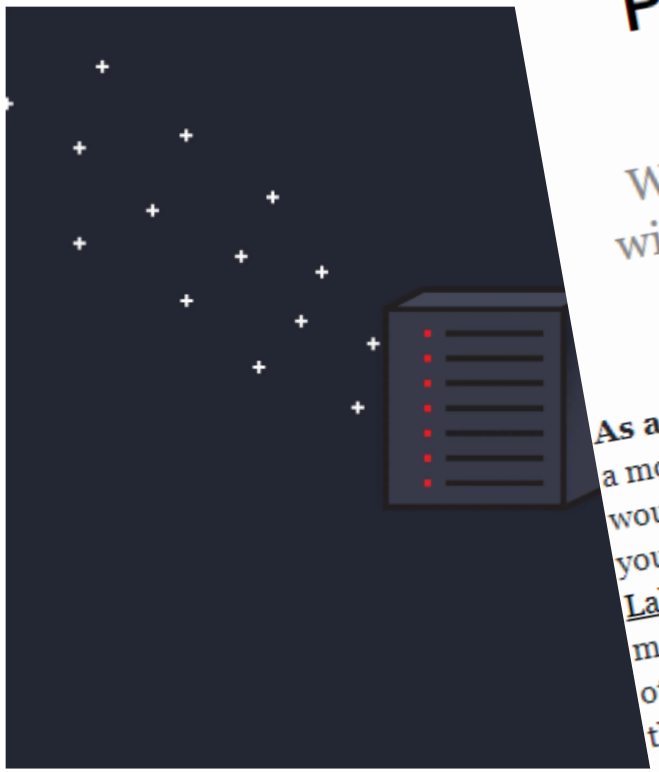
EMILY WAITE

# What if the Reliability is Compromised?

SARAH SCOLES SCIENCE 06.04.2018 07:00 AM

## Cosmic Ray Showers Crash Supercomputers. Here's About It

What happens when a national laboratory's superc



EMILY WAITE

## How To Kill A Supercomputer: Dirty Power, Cosmic Rays, and Bad Solder

Will future exascale supercomputers be able to withstand the steady onslaught of routine faults?

By Al Geist

**As a child**, were you ever afraid that a monster lurking in your bedroom would leap out of the dark and get you? My job at Oak Ridge National Laboratory is to worry about a similar monster, hiding in the steel cabinets of the supercomputers and threatening to crash the largest computing machines on the planet.

The monster is something

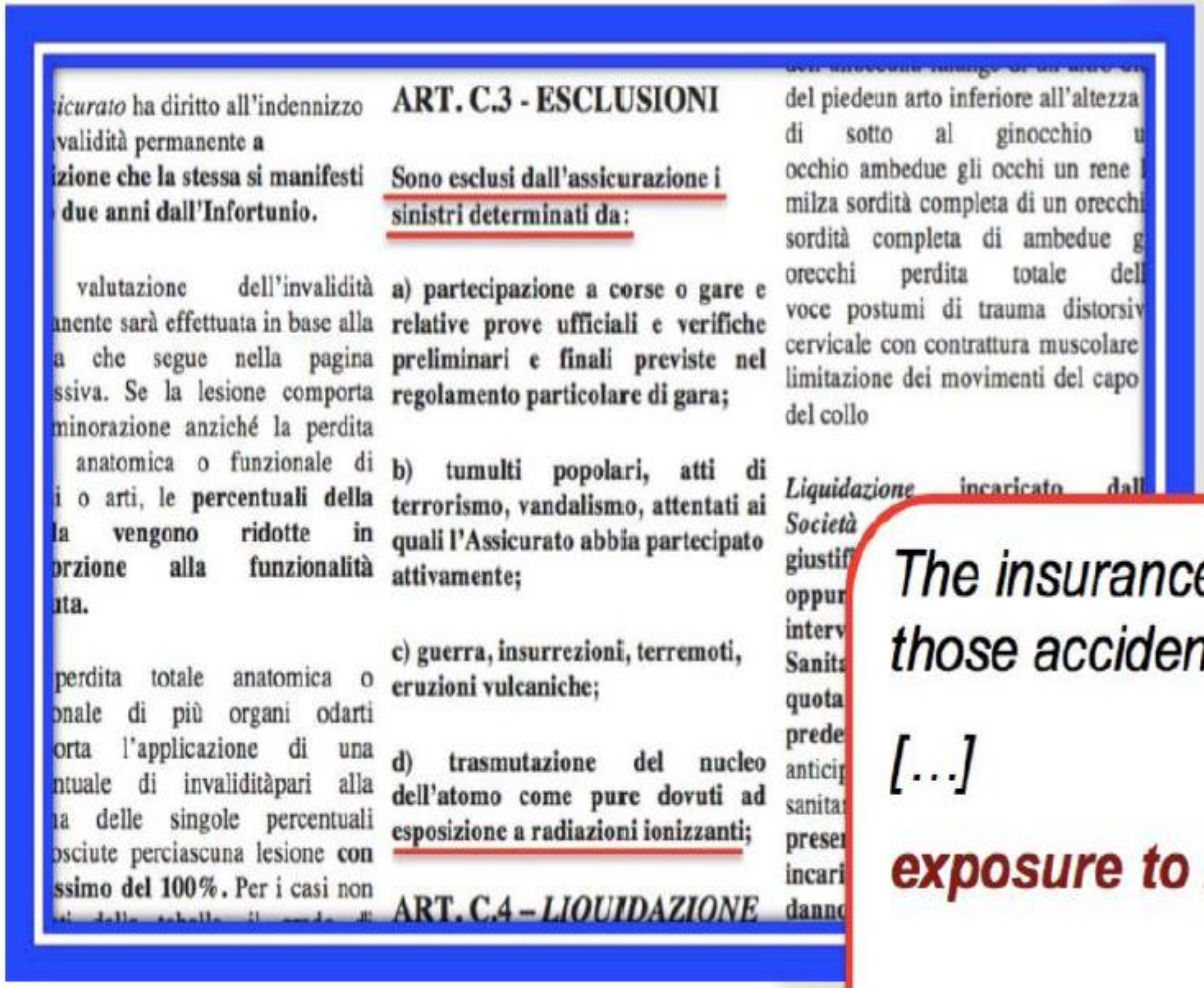
# What if the Reliability is Compromised?

## The Uber Crash Won't Be the Last Shocking Self-Driving Death

These systems will fail, and when the do, the result will look nothing like human-powered crashes.


KIEN HONG LE/BLOOMBERG/GETTY IMAGES

# Did you Read your Car Insurance Carefully?



*icurato* ha diritto all'indennizzo
validità permanente a
**izione che la stessa si manifesti**
**due anni dall'Infortunio.**

valutazione dell'invalidità
nente sarà effettuata in base alla
a che segue nella pagina
ssiva. Se la lesione comporta
minorazione anziché la perdita
anatomica o funzionale di
i o arti, le **percentuali della**
**la vengono ridotte in**
**orzione alla funzionalità**
**ita.**

perdita totale anatomica o
onale di più organi odarti
orta l'applicazione di una
ntuale di invalidità pari alla
a delle singole percentuali
osciute perciascuna lesione **con**
**ssimo del 100%.** Per i casi non

**ART. C.3 - ESCLUSIONI**

**Sono esclusi dall'assicurazione i sinistri determinati da:**

**a) partecipazione a corse o gare e relative prove ufficiali e verifiche preliminari e finali previste nel regolamento particolare di gara;**

**b) tumulti popolari, atti di terrorismo, vandalismo, attentati ai quali l'Assicurato abbia partecipato attivamente;**

**c) guerra, insurrezioni, terremoti, eruzioni vulcaniche;**

**d) trasmutazione del nucleo dell'atomo come pure dovuti ad esposizione a radiazioni ionizzanti;**

**ART. C.4 – *LIQUIDAZIONE***

del piedeun arto inferiore all'altezza
di sotto al ginocchio u
occhio ambedue gli occhi un rene
milza sordità completa di un orecchi
sordità completa di ambedue g
orecchi perdita totale dell
voce postumi di trauma distorsiv
cervicale con contrattura muscolare
limitazione dei movimenti del capo
del collo

***Liquidazione*** incaricato dall
***Società***
giustif
oppur
interv
Sanit
quota
prede
anticip
sanita
preser
incari
danno

*The insurance does not cover those accidents caused by:*

*[…]*

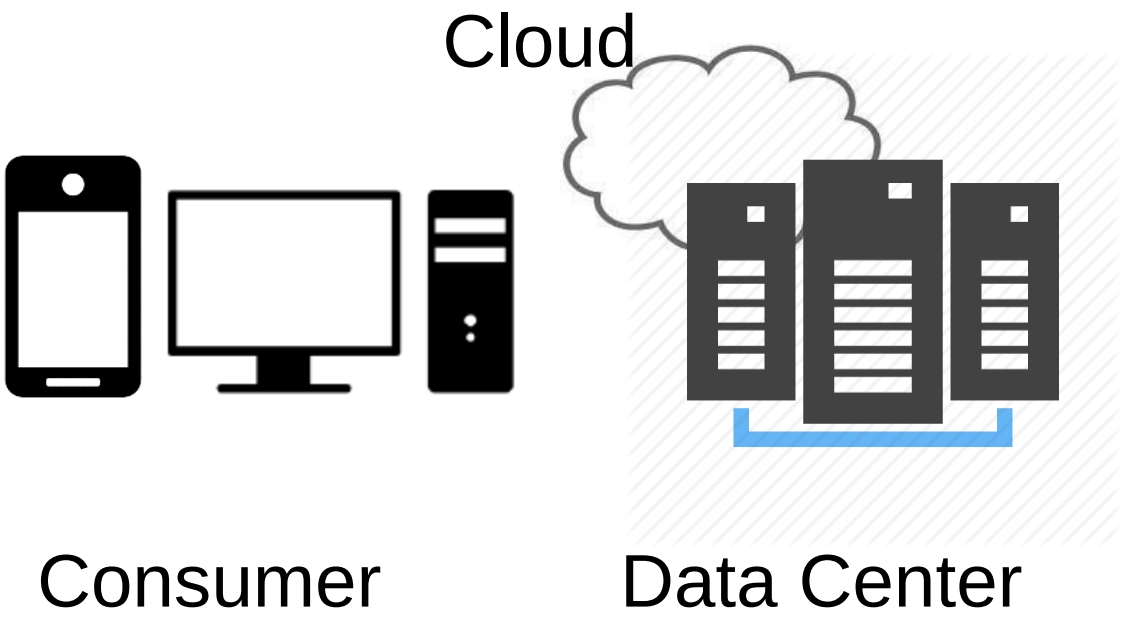
***exposure to ionizing radiation****

# Different Types of Computing Systems

# Different Types of Computing Systems

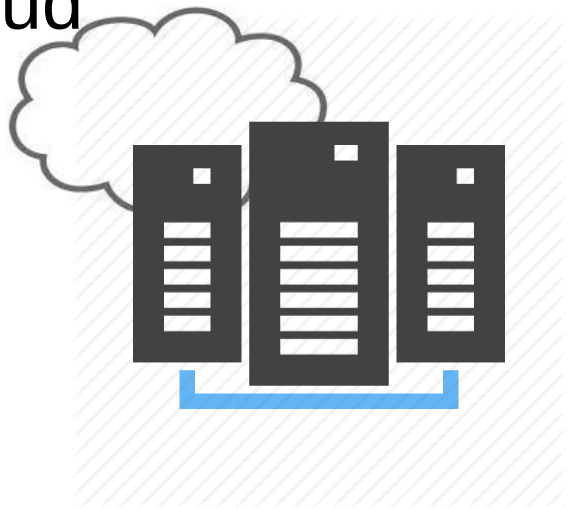
Consumer

# Different Types of Computing Systems



# Different Types of Computing Systems

Consumer




Data Center

Scientific Computing

# Different Types of Computing Systems

Consumer

Cloud

Data Center

Scientific Computing

Automotive

# Different Reliability Requirements

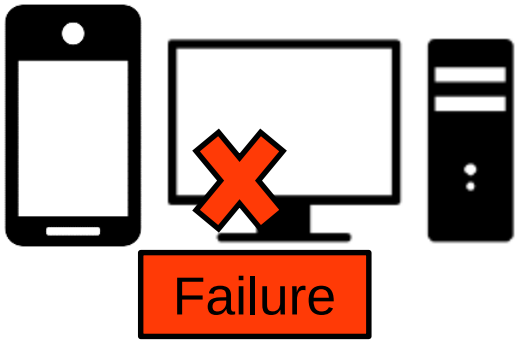
Consumer


Data Center

Scientific Computing

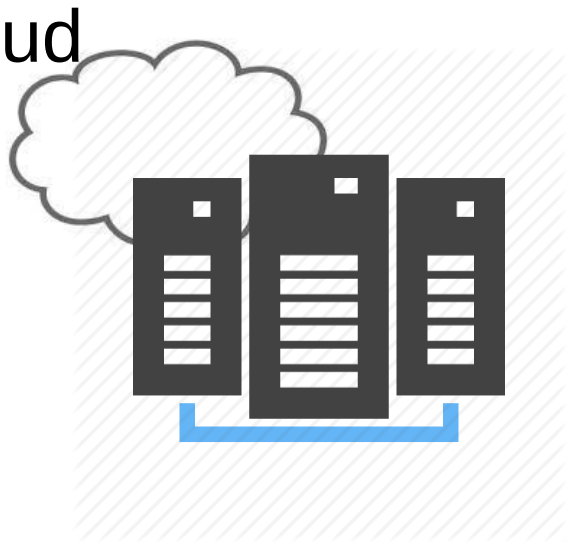
Automotive

# Different Reliability Requirements




Consumer

**Less critical**

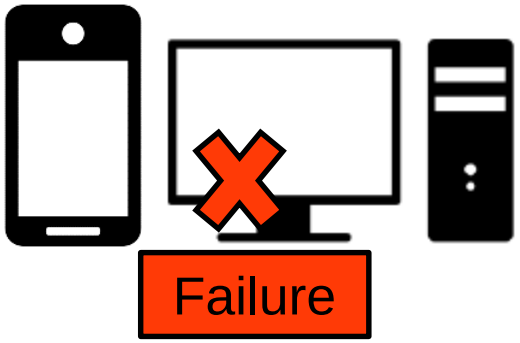



Data Center

Scientific Computing

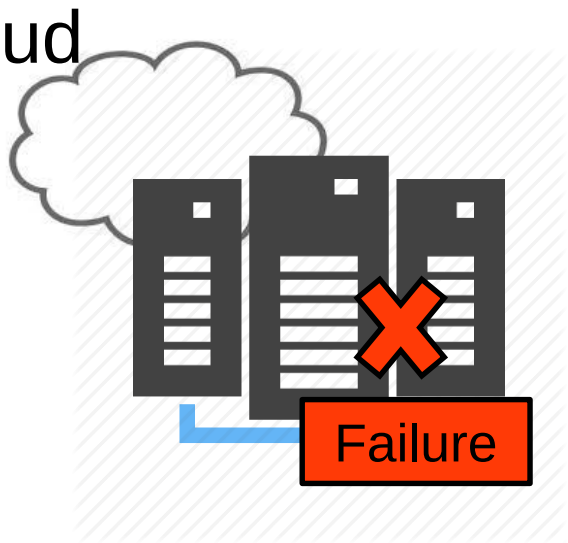
Automotive

# Different Reliability Requirements




Consumer

**Less critical**



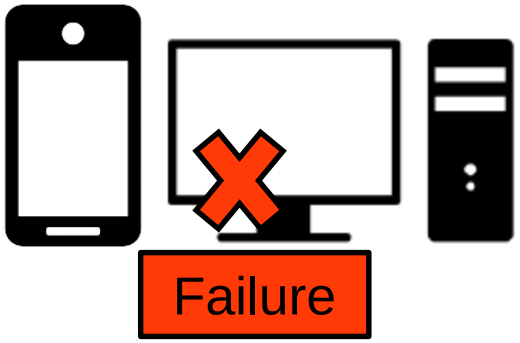

Data Center

**Critical**

Scientific Computing

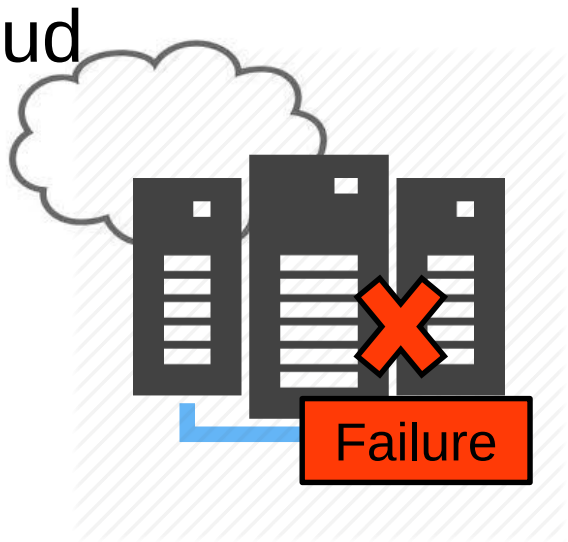
Automotive

# Different Reliability Requirements




Consumer

**Less critical**




Data Center

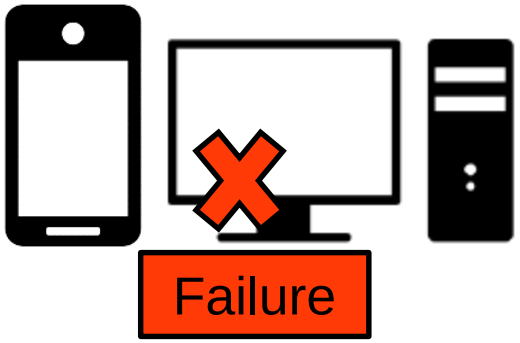
**Critical**


Scientific Computing

**More Critical**

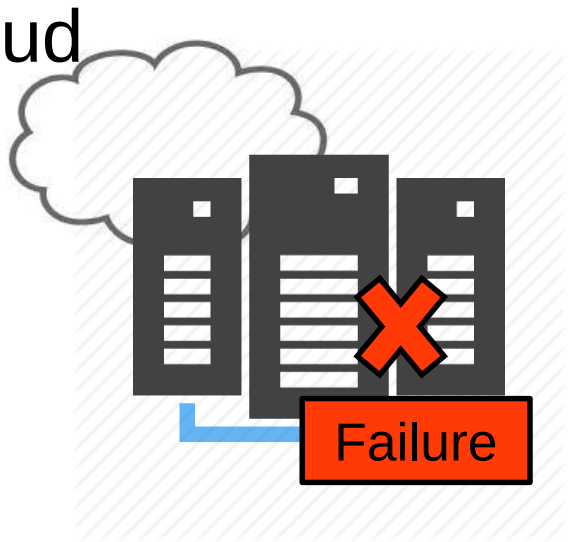
Automotive

# Different Reliability Requirements




Consumer

**Less critical**




Data Center

**Critical**

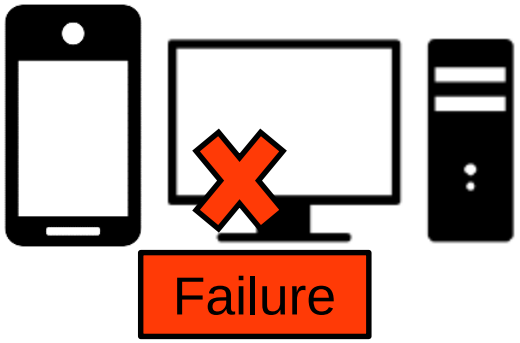

Scientific Computing

**More Critical**


Automotive

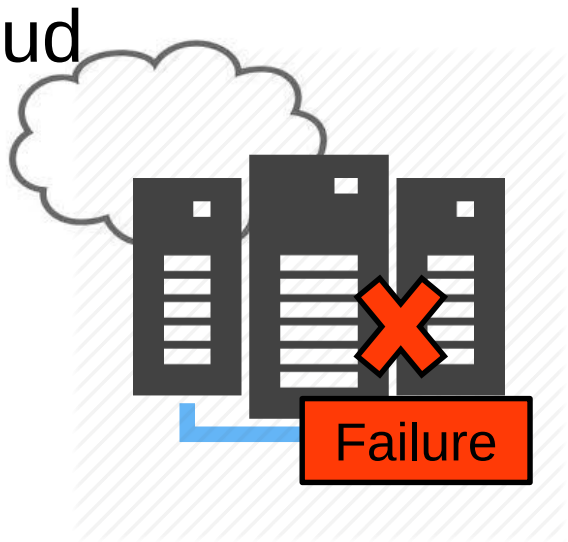
**Human life**

# Different Reliability Requirements




Consumer

**Less critical**




Data Center

**Critical**


Scientific Computing

**More Critical**

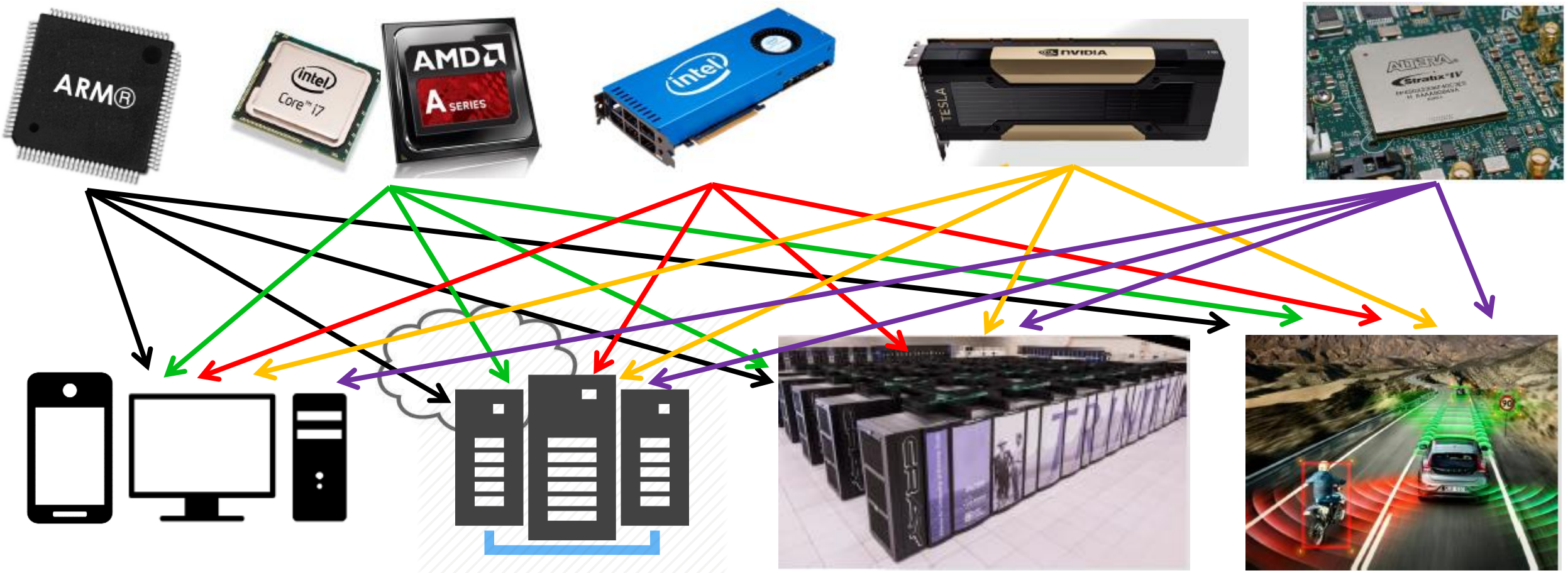

Automotive

**Human life**

**Reliability Requirements**

# Different Devices Running These Systems

ARM CPU Coprocessor GPU FPGA/SoC




Consumer Data Center Scientific Computing Automotive

[Arijit Biswas, SELSE 2018]

# Two Key Components: Processor and Memory

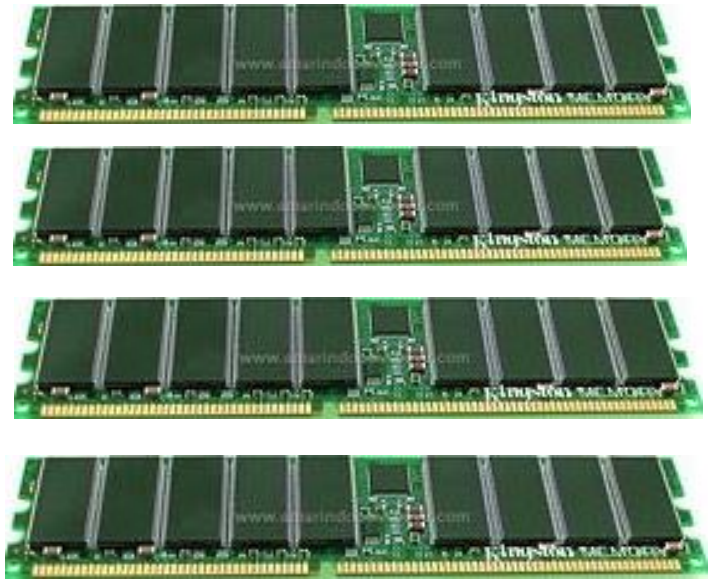
# Two Key Components: Processor and Memory

Processor

# Two Key Components: Processor and Memory

Processor


Memory

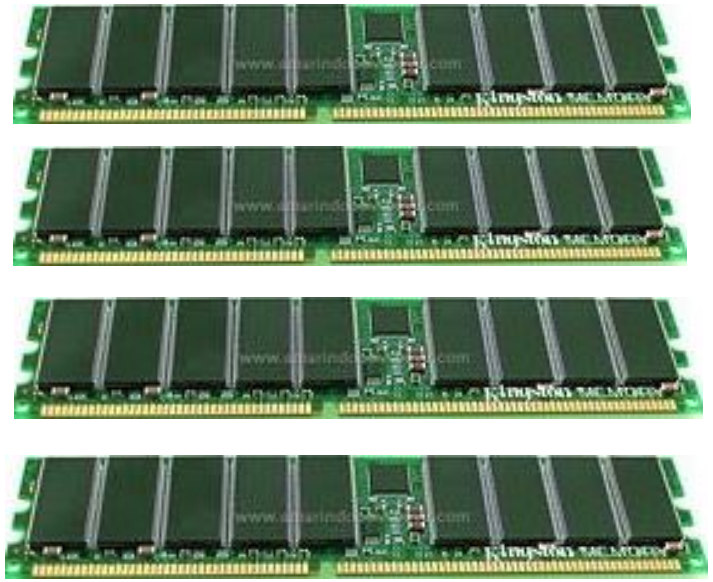


# Two Key Components: Processor and Memory

Processor


Read

Memory



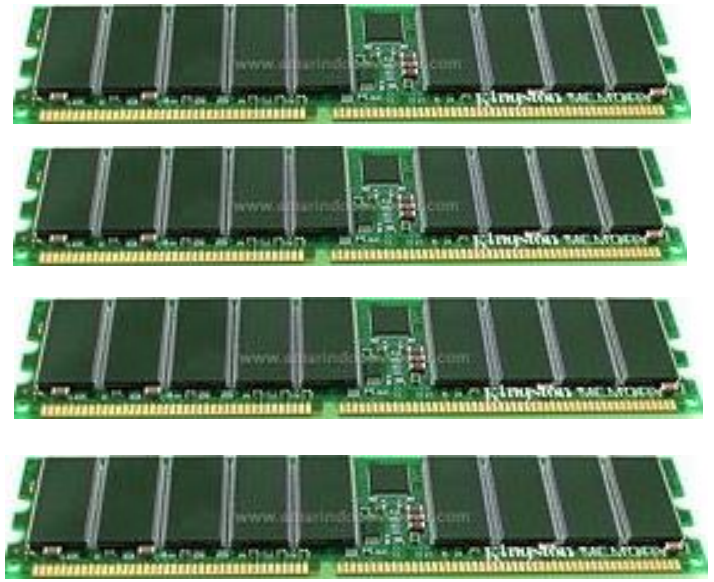
# Two Key Components: Processor and Memory

Processor


Compute

Read

Memory



# Two Key Components: Processor and Memory

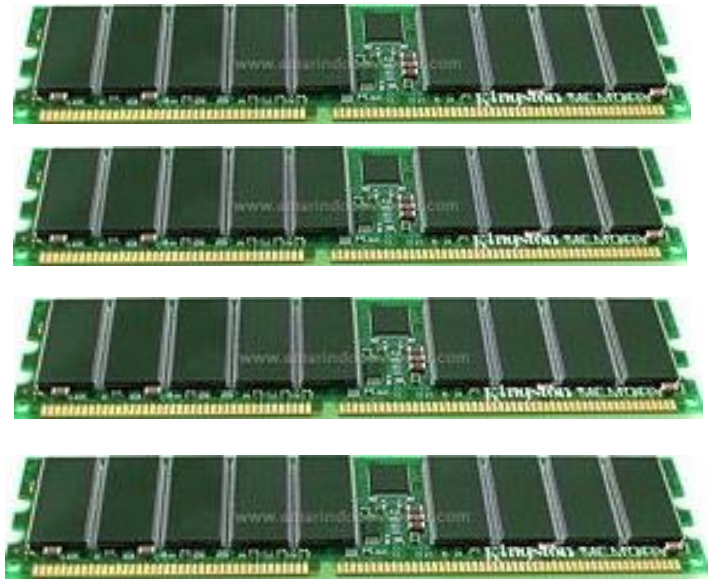
Processor


Compute

Read

Write

Memory



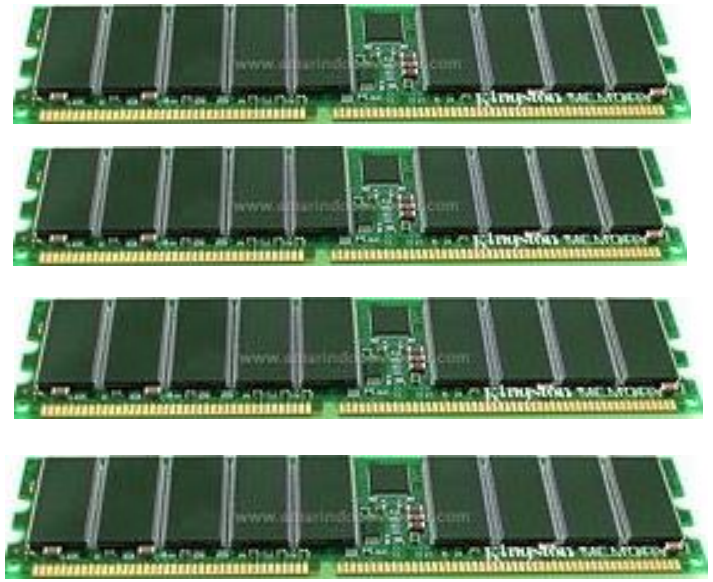
# Two Key Sources of Failures: Processor and Memory

Processor

Memory


Read

Write



Compute

# Two Key Sources of Failures: Processor and Memory

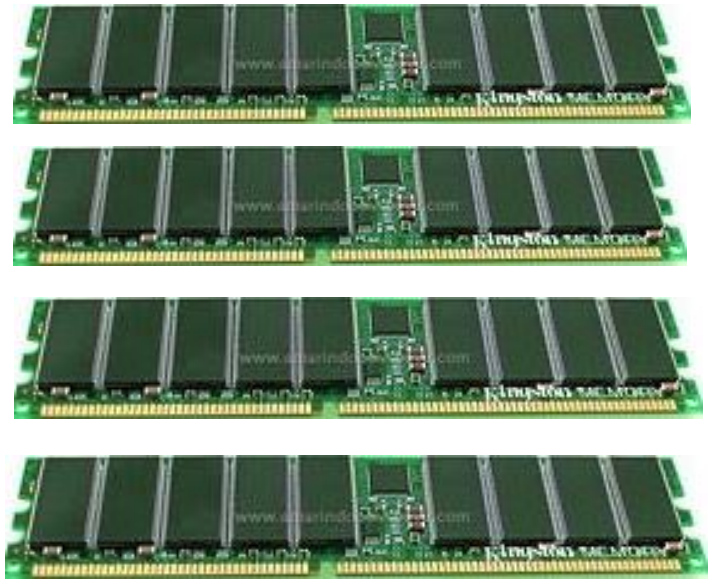
Processor

Memory

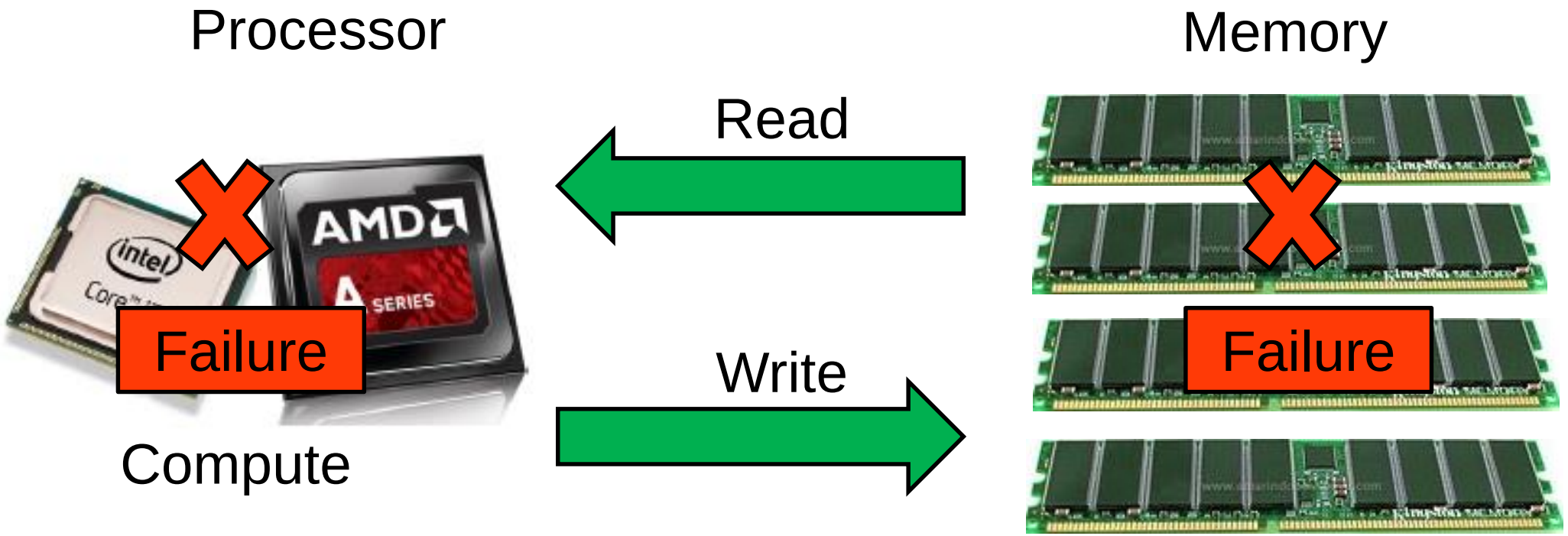

Compute

Read

Write

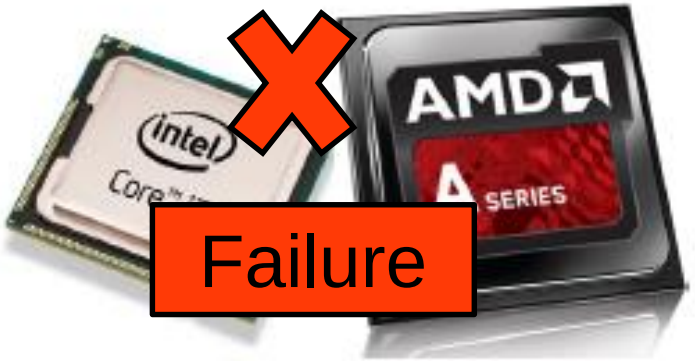


# Two Key Sources of Failures: Processor and Memory



# Two Key Sources of Failures: Processor and Memory

Processor

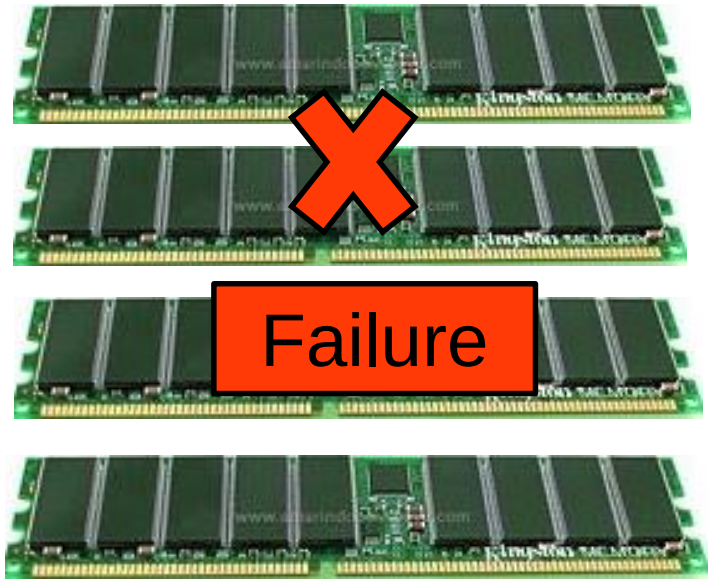



Compute

Read

Write

Memory

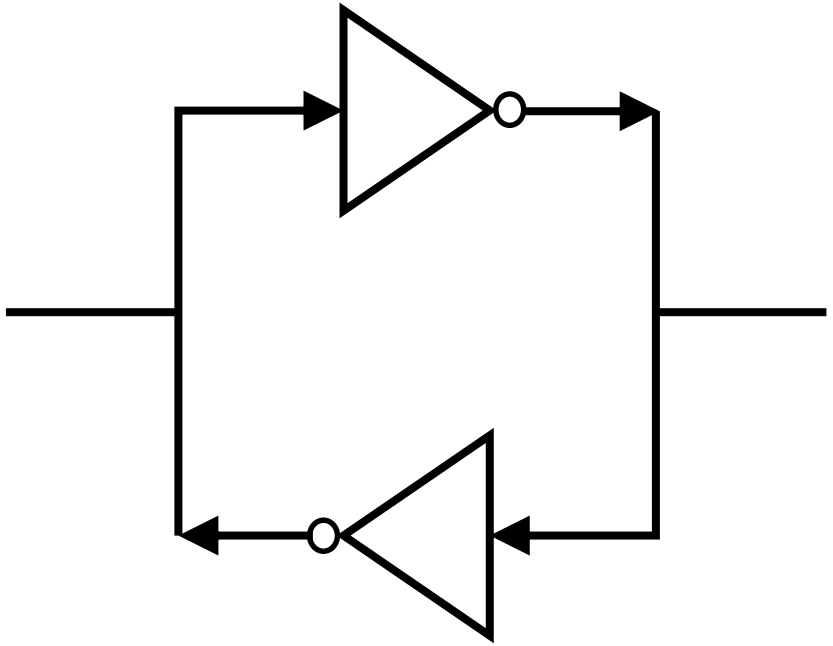



Error Correcting Code
(ECC)

# Processor Failures are a Result of Underlying Faults

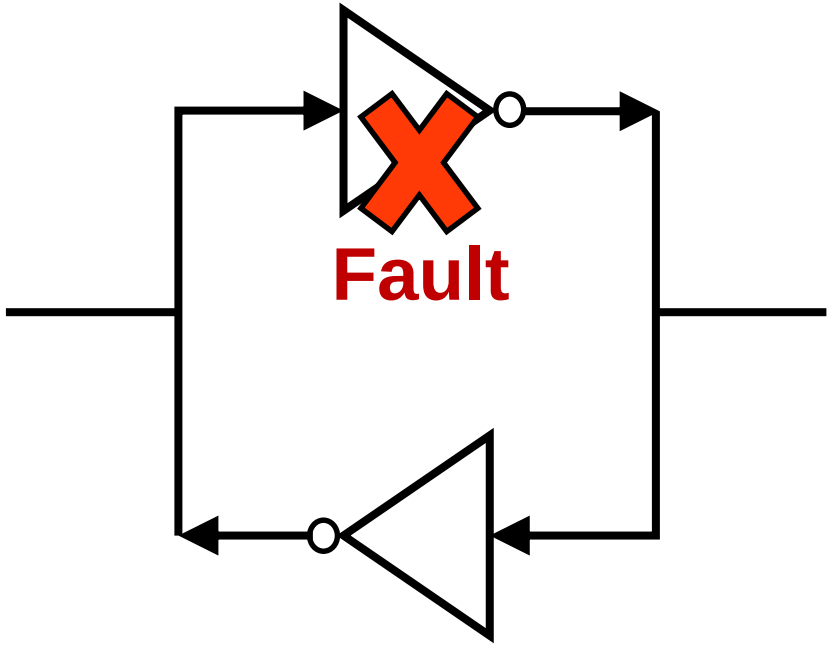
Processor

# Processor Failures are a Result of Underlying Faults



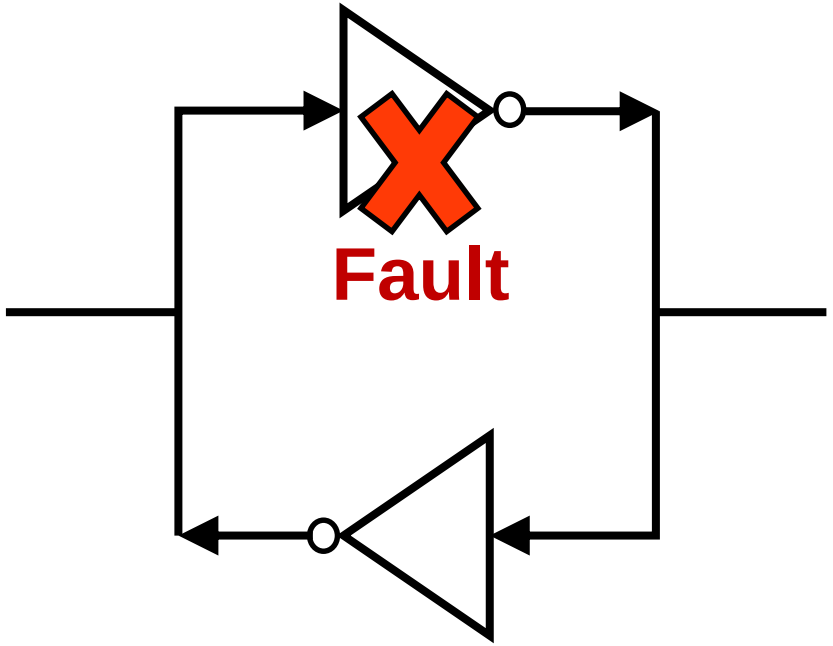
Processor

# Processor Failures are a Result of Underlying Faults




Processor

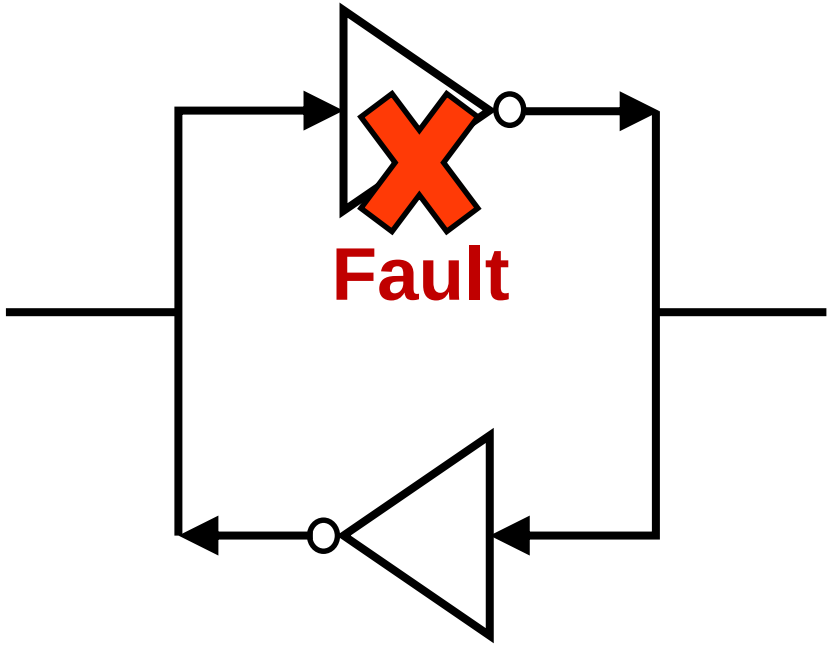
# Processor Failures are a Result of Underlying Faults




Processor


Example: non-functioning or defective transistor

# Processor Failures are a Result of Underlying Faults




Processor


Example: non-functioning or defective transistor

# Three Types of Faults

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed
   -- Wearout, Electromigration

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed
   -- Wearout, Electromigration

2. Intermittent – under specific conditions

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed
   -- Wearout, Electromigration

2. Intermittent – under specific conditions
   -- Elevated temperature

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed
   -- Wearout, Electromigration

2. Intermittent – under specific conditions
   -- Elevated temperature

3. Transient – random, temporary, non-reproducible

# Three Types of Faults

1. Permanent – to be fixed at hardware level or cannot be fixed
   -- Wearout, Electromigration

2. Intermittent – under specific conditions
   -- Elevated temperature

3. Transient – random, temporary, non-reproducible
   -- Energy particle strikes or radiations

# What are Radiation-Induced Faults?

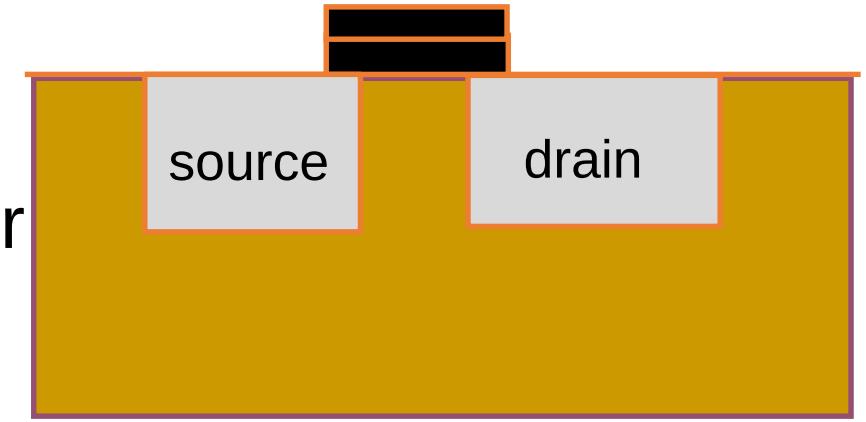
# What are Radiation-Induced Faults?

- Main source: neutrons from deep space

# What are Radiation-Induced Faults?

- Main source: neutrons from deep space

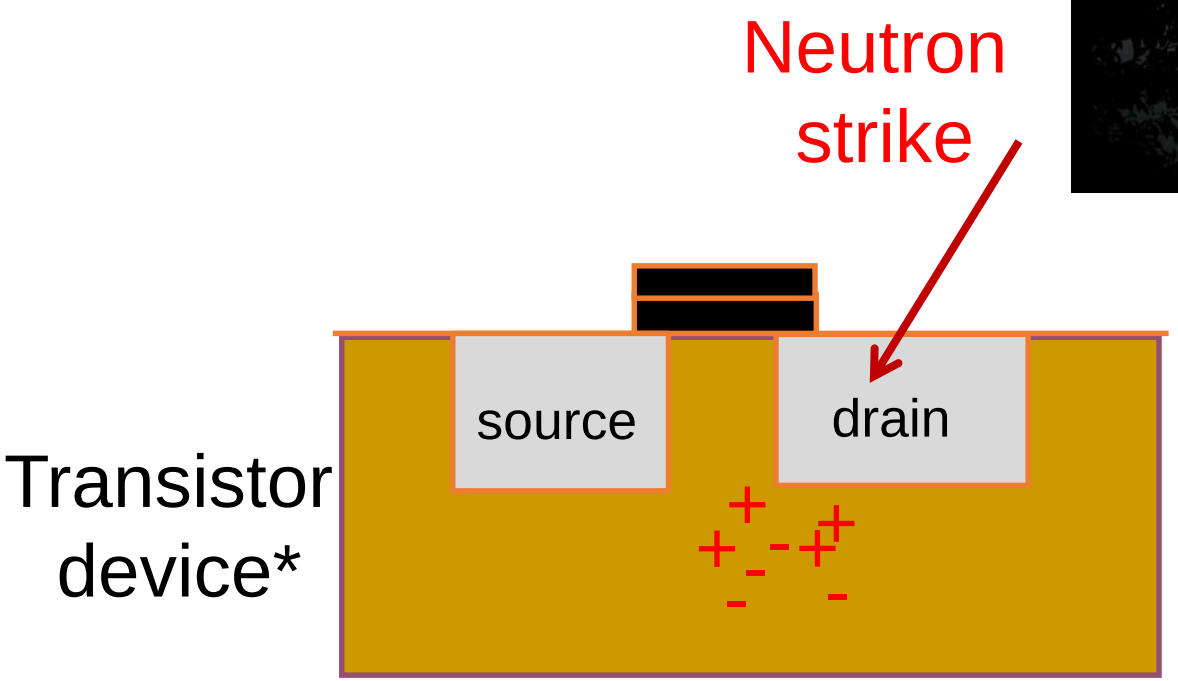
Transistor device*




*[Soft Error Problem, Shubhu Mukherjee, HPCA2005]

# What are Radiation-Induced Faults?

- Main source: neutrons from deep space



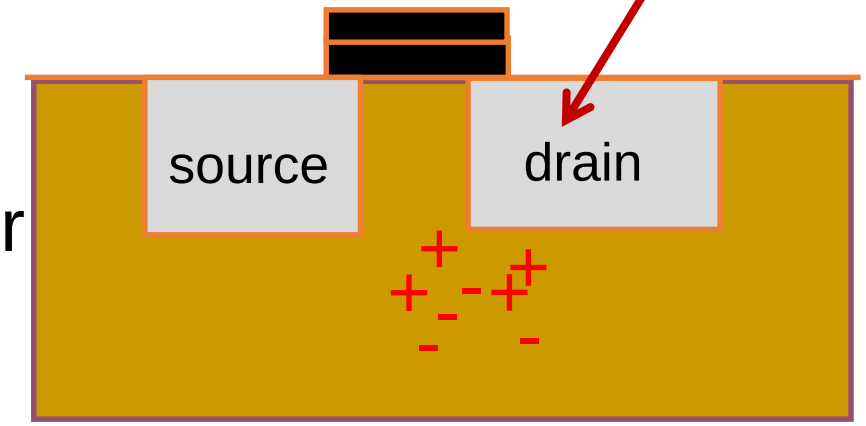

*[Soft Error Problem, Shubhu Mukherjee, HPCA2005]

# What are Radiation-Induced Faults?

- Main source: neutrons from deep space

Neutron strike

Transistor device*




**Temporarily changes device state**

*[Soft Error Problem, Shubhu Mukherjee, HPCA2005]

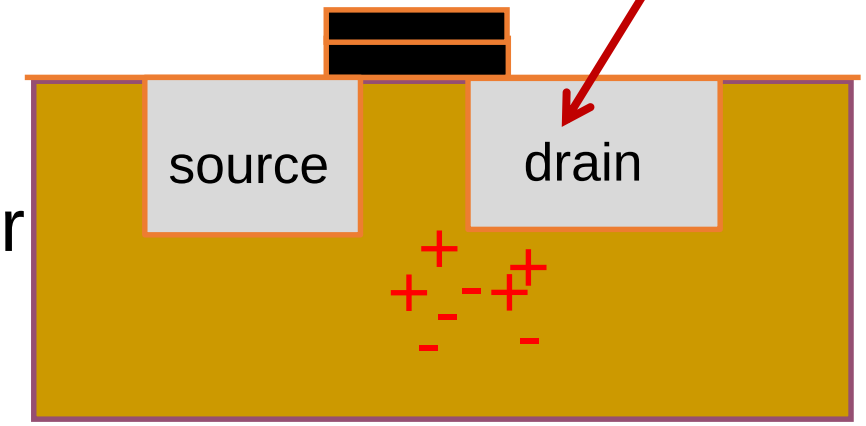
# What are Radiation-Induced Faults?

- Main source: neutrons from deep space

- Secondary source: alpha particles from packaging material

Neutron strike

Transistor device*




**Temporarily changes device state**

*[Soft Error Problem, Shubhu Mukherjee, HPCA2005]

# Faults → Errors → Failures

# Faults → Errors → Failures

Fault

# Faults → Errors → Failures

Fault

↓

Error

# Faults → Errors → Failures

Fault

↓

Error

↓

Failure

# Faults → Errors → Failures

Fault

Radiations

↓

Error

↓

Failure

# Faults → Errors → Failures

Fault

Radiations

Error

Soft error

Failure

# Faults → Errors → Failures

Fault

Radiations

↓

↓

Error

Soft error

↓

↓

Failure

Failure

# Faults → Errors → Failures

Fault

↓

Error

↓

Failure

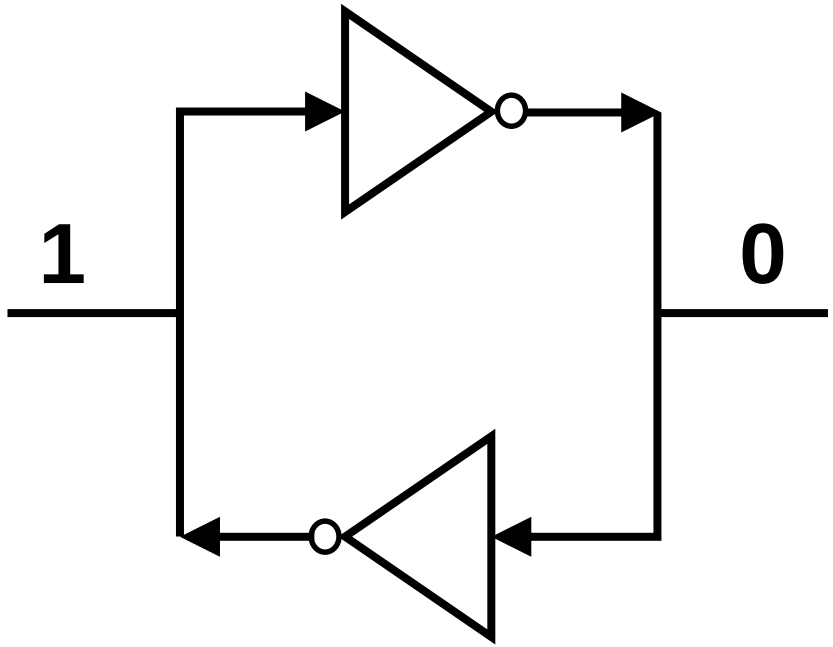
Radiations

↓

Soft error

↓

Failure



# Faults → Errors → Failures

Fault

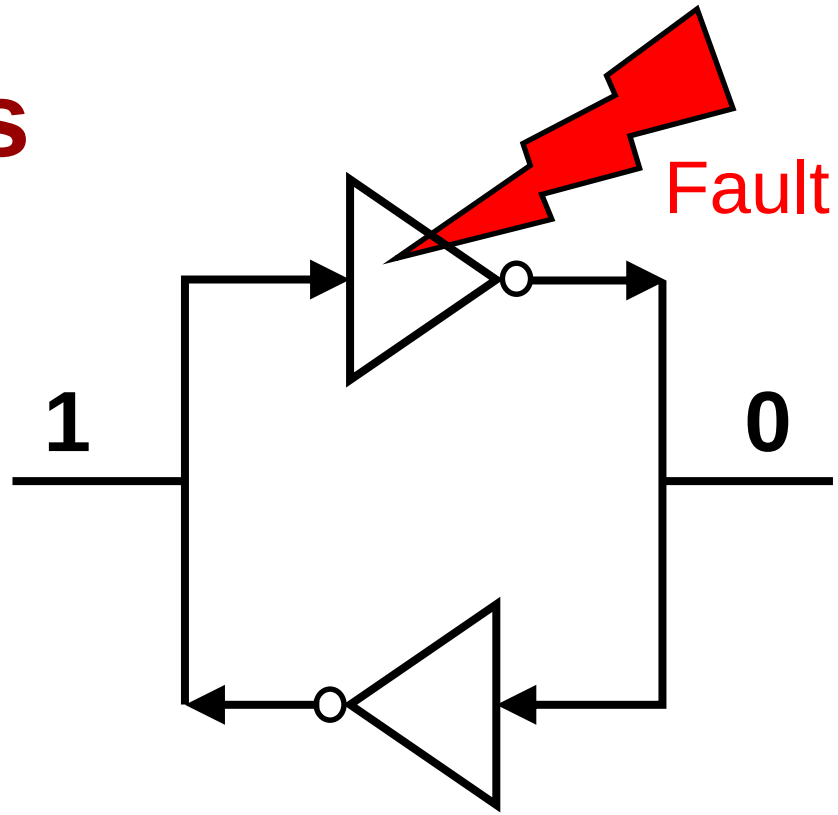
Error

Failure

Radiations

Soft error

Failure



# Faults → Errors → Failures

Fault

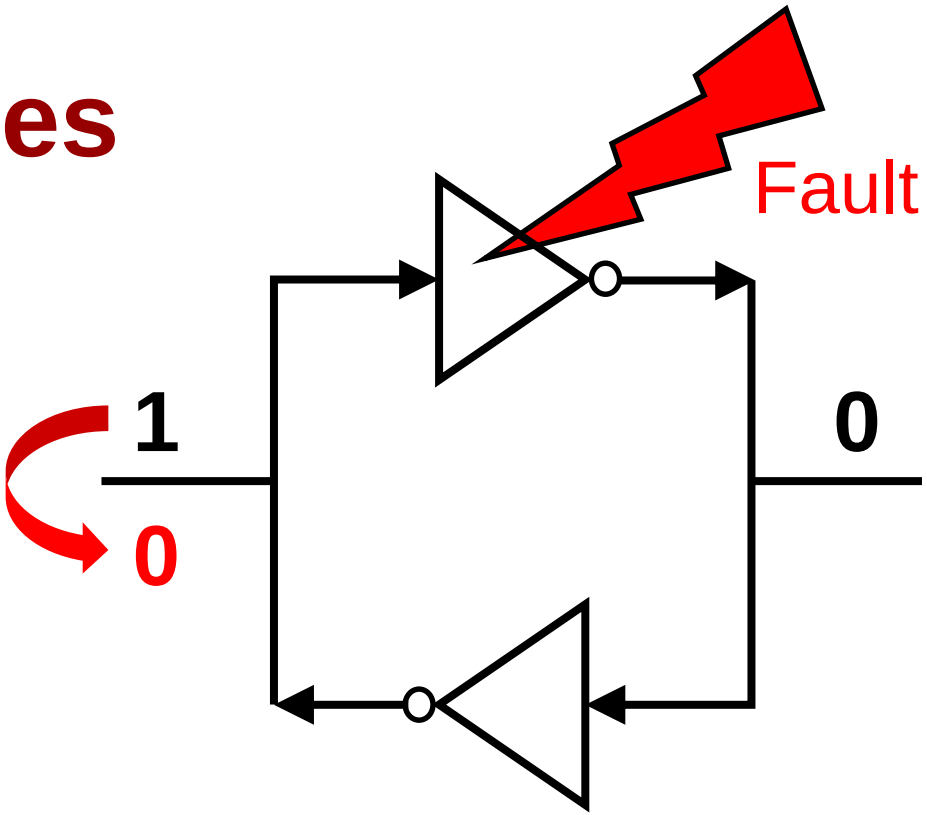
Error

Failure

Radiations

Soft error

Failure



# Faults → Errors → Failures

Fault

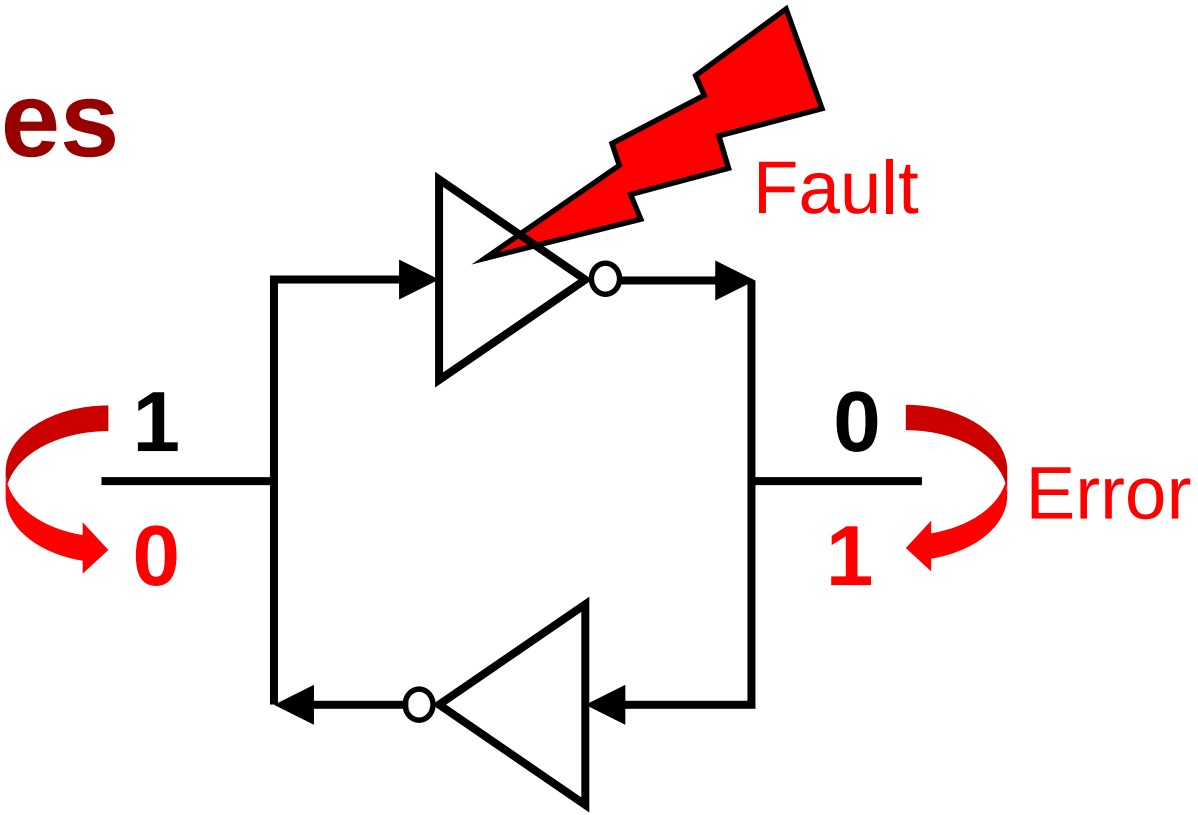
Error

Failure

Radiations

Soft error

Failure



# Faults → Errors → Failures

Fault

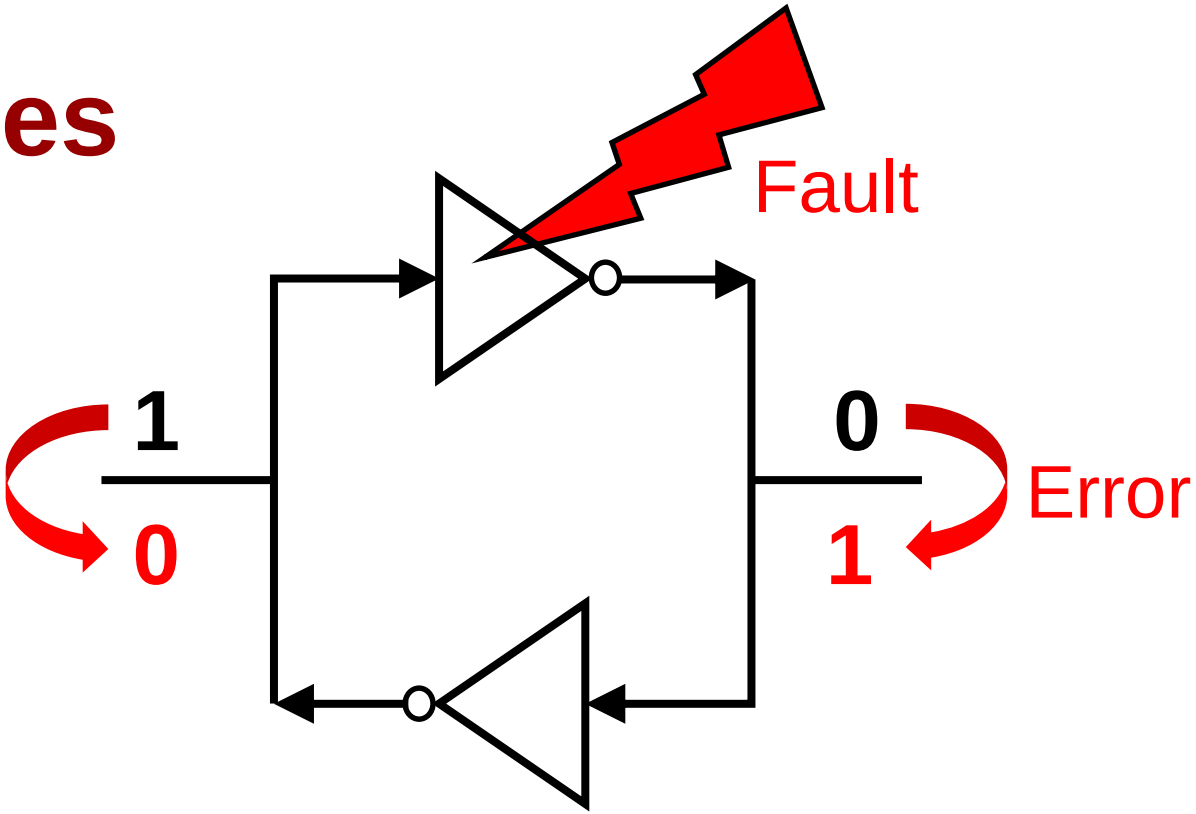
Error

Failure

Radiations

Soft error

Failure




# Why have Soft Errors Become a Critical Issue?

# Why have Soft Errors Become a Critical Issue?

High performance

# Why have Soft Errors Become a Critical Issue?

High performance

– 1988 → 2017 more than 1000 X increase

# Why have Soft Errors Become a Critical Issue?

High performance

– 1988 → 2017 more than 1000 X increase

Two key contributors

# Why have Soft Errors Become a Critical Issue?

High performance

– 1988 → 2017 more than 1000 X increase

Two key contributors

1. **Technology scaling** – smaller transistors → better performance

# Why have Soft Errors Become a Critical Issue?

High performance

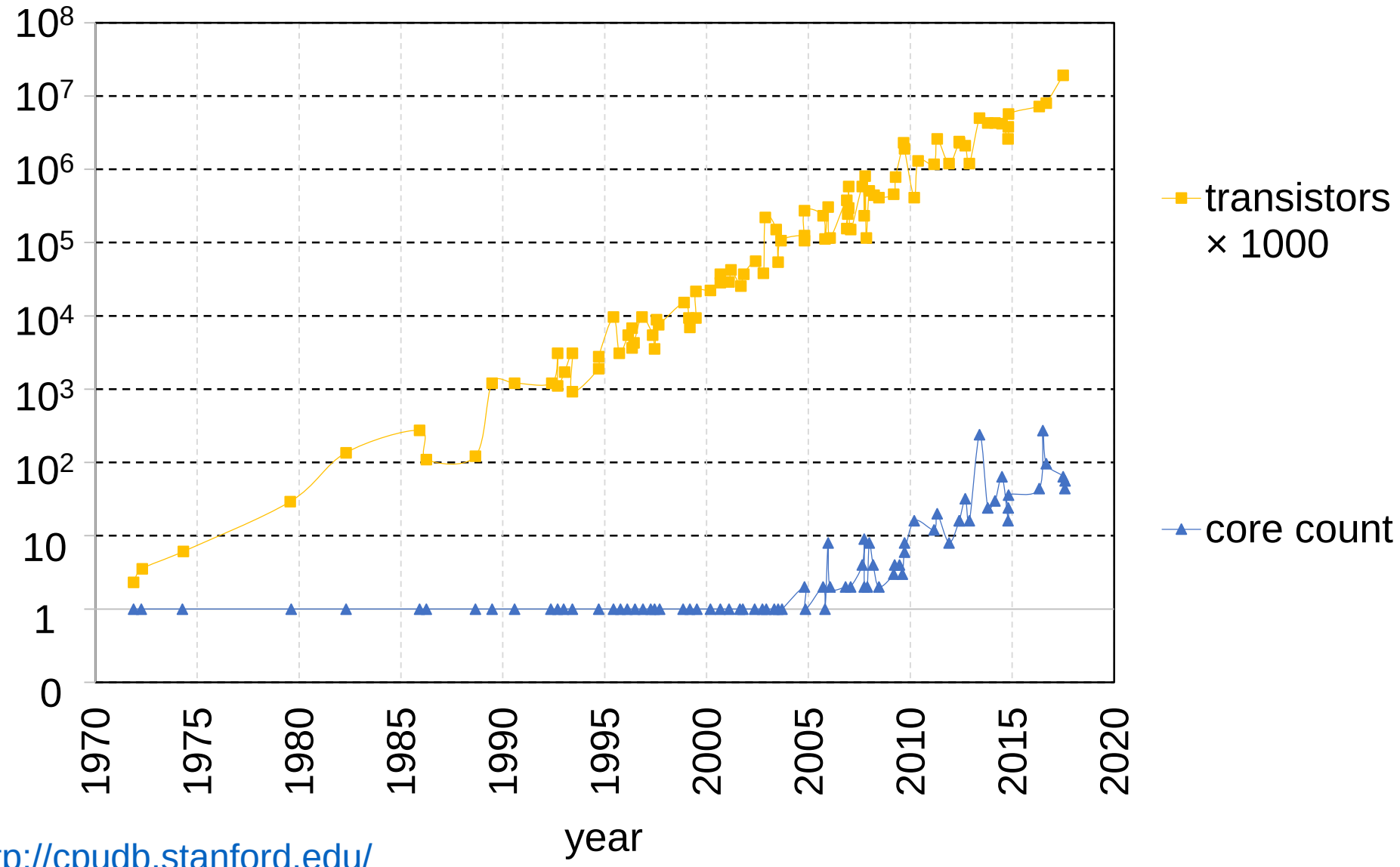
– 1988 → 2017 more than 1000 X increase

Two key contributors

1. **Technology scaling** – smaller transistors → better performance

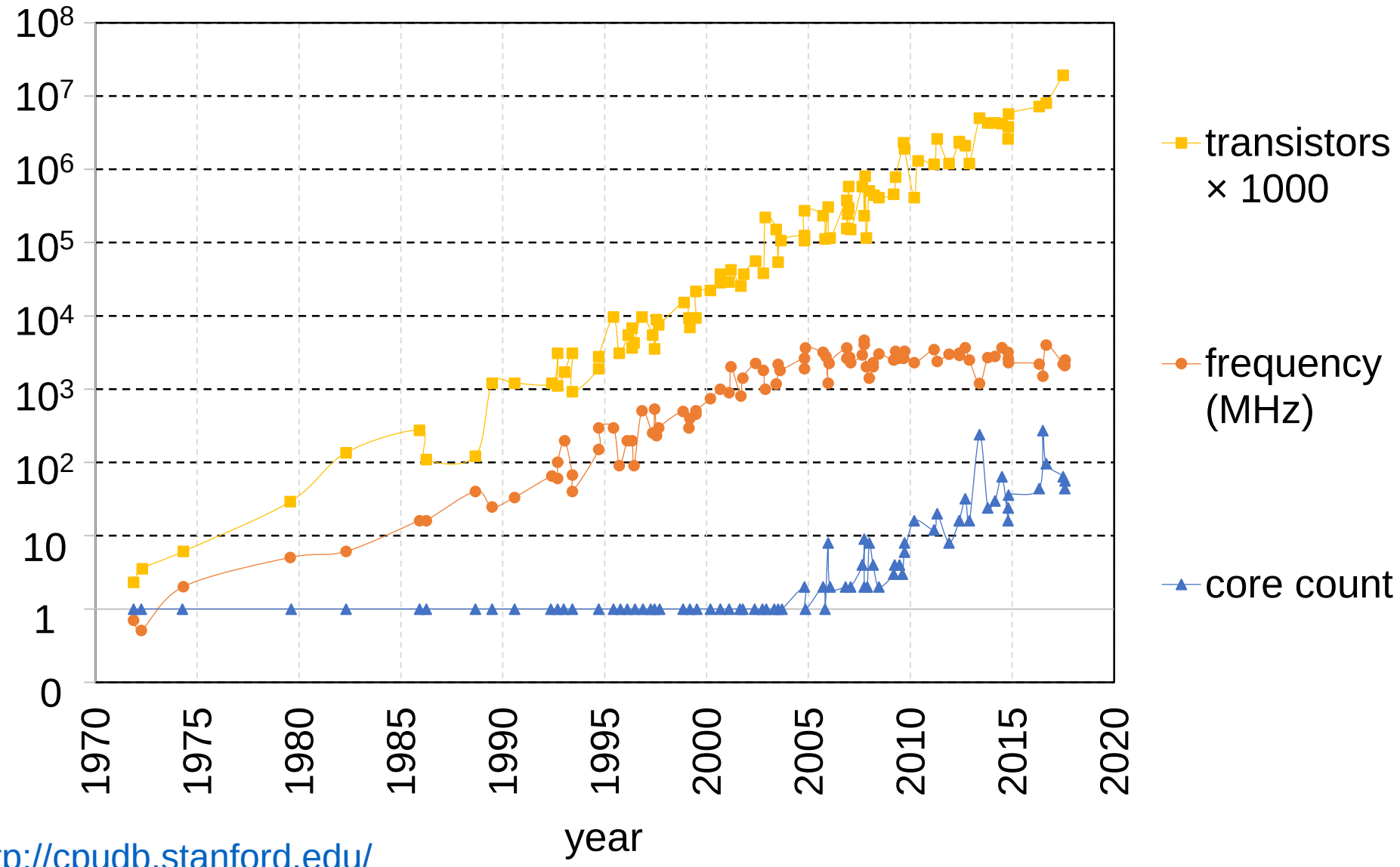
2. **Microarchitectural innovations** – pipelining, out-of-order execution, prefetching, branch prediction

# Consequences of Technology Scaling



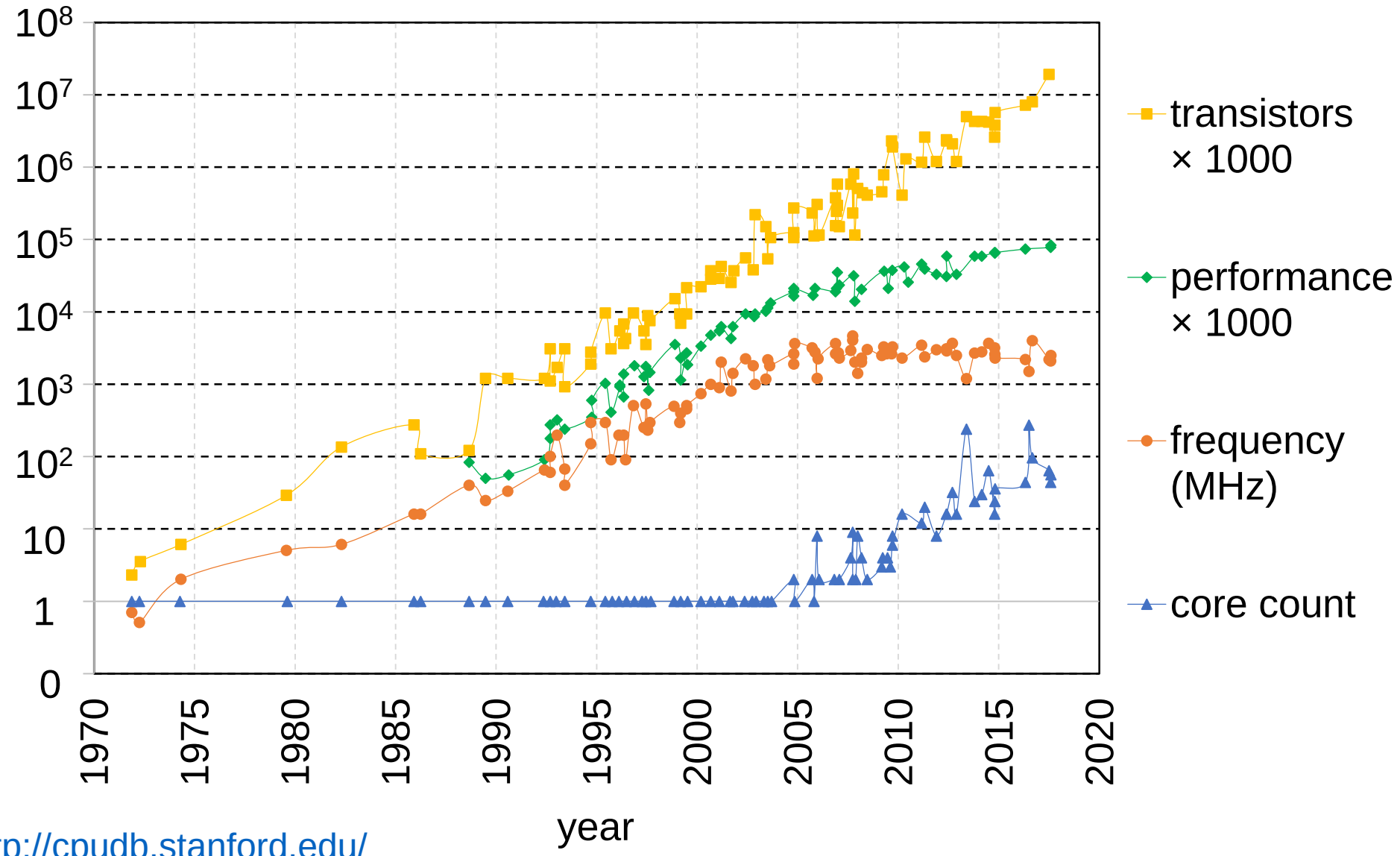

http://cpudb.stanford.edu/

# Consequences of Technology Scaling



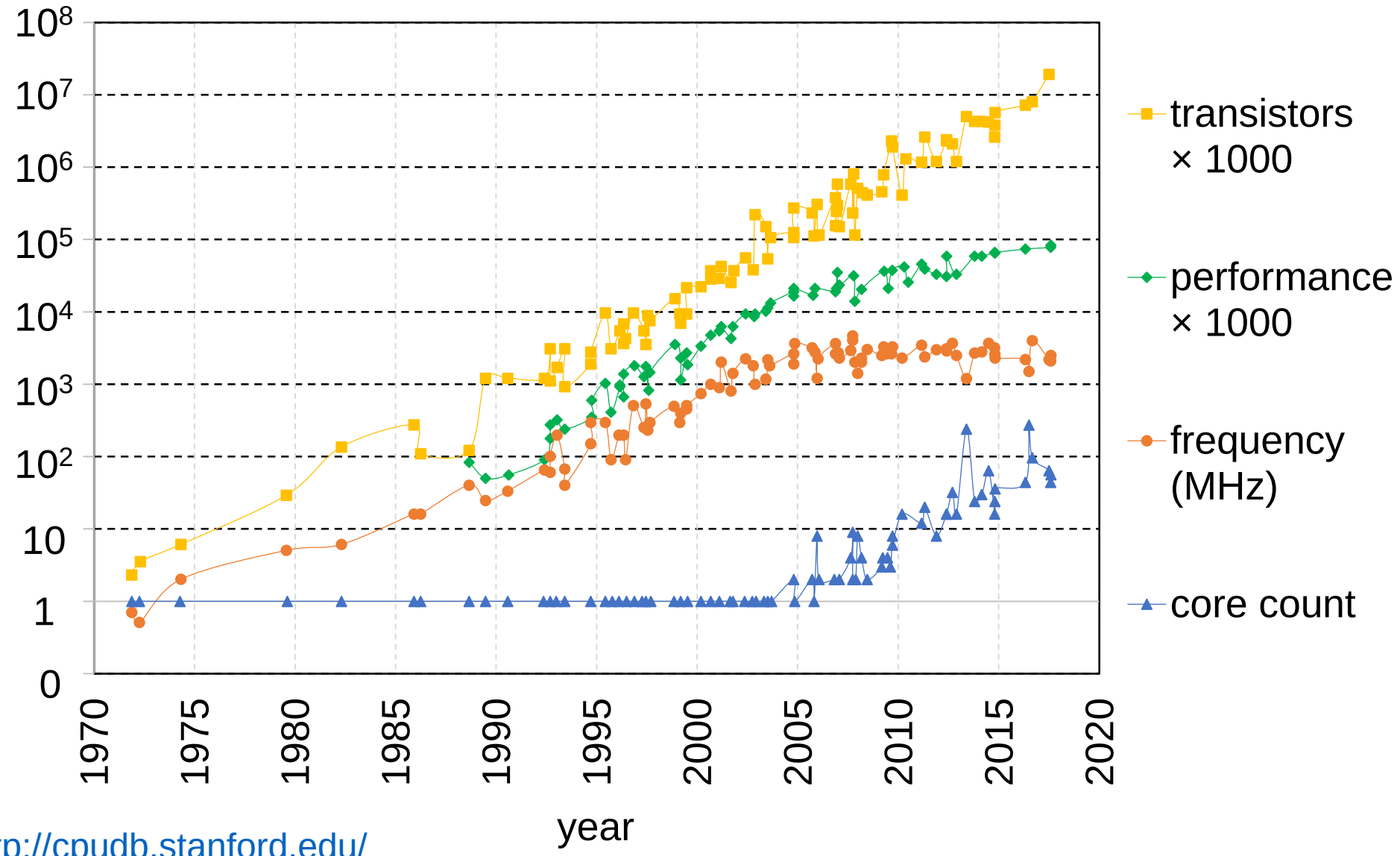

http://cpudb.stanford.edu/

# Consequences of Technology Scaling



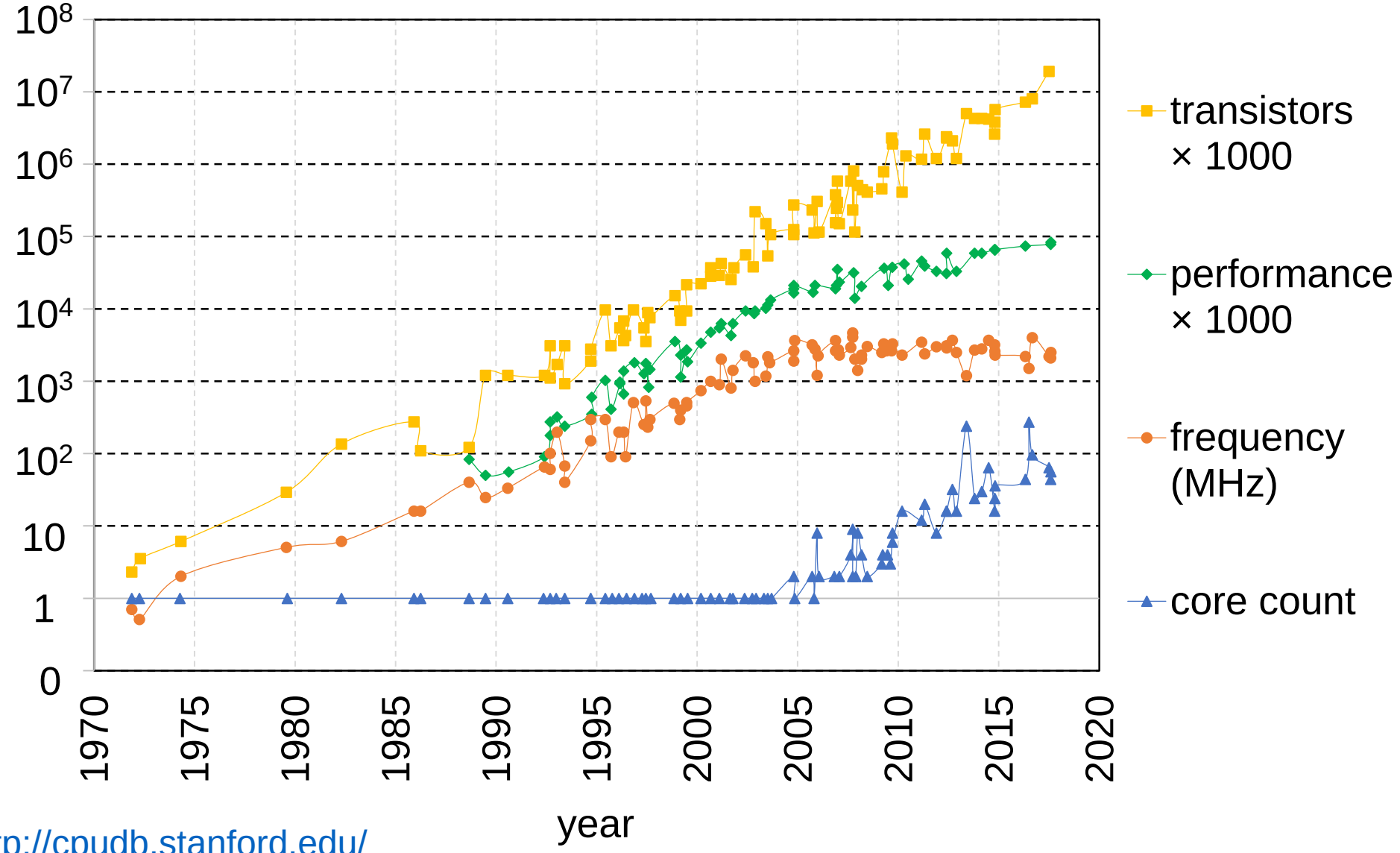

http://cpudb.stanford.edu/

# Consequences of Technology Scaling




http://cpudb.stanford.edu/

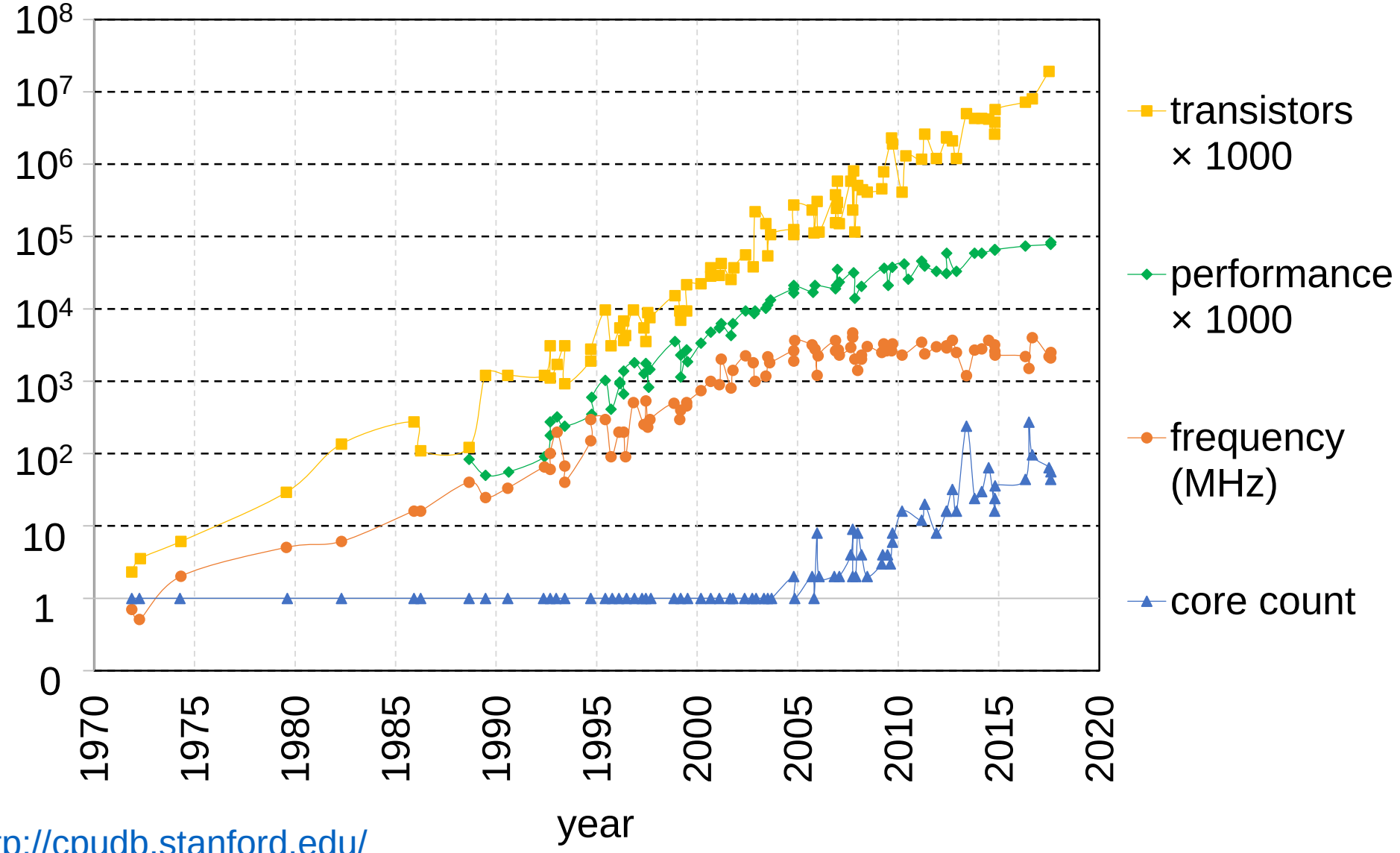
# Consequences of Technology Scaling




Size ↓ K

http://cpudb.stanford.edu/

# Consequences of Technology Scaling

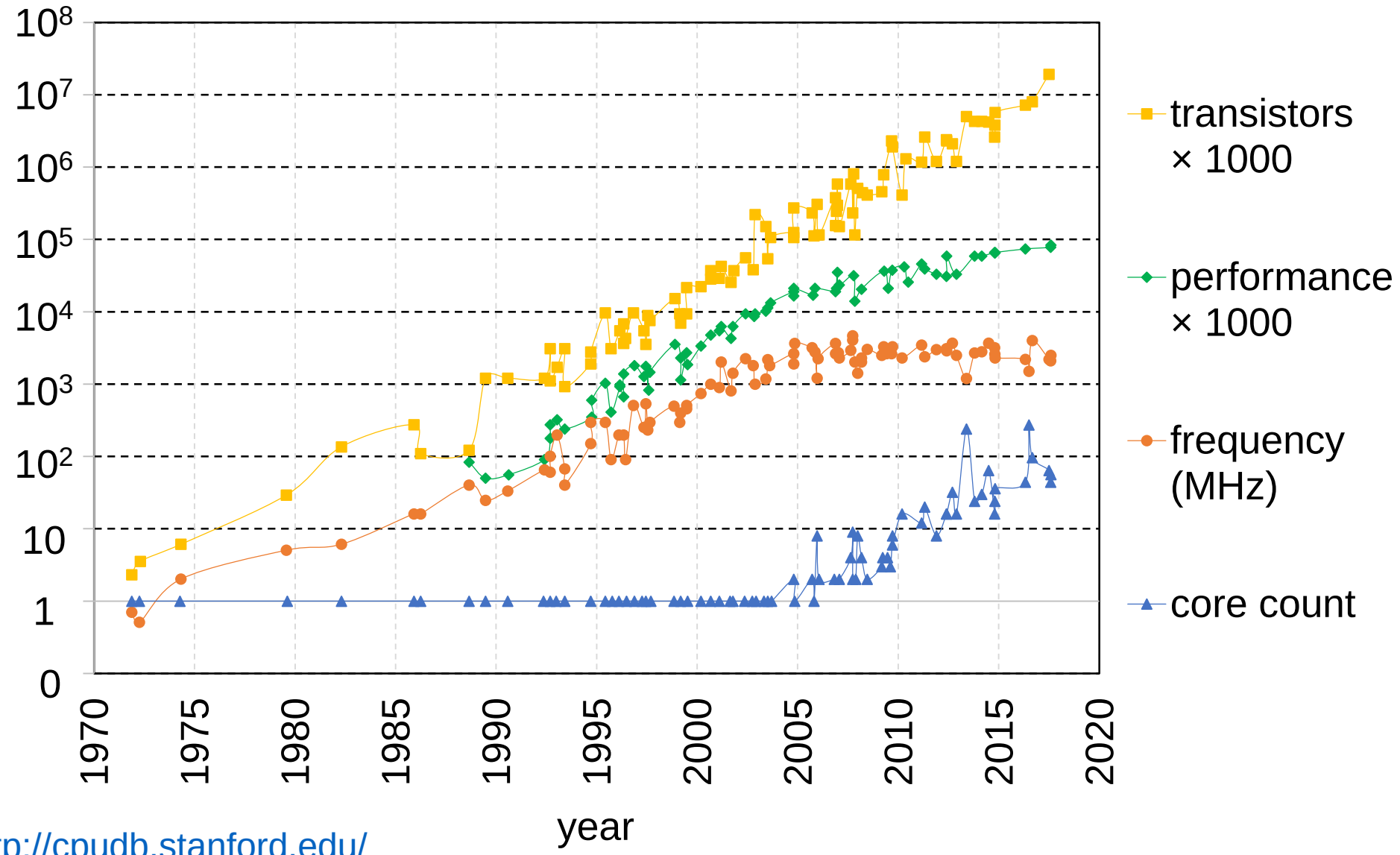



Size ↓ K

Circuit Delay ↓ K

http://cpudb.stanford.edu/

# Consequences of Technology Scaling



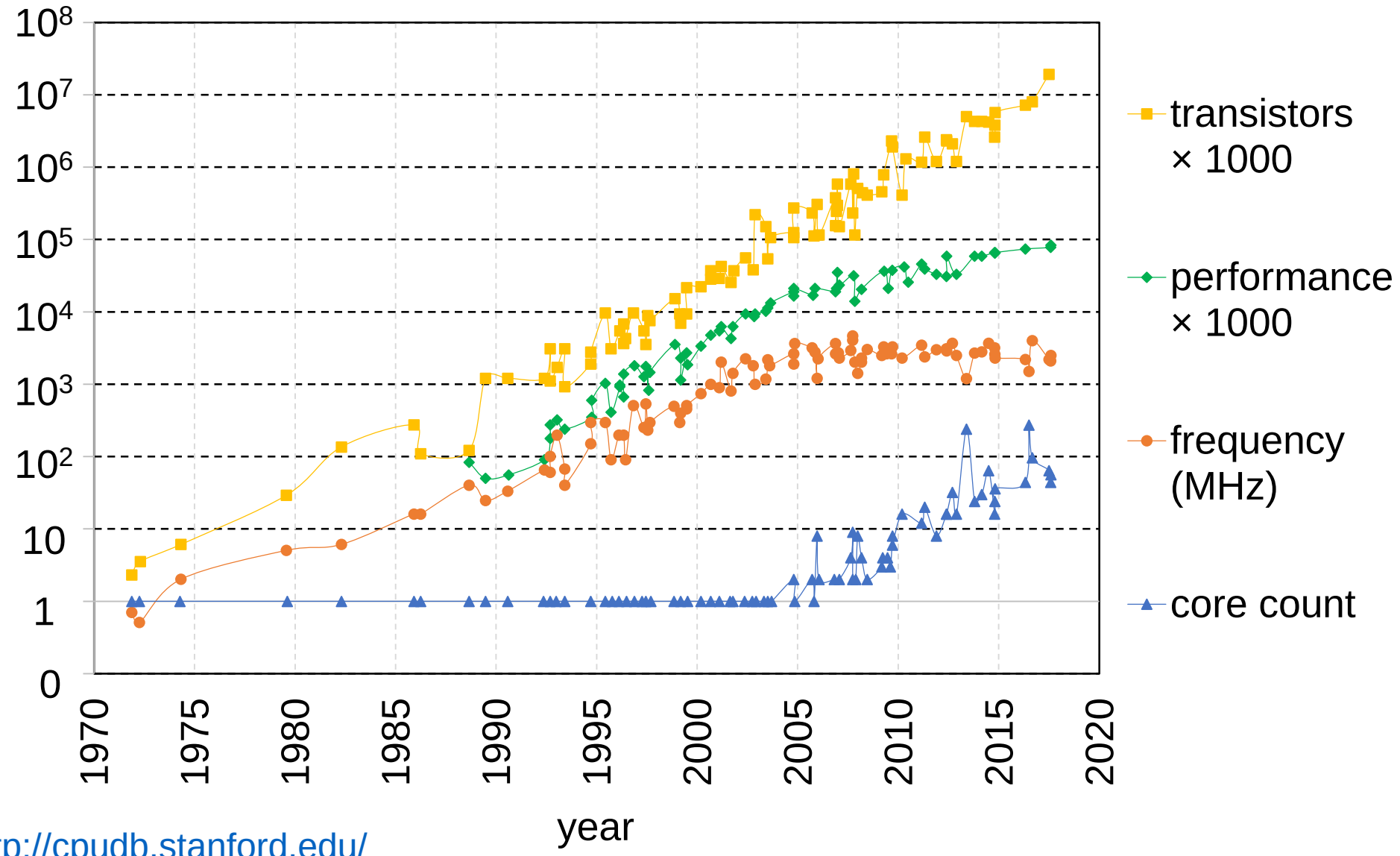

Size ↓ K

Circuit Delay ↓ K

Switching Charge ↓ K

http://cpudb.stanford.edu/

# Consequences of Technology Scaling




Size ↓ K

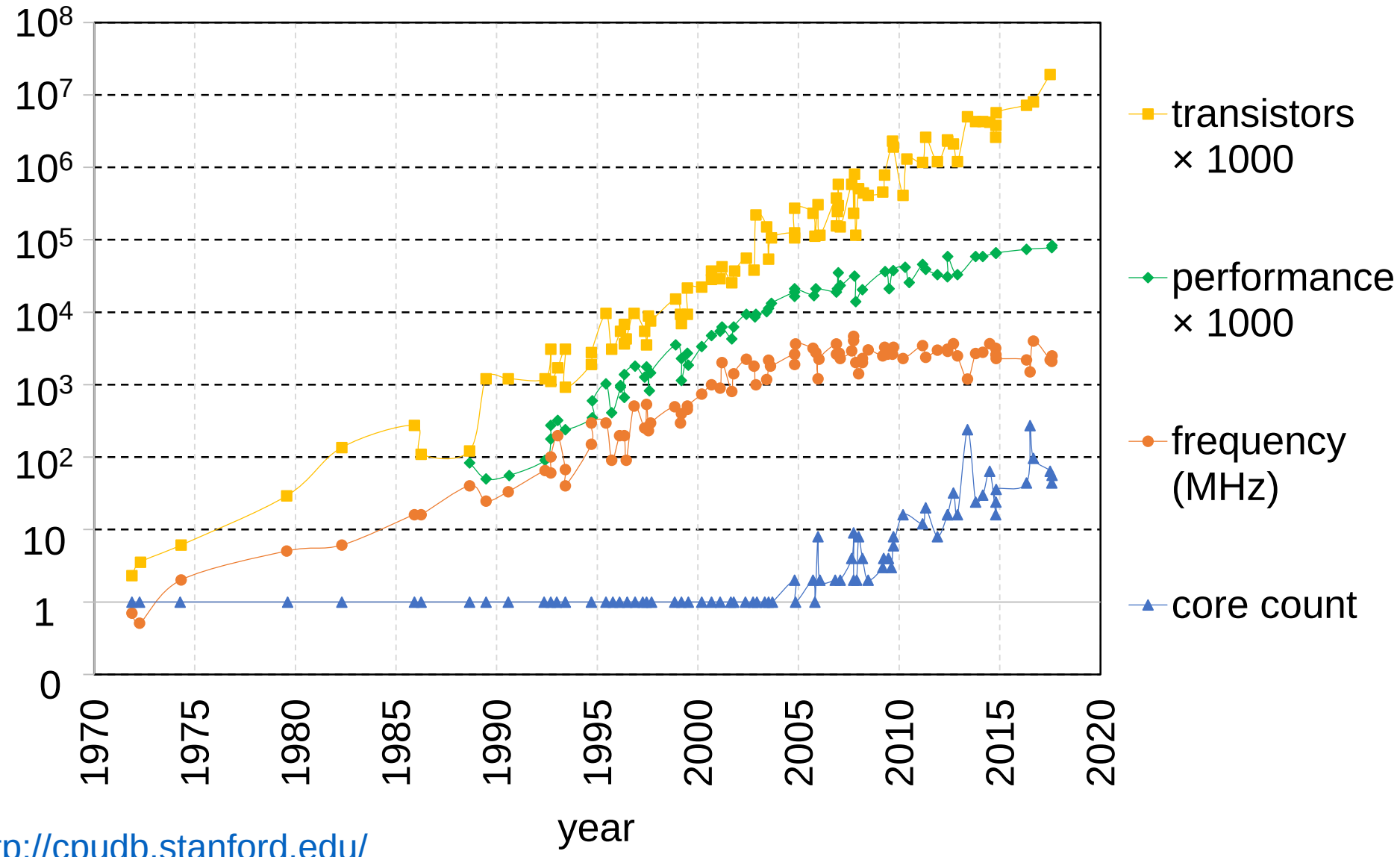
Circuit Delay ↓ K

Switching Charge ↓ K

Voltage ↓ K

http://cpudb.stanford.edu/

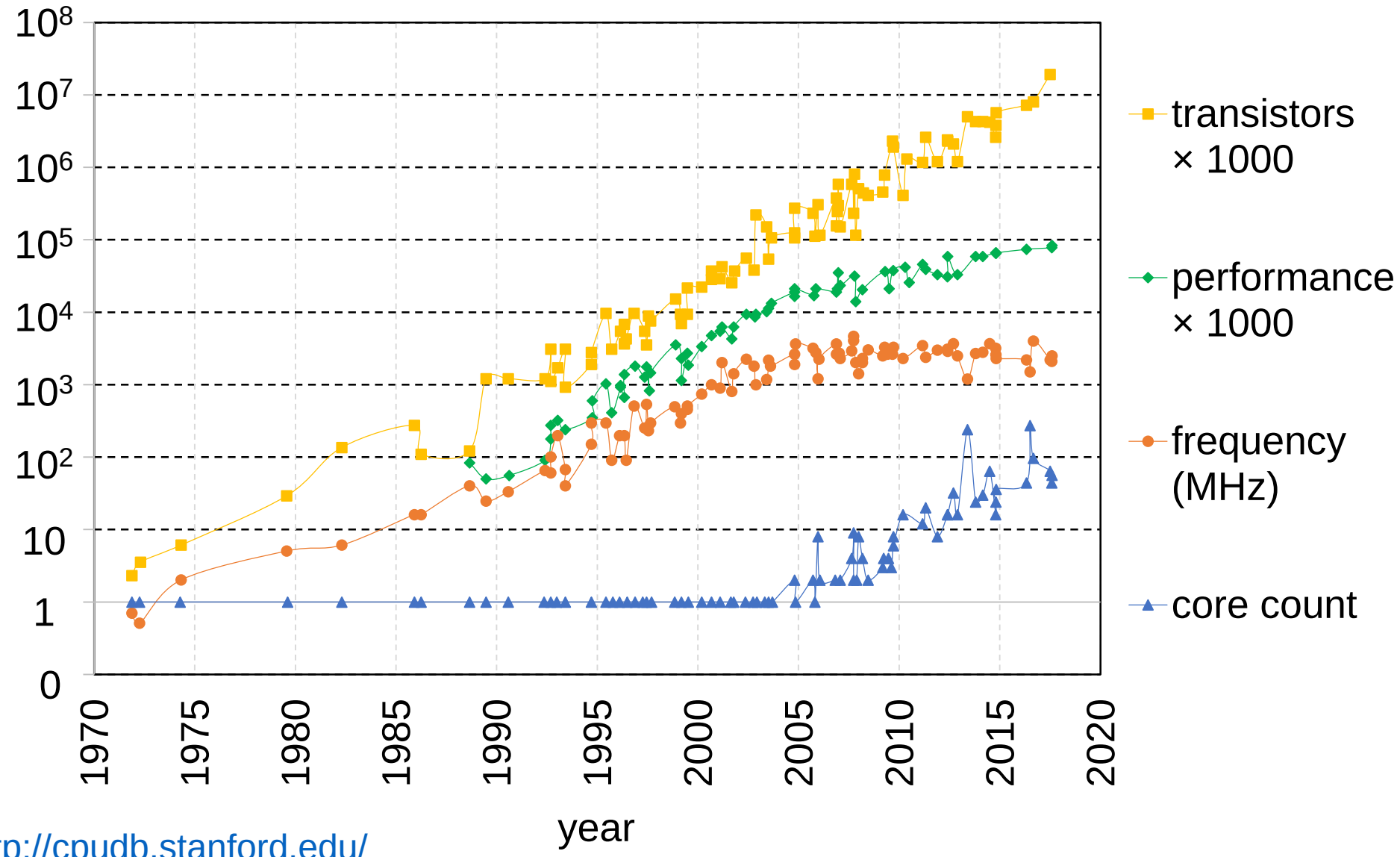
# Consequences of Technology Scaling




| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |

http://cpudb.stanford.edu/

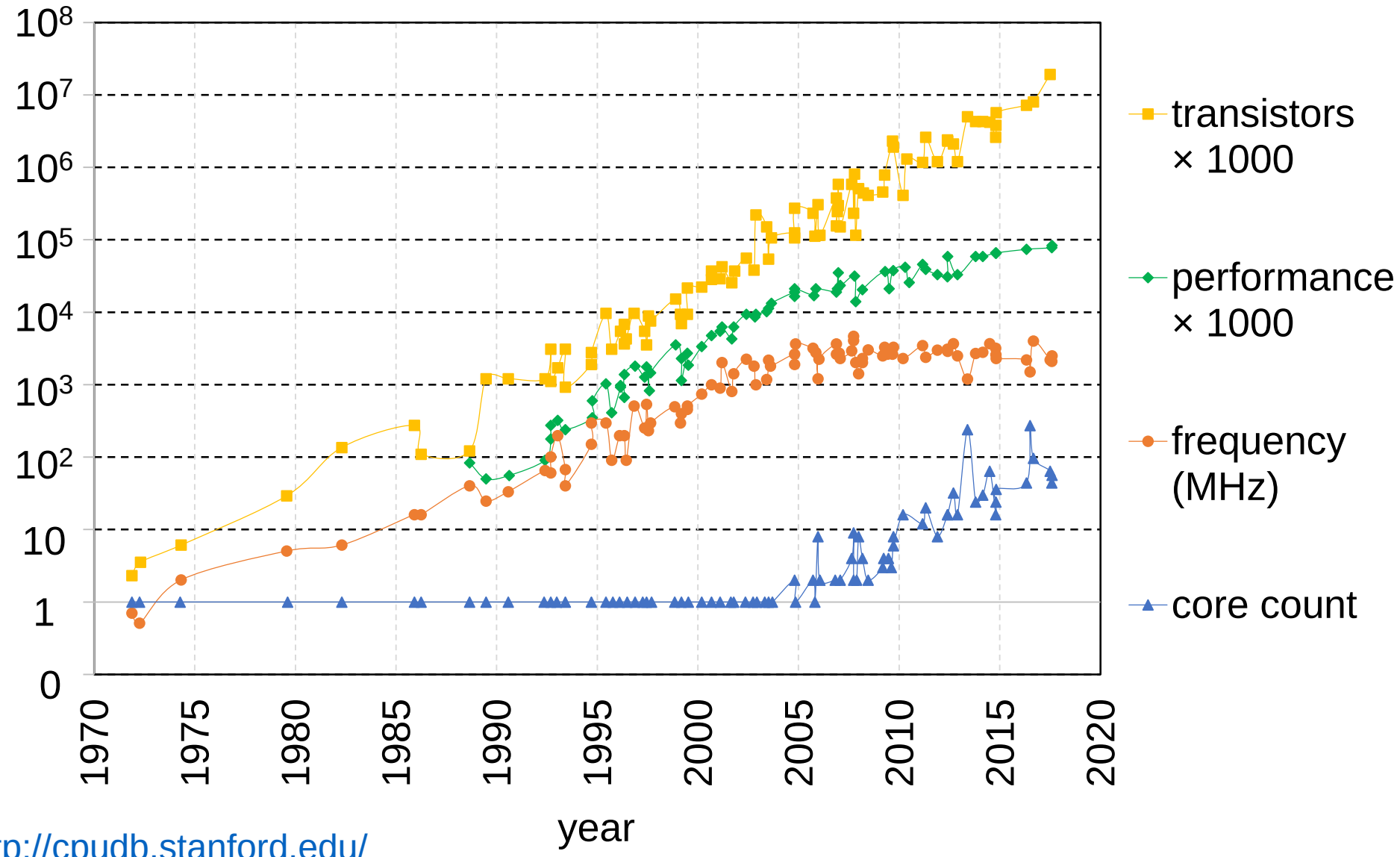
# Consequences of Technology Scaling




| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |

http://cpudb.stanford.edu/

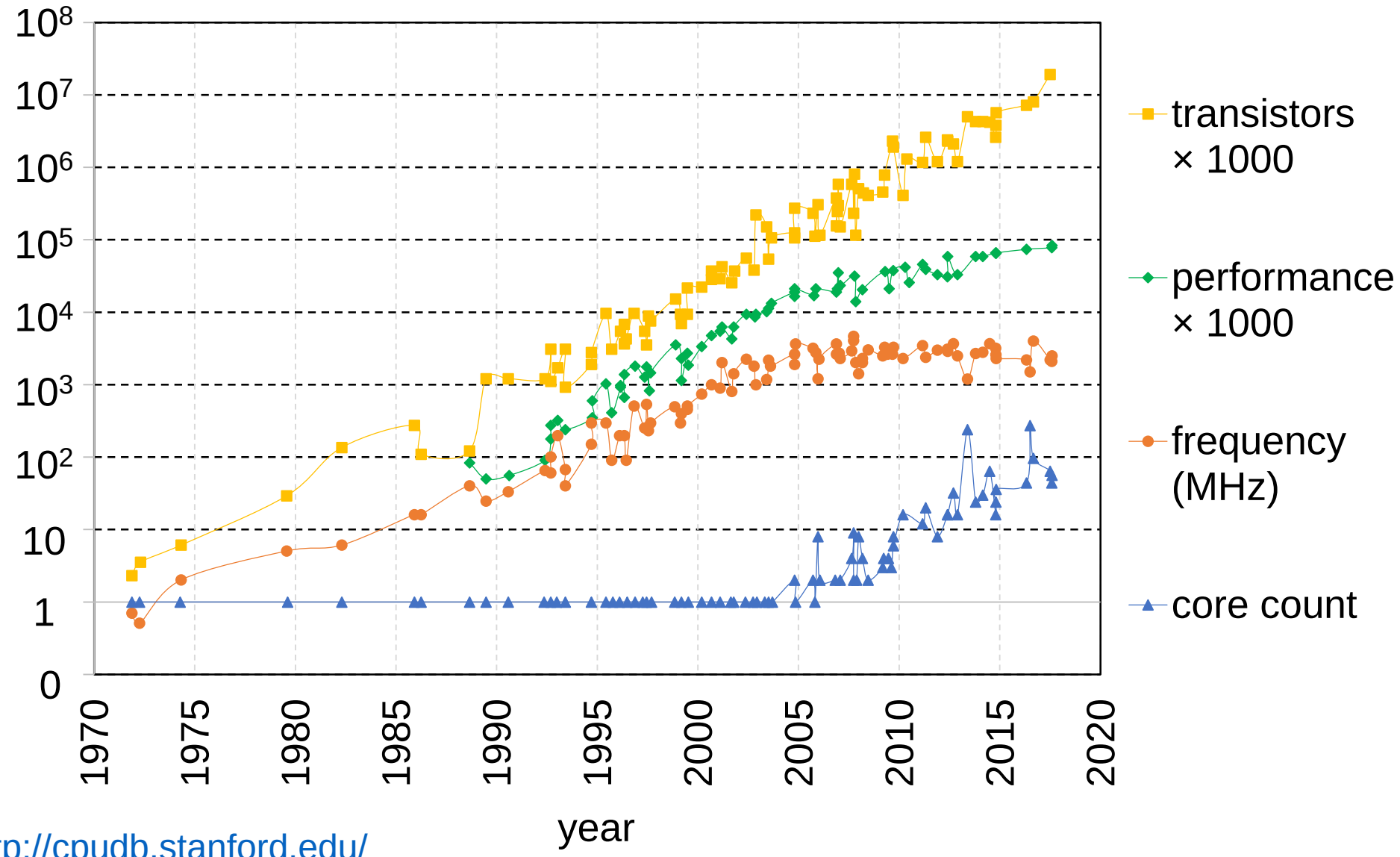
# Consequences of Technology Scaling




| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | |

http://cpudb.stanford.edu/

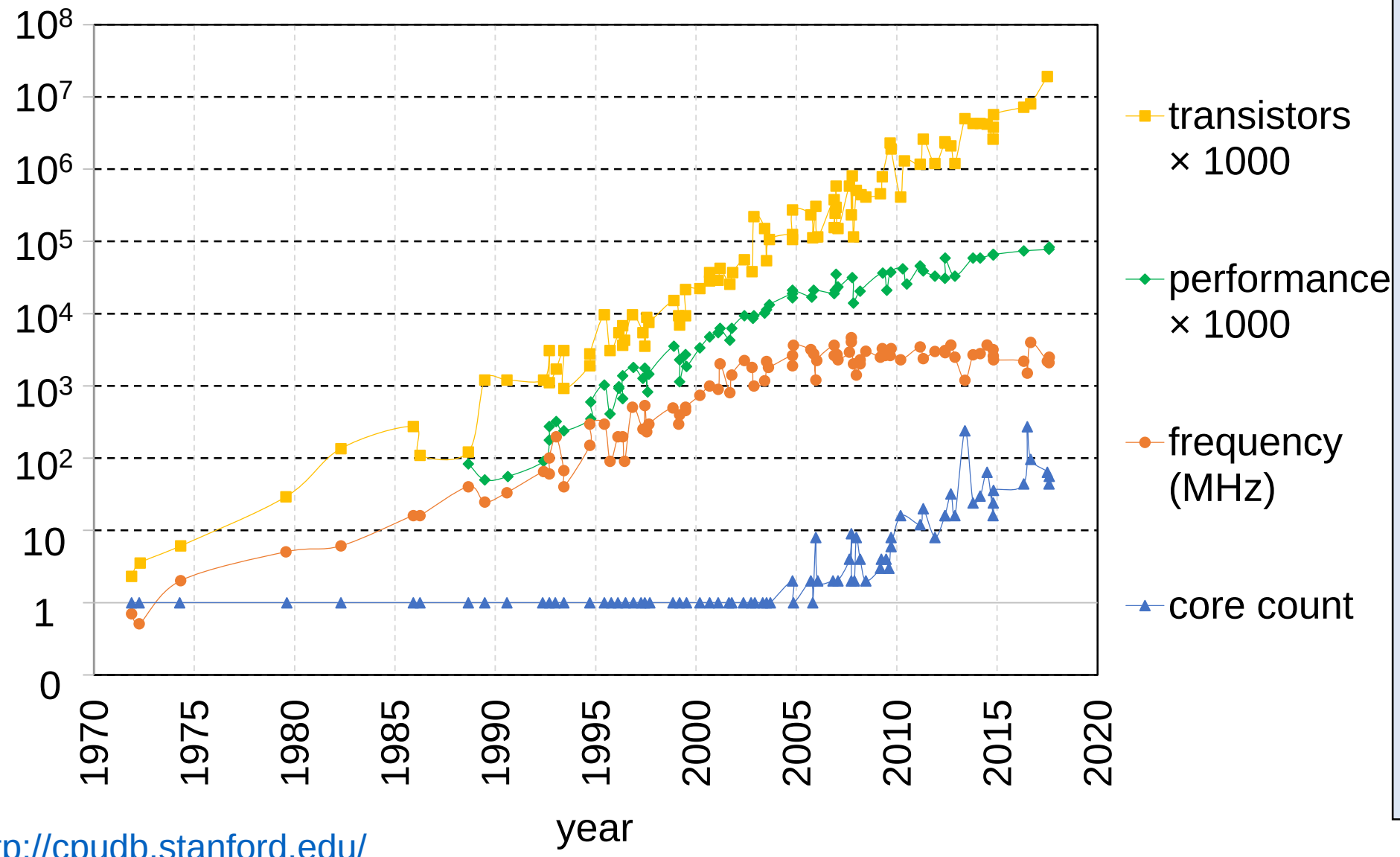
# Consequences of Technology Scaling




| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

http://cpudb.stanford.edu/

# Consequences of Technology Scaling

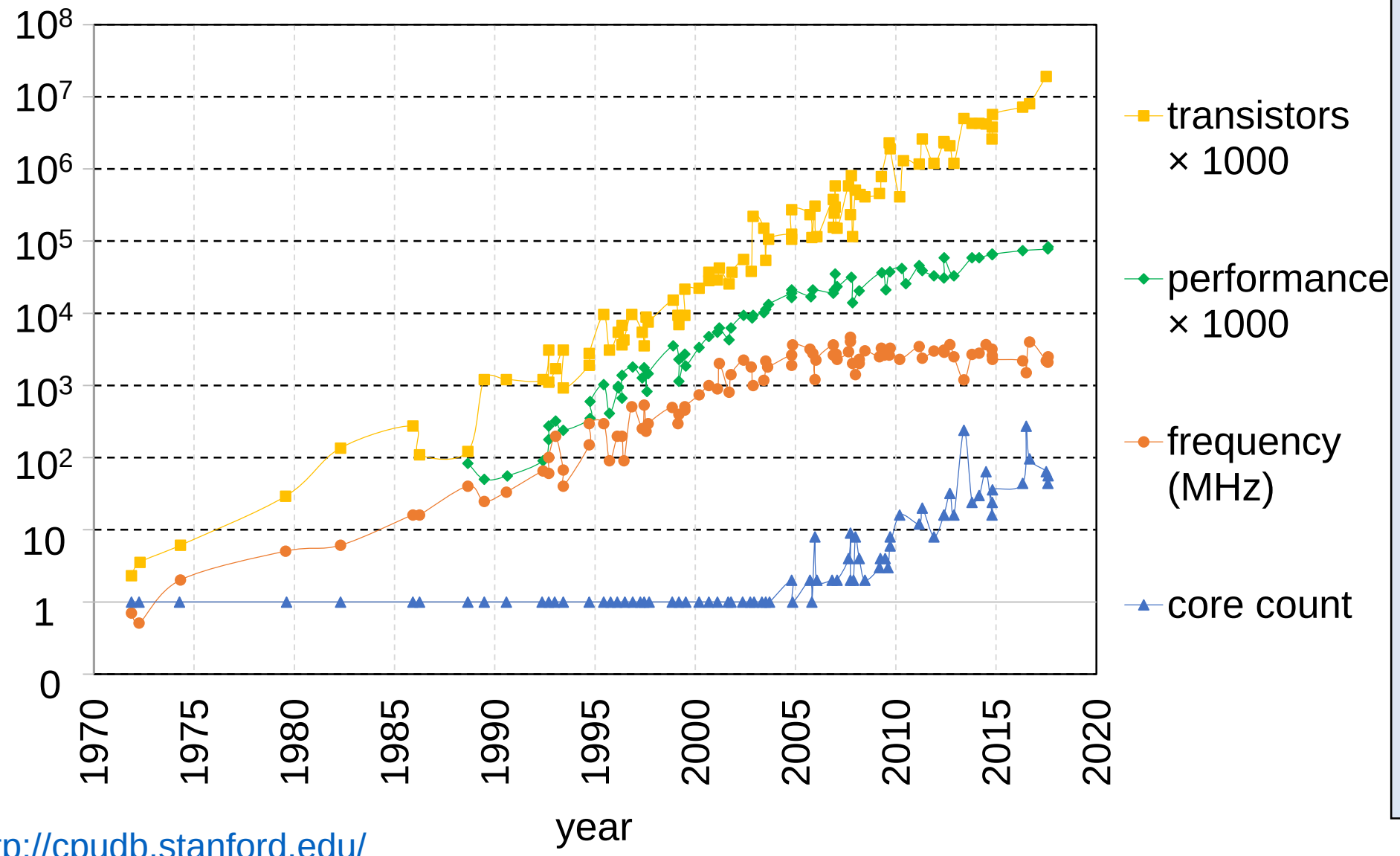



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

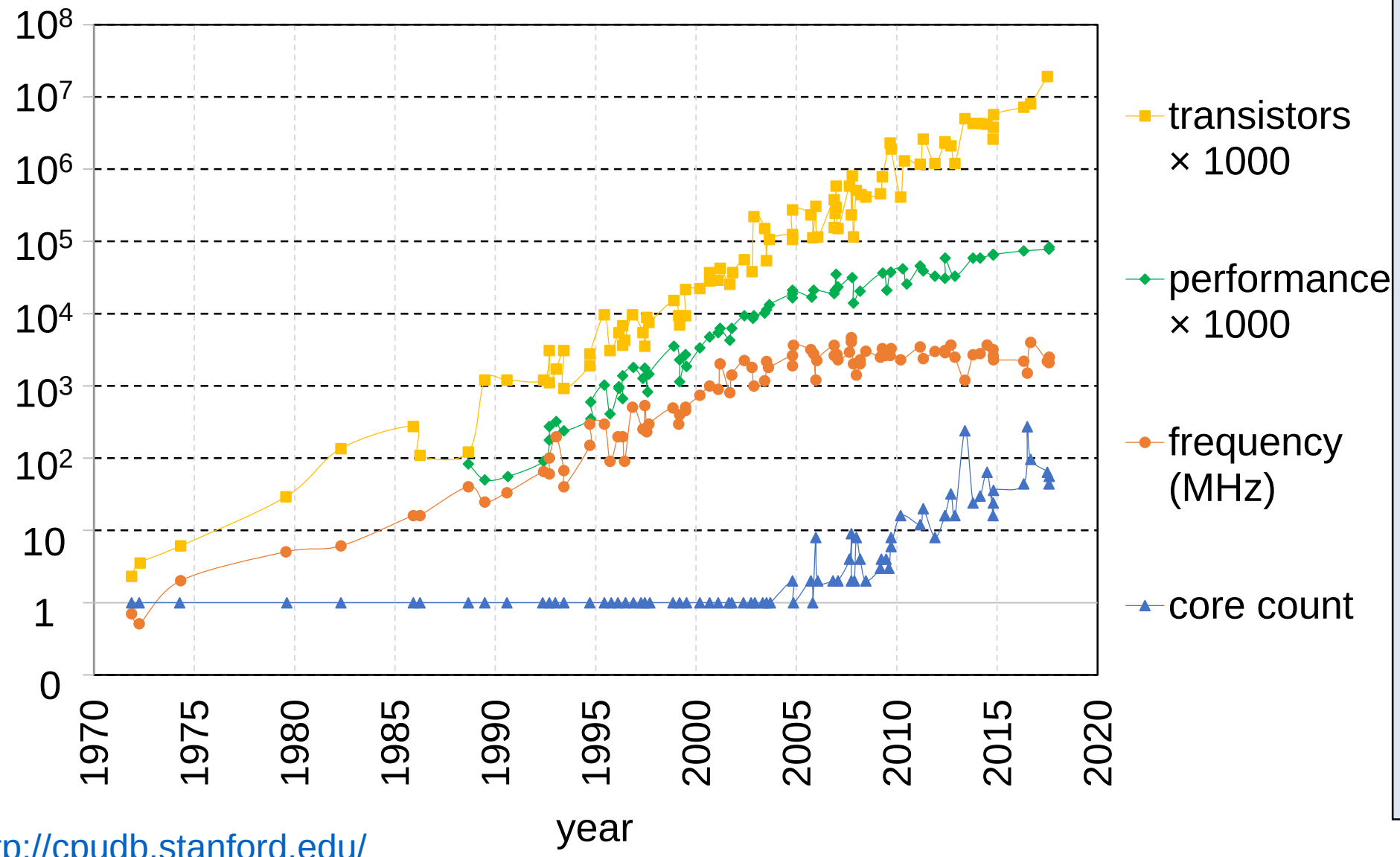



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

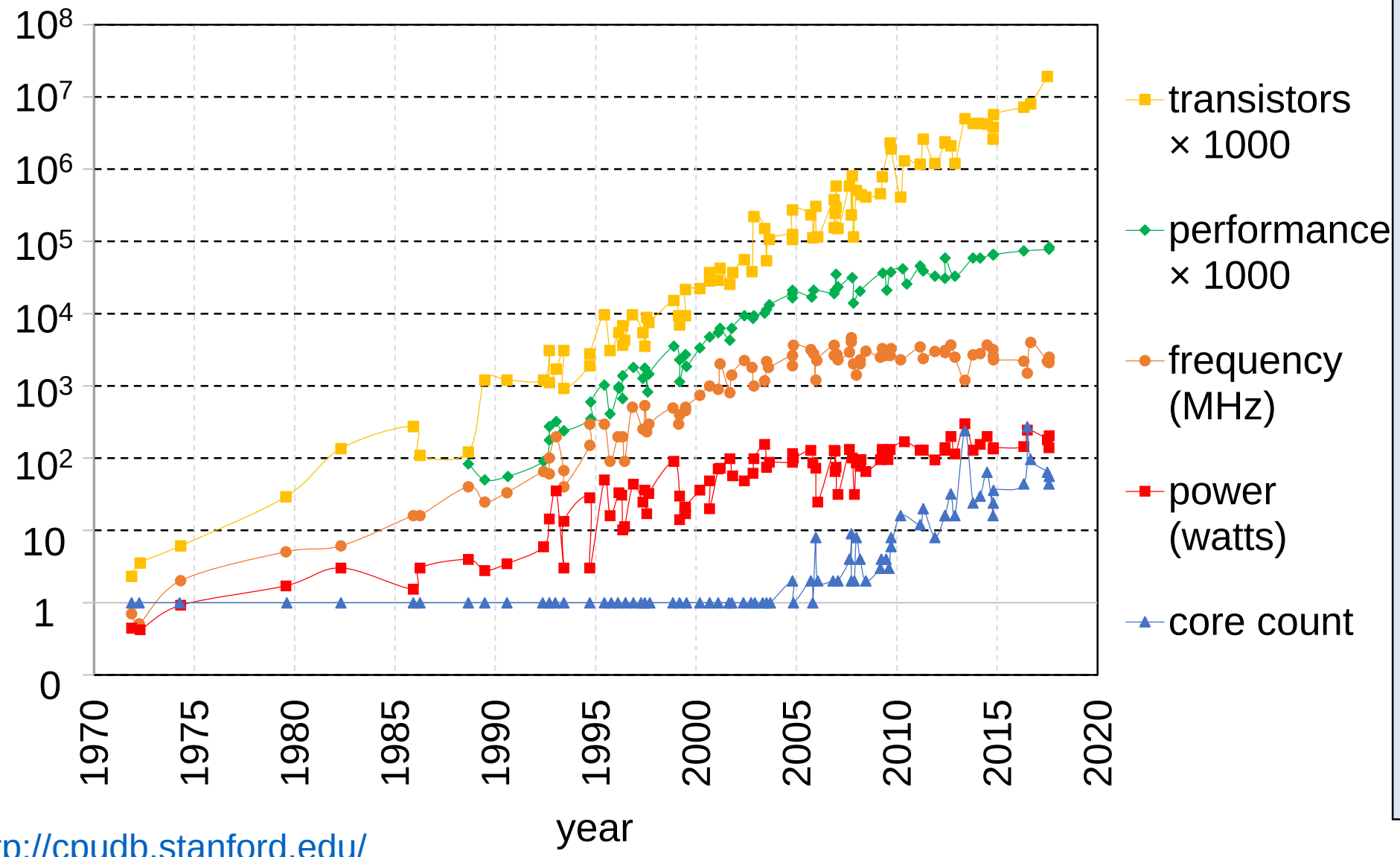



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ✗ ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

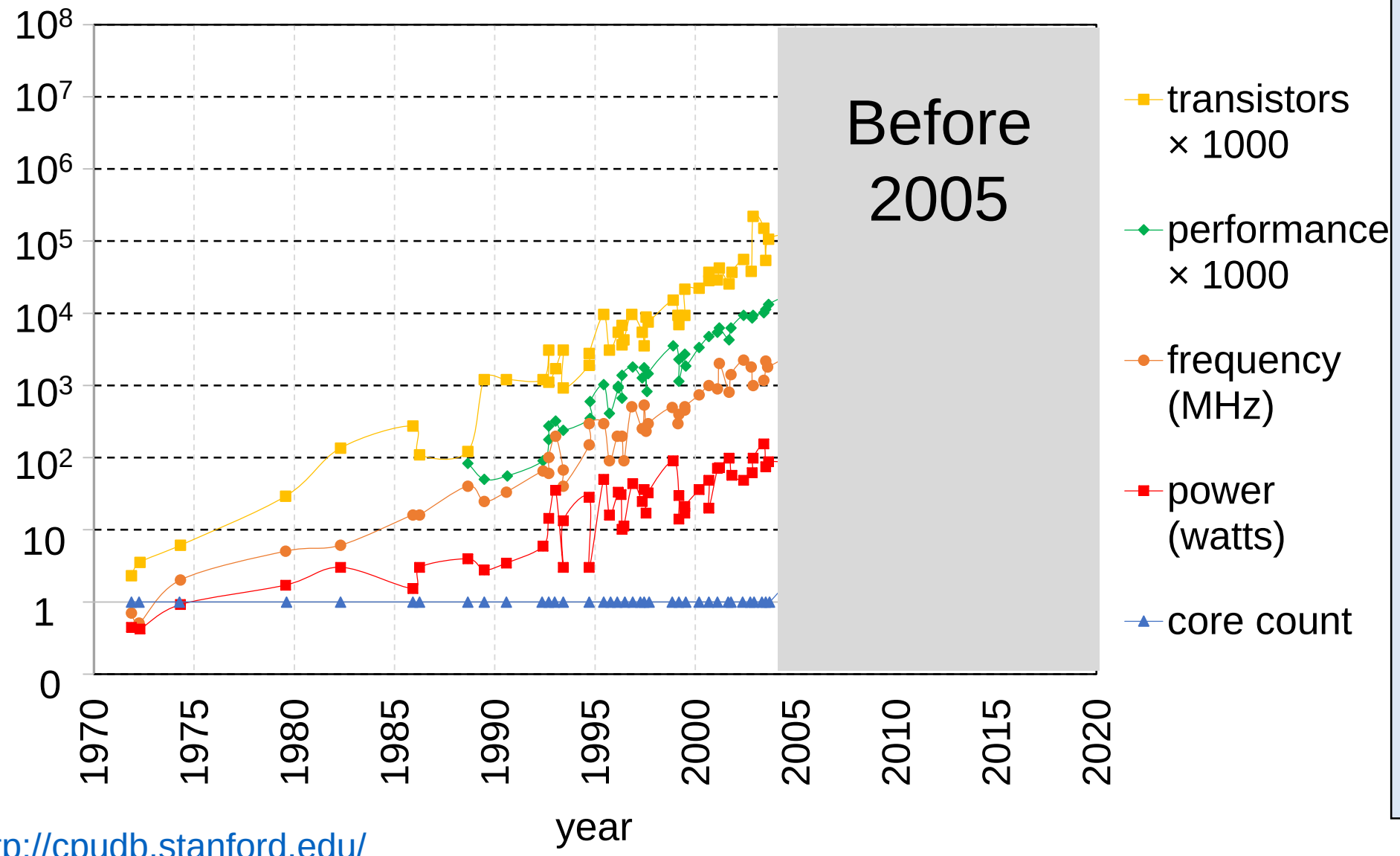



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

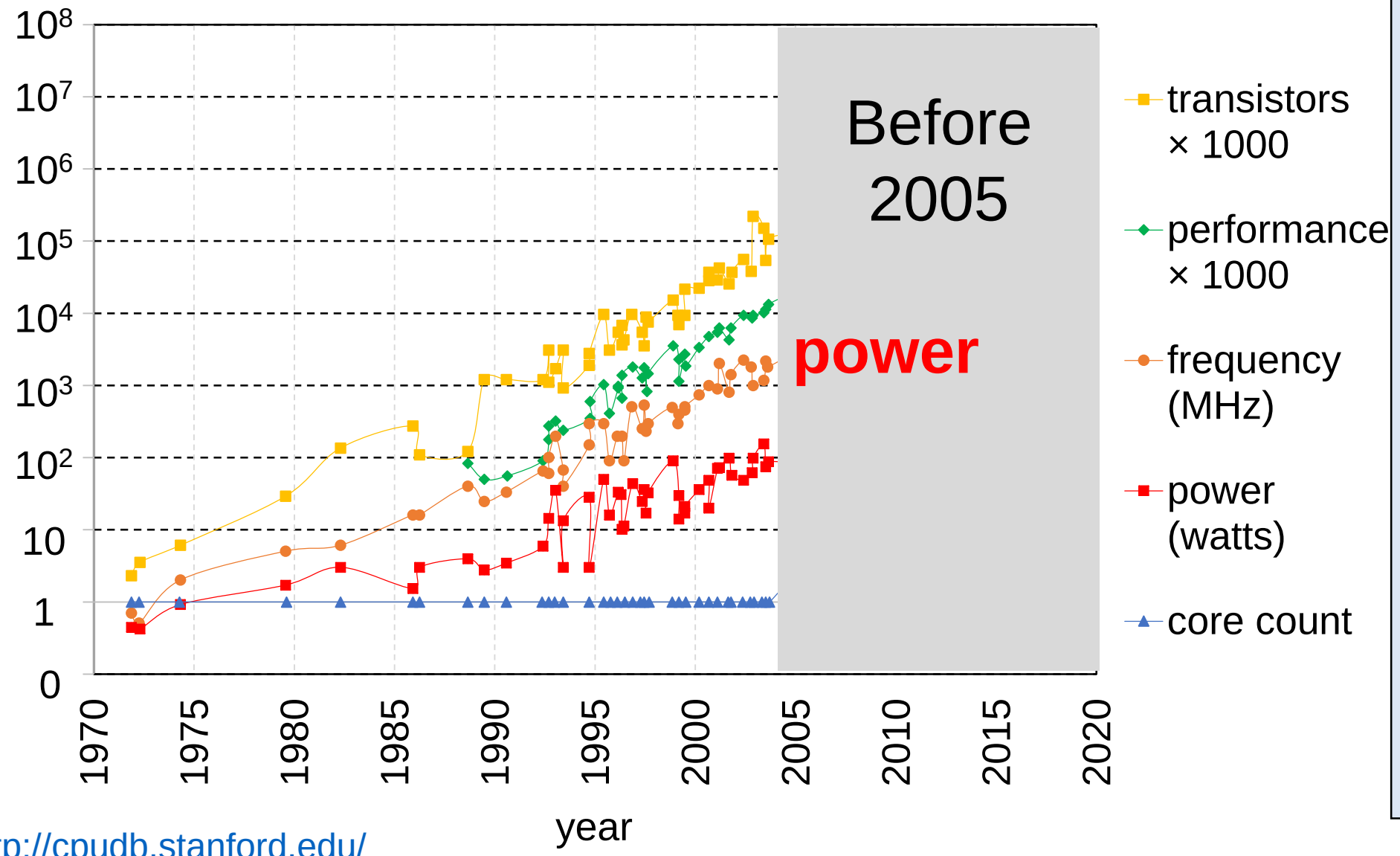



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

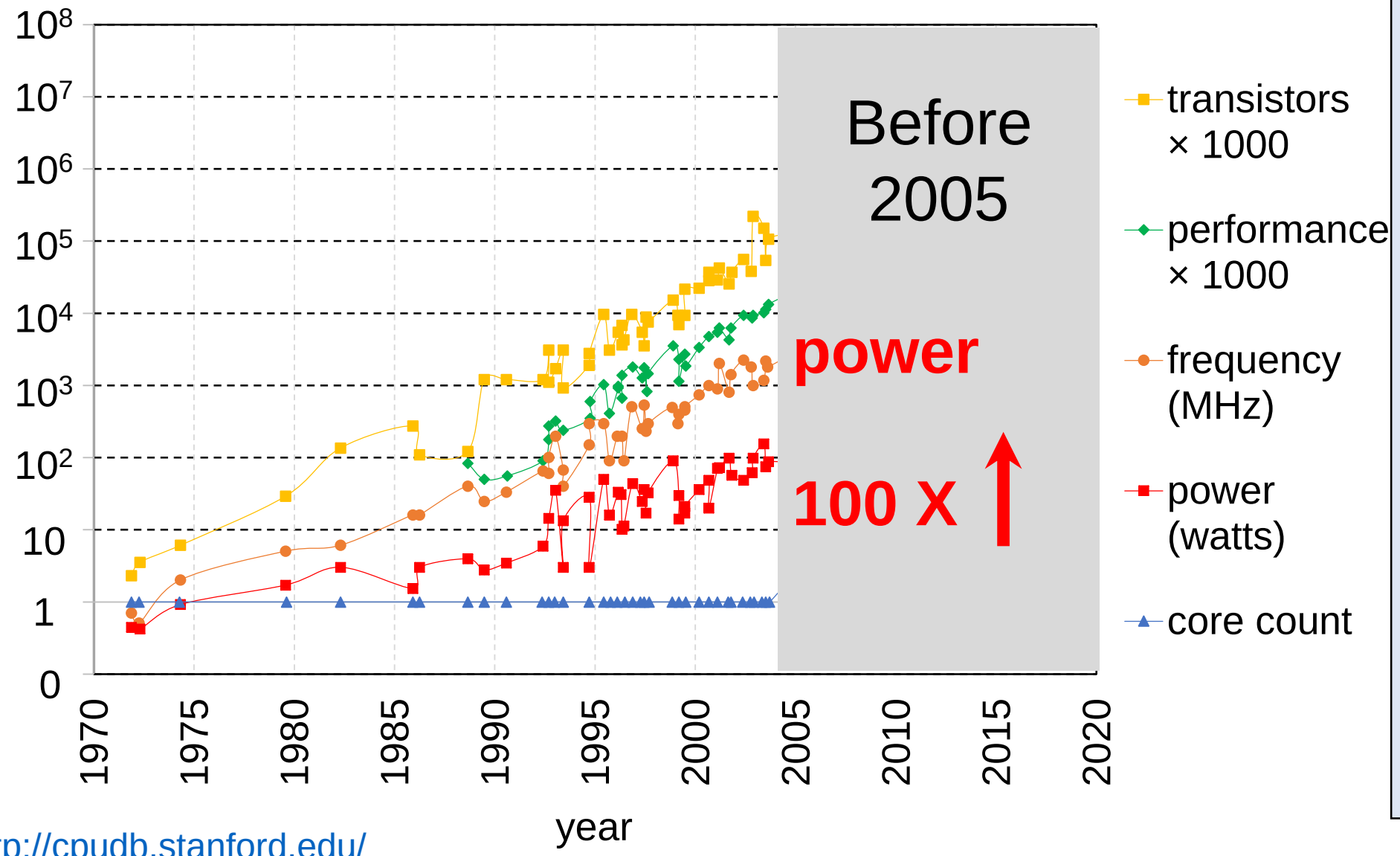



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling

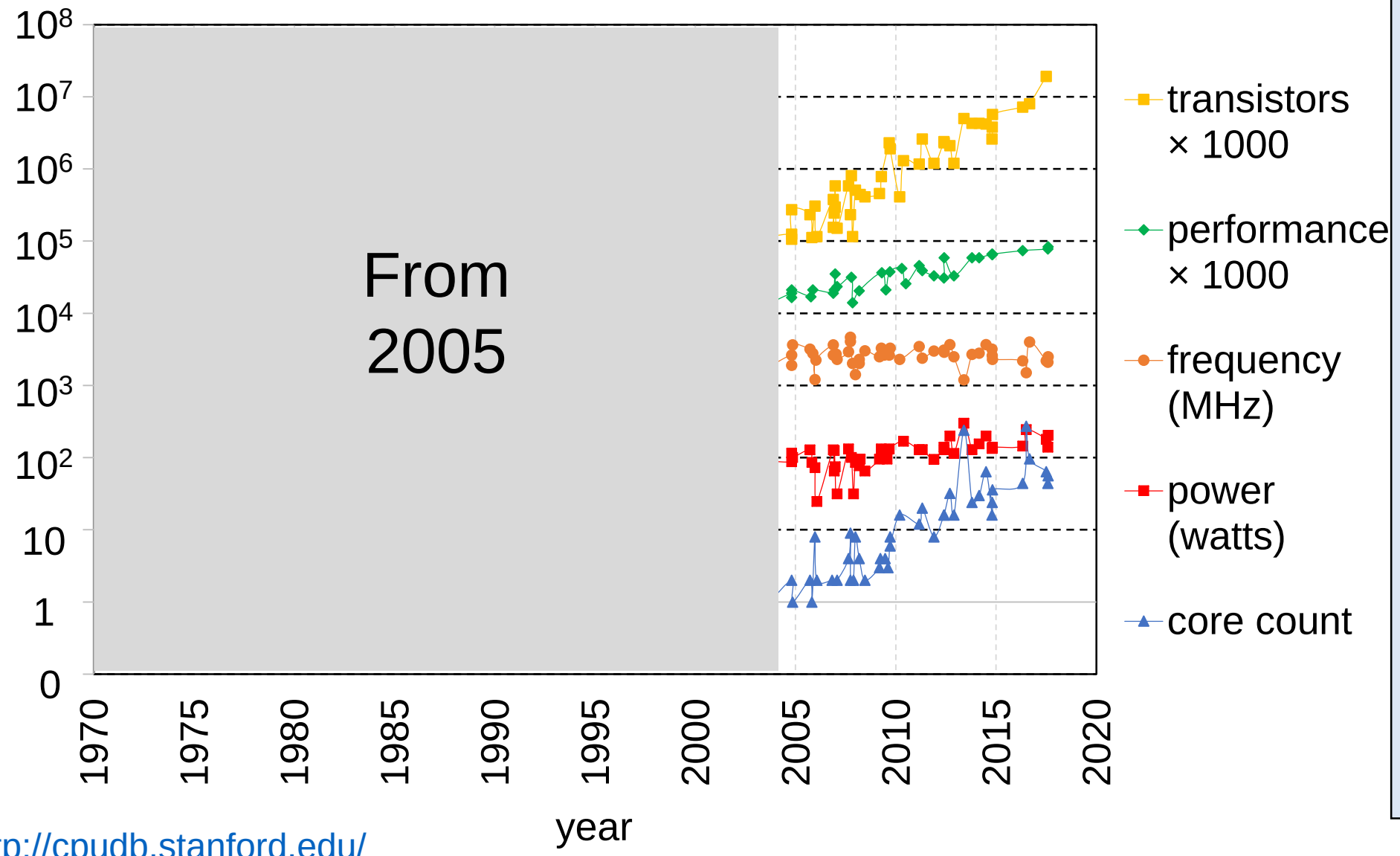



http://cpudb.stanford.edu/

## Dennard Scaling

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling




http://cpudb.stanford.edu/

## Dennard Scaling

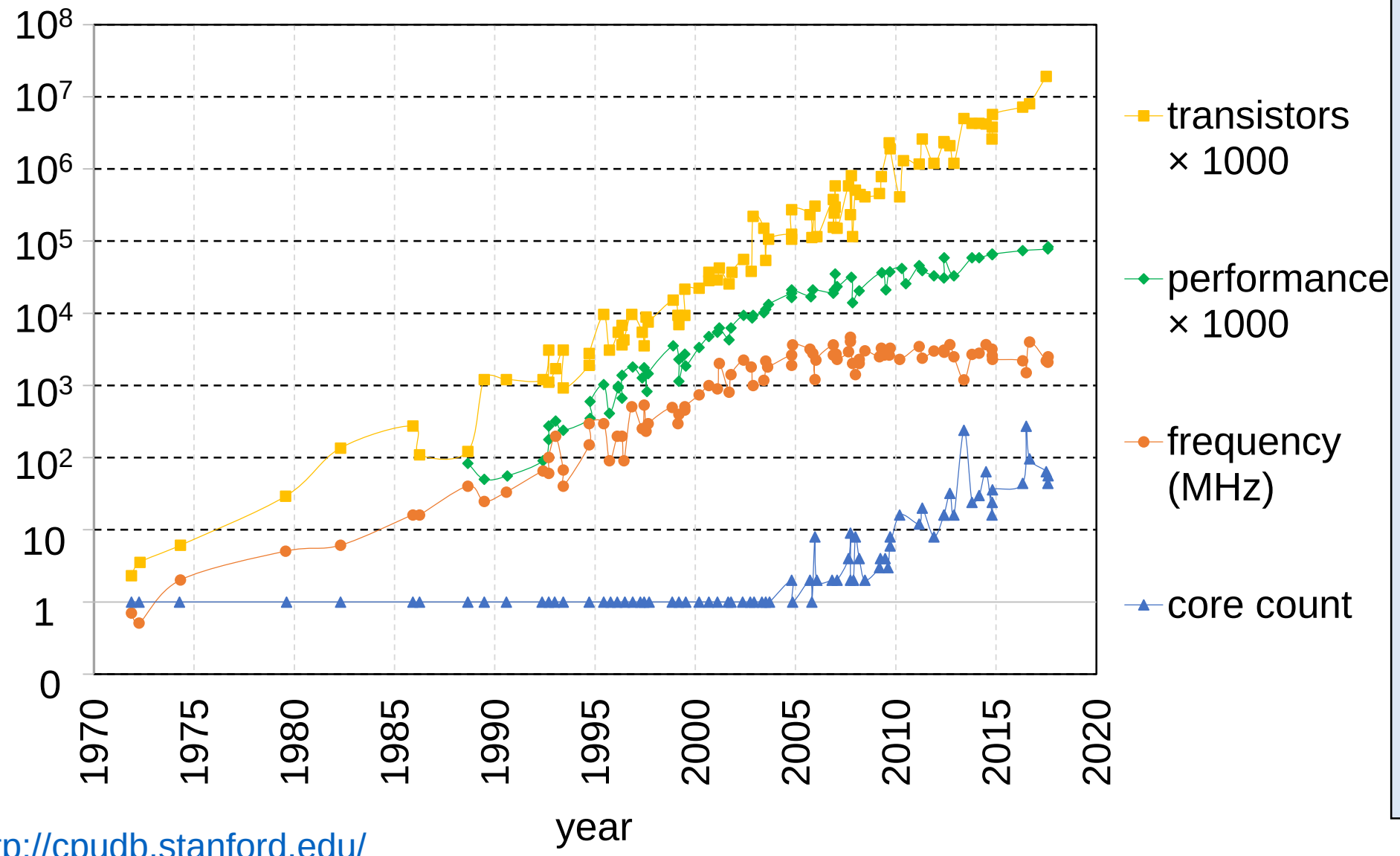
| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling on Soft Error Reliability

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

# Consequences of Technology Scaling

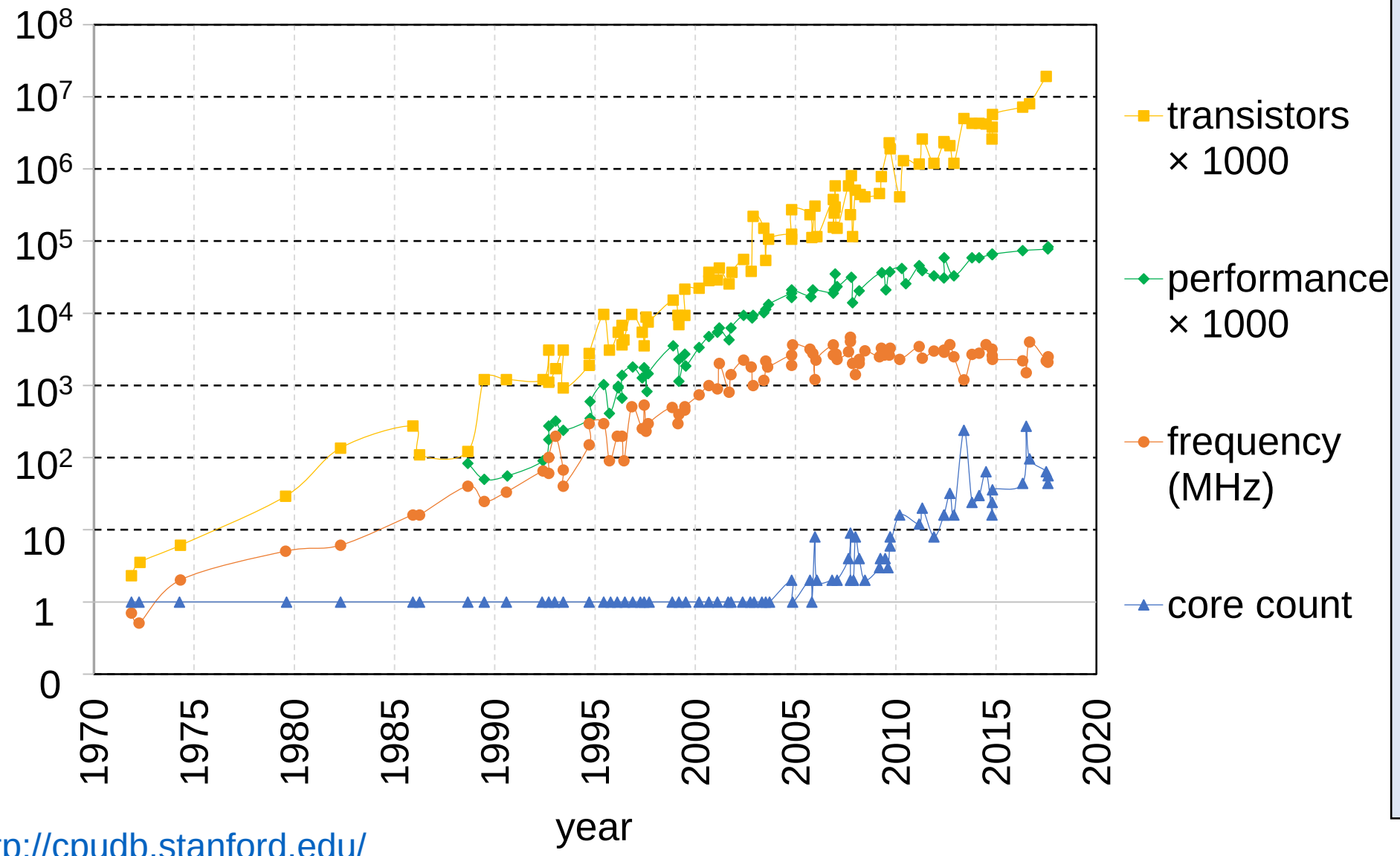



http://cpudb.stanford.edu/

**Dennard Scaling**

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage ✗ | ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling




http://cpudb.stanford.edu/

**Dennard Scaling**

| | |
|---|---|
| Size | ↓ K |
| Circuit Delay | ↓ K |
| Switching Charge | ↓ K |
| Voltage | ✗ ↓ K |
| Area | ↓ $K^2$ |
| Power | ↓ $K^2$ |
| Power Density | **constant** |

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

– Easily surpassed by neutrons

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

– Easily surpassed by neutrons

More transistors → More faults → More errors/failures

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

– Easily surpassed by neutrons

More transistors → More faults → More errors/failures

Technology scaling

# Consequences of Technology Scaling on Soft Error Reliability

Smaller transistors → Reduced charge

– Easily surpassed by neutrons

More transistors → More faults → More errors/failures

Technology scaling → Increased rate of soft errors

# Modern Processors are Increasingly Vulnerable to Soft Errors

# Modern Processors are Increasingly Vulnerable to Soft Errors

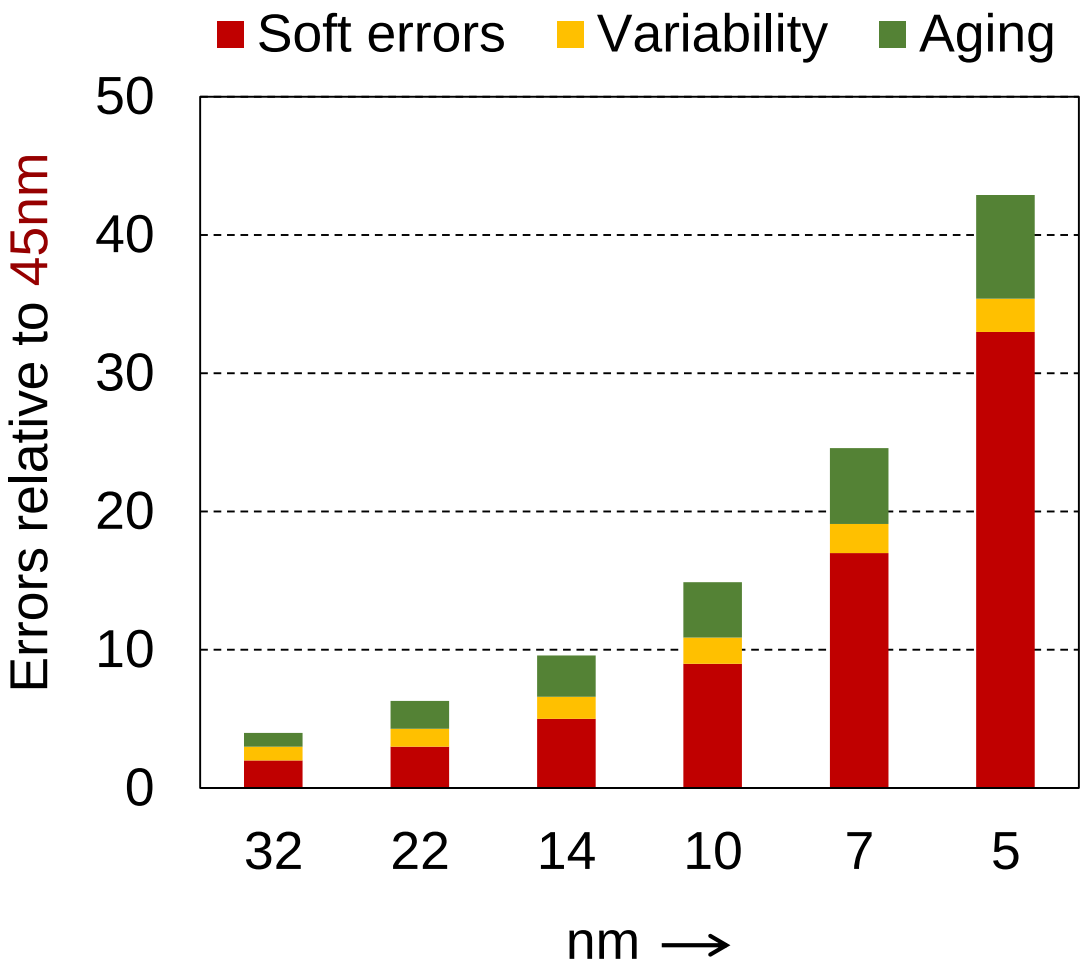
- Aggressive technology scaling

# Modern Processors are Increasingly Vulnerable to Soft Errors

- Aggressive technology scaling
- Large microarchitectural state

# Modern Processors are Increasingly Vulnerable to Soft Errors

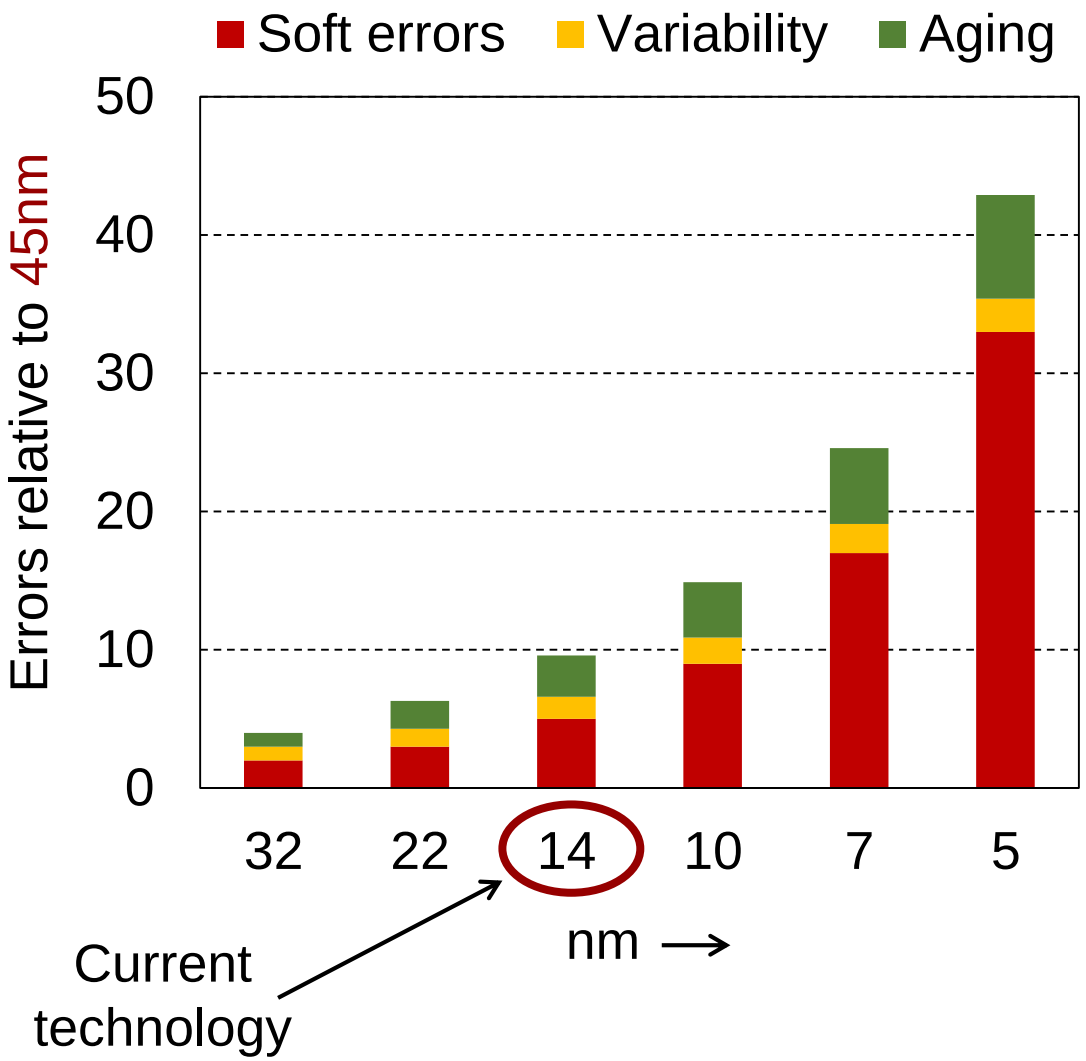
- Aggressive technology scaling
- Large microarchitectural state




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

# Modern Processors are Increasingly Vulnerable to Soft Errors

- Aggressive technology scaling
- Large microarchitectural state




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

# This Dissertation: Three New Contributions

# This Dissertation: Three New Contributions

1. Exploiting core heterogeneity to improve reliability
   [HPCA 2017, TC 2018]

# This Dissertation: Three New Contributions

1. Exploiting core heterogeneity to improve reliability
   [HPCA 2017, TC 2018]

2. Dispatch Halting: Improving out-of-order core reliability
   [DSN 2020, Under review]

# This Dissertation: Three New Contributions

1. Exploiting core heterogeneity to improve reliability
   [HPCA 2017, TC 2018]

2. Dispatch Halting: Improving out-of-order core reliability
   [DSN 2020, Under review]

3. Precise Runahead Execution: Improving performance and evaluating reliability
   [CAL 2019, HPCA 2020]

# Contribution #1 [HPCA 2017, TC 2018]

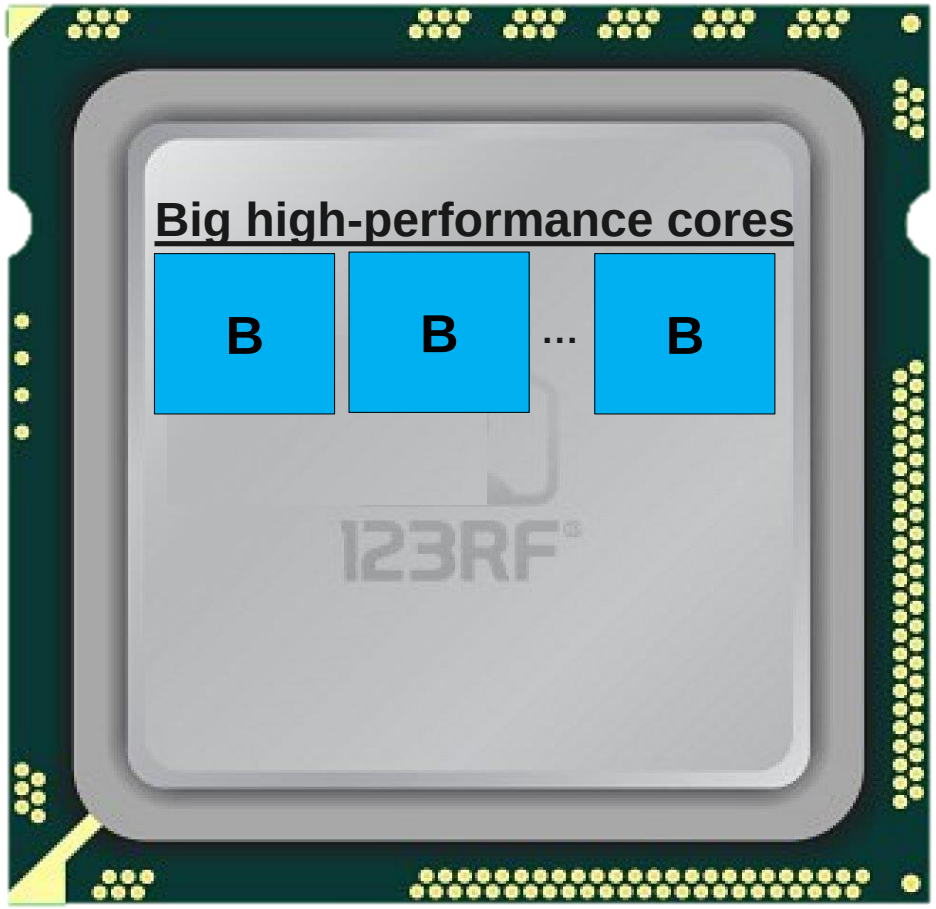
# Exploiting Core Heterogeneity to Improve Reliability

# Benefits of Heterogeneous Multicores

- Multiple core types

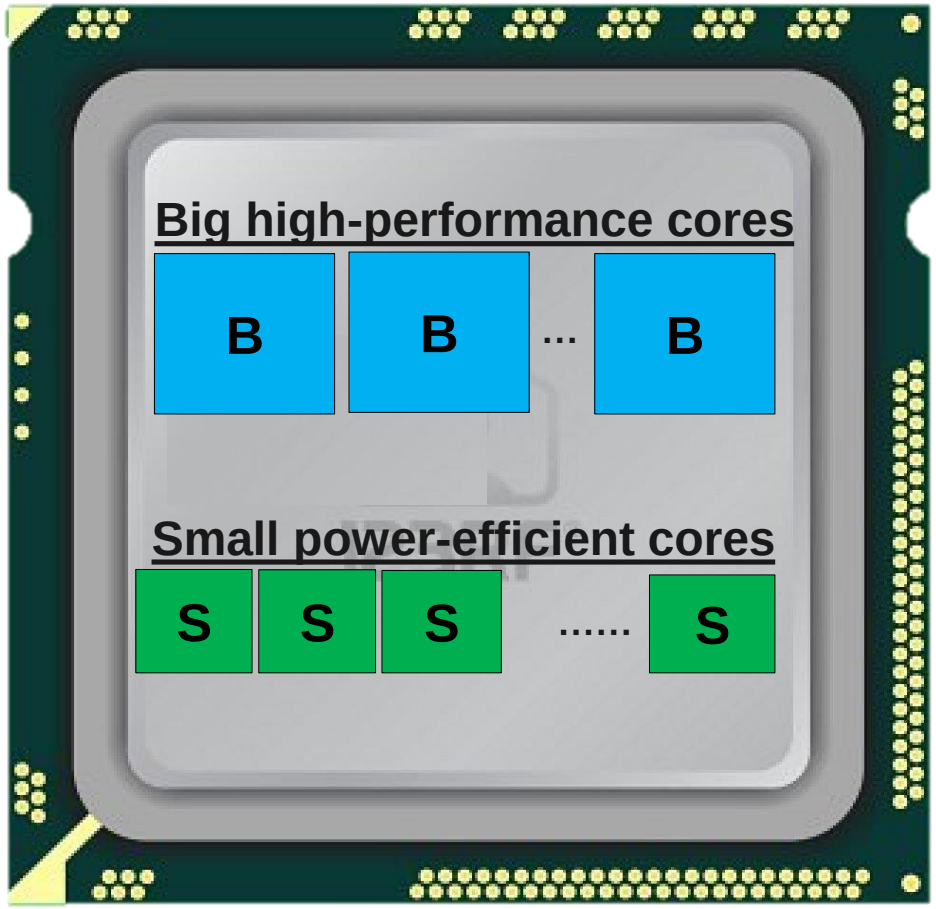
# Benefits of Heterogeneous Multicores

- Multiple core types



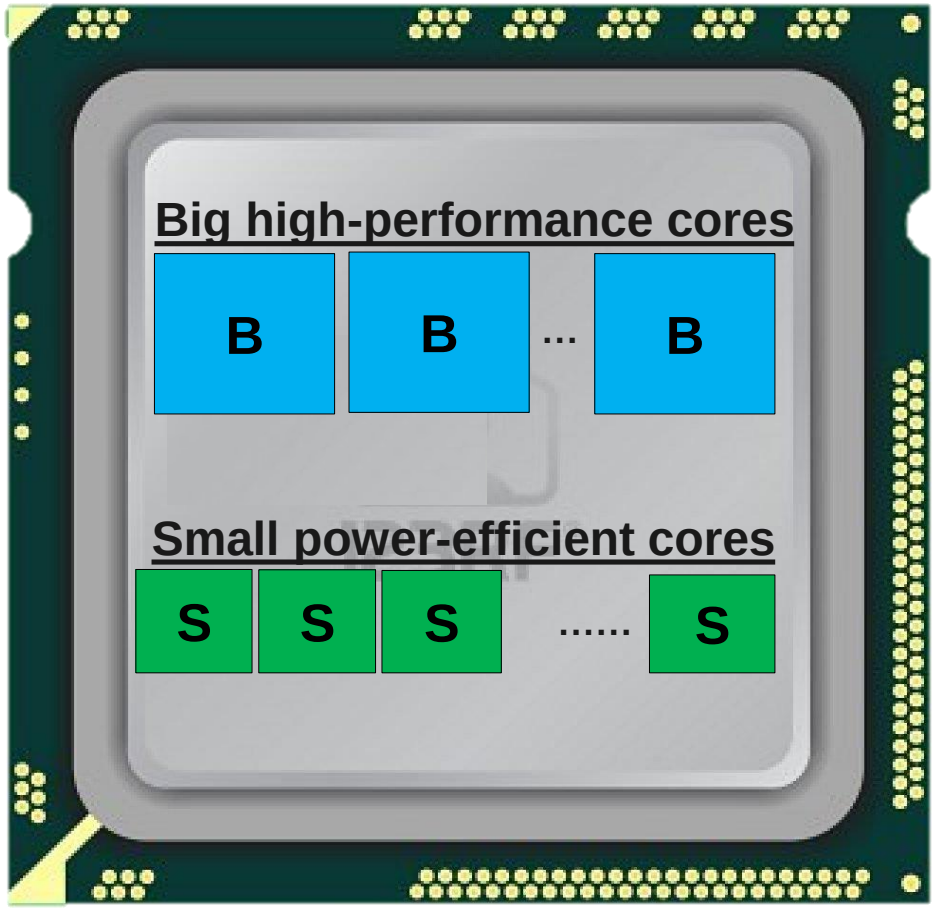
# Benefits of Heterogeneous Multicores

- Multiple core types



# Benefits of Heterogeneous Multicores

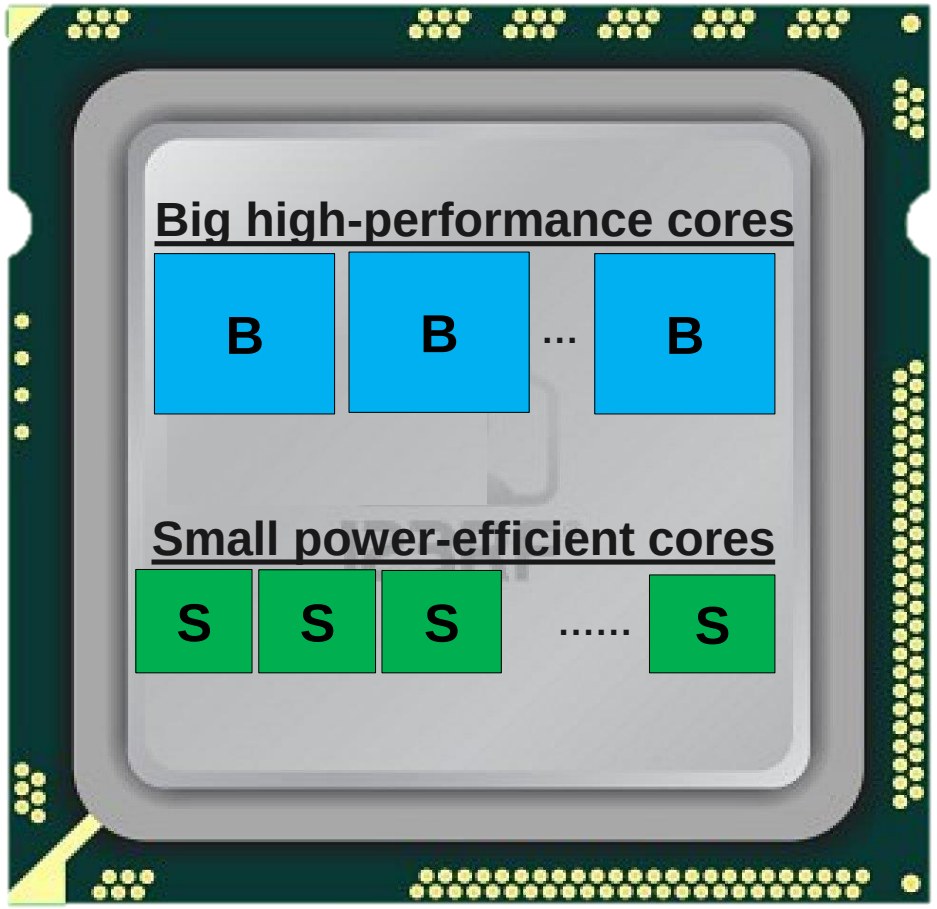
- Multiple core types
- Well-established power benefits*




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

# Benefits of Heterogeneous Multicores

- Multiple core types
- Well-established power benefits*
- Well-established scheduling techniques**

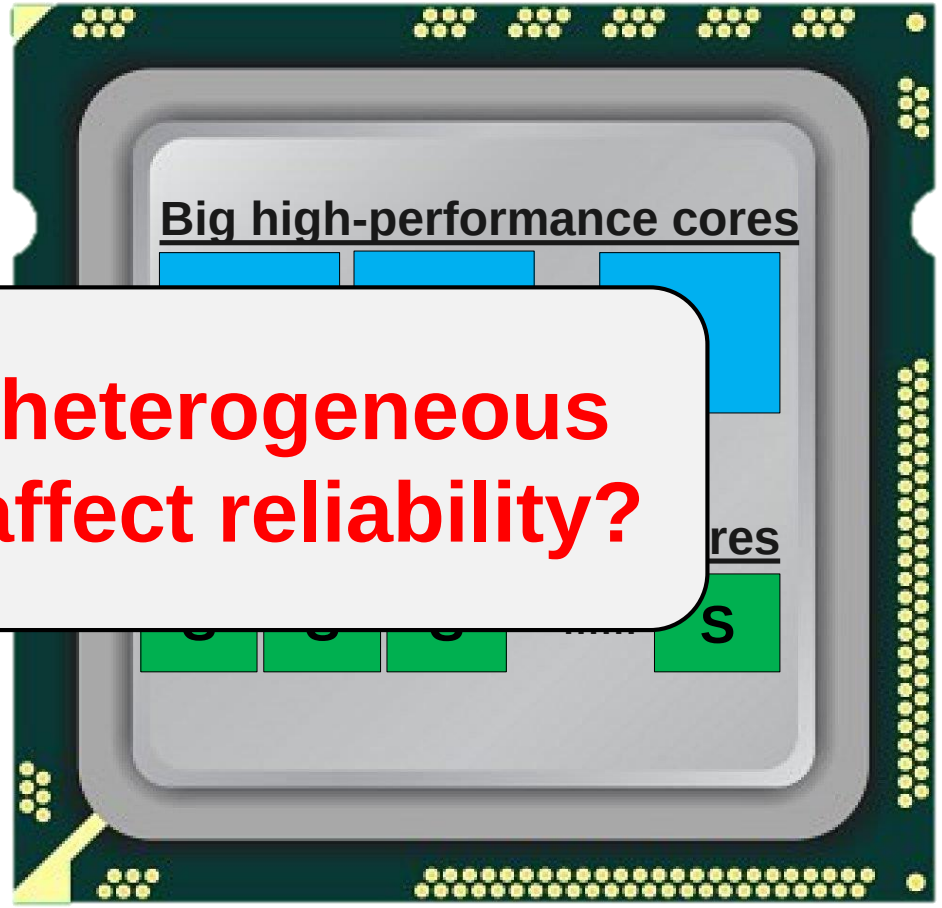



e.g. Big.LITTLE and Kal-El

*[Kumar et al. MICRO'03, ISCA'04] **[Van Craeynest et al. ISCA'12]

# Benefits of Heterogeneous Multicores

- Multiple core types
- Well-esta
- Well-esta
  techniques**




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

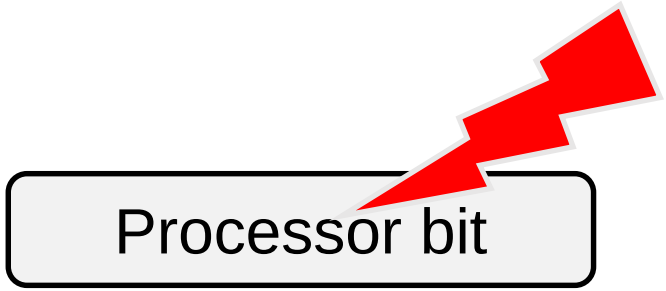
e.g. Big.LITTLE and Kal-El

*[Kumar et al. MICRO'03, ISCA'04] **[Van Craeynest et al. ISCA'12]

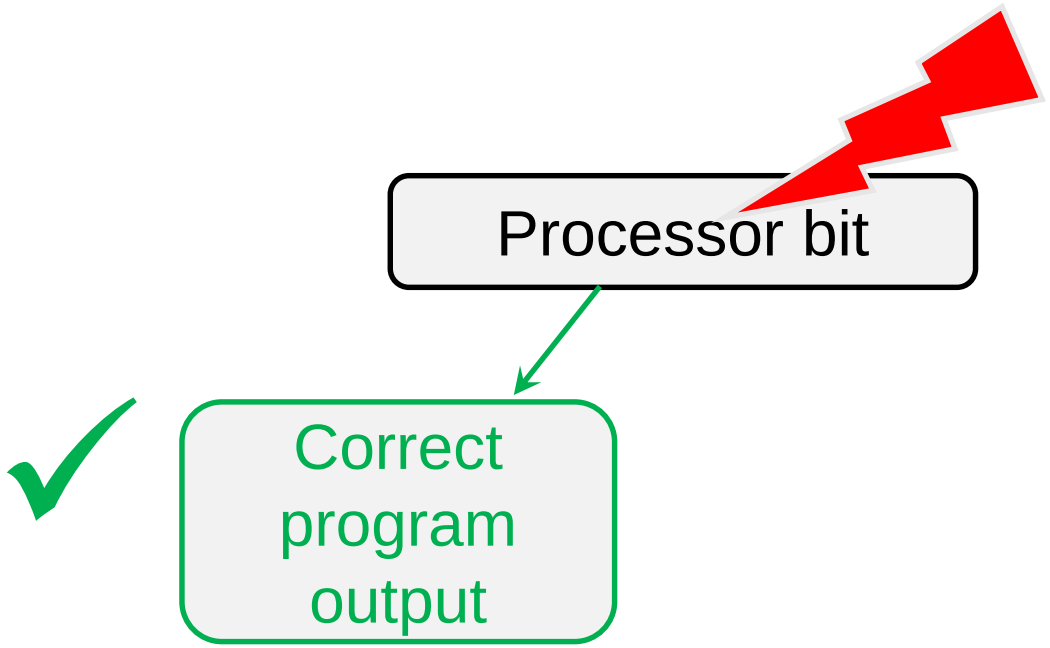
# ACE Analysis for Estimating Soft Errors

Processor bit

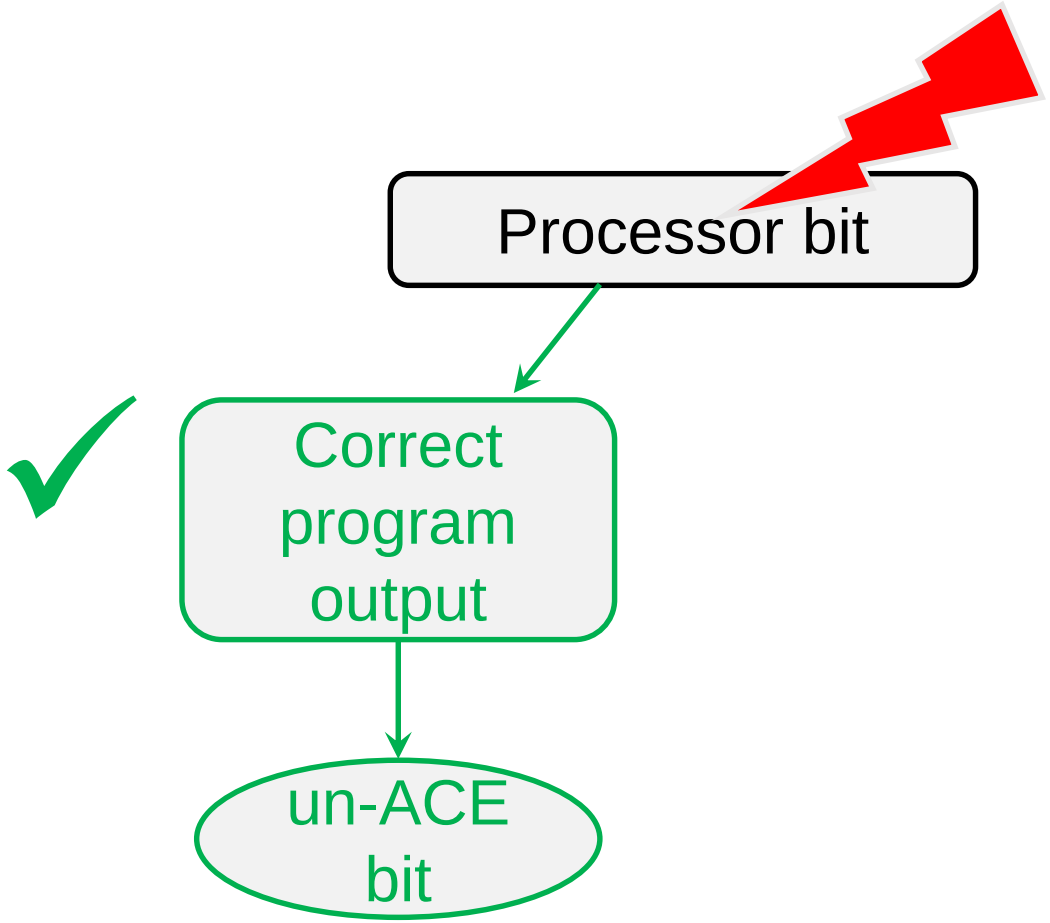
# ACE Analysis for Estimating Soft Errors



# ACE Analysis for Estimating Soft Errors

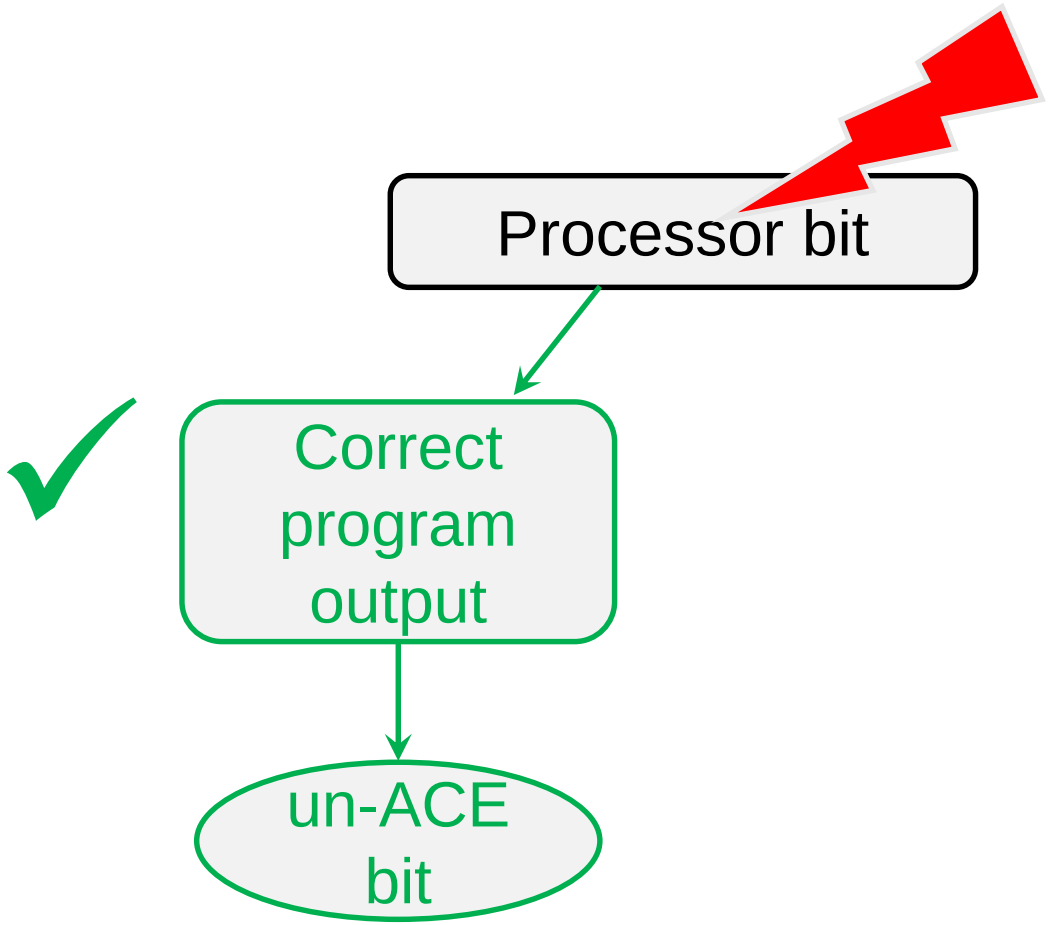


# ACE Analysis for Estimating Soft Errors




**ACE** →
**A**rchitecturally
**C**orrect
**E**xecution

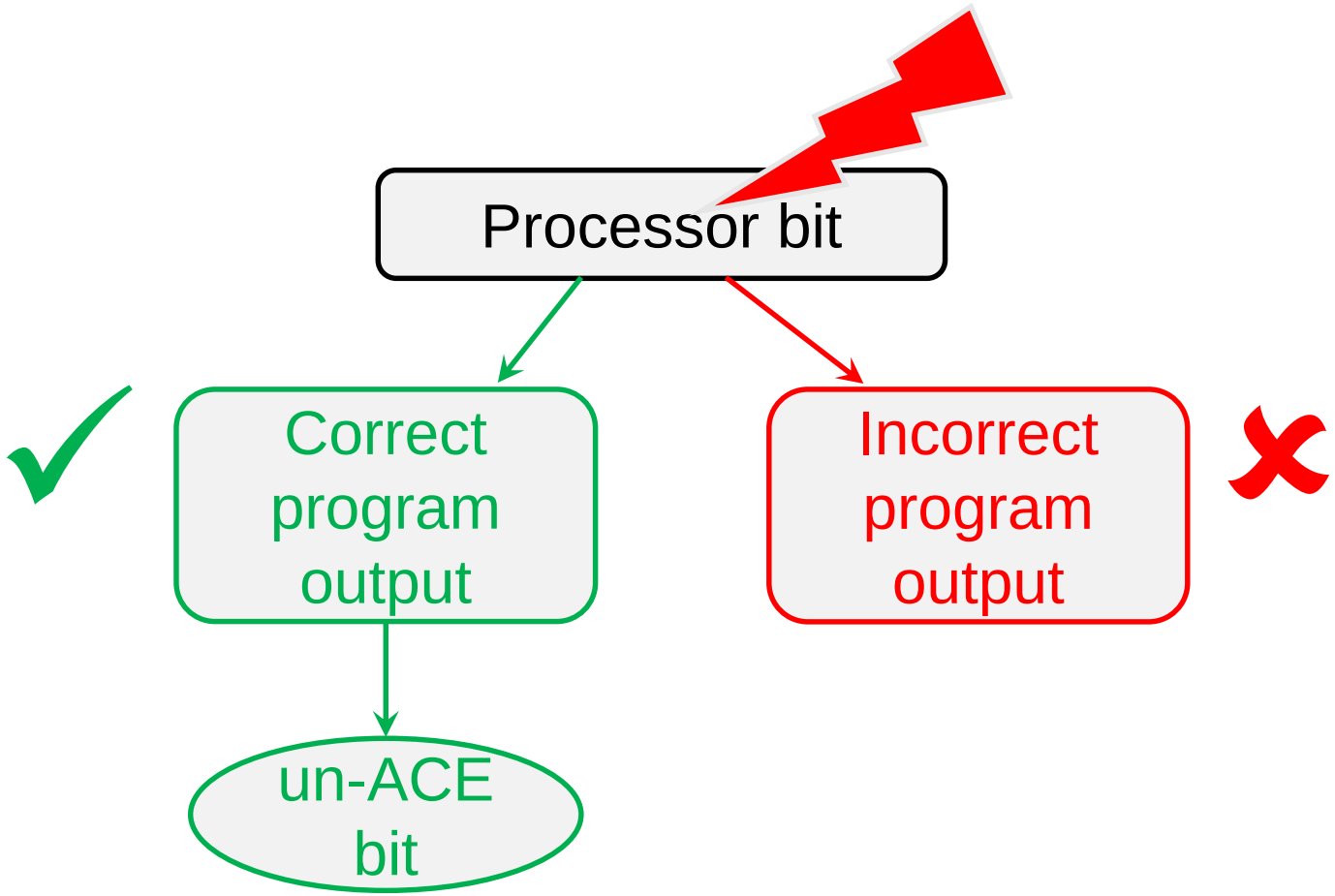
# ACE Analysis for Estimating Soft Errors




Branch predictor, wrong-path instructions

**ACE** →
**A**rchitecturally
**C**orrect
**E**xecution

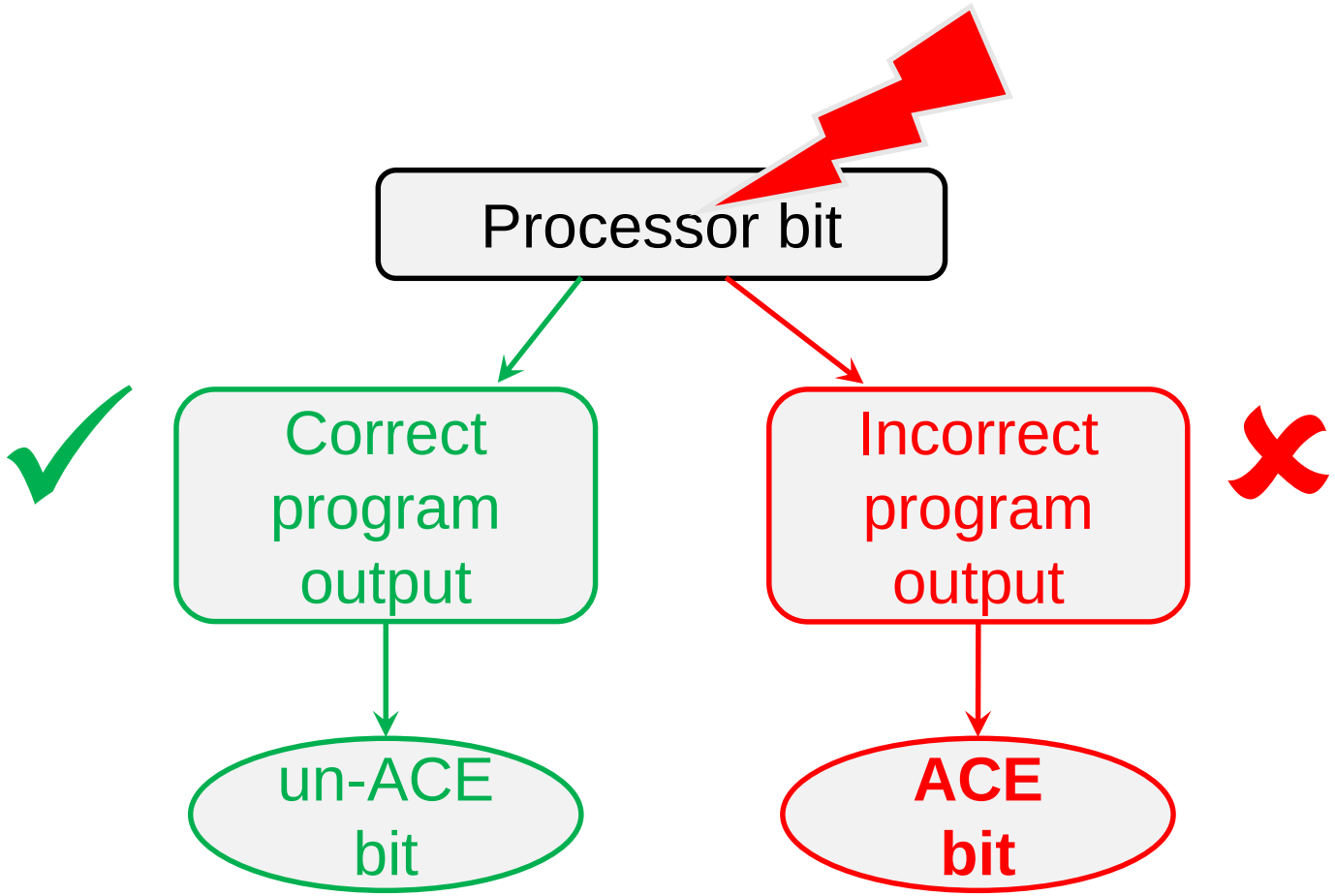
# ACE Analysis for Estimating Soft Errors




Branch predictor,
wrong-path instructions

**ACE** →
**A**rchitecturally
**C**orrect
**E**xecution

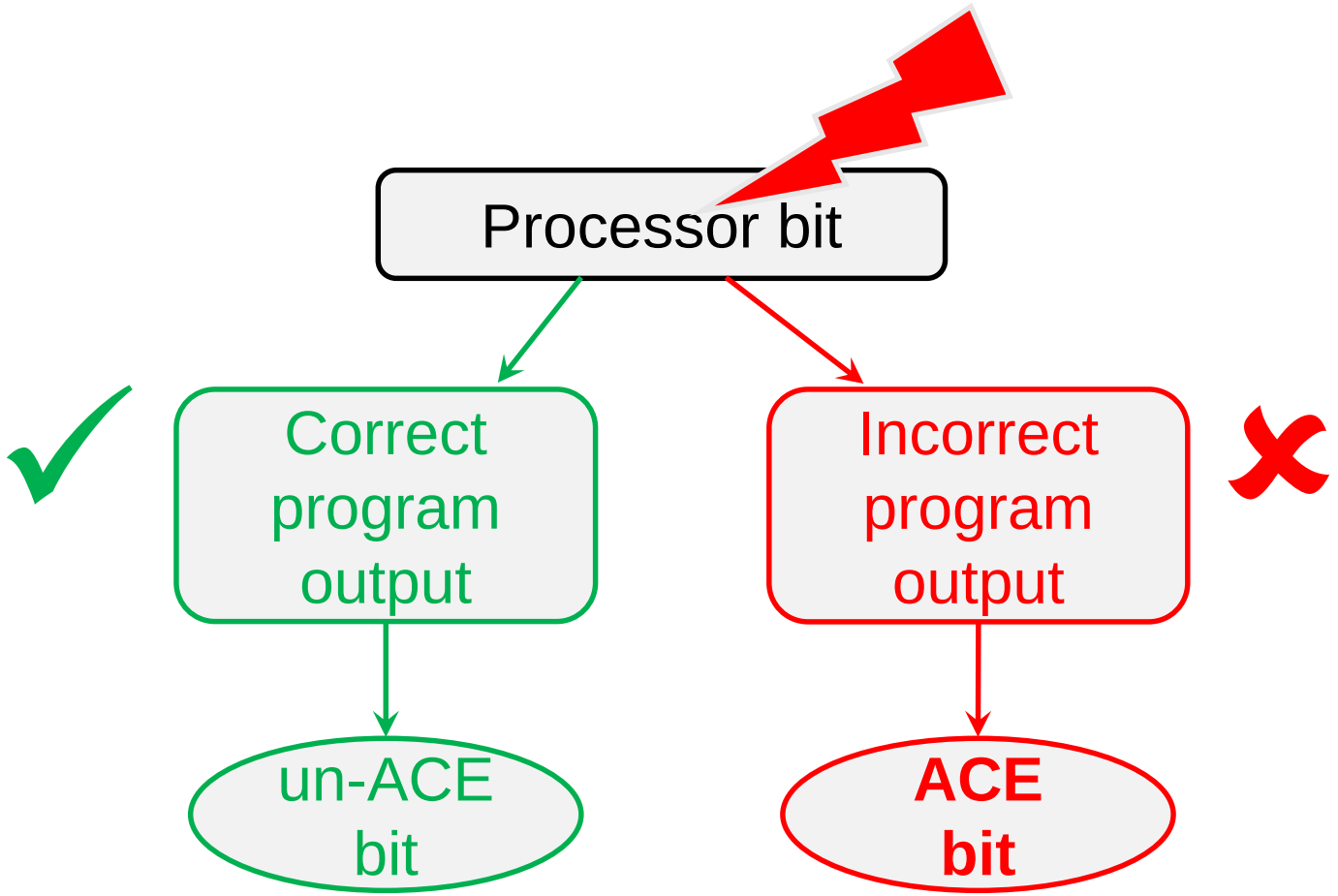
# ACE Analysis for Estimating Soft Errors




Branch predictor, wrong-path instructions

**ACE** →
**A**rchitecturally
**C**orrect
**E**xecution

# ACE Analysis for Estimating Soft Errors

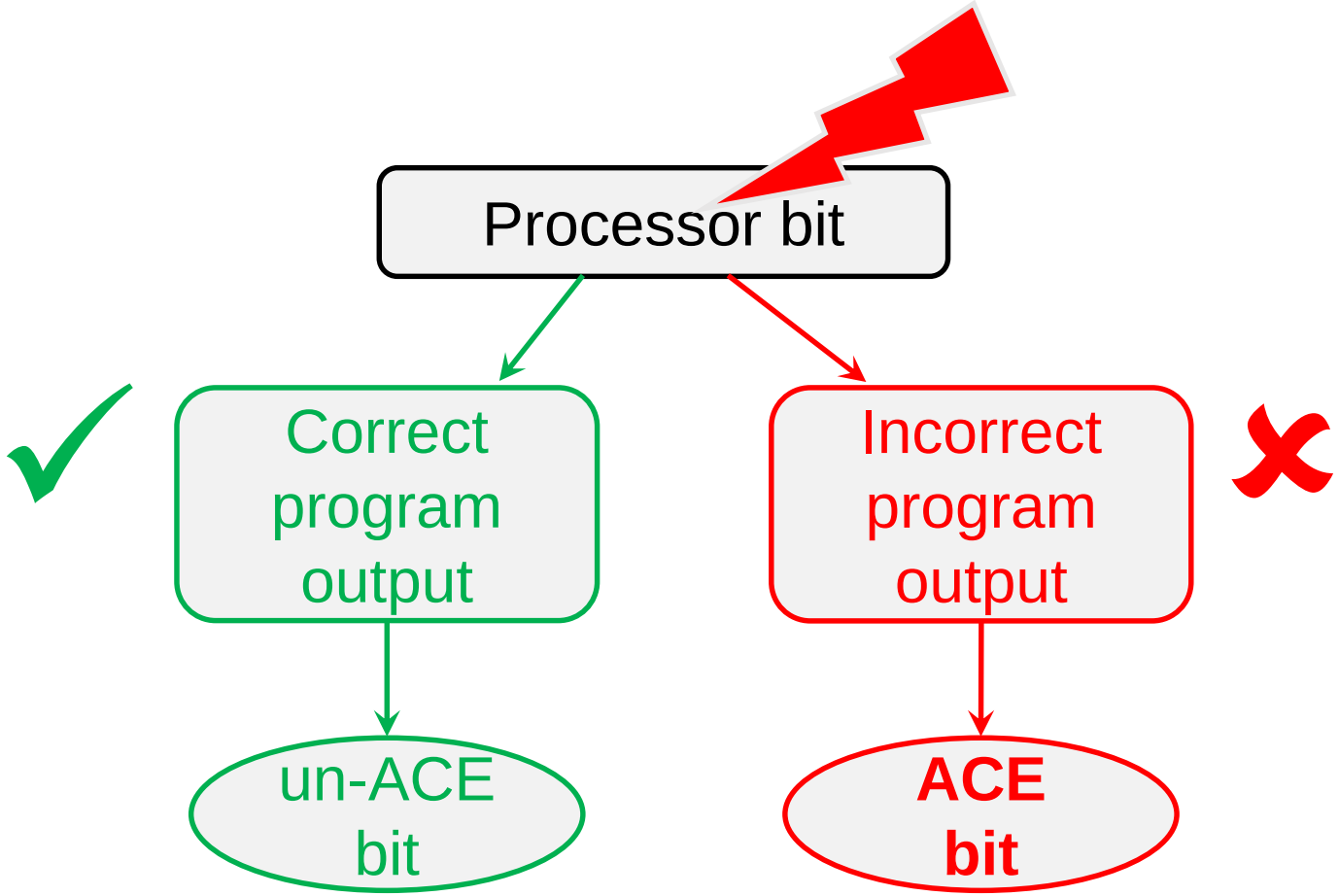



Branch predictor, wrong-path instructions

Program counter, correct-path instructions

**ACE** →
**A**rchitecturally
**C**orrect
**E**xecution

# ACE Analysis for Estimating Soft Errors




Branch predictor, wrong-path instructions

Program counter, correct-path instructions

**ACE** → **A**rchitecturally **C**orrect **E**xecution

**For correct program execution, all ACE bits must be correct**

# ACE Analysis for Estimating Soft Errors

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

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

IFR = Probability of a soft error

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

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

IFR = Probability of a soft error

For the same execution time

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

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

IFR = Probability of a soft error

For the same execution time

ABC

SER

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

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

IFR = Probability of a soft error

For the same execution time

ABC ↑ SER

# ACE Analysis for Estimating Soft Errors

ACE Cycles = Total cycles a bit was ACE

ACE Bit Count (ABC) = Sum of ACE cycles for all committed instructions

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

IFR = Probability of a soft error

For the same execution time

ABC ↑

SER ↑

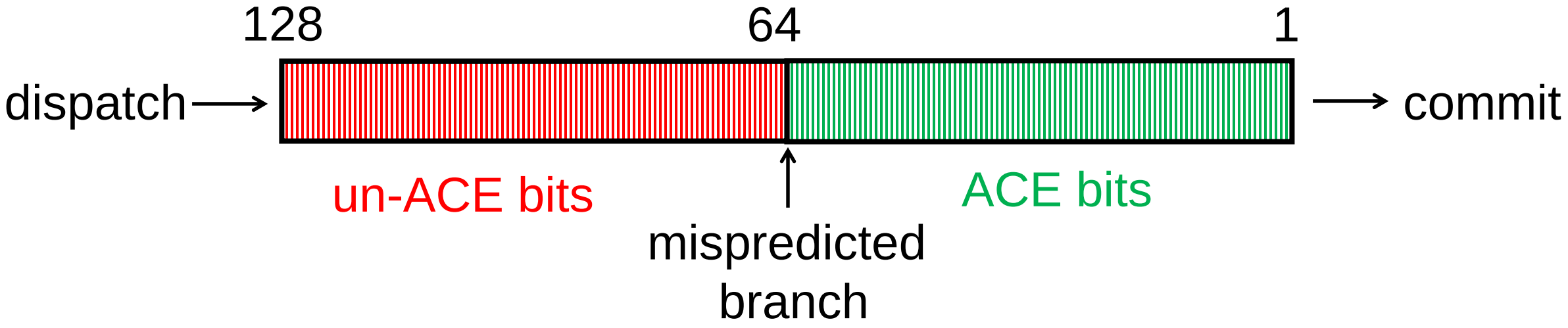
# Example: Reorder Buffer (ROB)

# Example: Reorder Buffer (ROB)

Size = 128, each entry = 100 bits, total bits = 12800

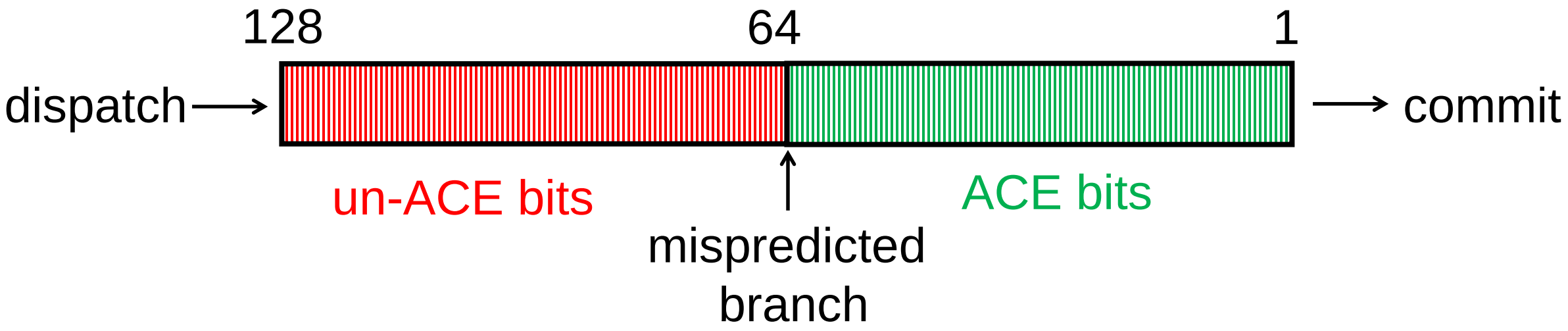
# Example: Reorder Buffer (ROB)

Size = 128, each entry = 100 bits, total bits = 12800



# Example: Reorder Buffer (ROB)

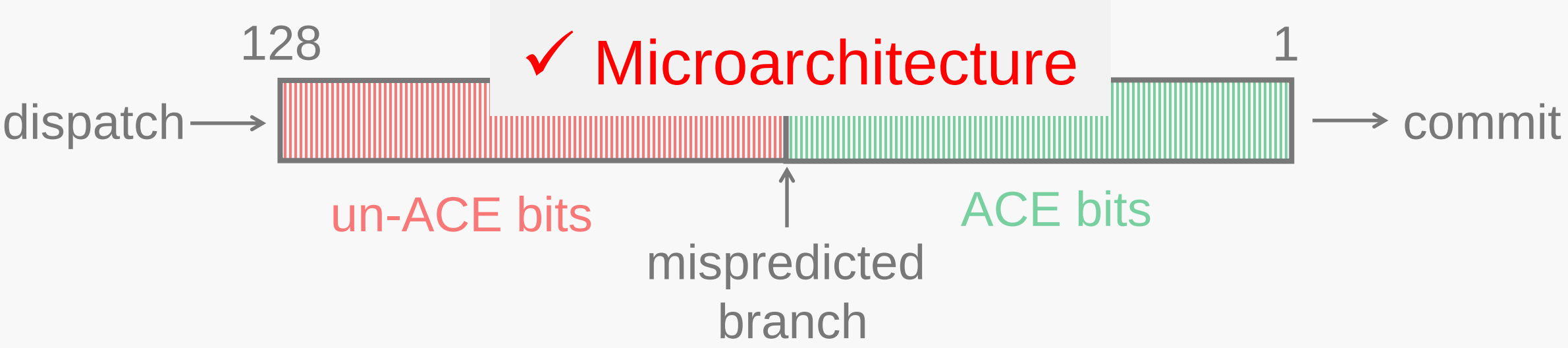
Size = 128, each entry = 100 bits, total bits = 12800




ROB ACE bits = 64 × 100 = 6400

# Example: Reorder Buffer (ROB)

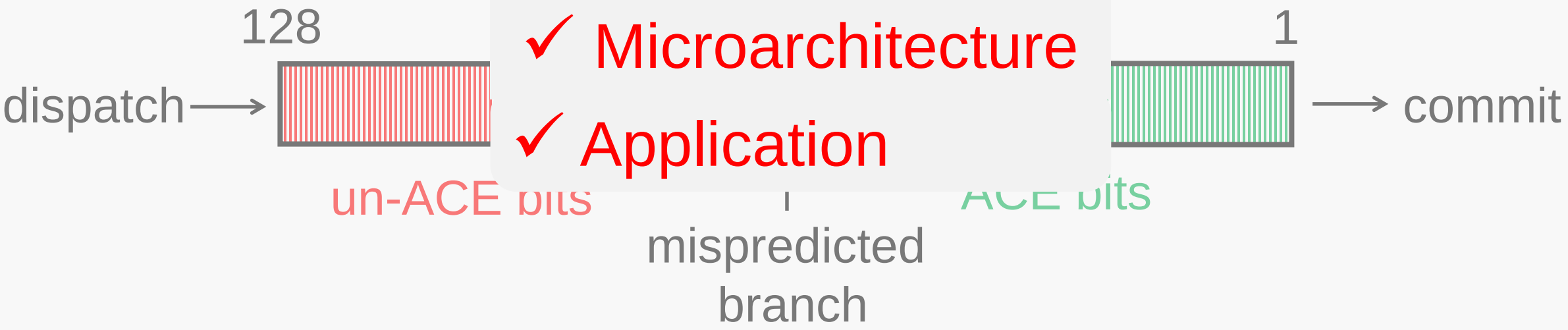
Size = 128, each entry = 100 bits, total bits = 12800




ROB ACE bits = 64 × 100 = 6400

# Example: Reorder Buffer (ROB)

Size = 128, each entry = 100 bits, total bits = 12800



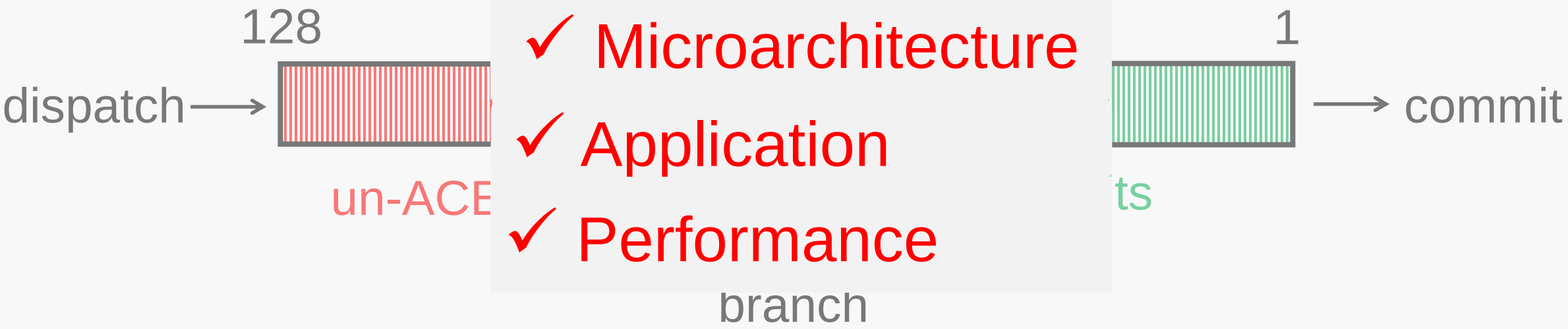

✓ Microarchitecture

✓ Application

ROB ACE bits = 64 × 100 = 6400

# Example: Reorder Buffer (ROB)

Size = 128, each entry = 100 bits, total bits = 12800

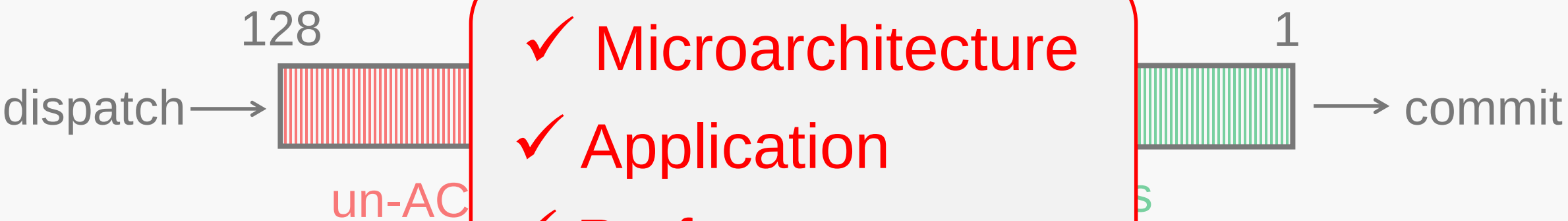



- ✓ Microarchitecture
- ✓ Application
- ✓ Performance

ROB ACE bits = 64 × 100 = 6400

# Example: Reorder Buffer (ROB)

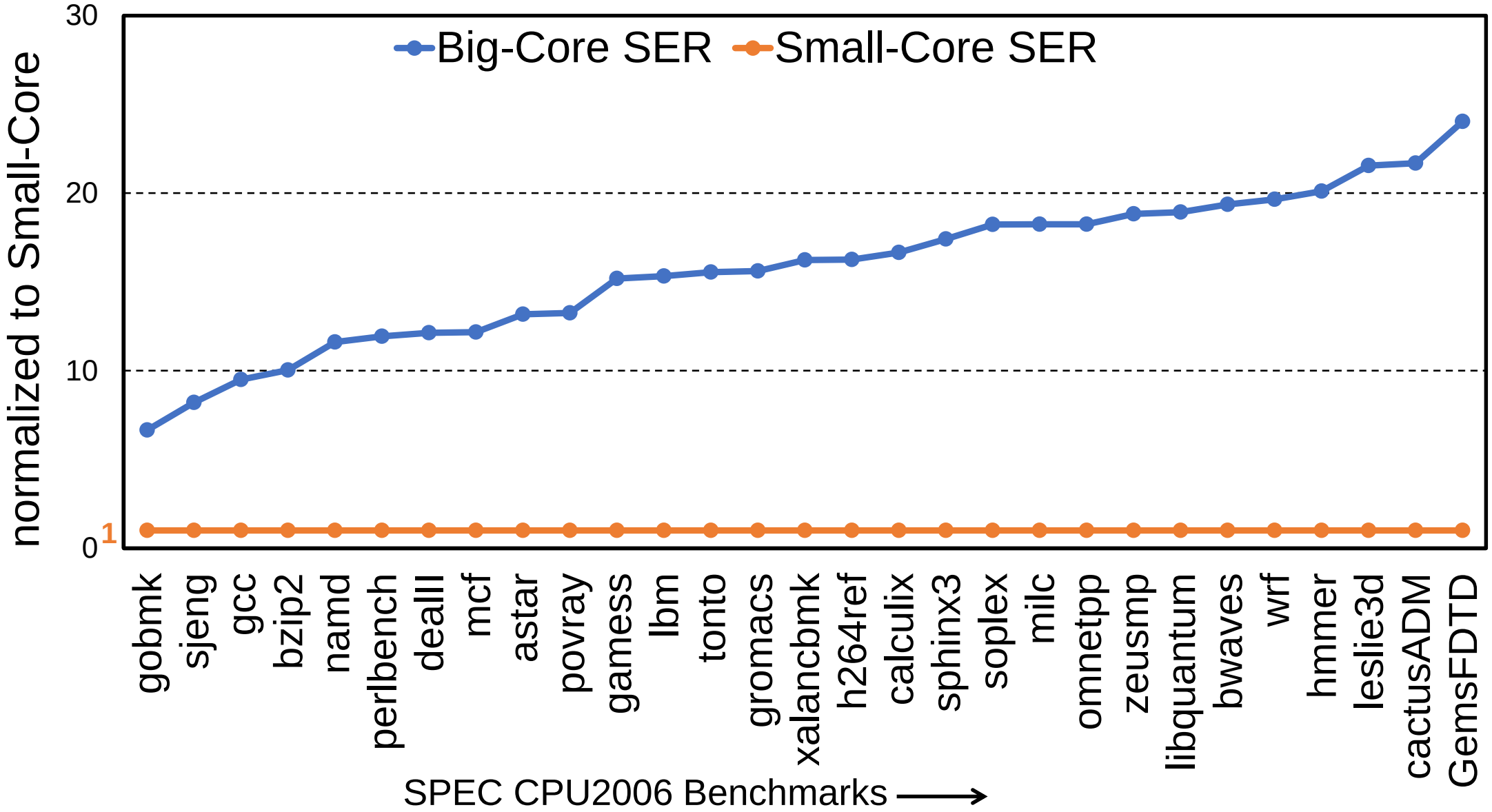
Size = 128, each entry = 100 bits, total bits = 12800



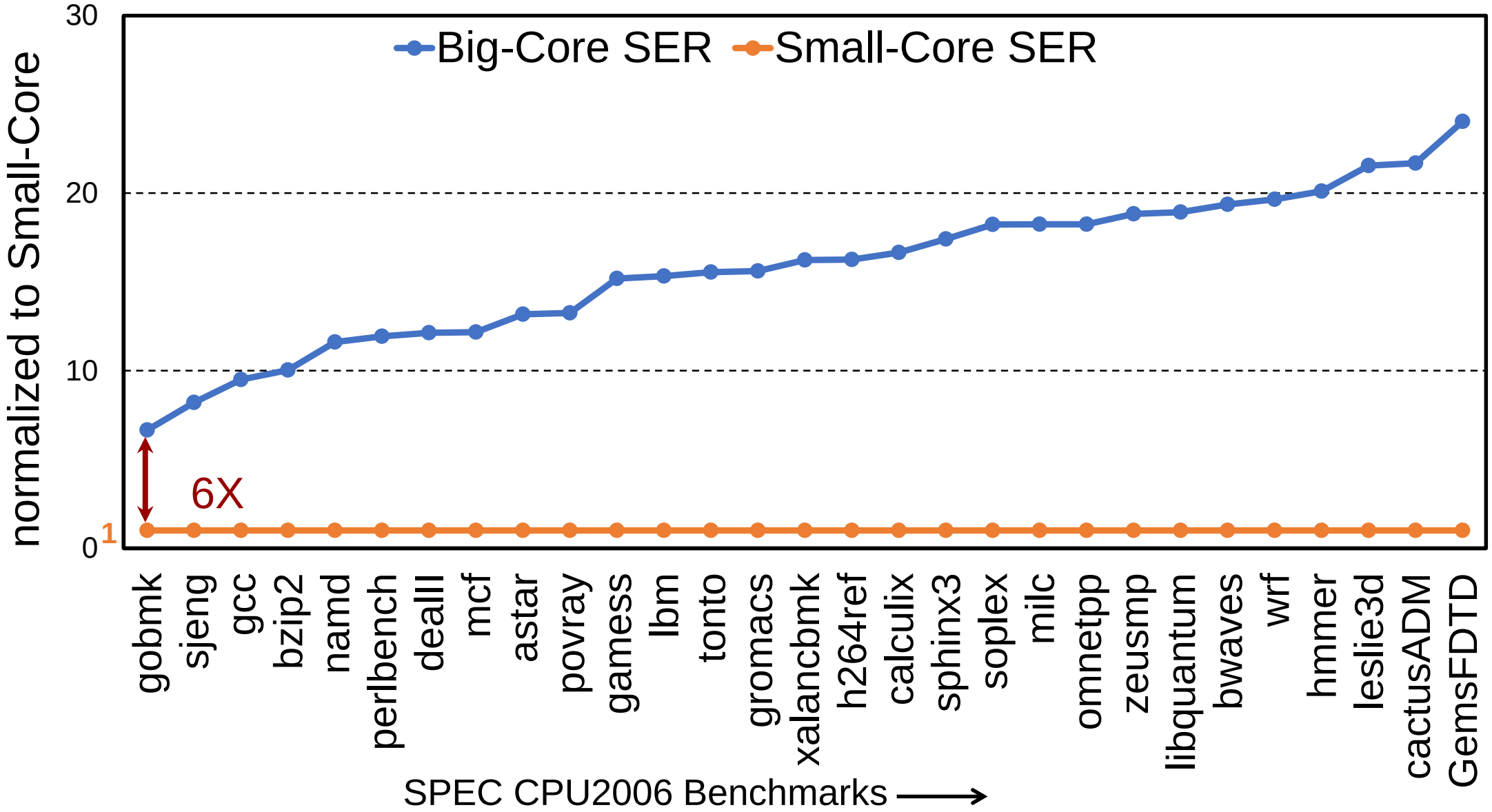

- ✓ Microarchitecture
- ✓ Application
- ✓ Performance

ROB ACE bits = 64 × 100 = 6400

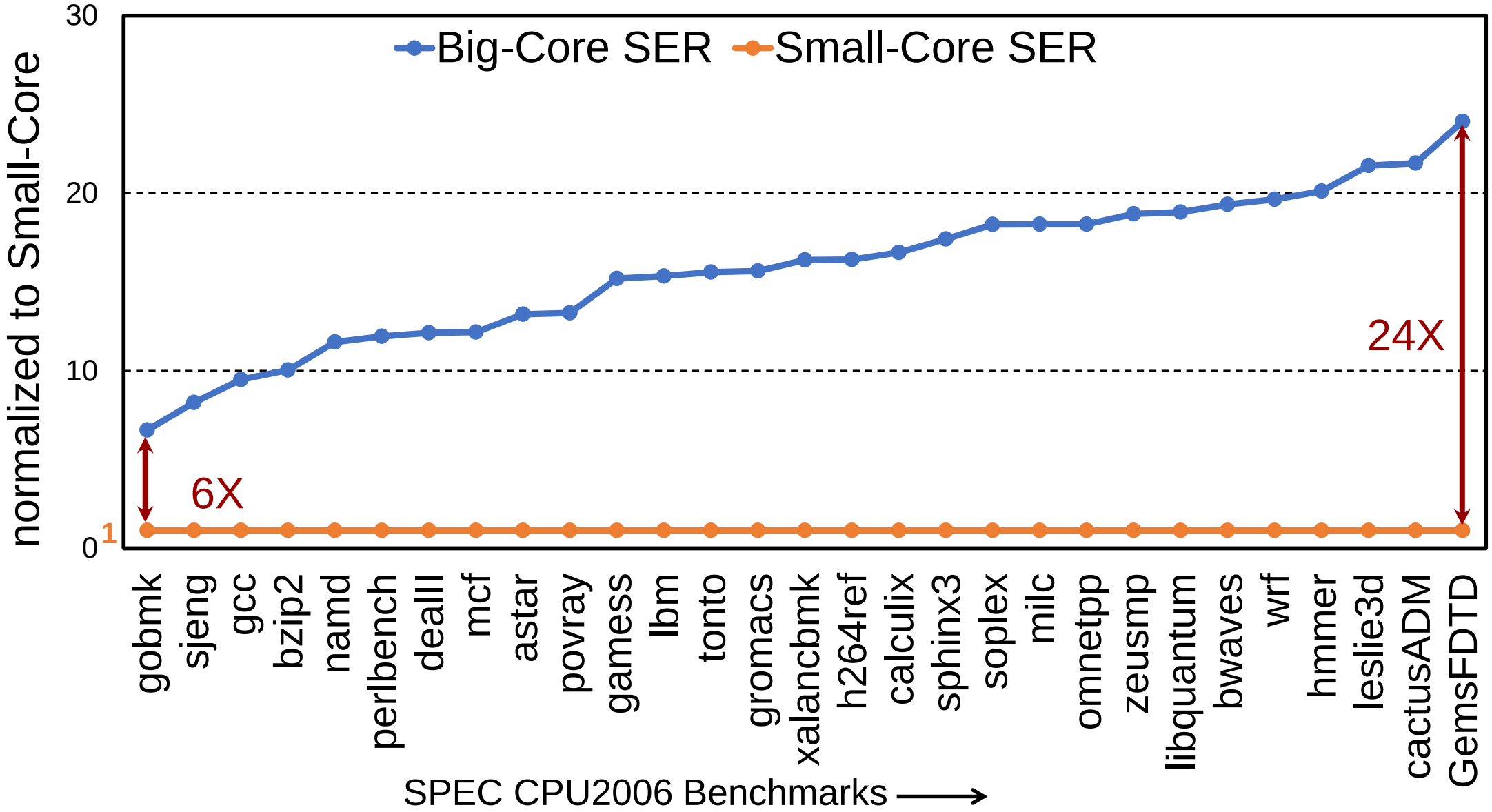
# Vulnerability vs. Core Type



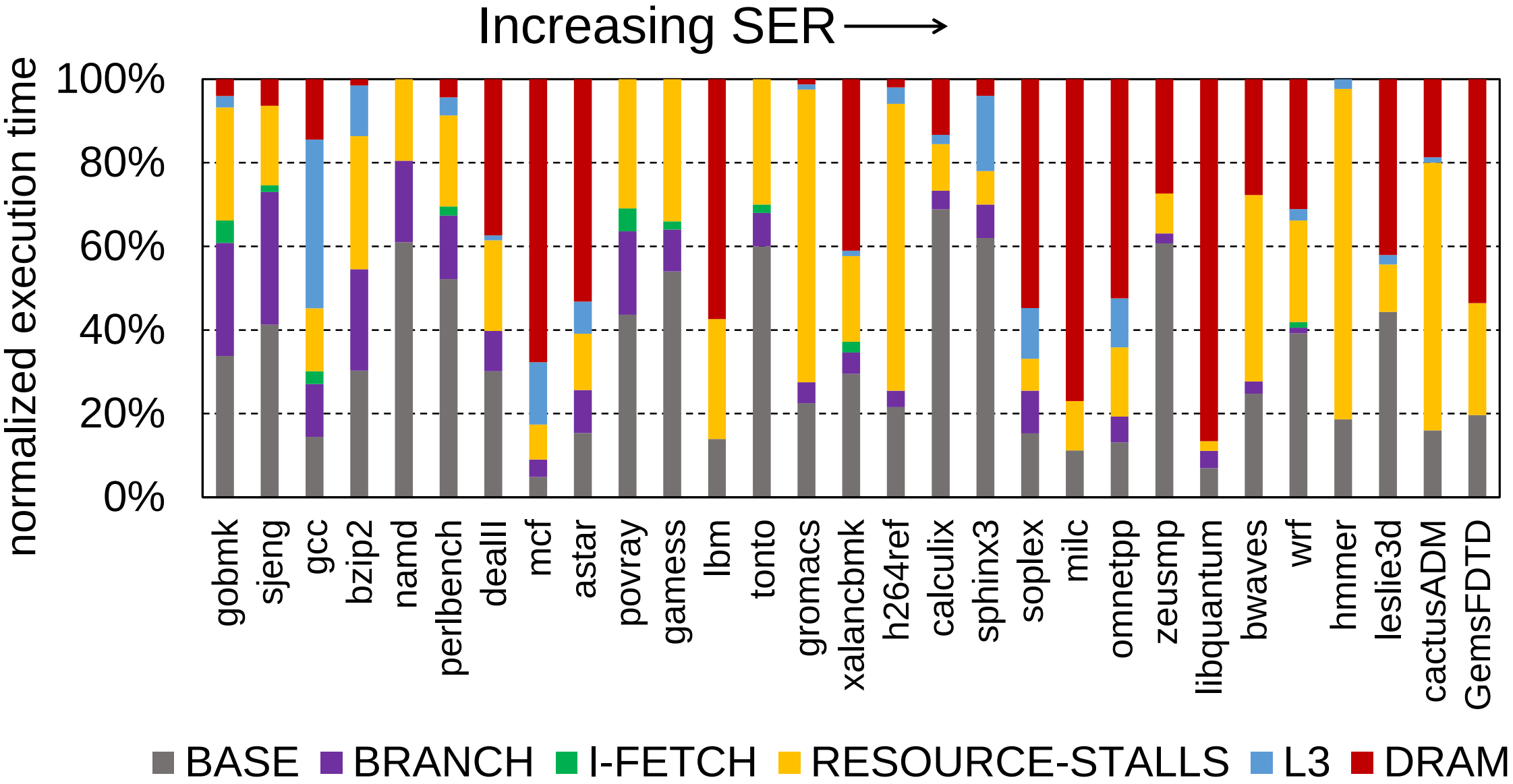
# Vulnerability vs. Core Type



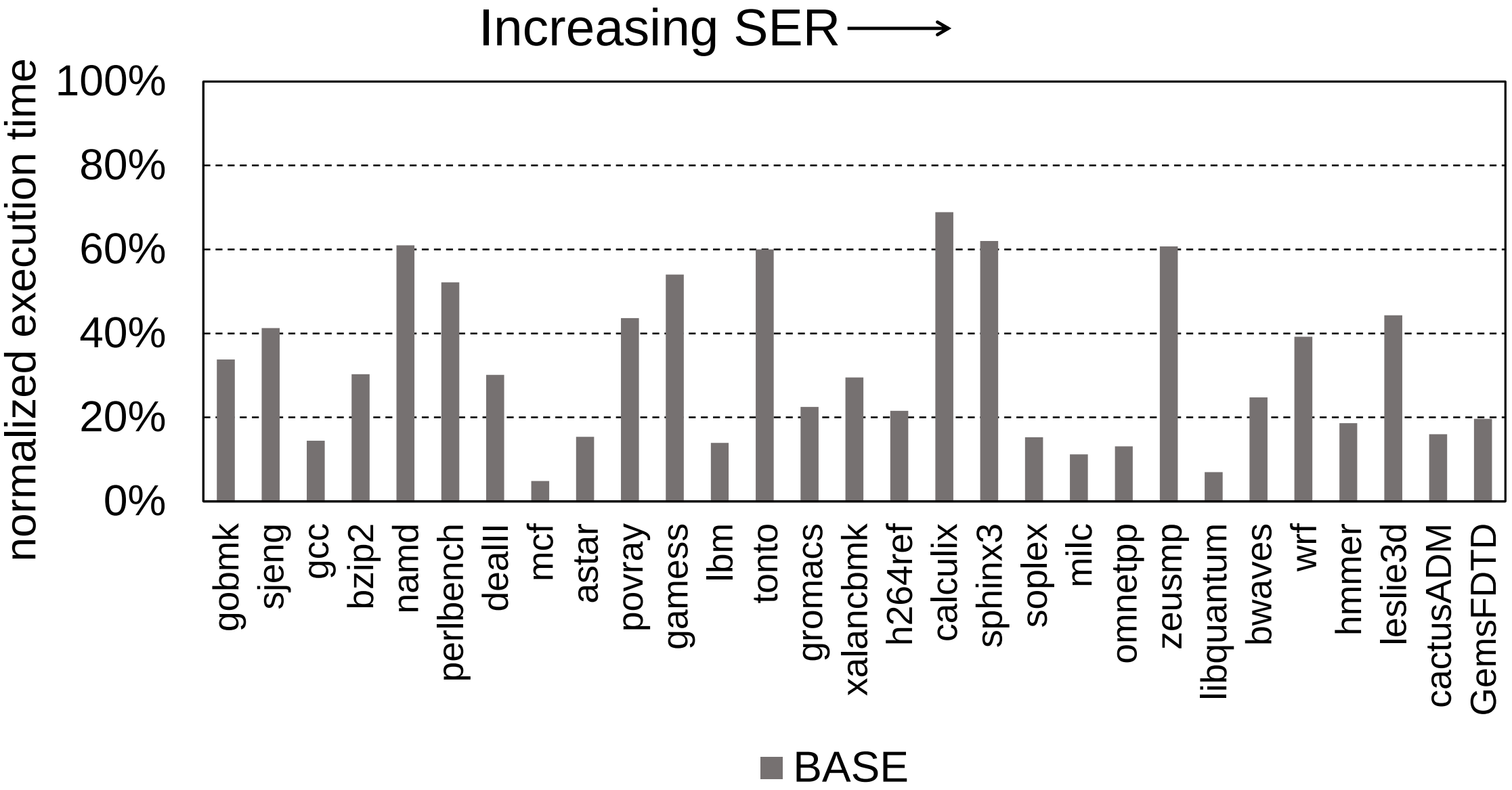
# Vulnerability vs. Core Type



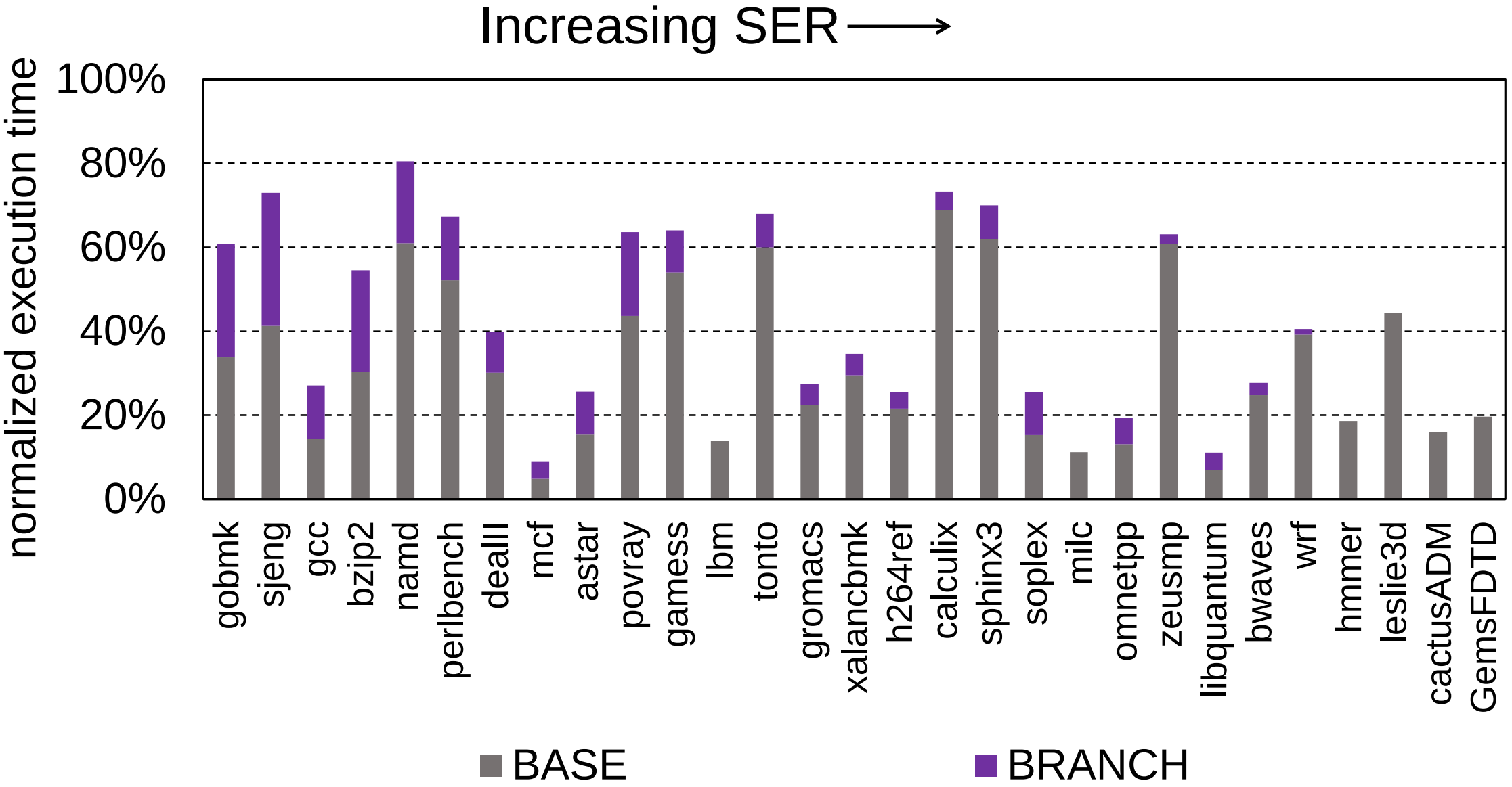
# Vulnerability vs. CPI Stacks



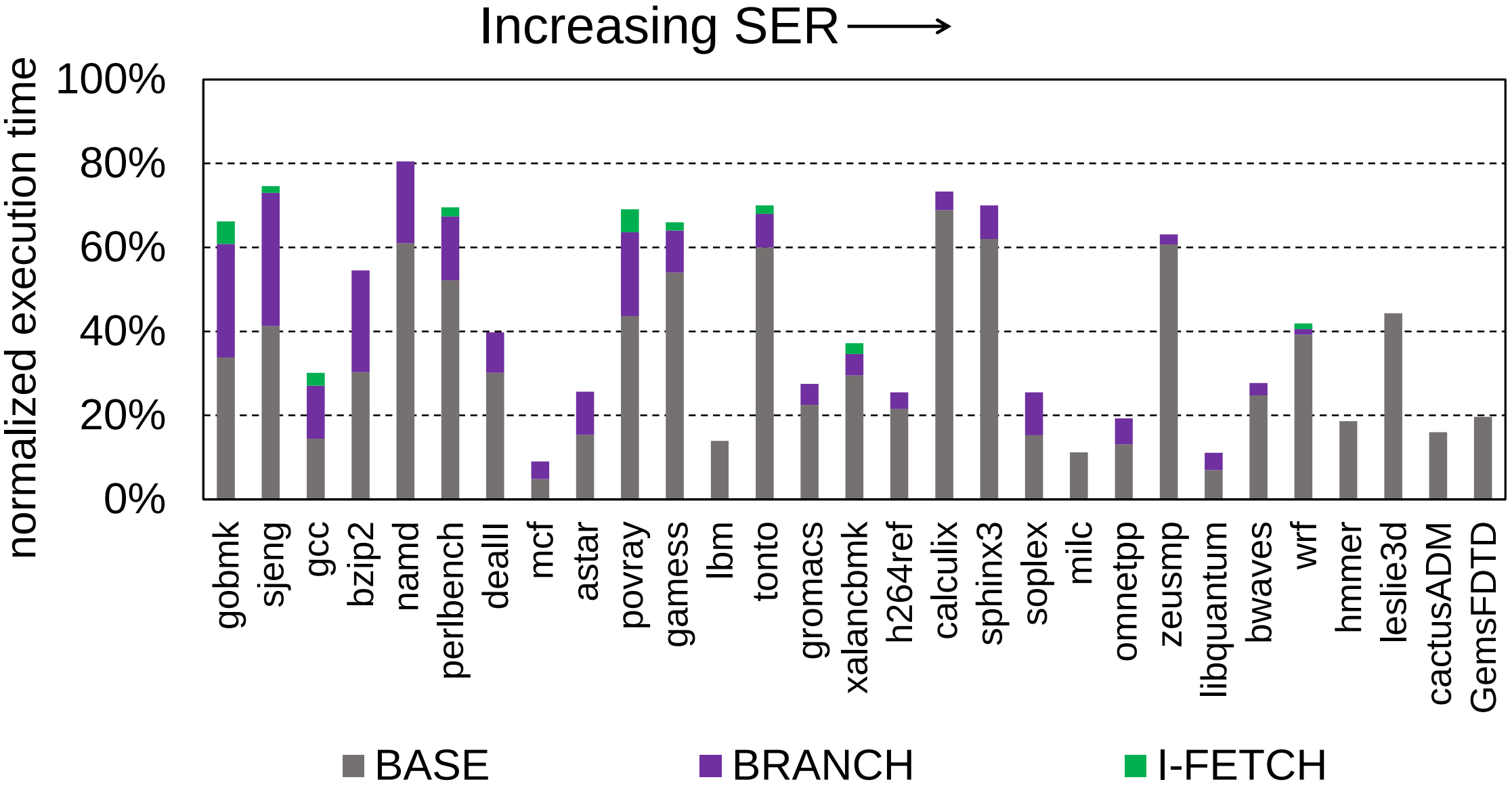
# Vulnerability vs. CPI Stacks



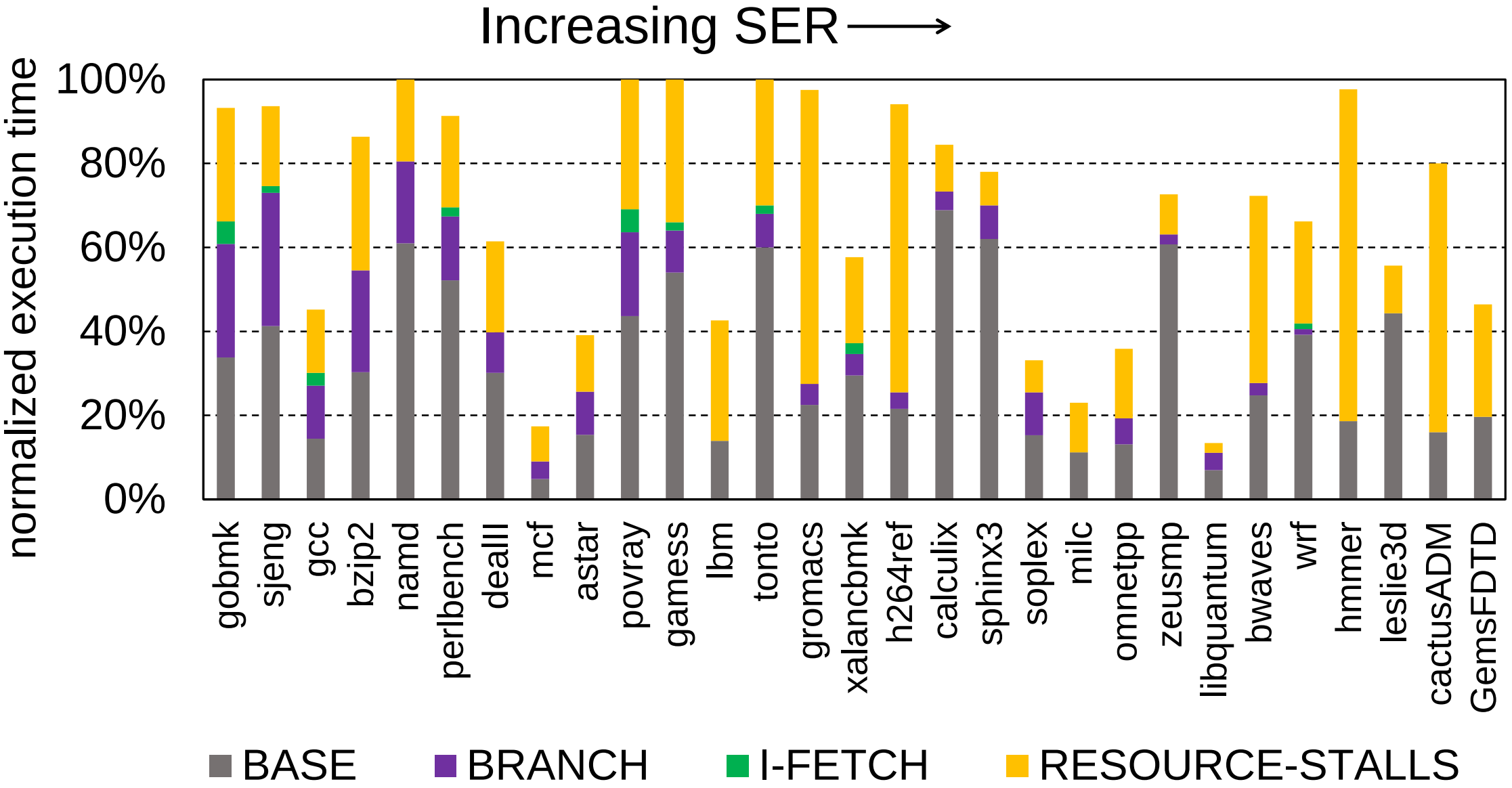
# Vulnerability vs. CPI Stacks



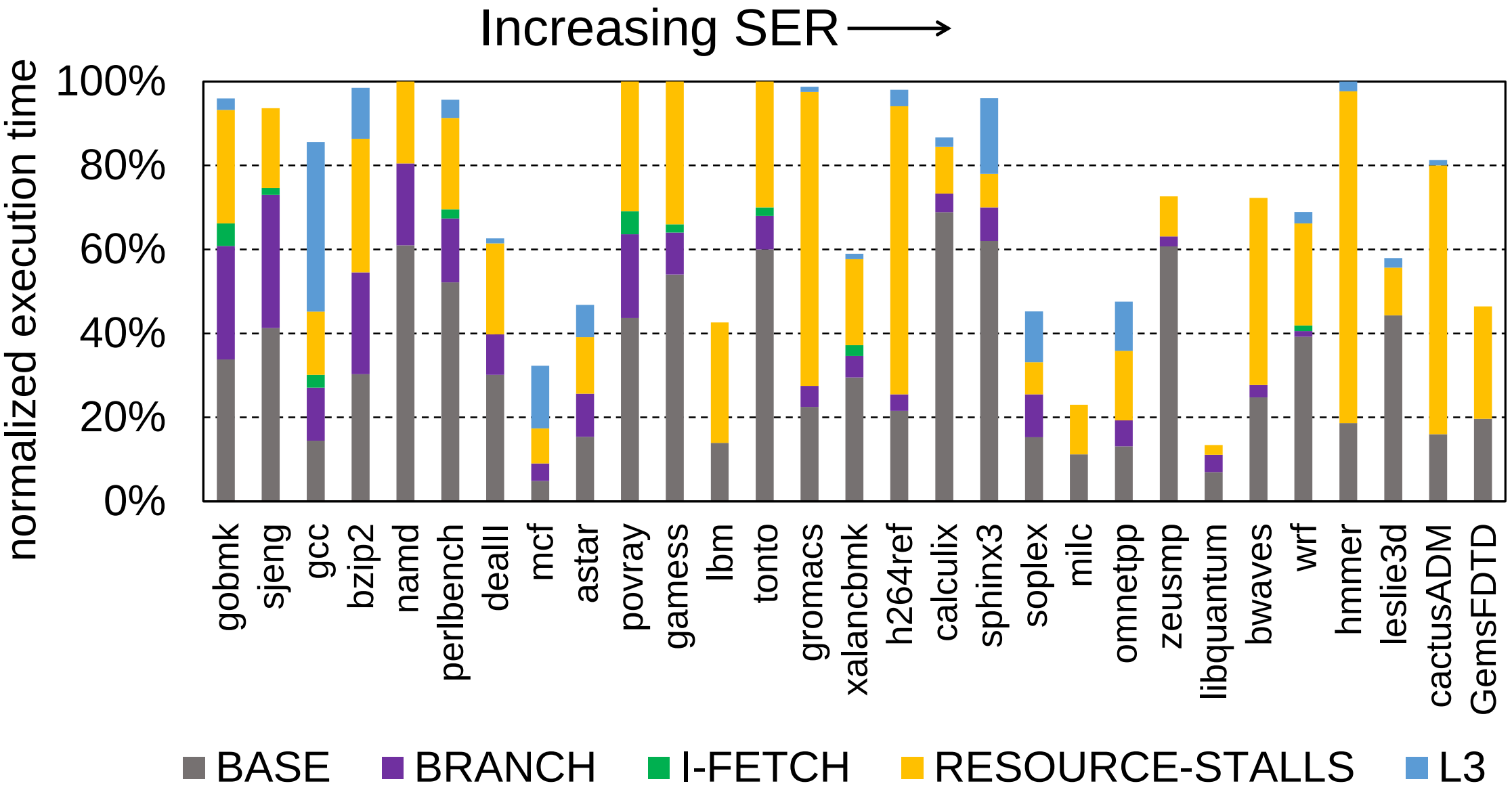
# Vulnerability vs. CPI Stacks



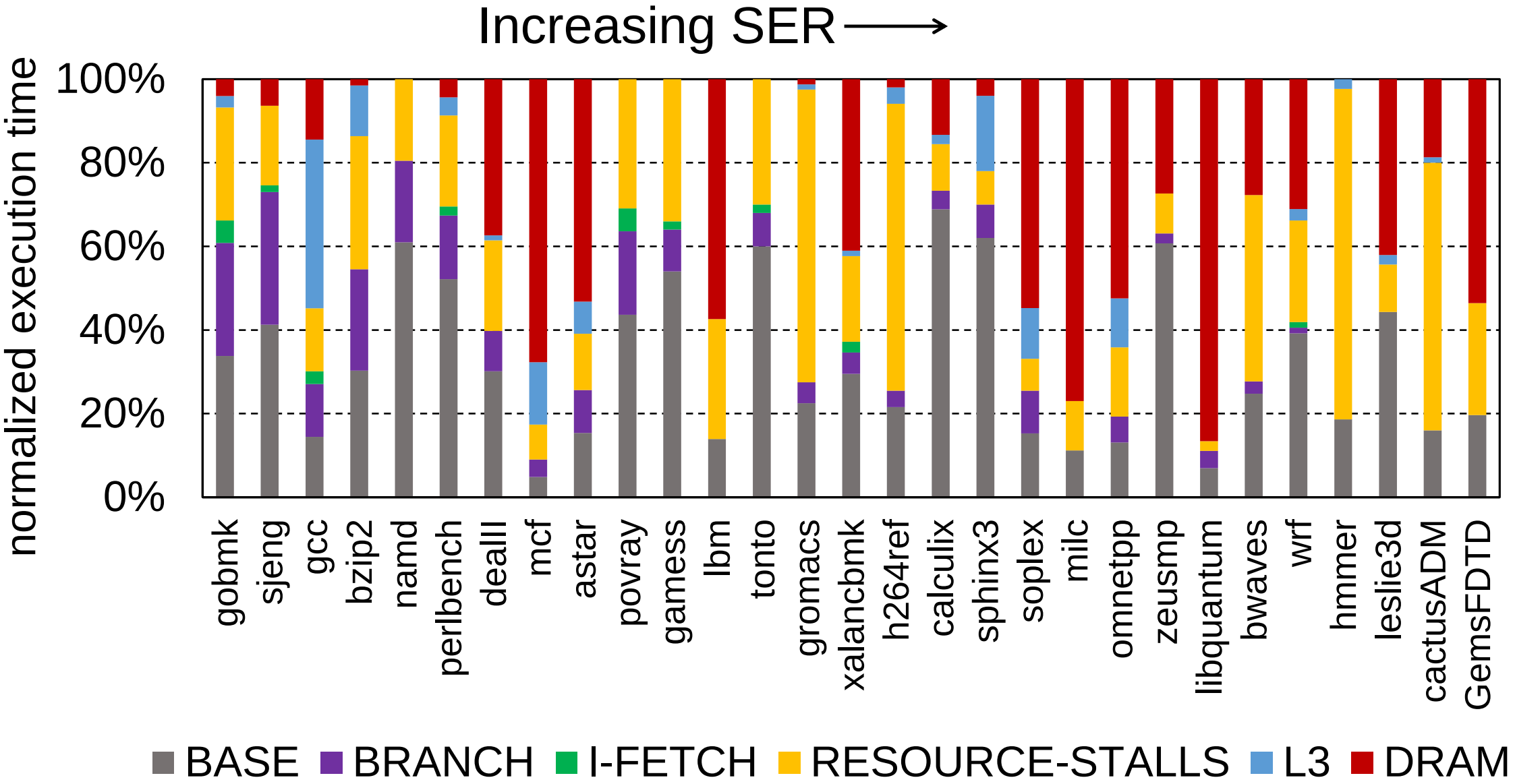
# Vulnerability vs. CPI Stacks



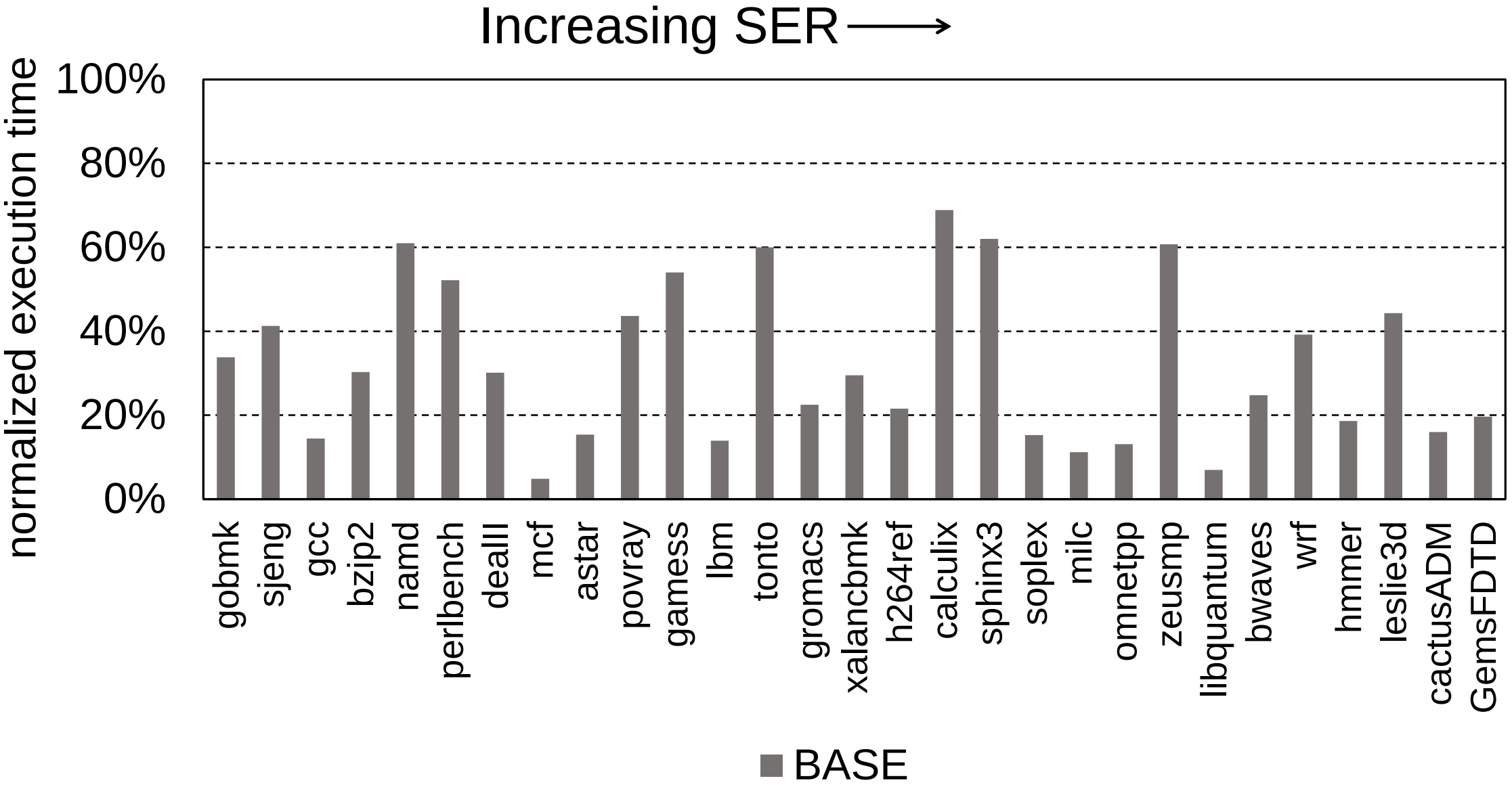
# Vulnerability vs. CPI Stacks



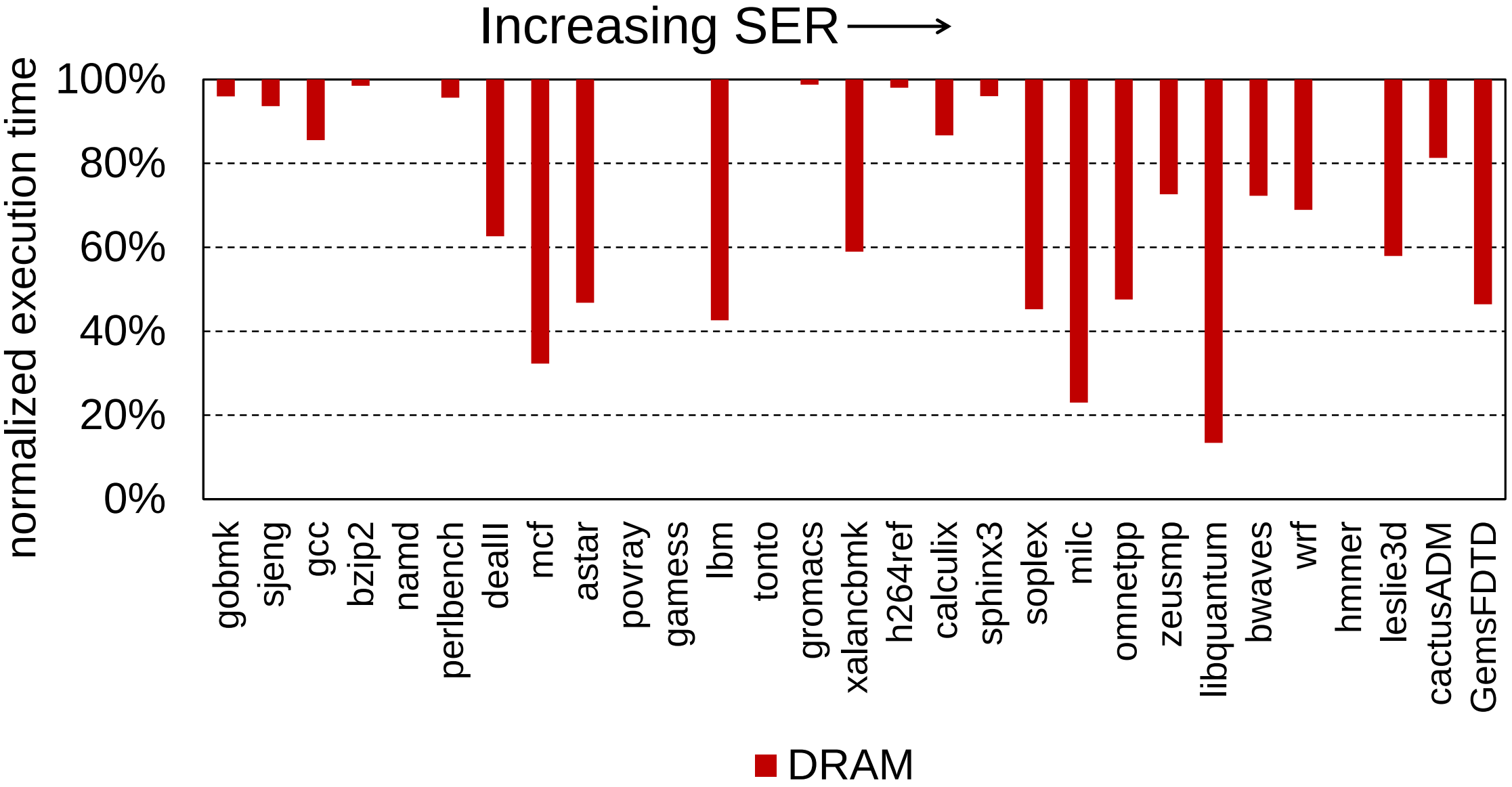
# Vulnerability vs. CPI Stacks



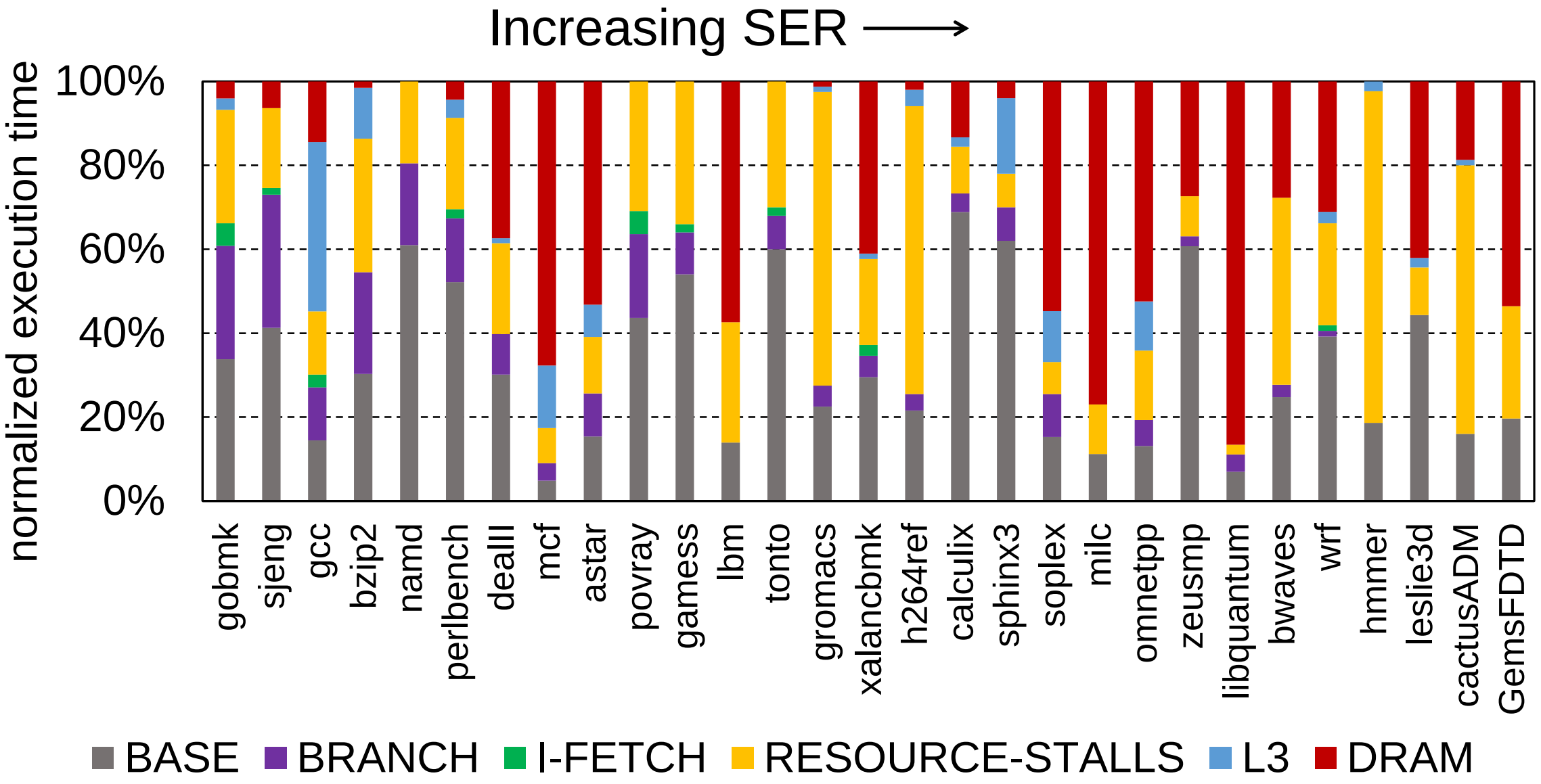
# Vulnerability vs. Compute-Intensity



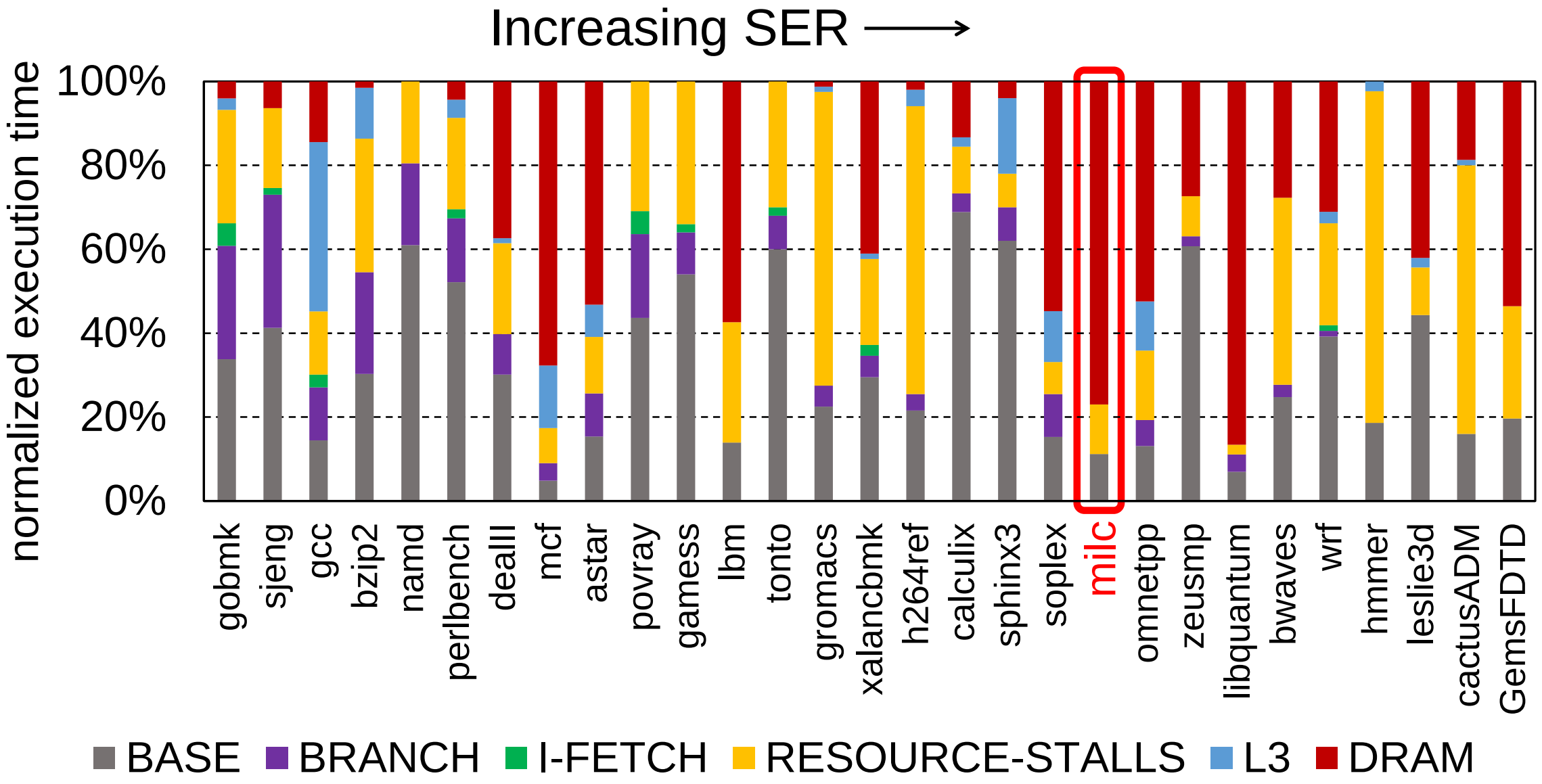
# Vulnerability vs. Memory-Intensity



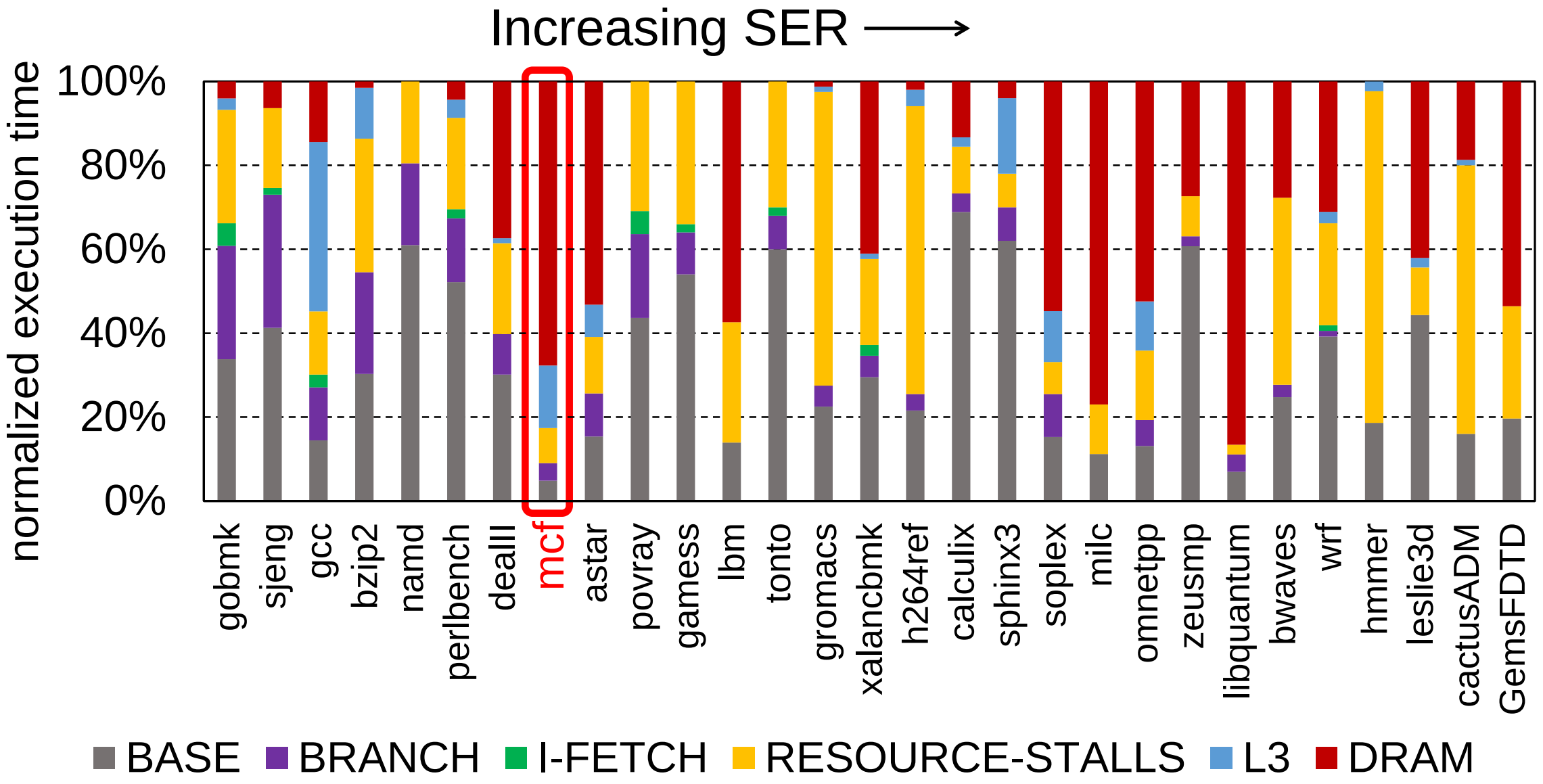
# Vulnerability vs. CPI Stacks



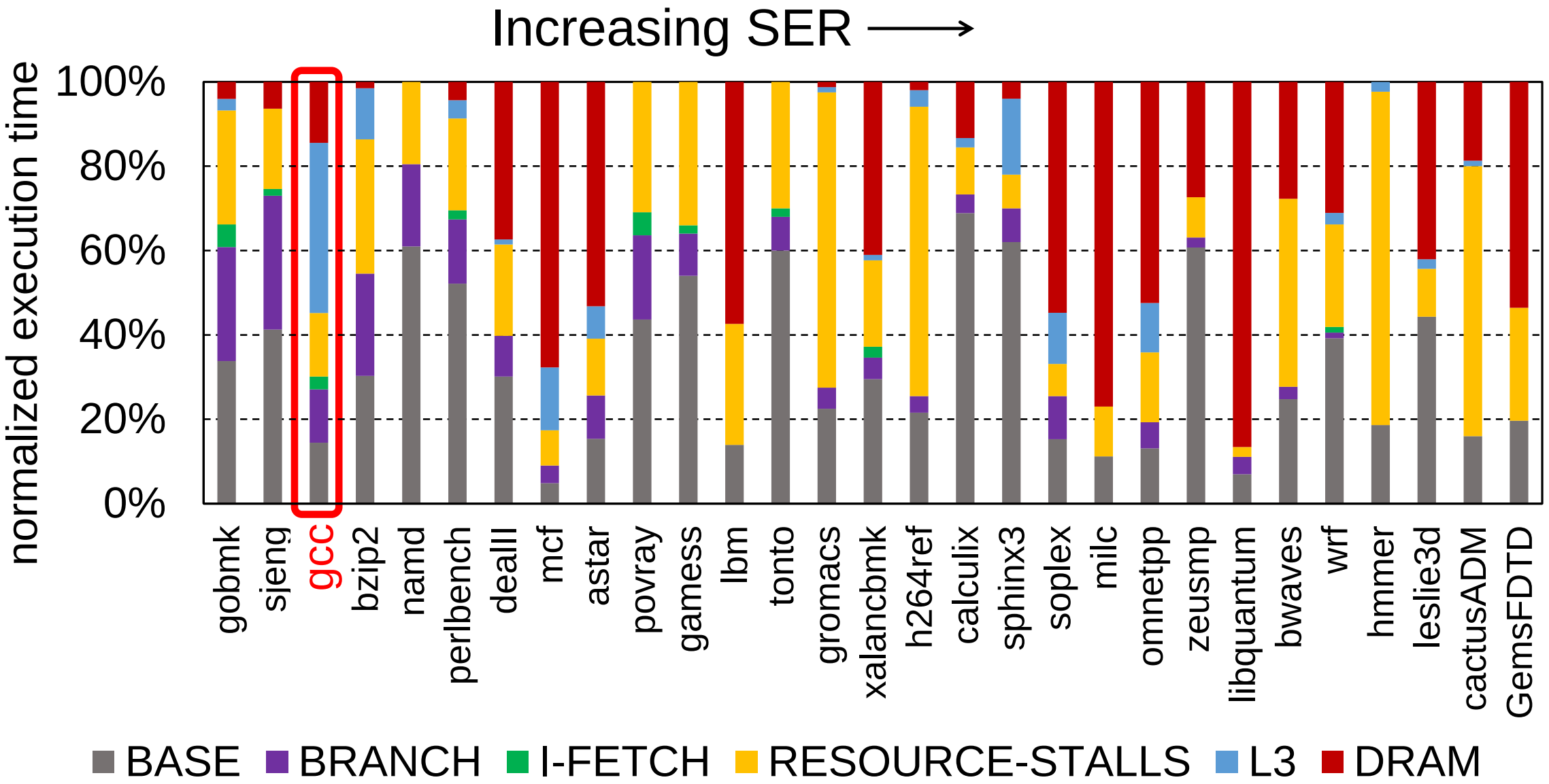
# Vulnerability vs. CPI Stacks



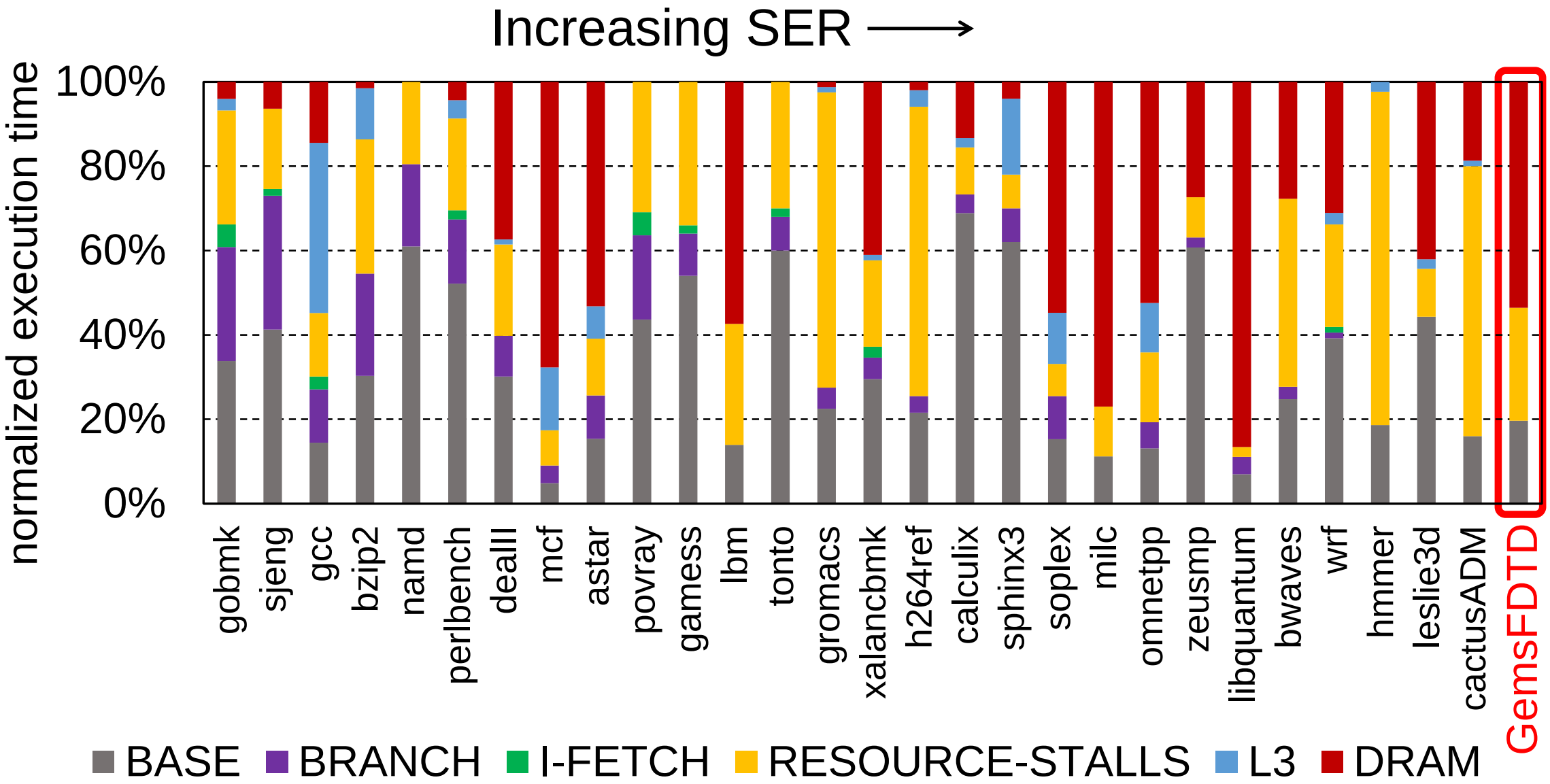
# Vulnerability vs. CPI Stacks



# Vulnerability vs. CPI Stacks



# Vulnerability vs. CPI Stacks



# Reliability vs. Application Type

# Reliability vs. Application Type

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

# Reliability vs. Application Type

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

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

*[Nair et al. ISCA'12]

# Reliability vs. Application Type

- No simple workload characteristic
  -- For example, memory-intensity or compute-intensity
- Predicting SER is not straightforward*
  -- Complex interaction of various workload characteristics
- A dynamic mechanism required
  -- To monitor reliability on either core type

*[Nair et al. ISCA'12]

# Reliability vs. Application Type

- No simple workload characteristic
  -- For example, memory-intensity or compute-intensity
- Predicting S
  -- Comple aracteristics
- A dynamic
  -- To monitor reliability on either core type

**Reliability-Aware Scheduling**

*[Nair et al. ISCA'12]

# Reliability Metric for Multiprogram Workloads

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

**A** Time = 1

SER = 4

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

| | | |
|---|---|---|
| **A** | Time = 1 | SER = 4 |
| **B** | Time = 1 | SER = 1 |

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

| | | |
|---|---|---|
| **A** | Time = 1 | SER = 4 |
| **B** | Time = 1 | SER = 1 |

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

| | | |
|---|---|---|
| **A** | Time = 1 | SER = 4 |

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 2 | SER = 1 |

# Reliability Metric for Multiprogram Workloads

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| | | + |
| B | Time = 2 | SER = 1 |

Total SER = 5

# Reliability Metric for Multiprogram Workloads

Adding SER

# Reliability Metric for Multiprogram Workloads

Adding SER → More weight to high-SER applications

# Reliability Metric for Multiprogram Workloads

Adding SER

- More weight to high-SER applications
- Ignores slowdown

# Reliability Metric for Multiprogram Workloads

Adding SER

- More weight to high-SER applications
- Ignores slowdown

Problem

Ignores the impact of performance on SER

# Reliability Metric for Multiprogram Workloads

Adding SER

- More weight to high-SER applications
- Ignores slowdown

Problem

Ignores the impact of performance on SER

Solution

New metric: System Soft Error Rate (SSER)

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| | | + |
| B | Time = 2 | SER = 1 |

Total SER = 5

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | |
|---|---|---|---|
| A | Time = 1 | SER = 4 | $1 \times 4$ |
| | | + | |
| B | Time = 2 | SER = 1 | |

Total SER = 5

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | |
|---|---|---|---|
| A | Time = 1 | SER = 4 | 1 × 4 Weighted SER = 4 |
| | | + | |
| B | Time = 2 | SER = 1 | |

Total SER = 5

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | | |
|---|---|---|---|---|
| A | Time = 1 | SER = 4 | 1 × 4 | Weighted SER = 4 |
| | | + | | |
| B | Time = 2 | SER = 1 | **2** × 1 | |

Total SER = 5

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | | |
|---|---|---|---|---|
| A | Time = 1 | SER = 4 | 1 × 4 | Weighted SER = 4 |
| | | + | | |
| B | Time = 2 | SER = 1 | **2** × 1 | Weighted SER = 2 |

Total SER = 5

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | |
|---|---|---|---|
| A | Time = 1 | SER = 4 | Weighted SER = 4 |
| | | + | + |
| B | Time = 2 | SER = 1 | Weighted SER = 2 |

Total SER = 5

SSER = 6

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | |
|---|---|---|---|
| A | Time = 1 | SER = 4 | Weighted SER = 4 |
| | | + | + |
| B | Time = 2 | SER = 1 | Weighted SER = 2 |

Total SER = 5 ✗

SSER = 6

# System Soft Error Rate (SSER) Correctly Accounts for the Slowdown

In isolation on one core:

| | | |
|---|---|---|
| A | Time = 1 | SER = 4 |
| B | Time = 1 | SER = 1 |

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

| | | | |
|---|---|---|---|
| A | Time = 1 | SER = 4 | Weighted SER = 4 |
| | | + | + |
| B | Time = 2 | SER = 1 | Weighted SER = 2 |

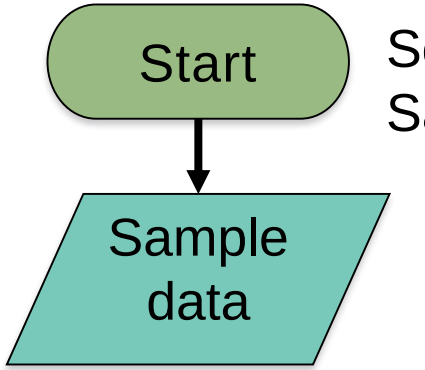
Total SER = 5 ✗

SSER = 6 ✓

# Scheduling Algorithm

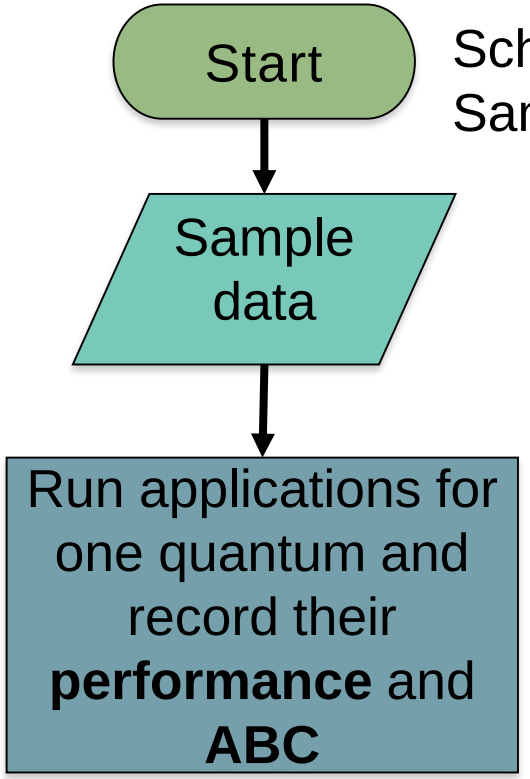
# Scheduling Algorithm

# Scheduling Algorithm



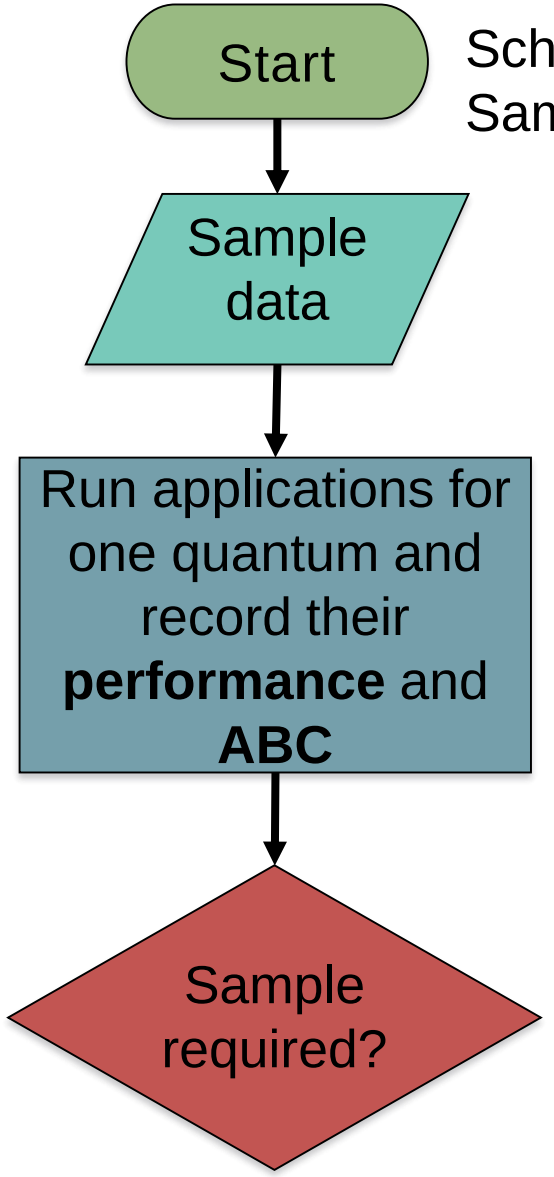

Scheduler quantum = 1ms
Sampling for 0.1ms

# Scheduling Algorithm



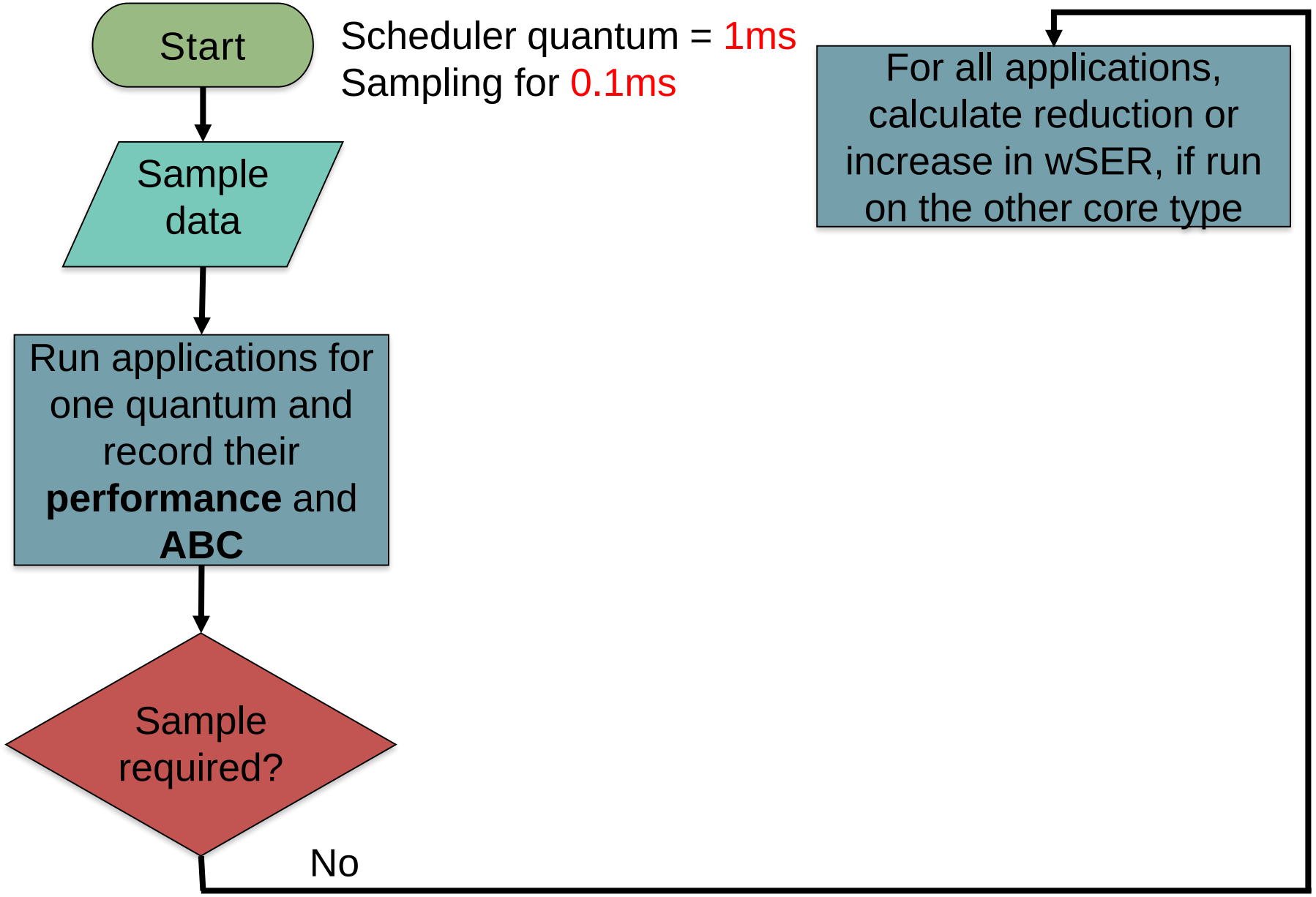

Scheduler quantum = 1ms
Sampling for 0.1ms

# Scheduling Algorithm

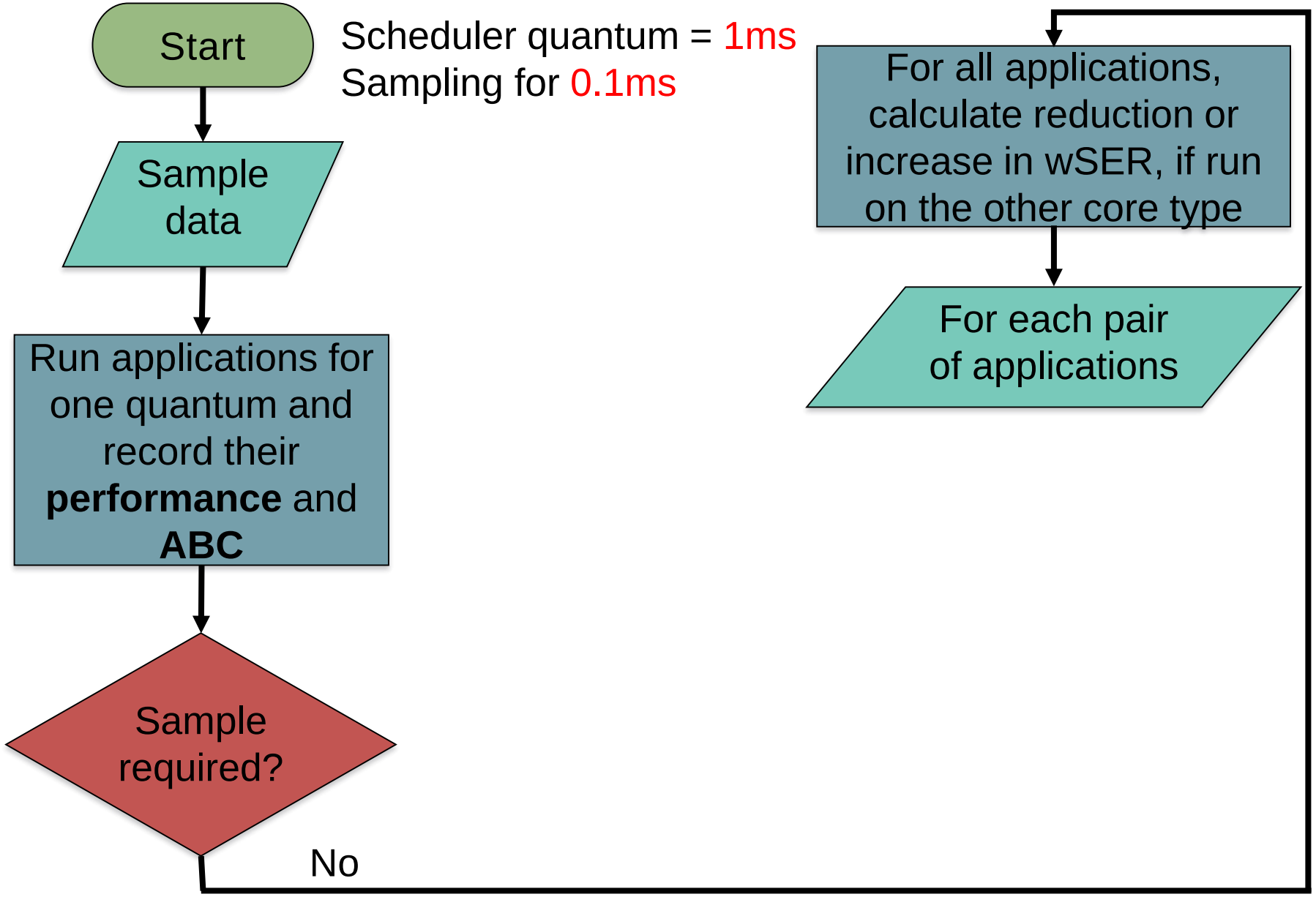



Scheduler quantum = 1ms
Sampling for 0.1ms

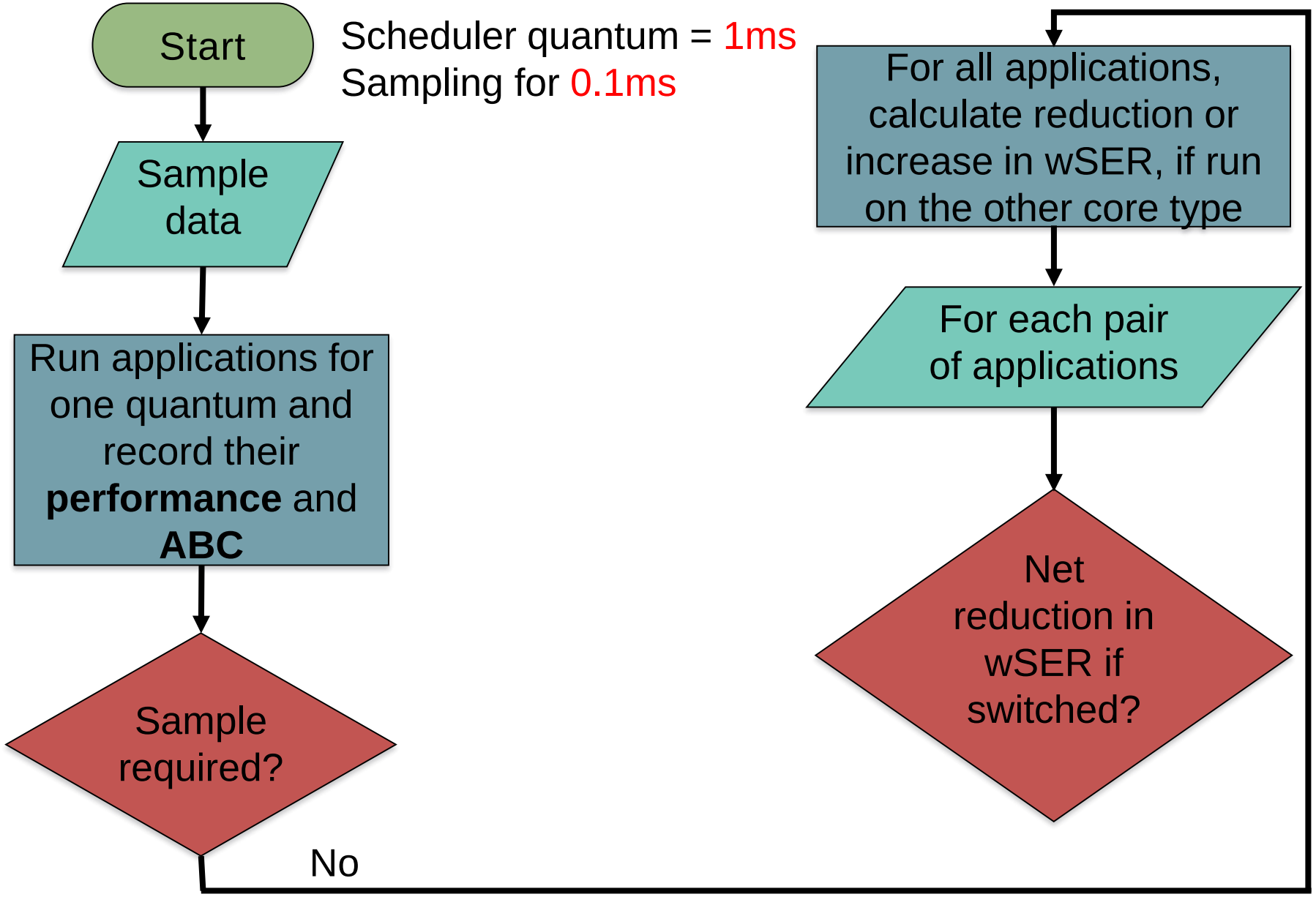
# Scheduling Algorithm



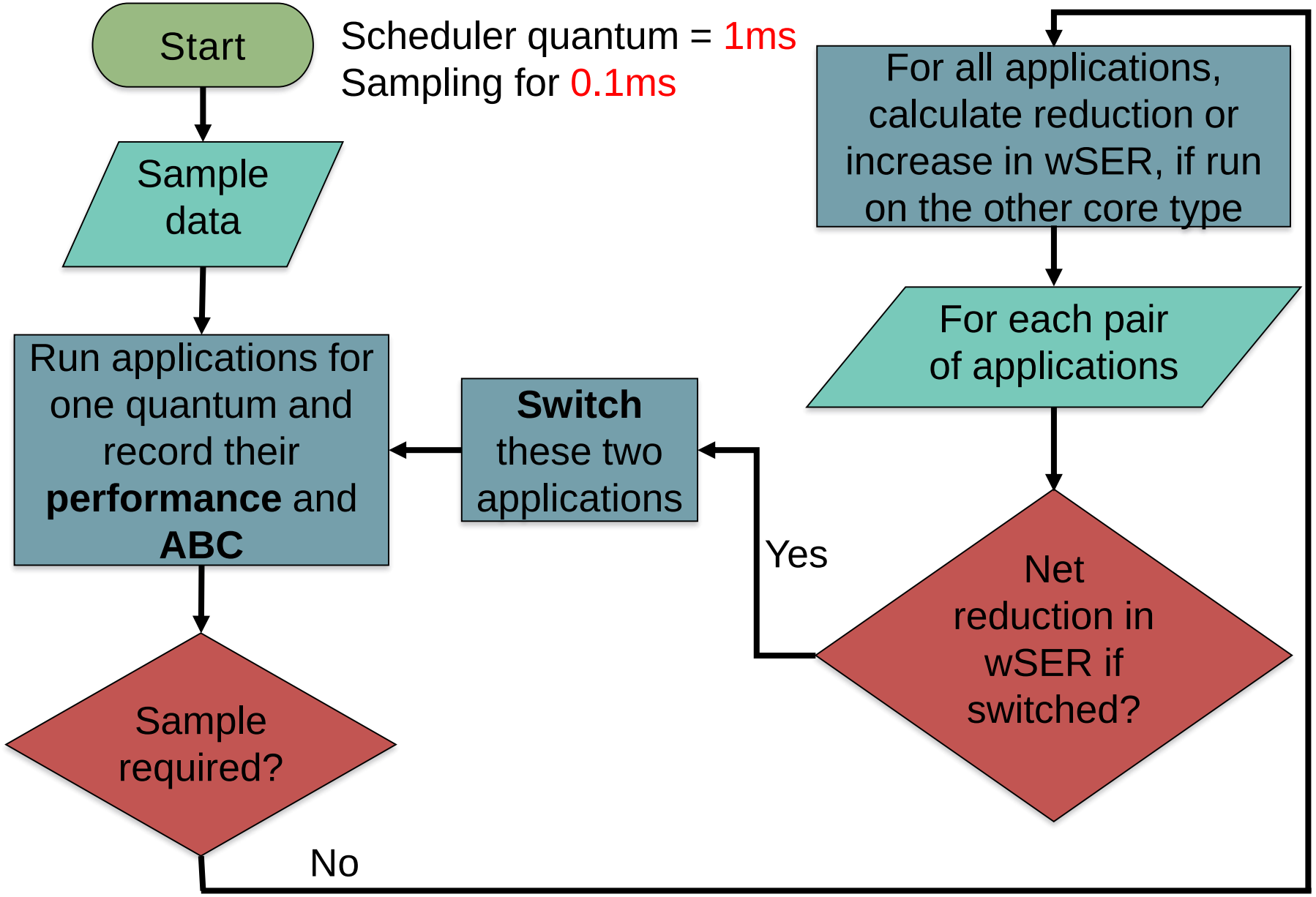
# Scheduling Algorithm



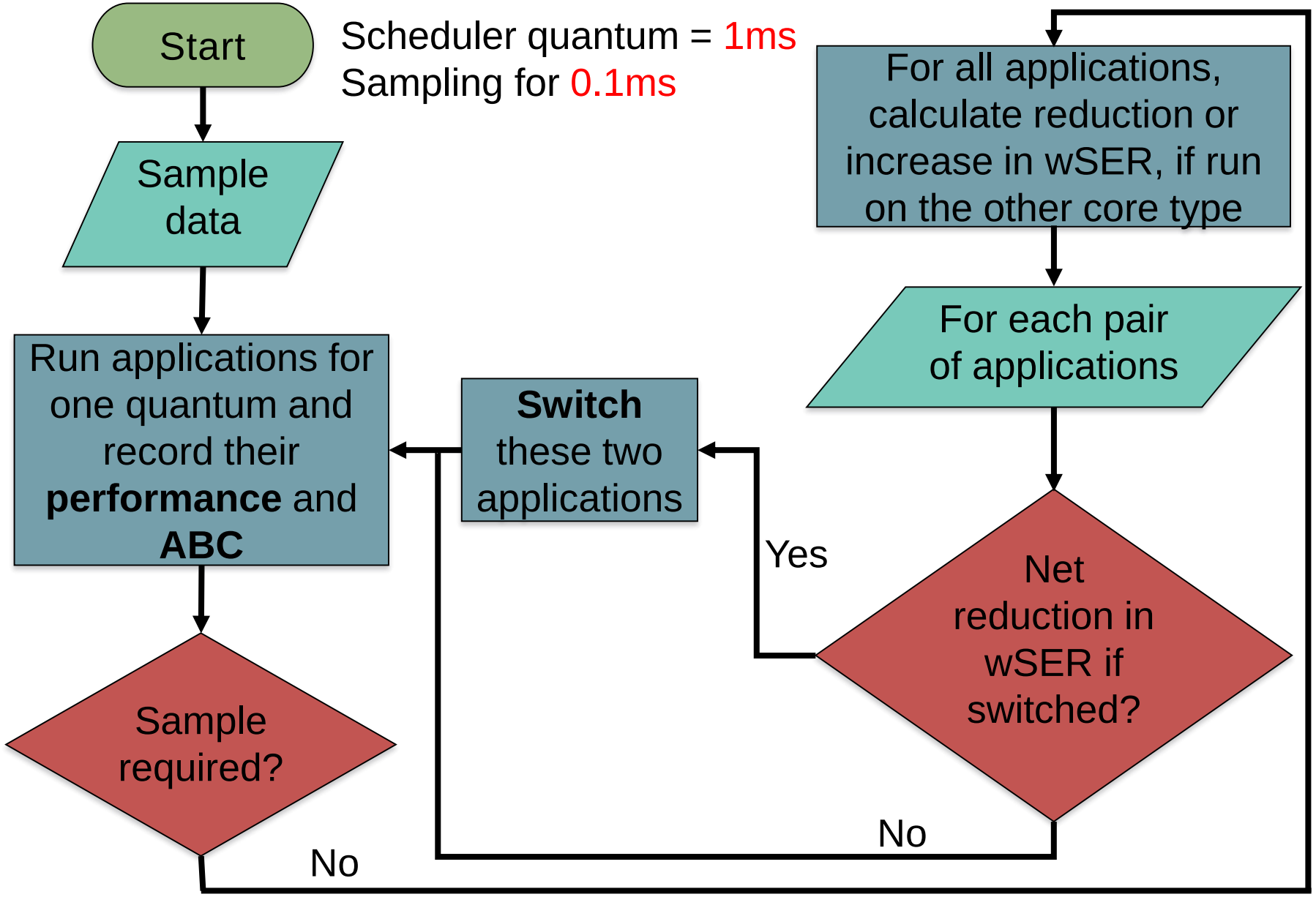
# Scheduling Algorithm



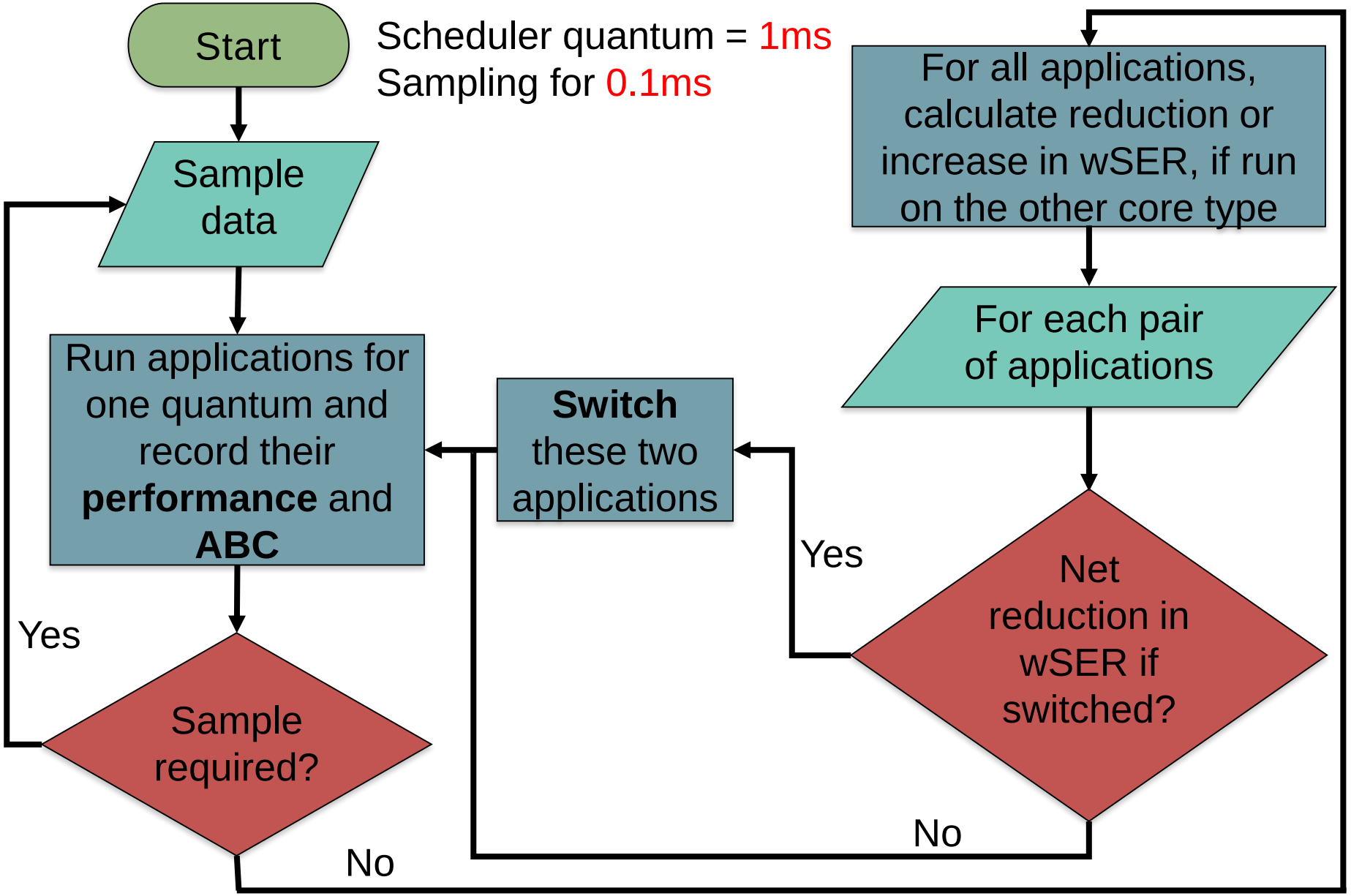
# Scheduling Algorithm



# Scheduling Algorithm



# Scheduling Algorithm



# Experimental Setup

# Experimental Setup

Sniper 6.0

-- Augmented with ACE Bit Counters

# Experimental Setup

Sniper 6.0

-- Augmented with ACE Bit Counters

SPEC CPU2006

-- 1B instruction SimPoints

-- Based on big-core SER

- ✓ L→ Low SER
- ✓ M → Medium SER
- ✓ H → High SER

# Experimental Setup

- ✓ L→ Low SER
- ✓ M → Medium SER
- ✓ H → High SER

## Sniper 6.0

-- Augmented with ACE Bit Counters

## SPEC CPU2006

-- 1B instruction SimPoints

-- Based on big-core SER

## 4-program workloads

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

# Experimental Setup

Sniper 6.0

-- Augmented with ACE Bit Counters

SPEC CPU2006

-- 1B instruction SimPoints

-- Based on big-core SER

4-program workloads

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

✓ L→ Low SER

✓ M → Medium SER

✓ H → High SER

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

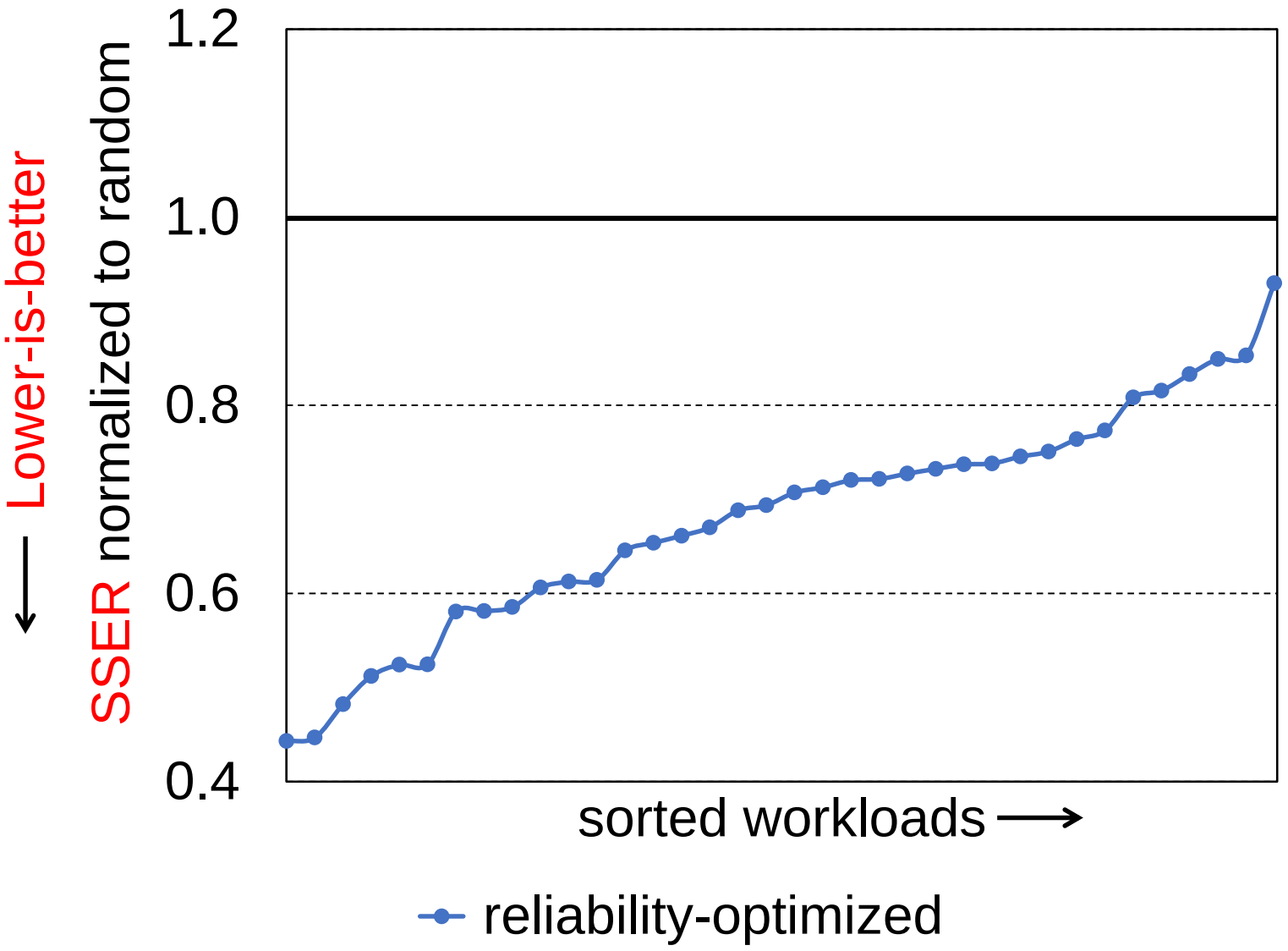
Performance-Optimized Scheduler

-- Optimizes System Throughput (STP)

Random Scheduler

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

# 2B2S Results – Reliability

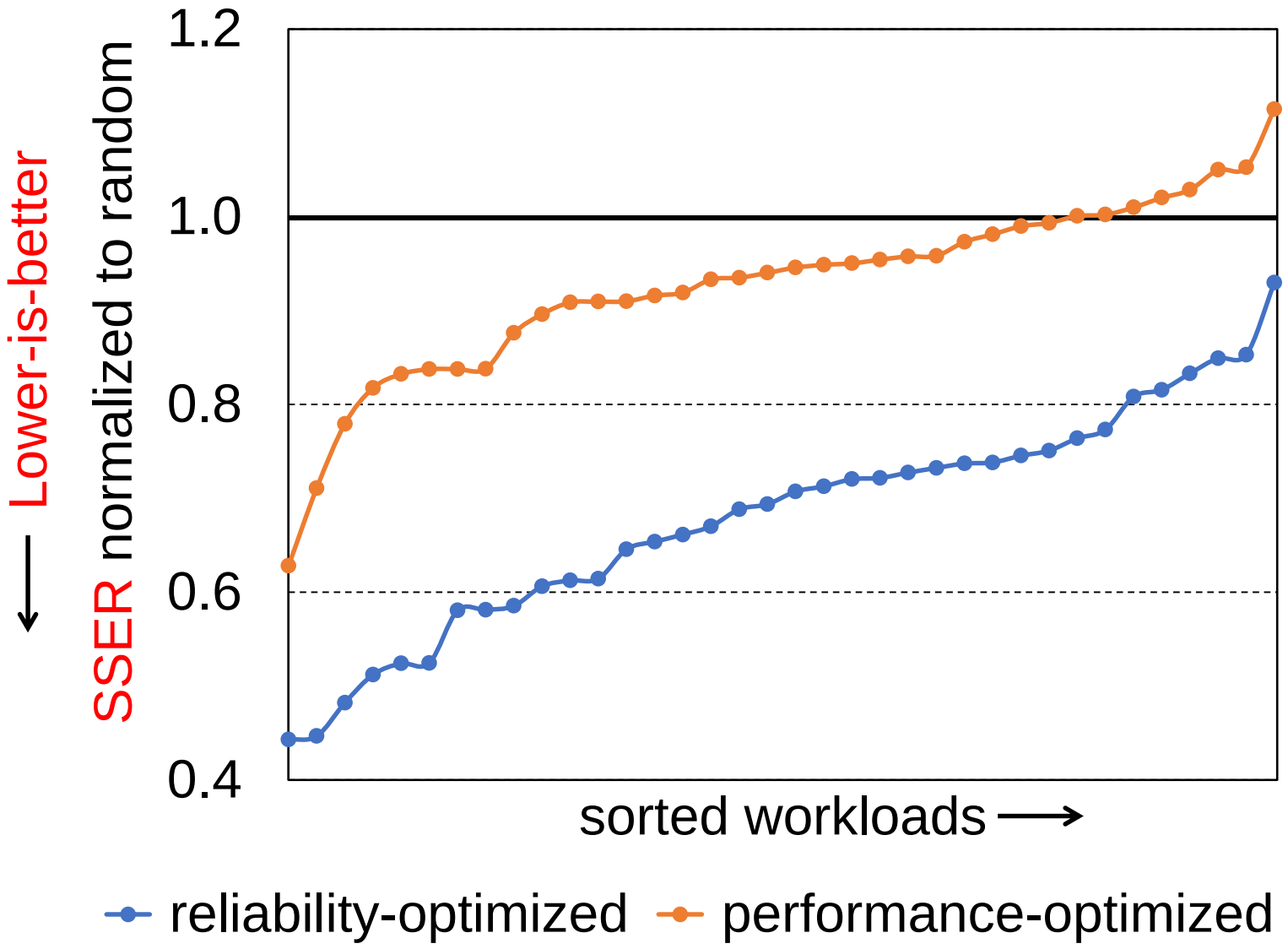



Compared to random

Reliability-optimized
-- improves SSER by 32%

-- up to 55.6%

# 2B2S Results – Reliability

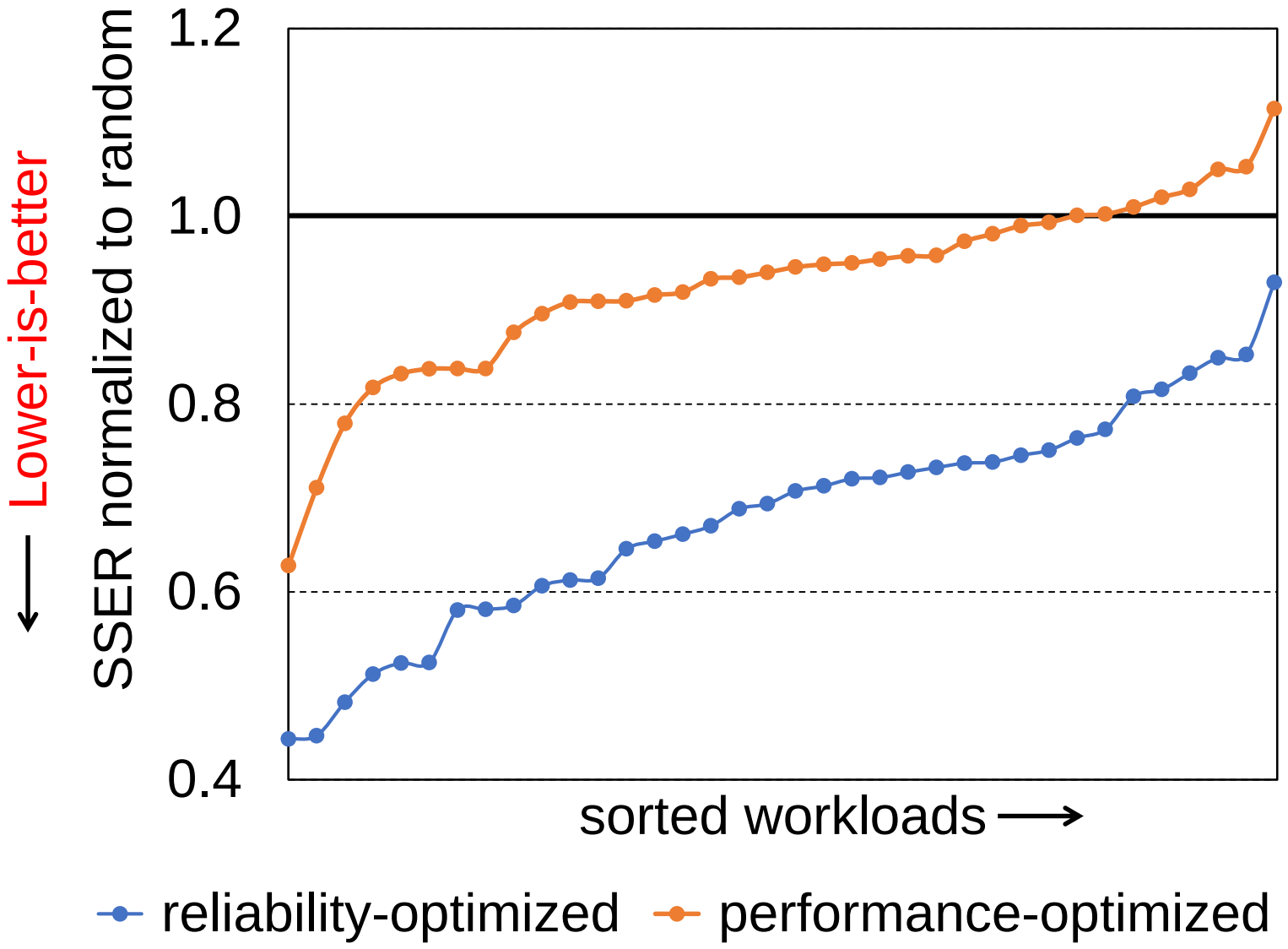



Compared to random

Reliability-optimized
-- improves SSER by 32%
-- up to 55.6%

Performance-optimized
-- improves SSER by 7.3%

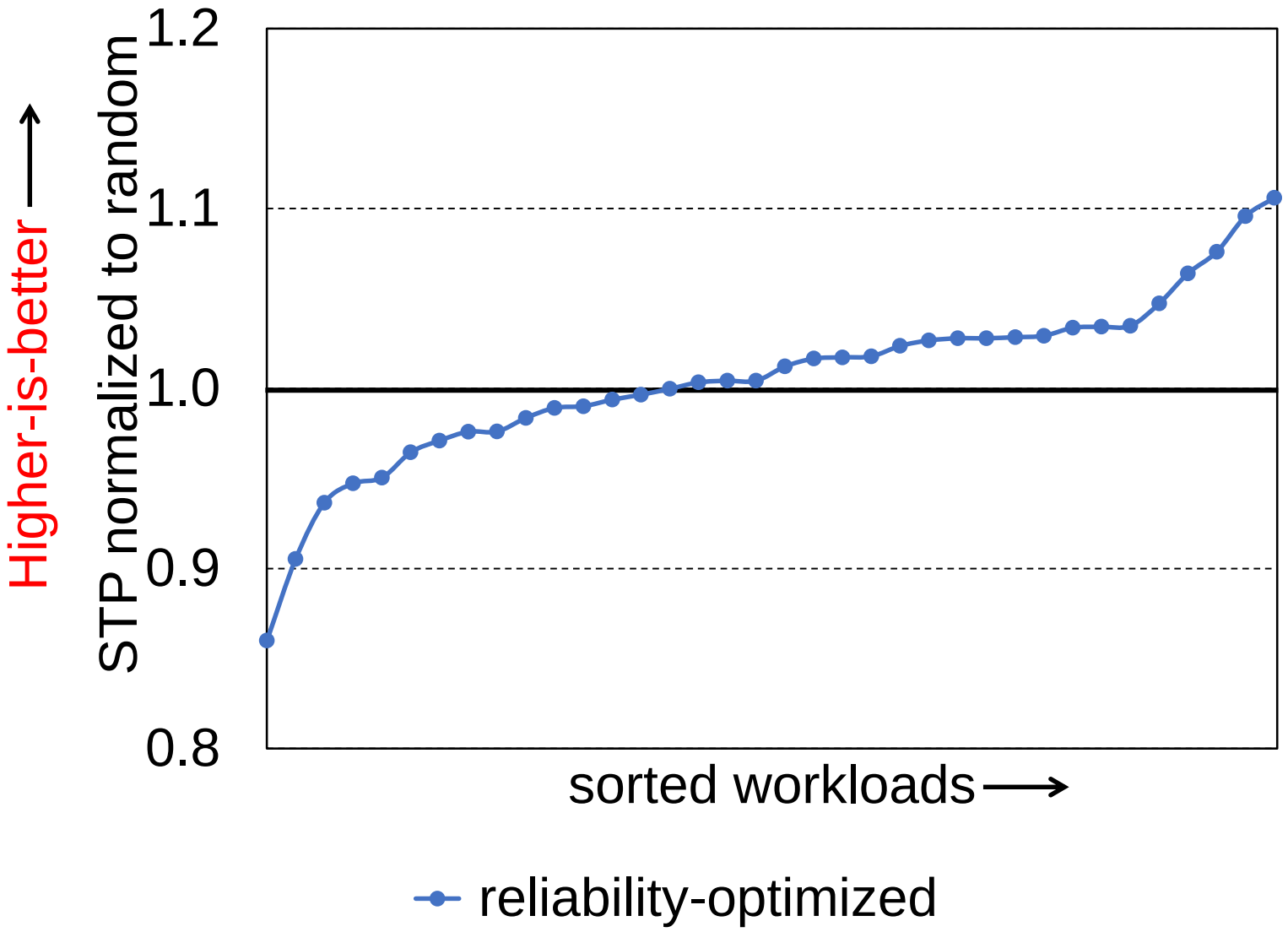
# 2B2S Results – Reliability




Compared to performance-optimized

Reliability-optimized
-- improves SSER by 25.4%
-- up to 60.2%

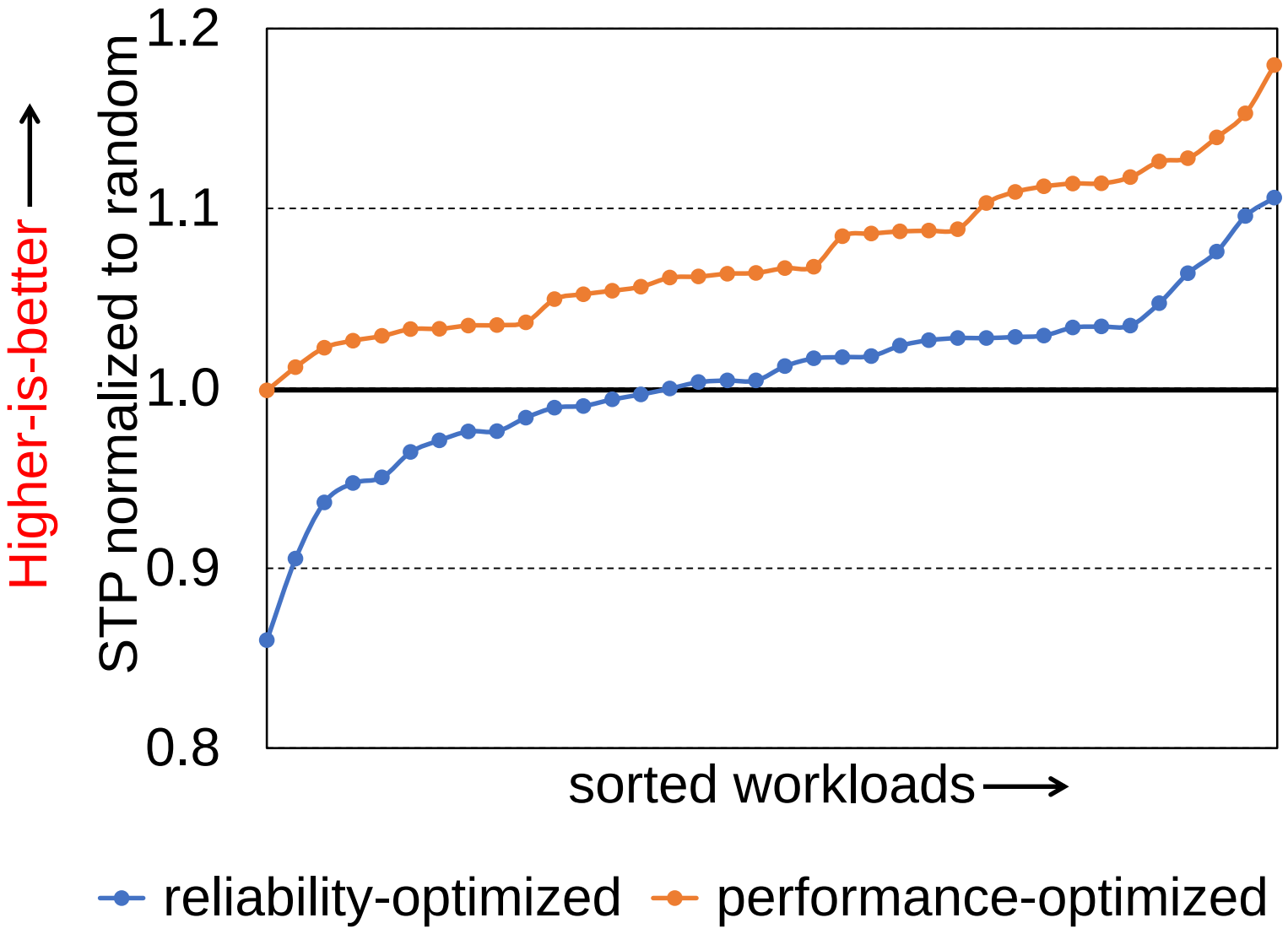
# 2B2S Results – Performance




Compared to random

Reliability-optimized
-- does not degrade STP

# 2B2S Results – Performance



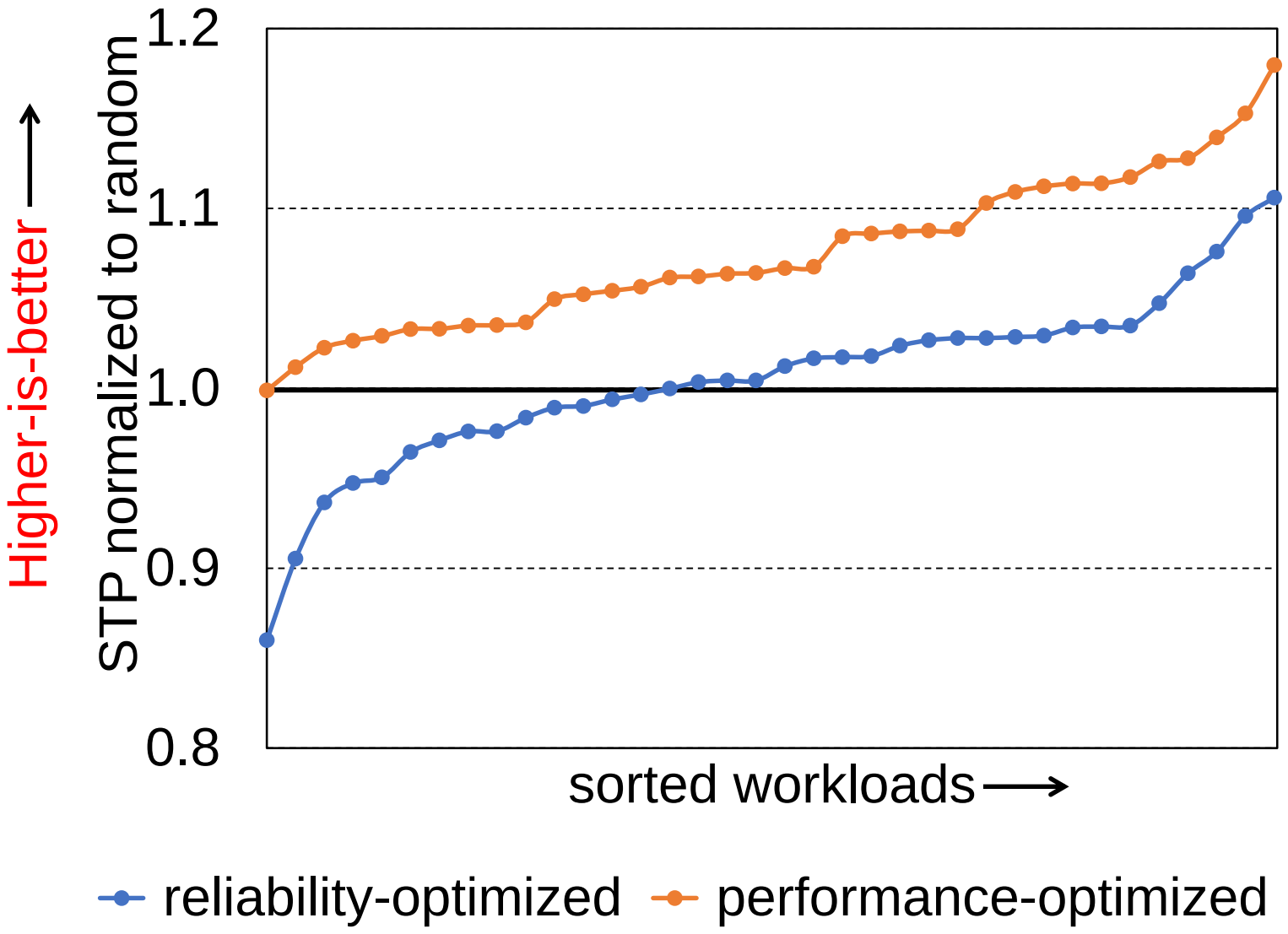

Compared to random

Reliability-optimized
-- does not degrade STP

Performance-optimized
-- improves STP by 6.3%

-- up to 18.7%

# 2B2S Results – Performance

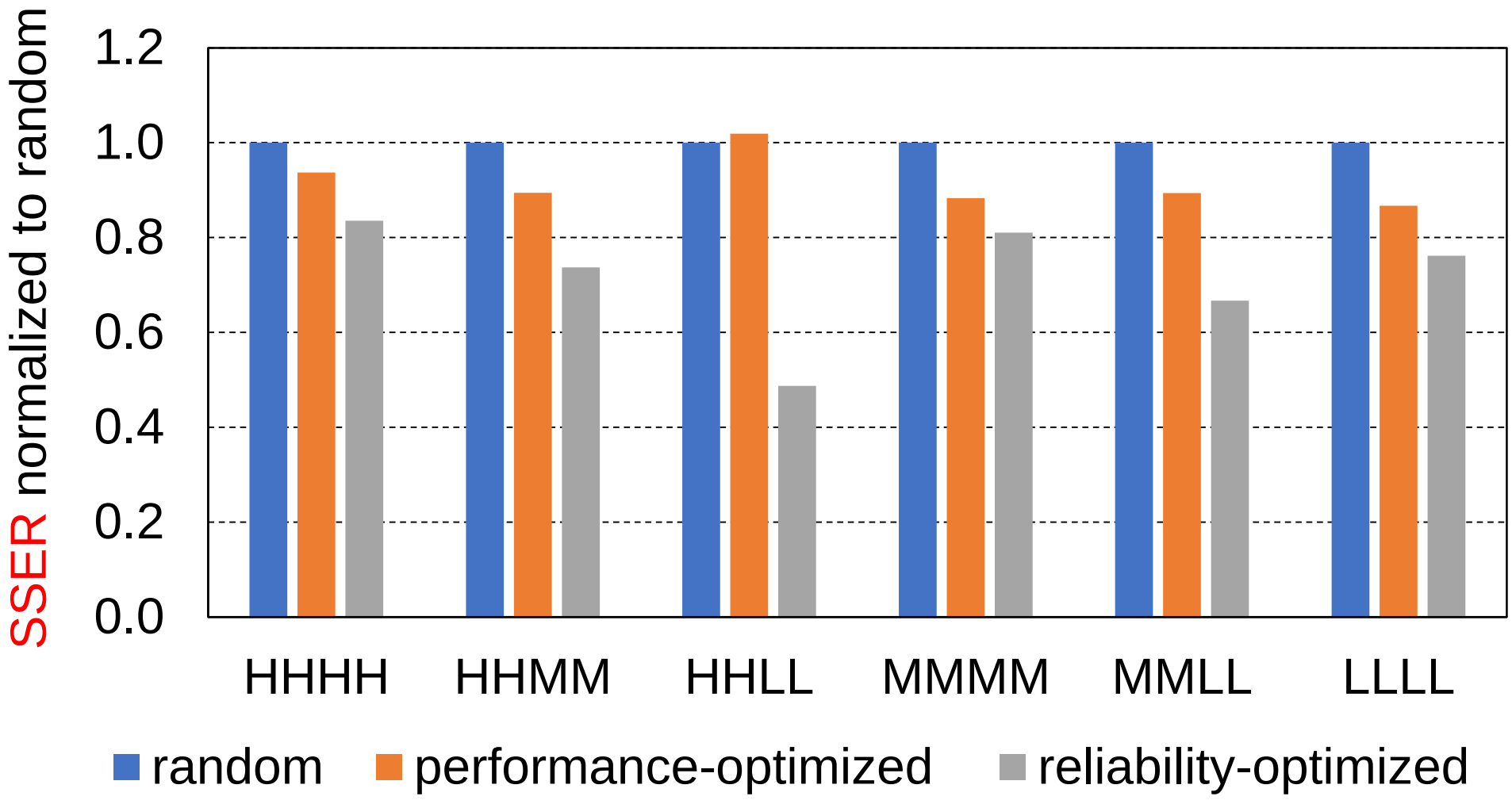



Compared to performance-optimized

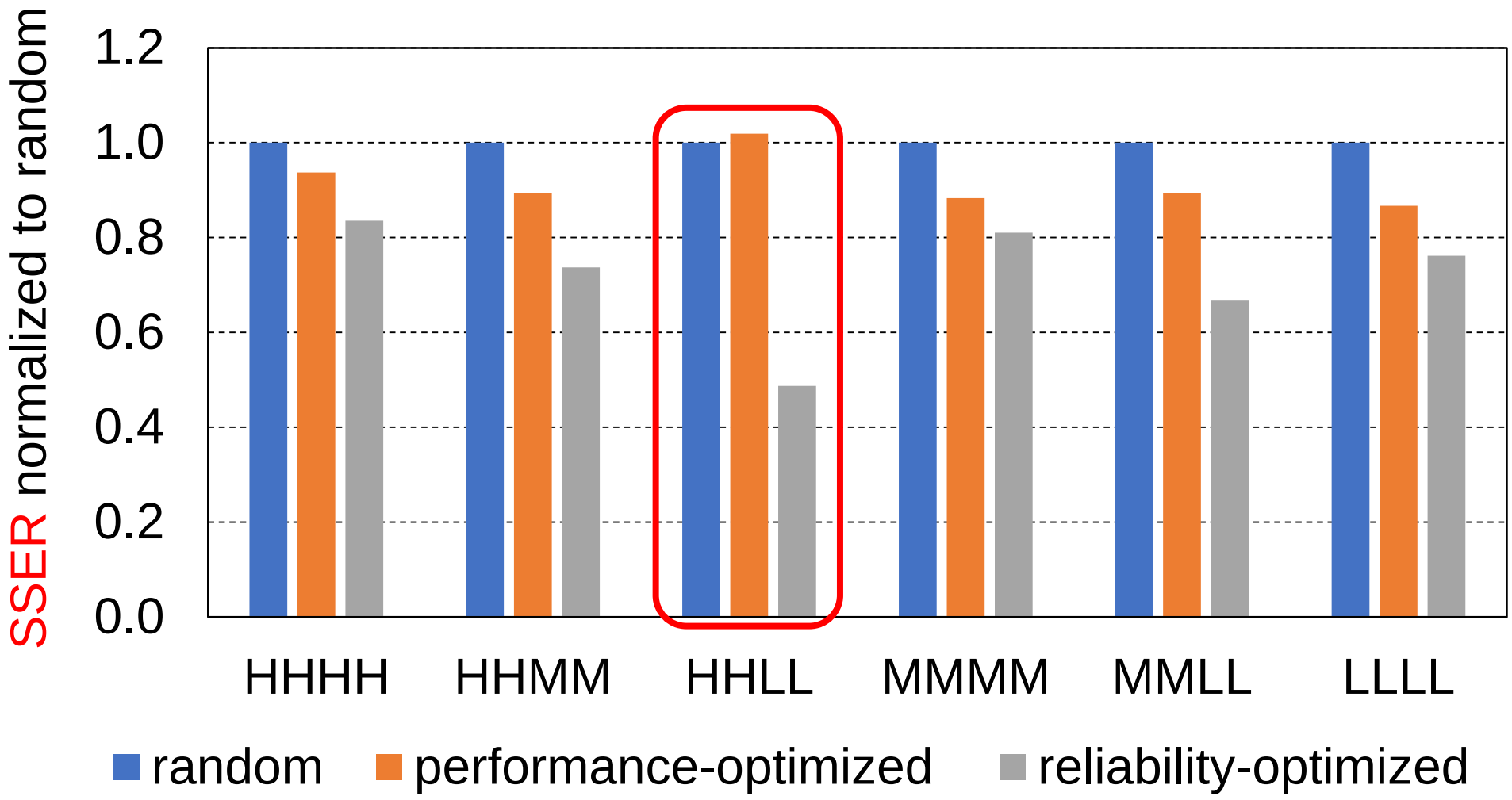
Reliability-optimized
-- degrades STP by 6.3%

-- up to 18.7%

# 2B2S Results – by Workload Category

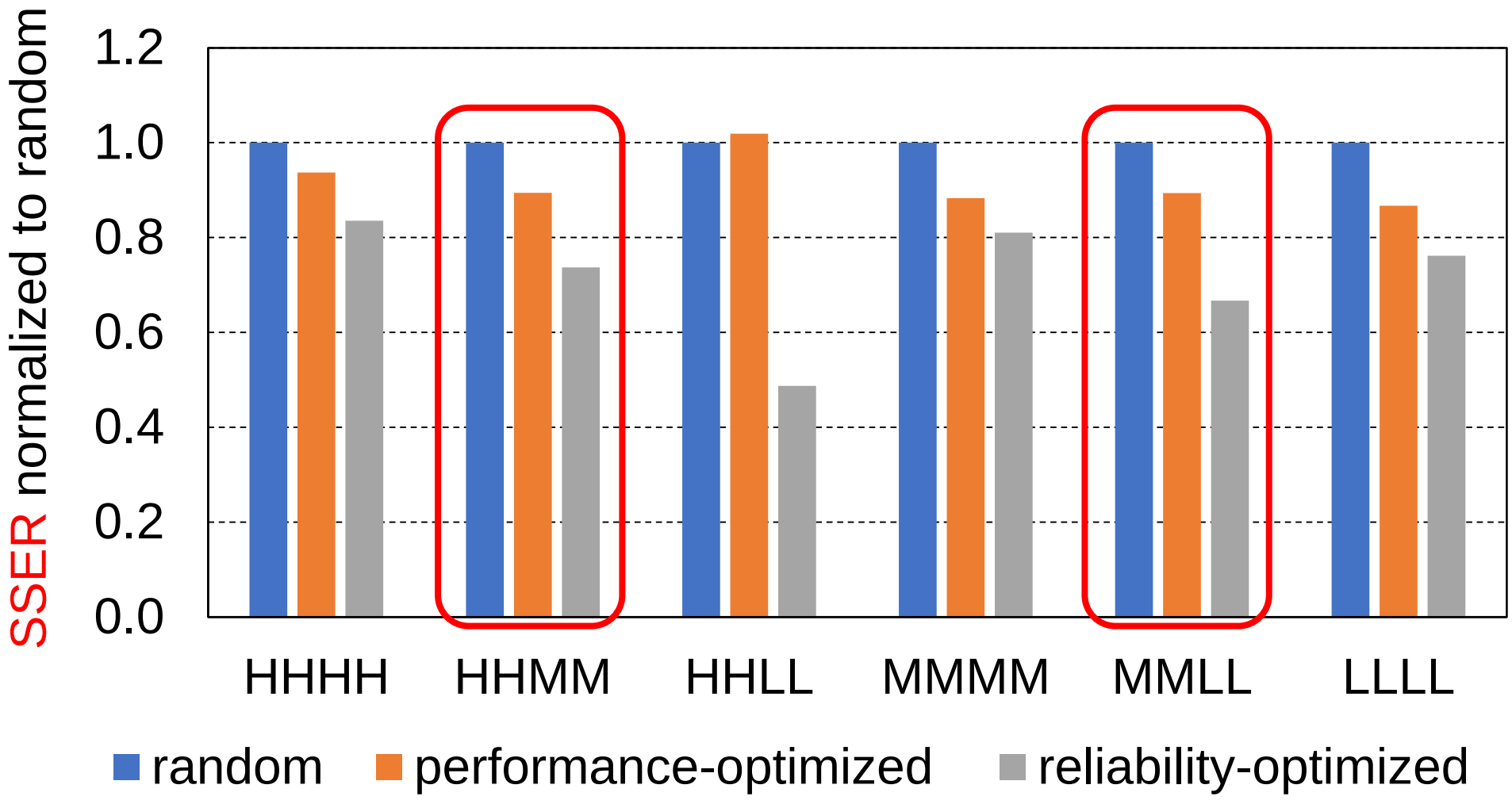


# 2B2S Results – by Workload Category




HHLL 53.2%

# 2B2S Results – by Workload Category

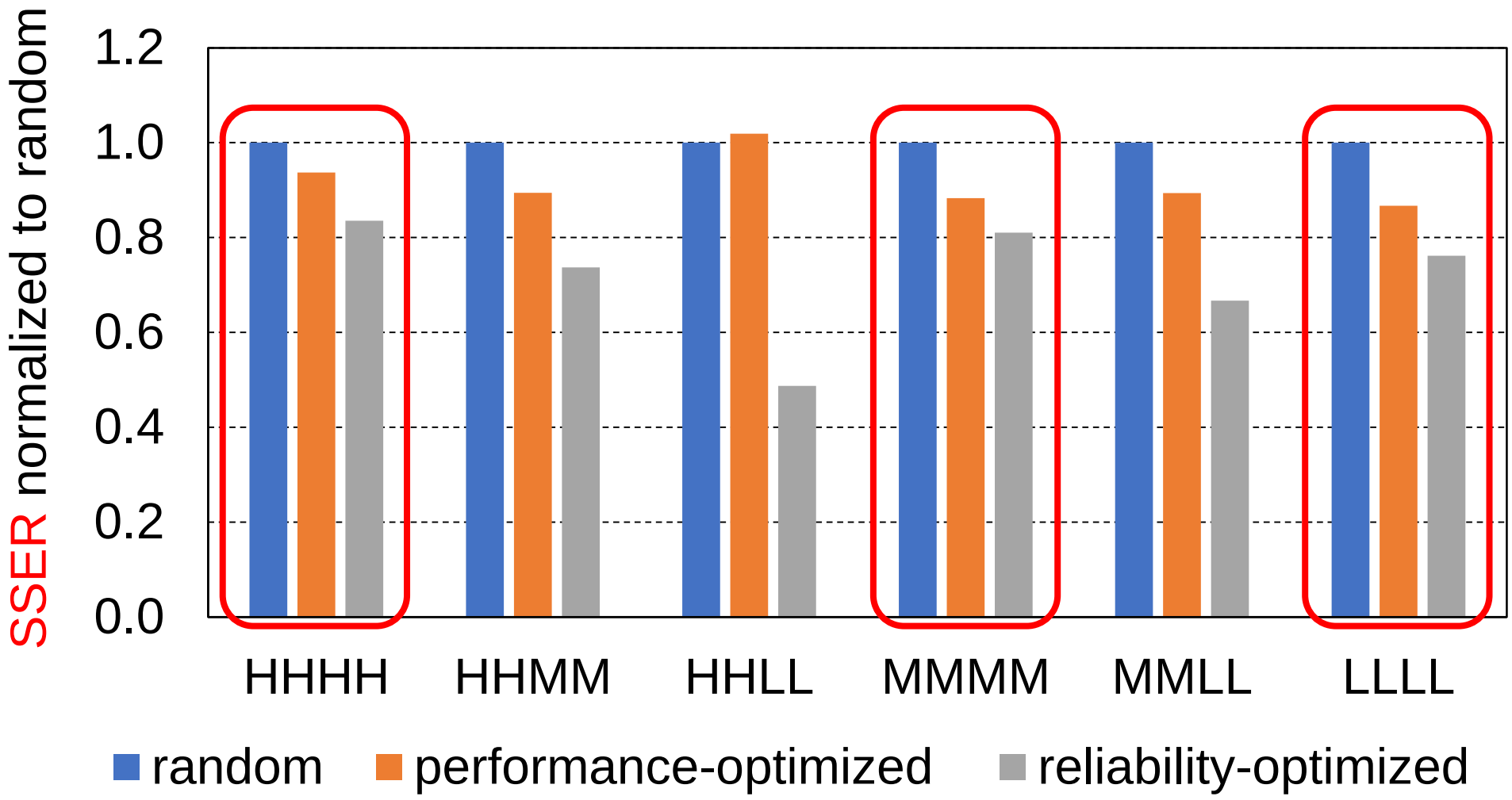



HHLL 53.2%

HHMM 15.8%

MMLL 22.7%

# 2B2S Results – by Workload Category




HHLL 53.2%

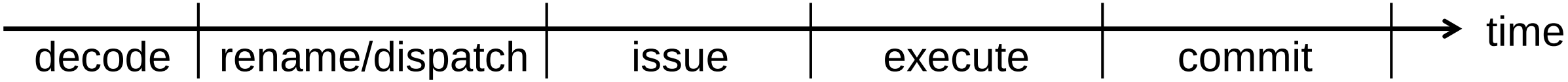
HHMM 15.8%
MMLL 22.7%

HHHH 10.2%
MMMM 7.3%
LLLL 10.6%

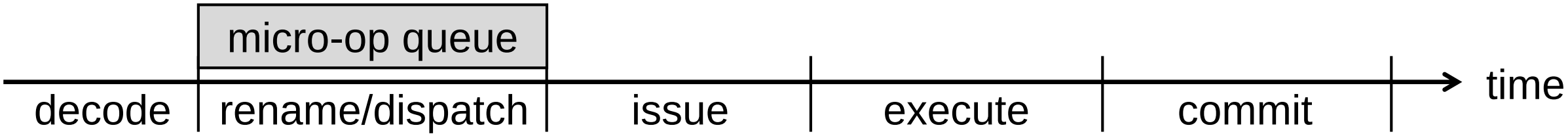
# Contribution #2 [DSN 2020, Under review]

# Dispatch Halting to Improve Out-of-Order Core Reliability

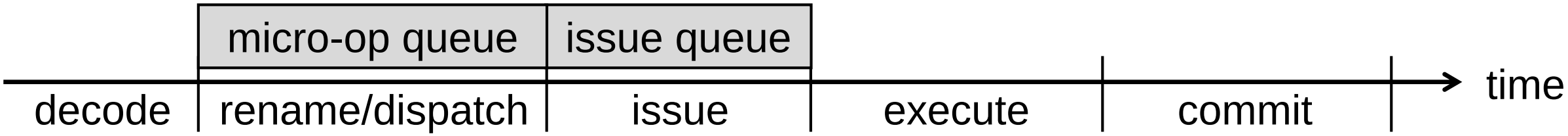
# Vulnerability of Out-of-Order Core Resources



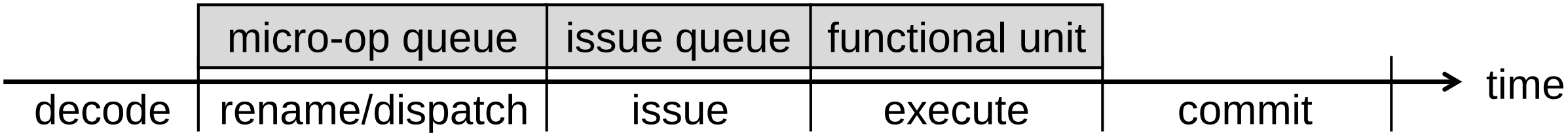
# Vulnerability of Out-of-Order Core Resources



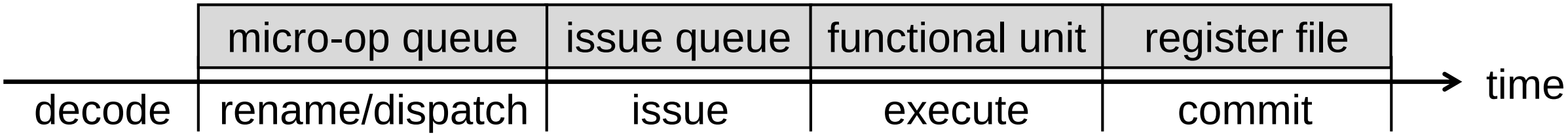
# Vulnerability of Out-of-Order Core Resources



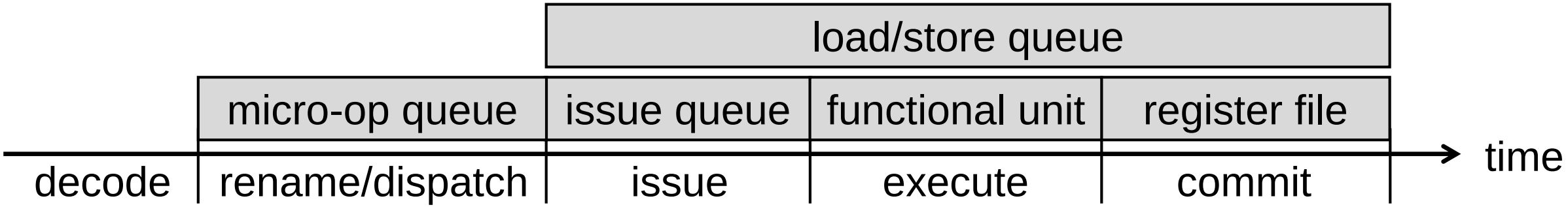
# Vulnerability of Out-of-Order Core Resources



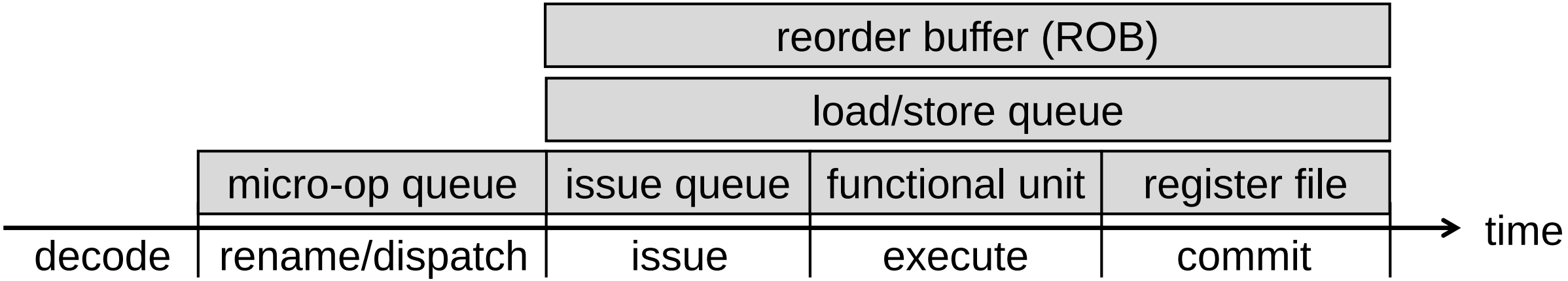
# Vulnerability of Out-of-Order Core Resources



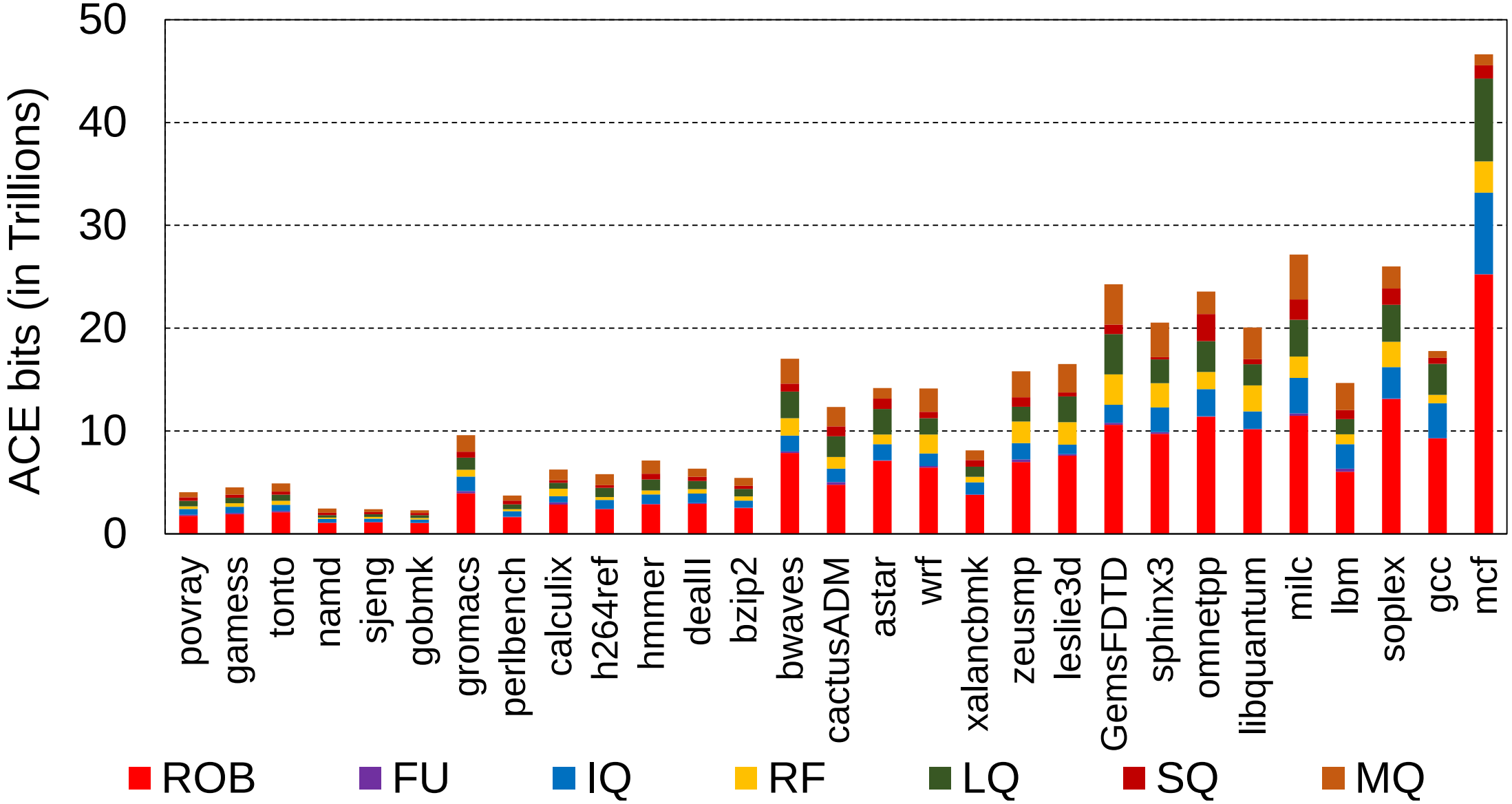
# Vulnerability of Out-of-Order Core Resources



# Vulnerability of Out-of-Order Core Resources



# ACE Bit Count for OoO Resources

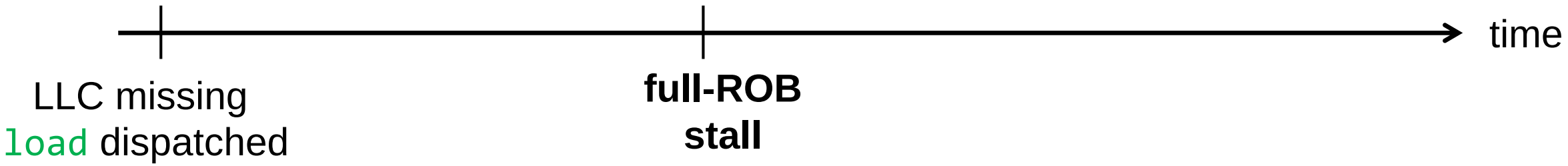


# Vulnerable State under Full-ROB Stalls

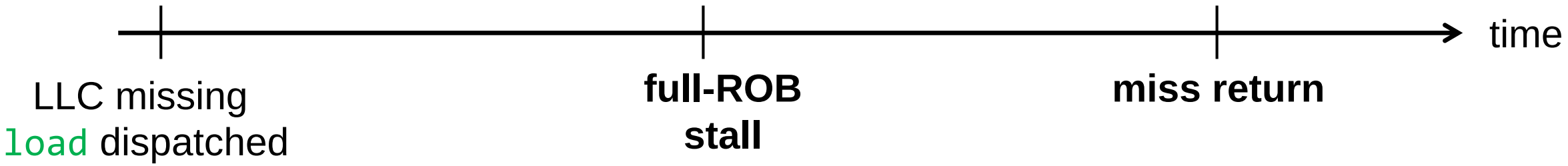
# Vulnerable State under Full-ROB Stalls

# Vulnerable State under Full-ROB Stalls

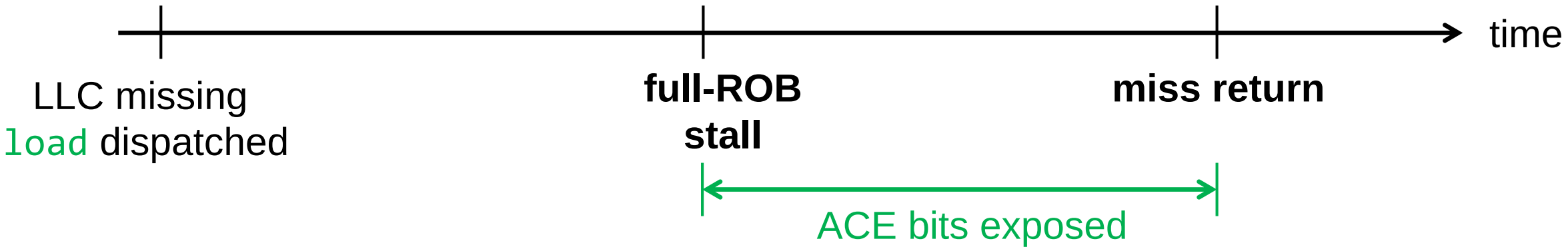
# Vulnerable State under Full-ROB Stalls



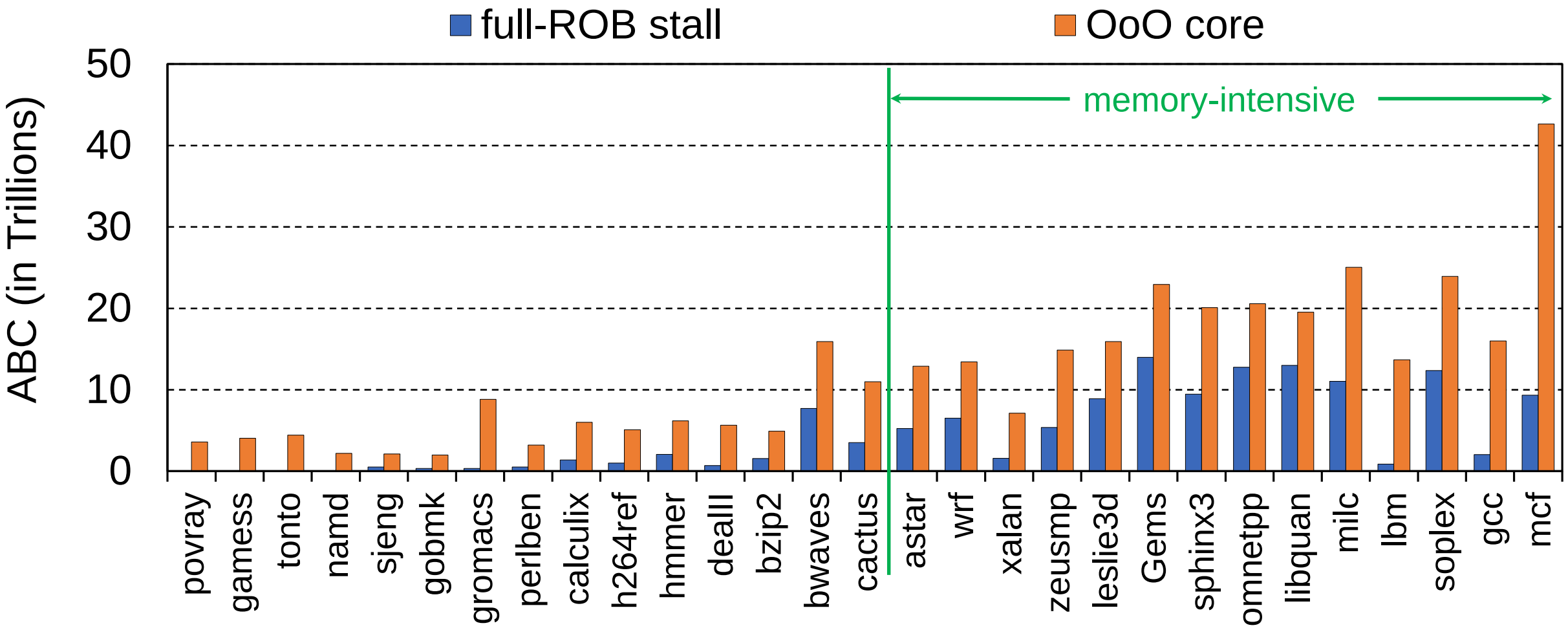
# Vulnerable State under Full-ROB Stalls



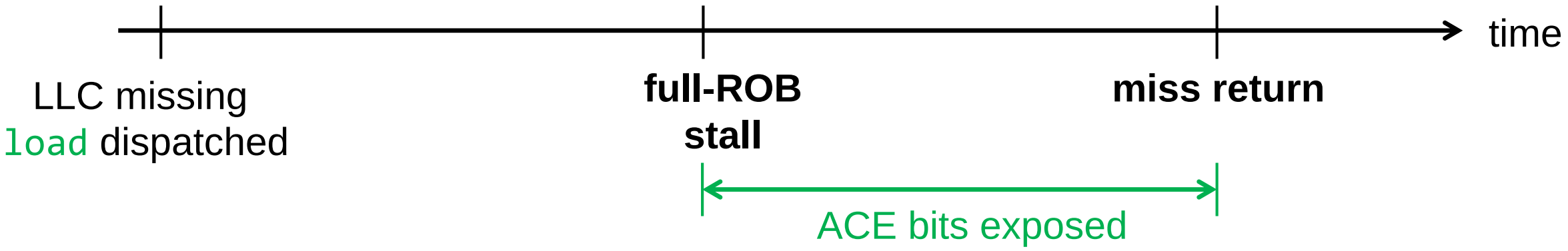
# Vulnerable State under Full-ROB Stalls



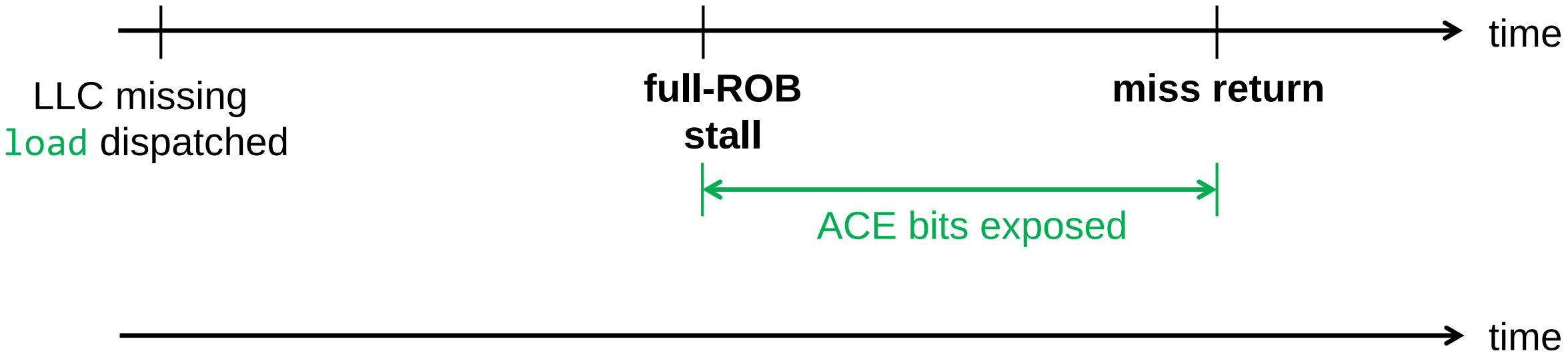
# Vulnerable State under Full-ROB Stalls



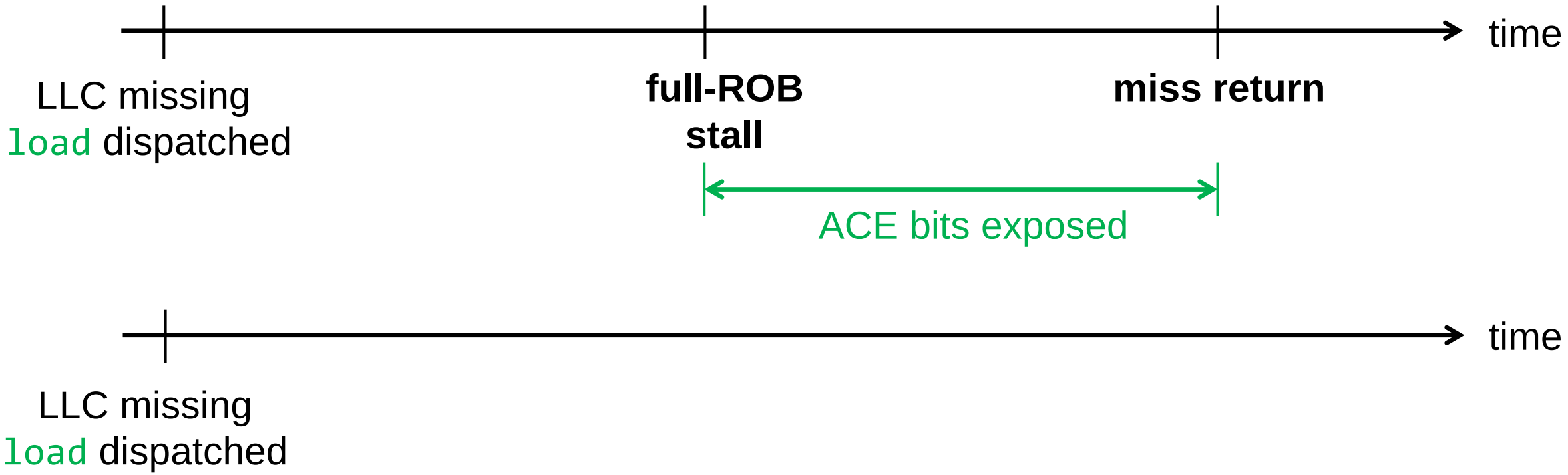
# Vulnerable State under Blocked ROB Head



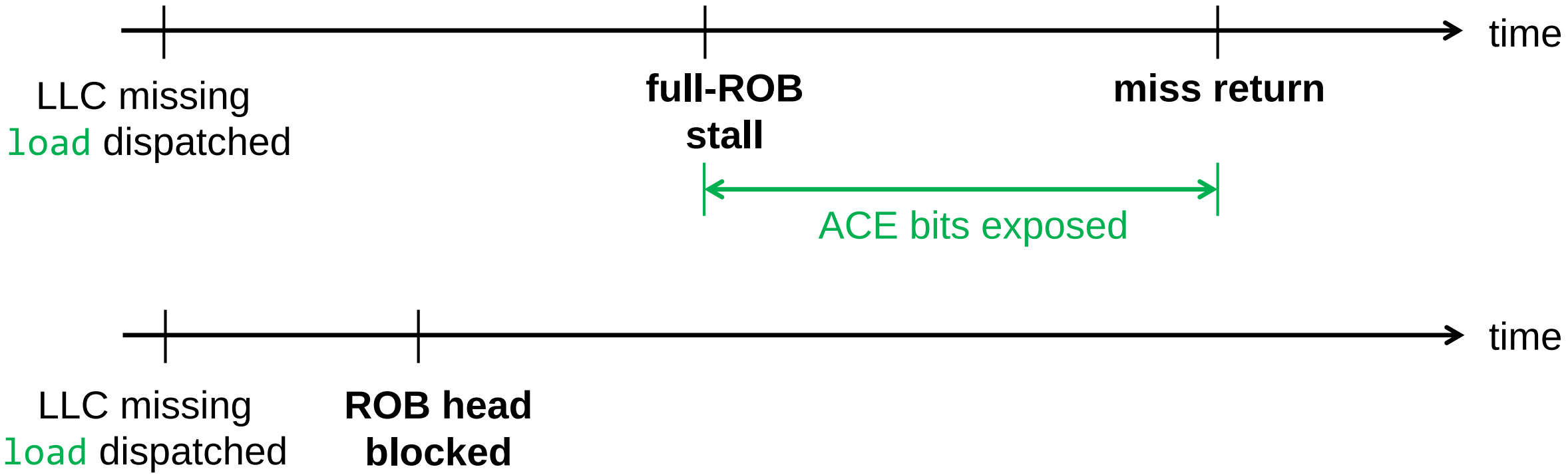
# Vulnerable State under Blocked ROB Head



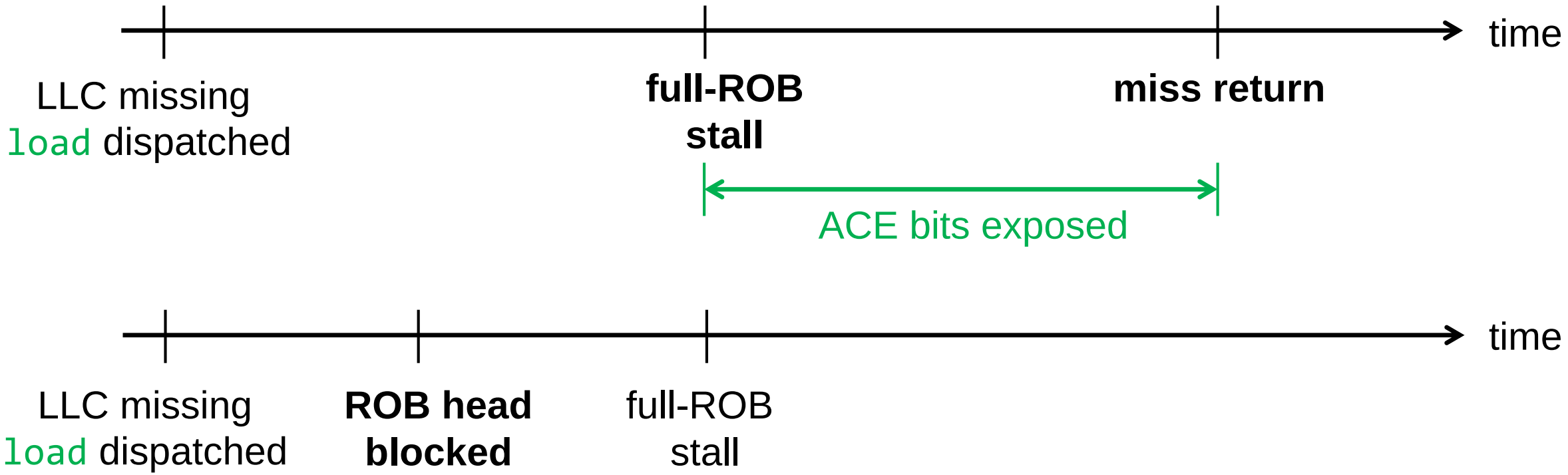
# Vulnerable State under Blocked ROB Head



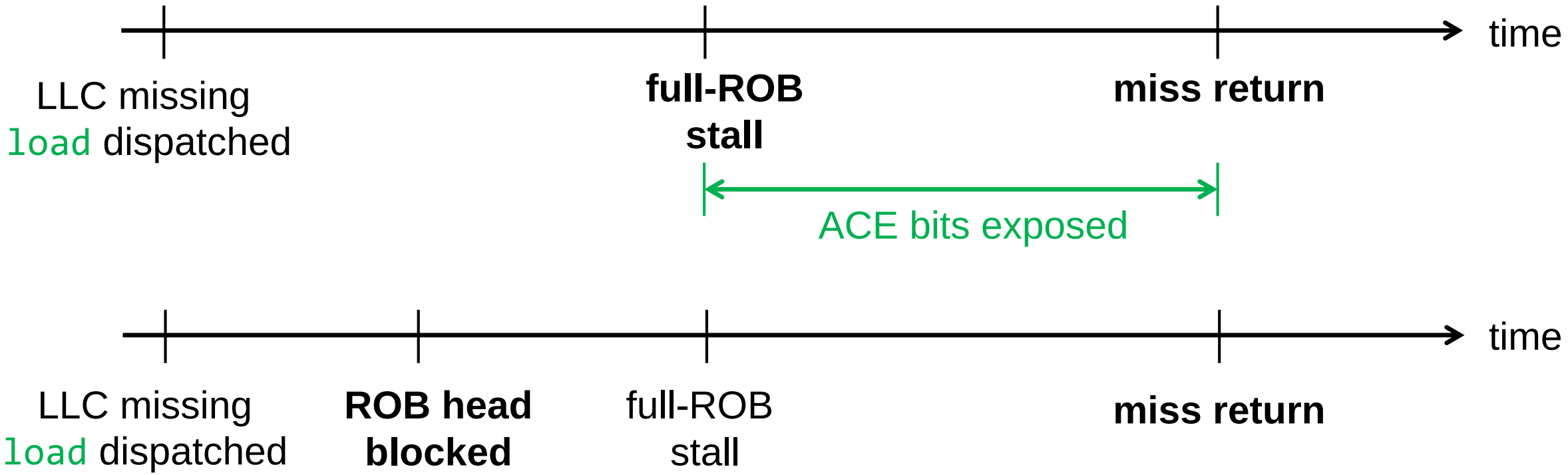
# Vulnerable State under Blocked ROB Head



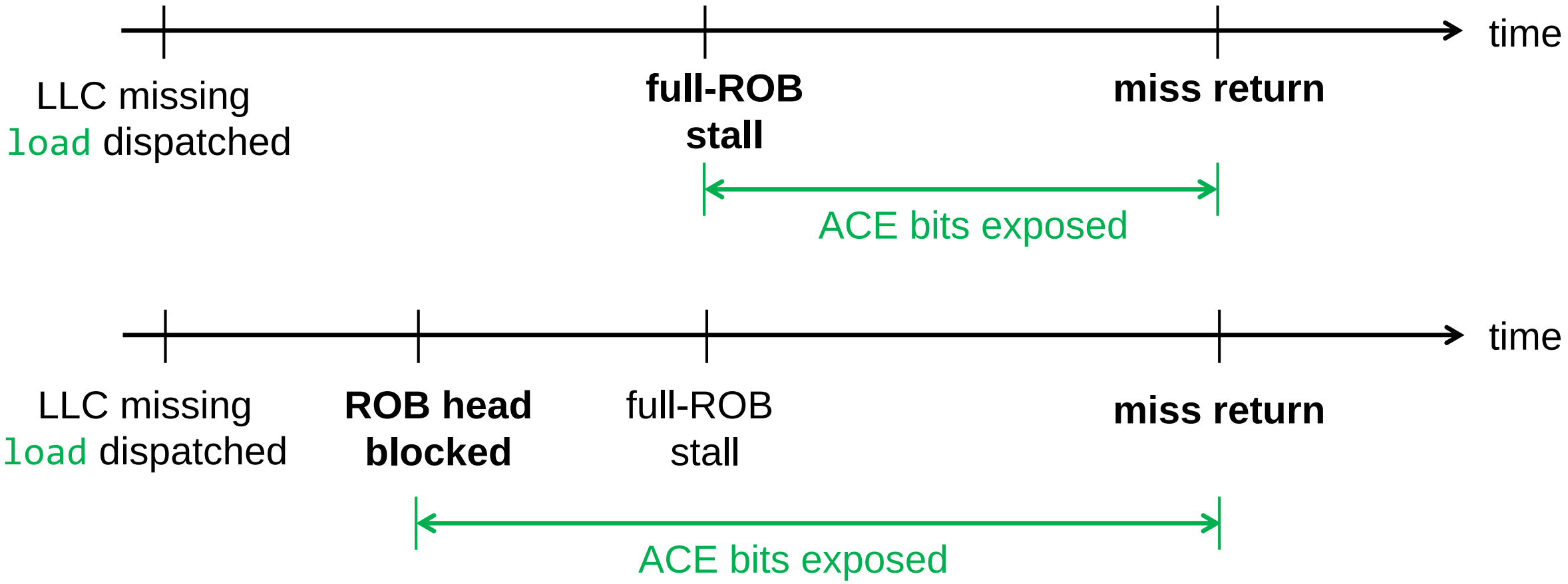
# Vulnerable State under Blocked ROB Head



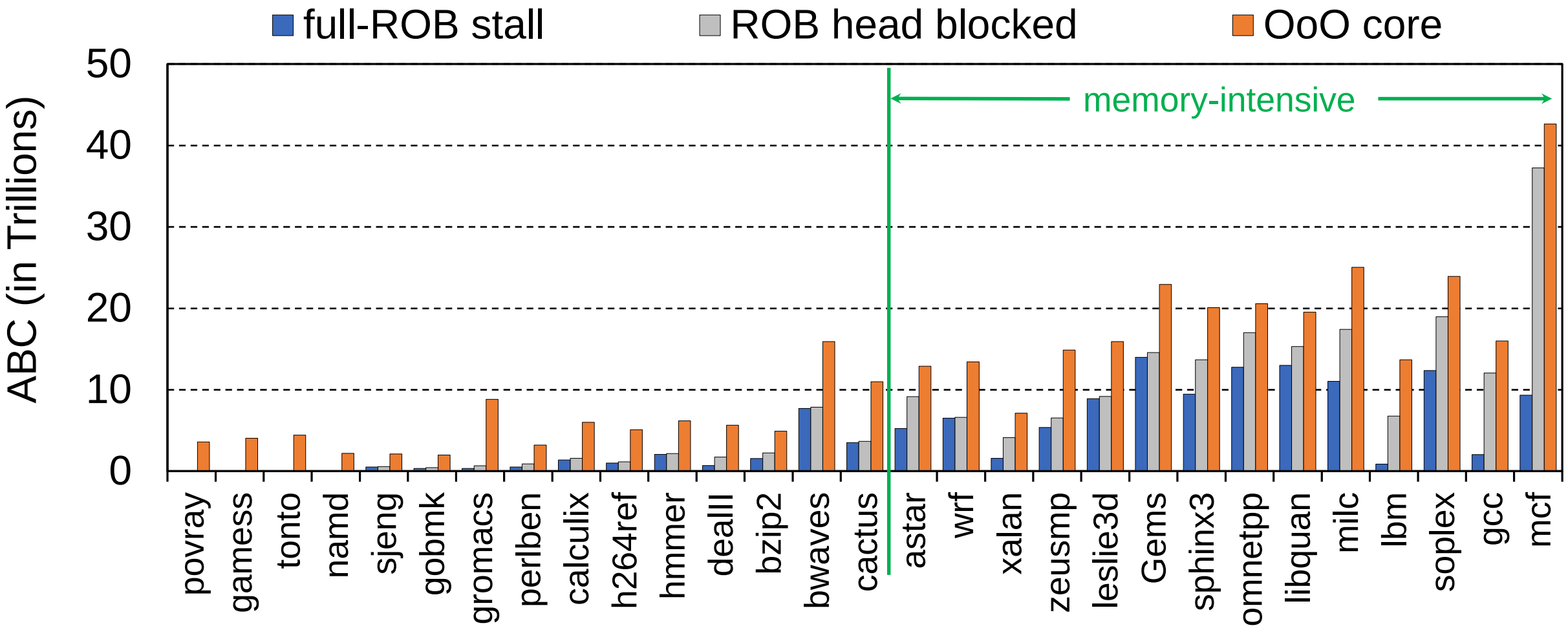
# Vulnerable State under Blocked ROB Head



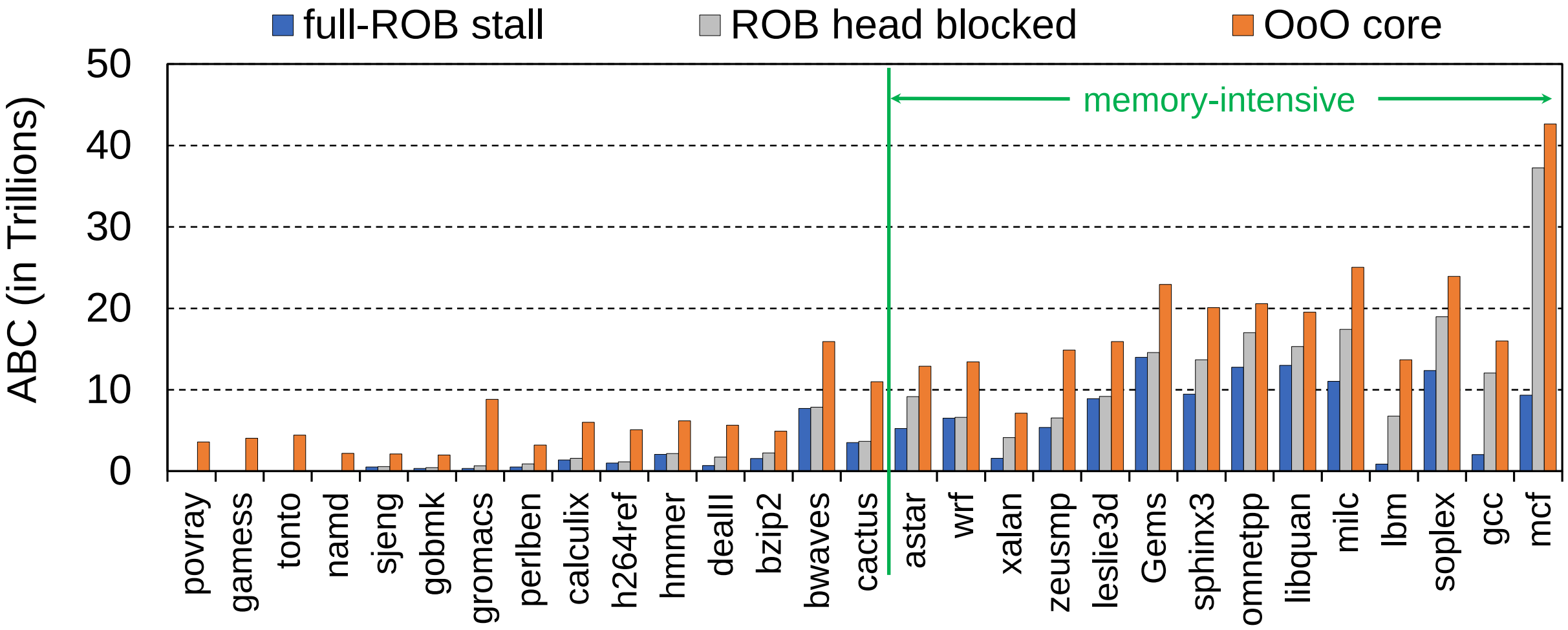
# Vulnerable State under Blocked ROB Head



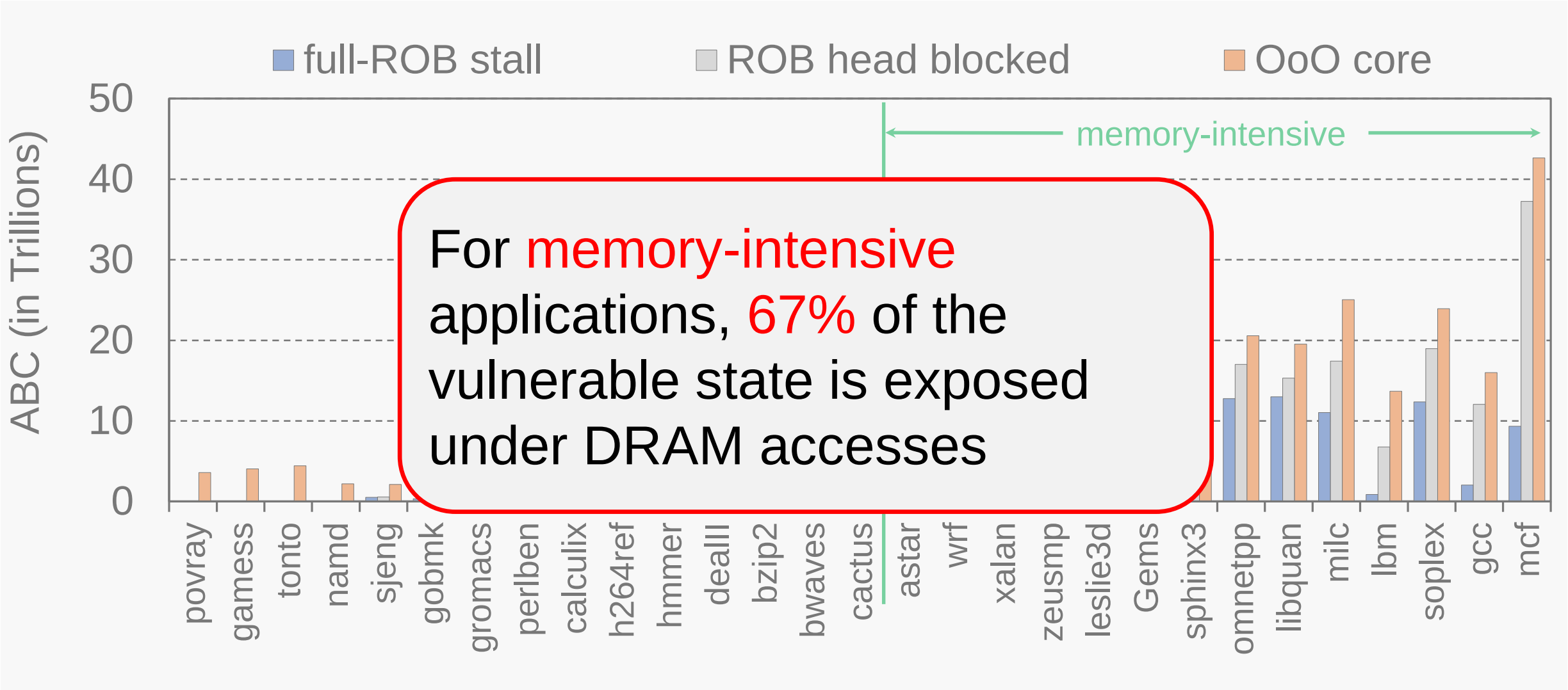
# Vulnerable State under Blocked ROB Head



# Quantifying the Potential



# Quantifying the Potential




For memory-intensive applications, 67% of the vulnerable state is exposed under DRAM accesses

# Dispatch Halting to Improve Reliability

# Dispatch Halting to Improve Reliability

Proactively prevent instructions following an LLC missing load to allocate entries in the ROB (or back-end structures)

# Dispatch Halting to Improve Reliability

Proactively prevent instructions following an LLC missing load to allocate entries in the ROB (or back-end structures)

⇒

Proactive Dispatch Halting

# Dispatch Halting to Improve Reliability

Proactively prevent instructions following an LLC missing load to allocate entries in the ROB (or back-end structures)

⇒ Proactive Dispatch Halting

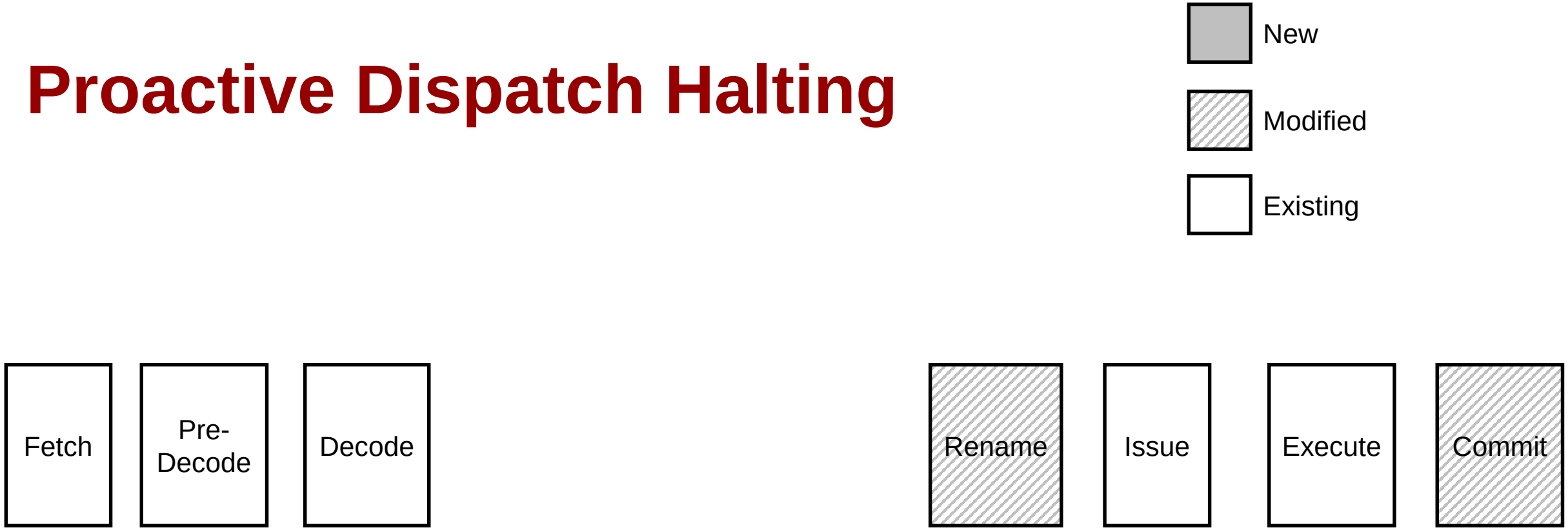
Reactively mark the back-end speculative after a blocked ROB head

# Dispatch Halting to Improve Reliability

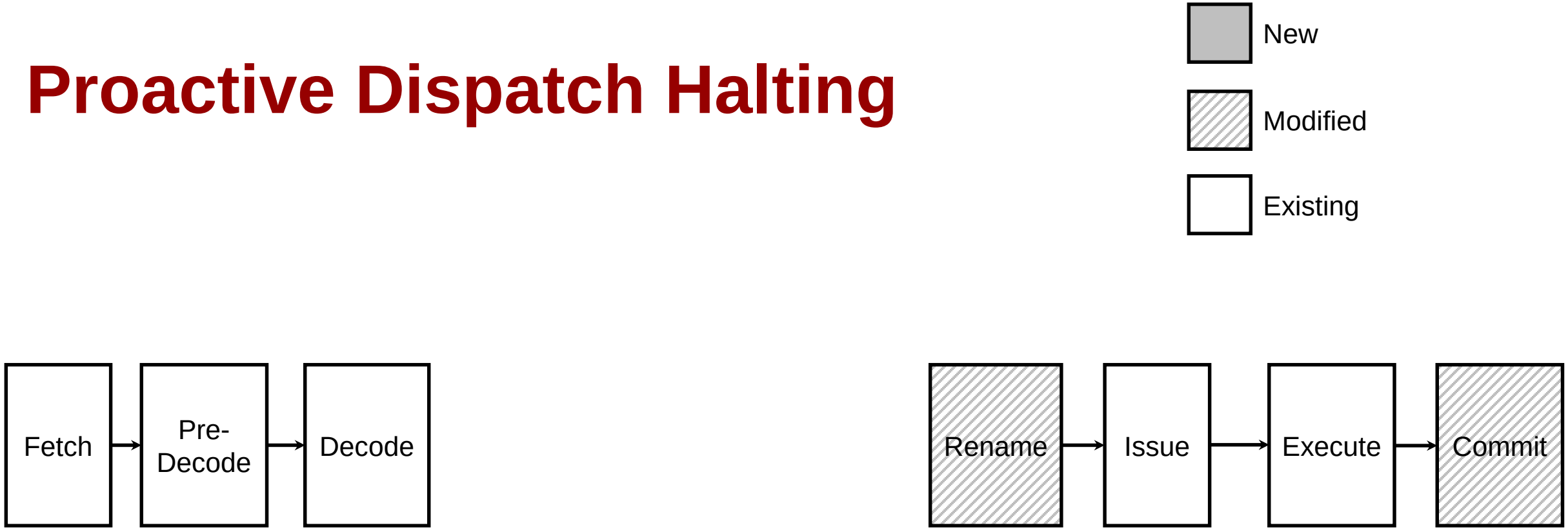
Proactively prevent instructions following an LLC missing load to allocate entries in the ROB (or back-end structures) ⇒ Proactive Dispatch Halting

Reactively mark the back-end speculative after a blocked ROB head ⇒ Reactive Dispatch Halting

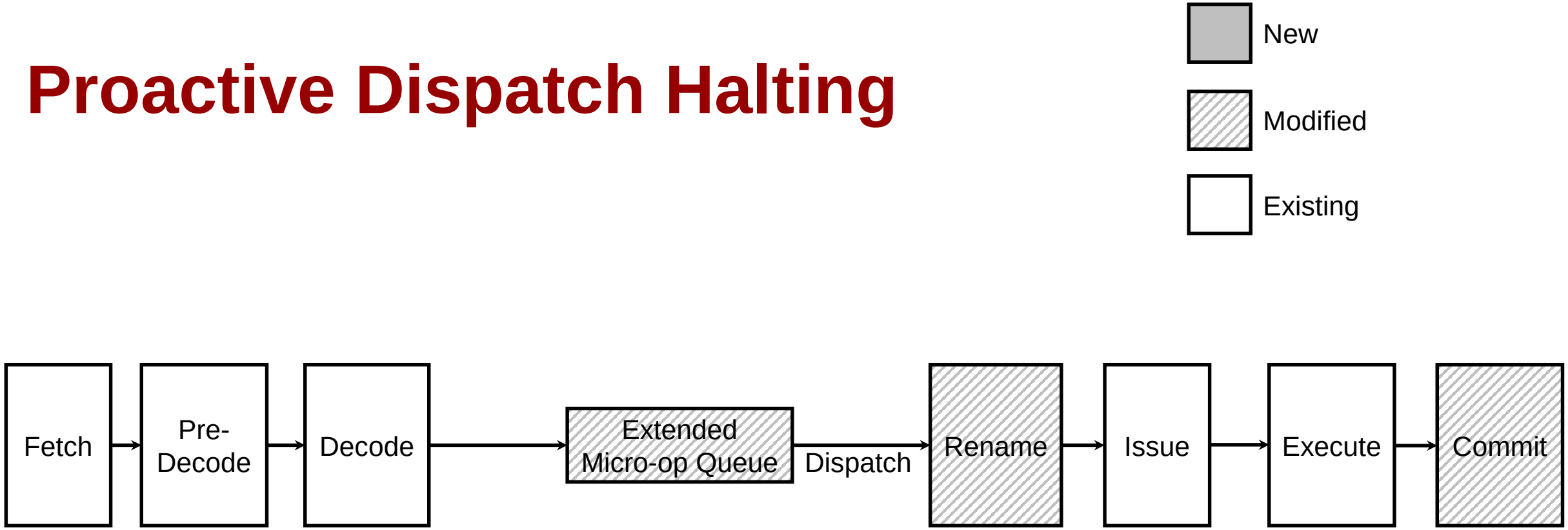
# Proactive Dispatch Halting



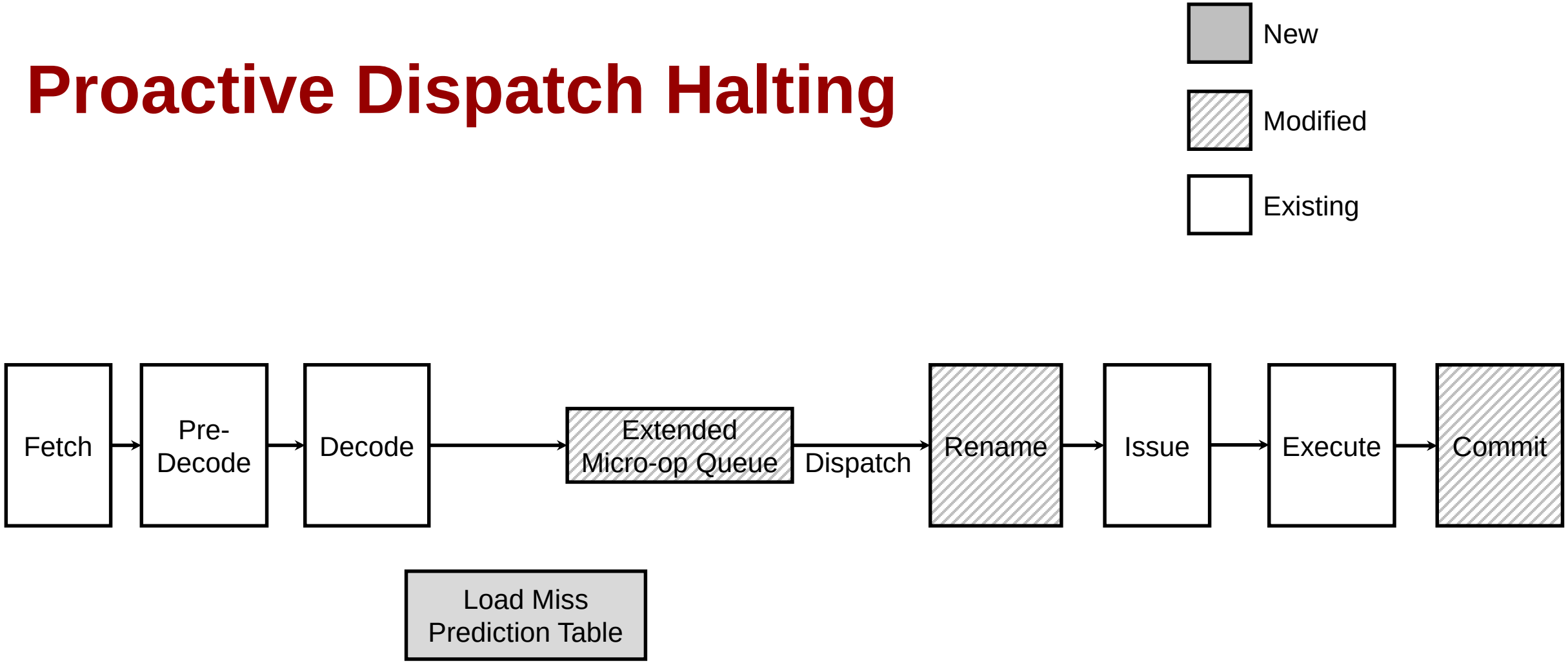
# Proactive Dispatch Halting



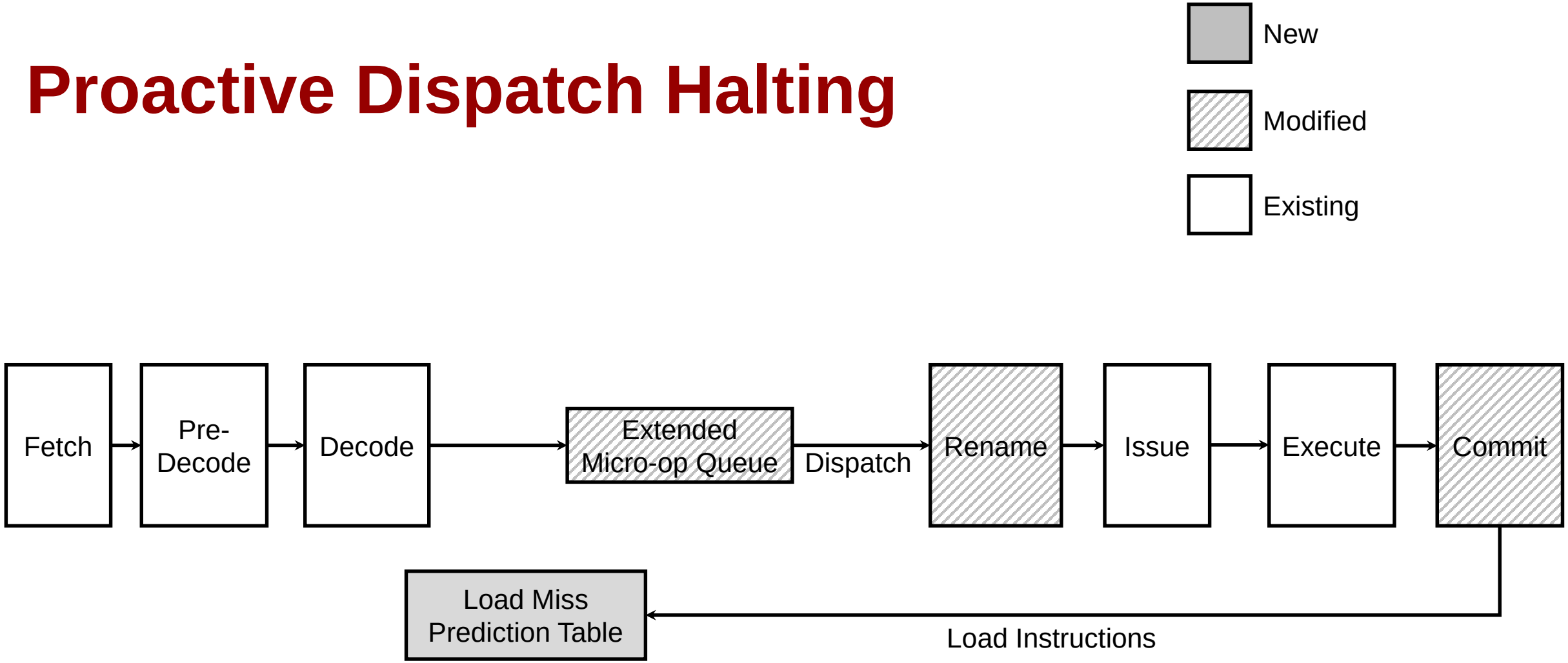
# Proactive Dispatch Halting



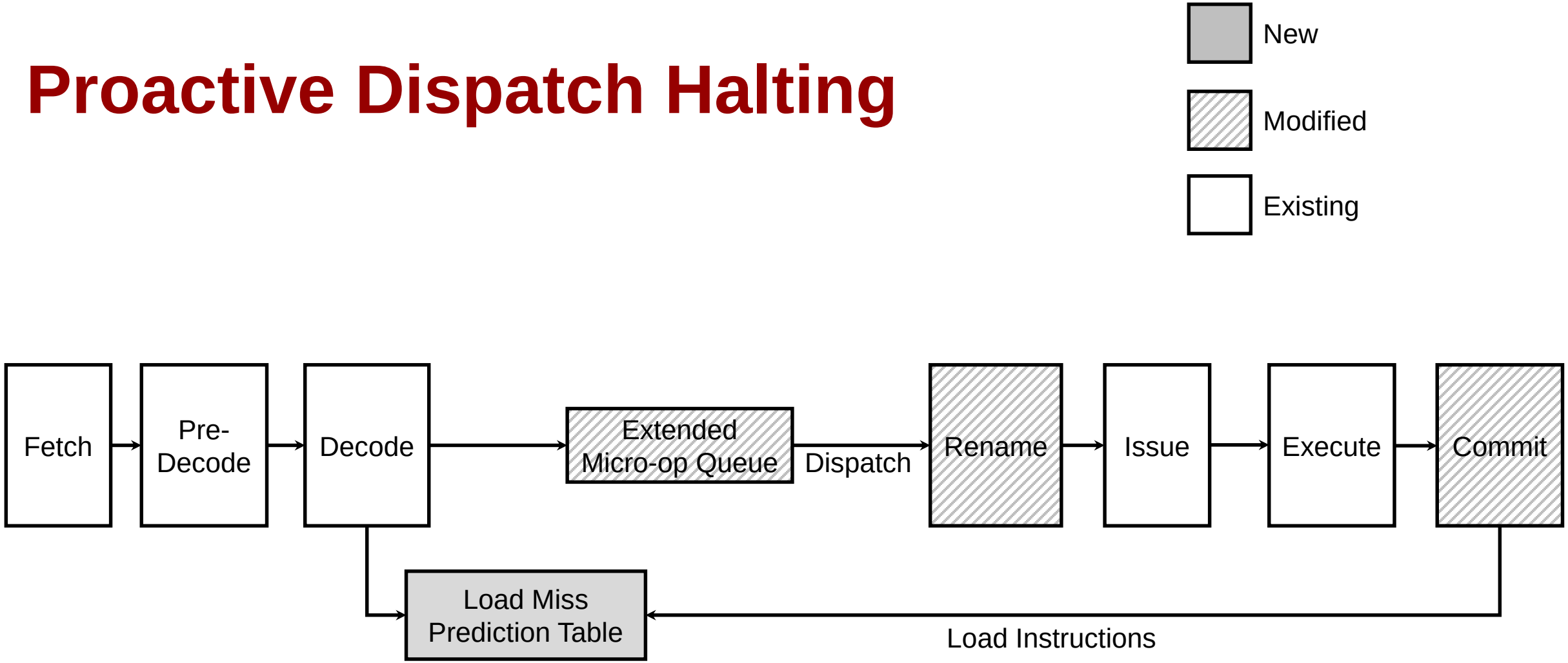
# Proactive Dispatch Halting



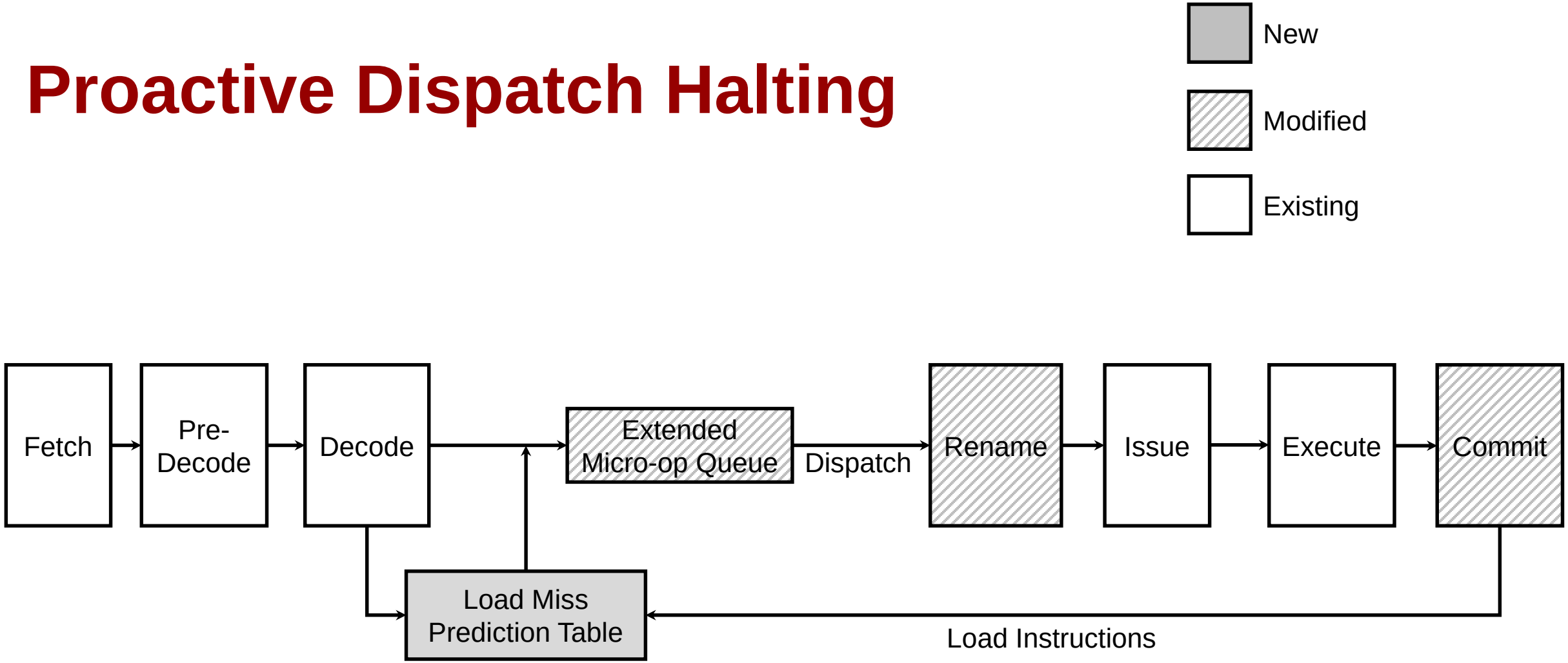
# Proactive Dispatch Halting



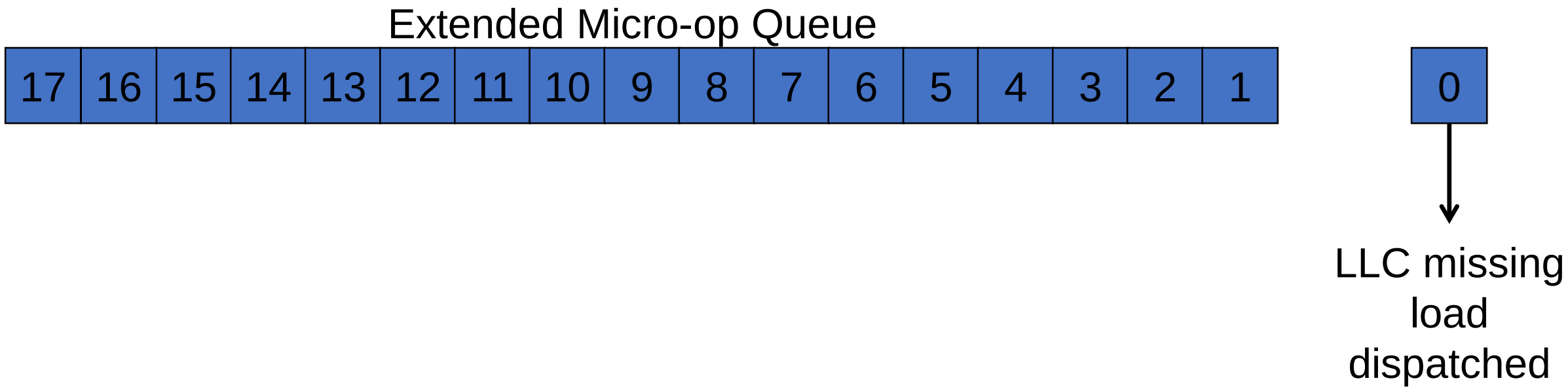
# Proactive Dispatch Halting



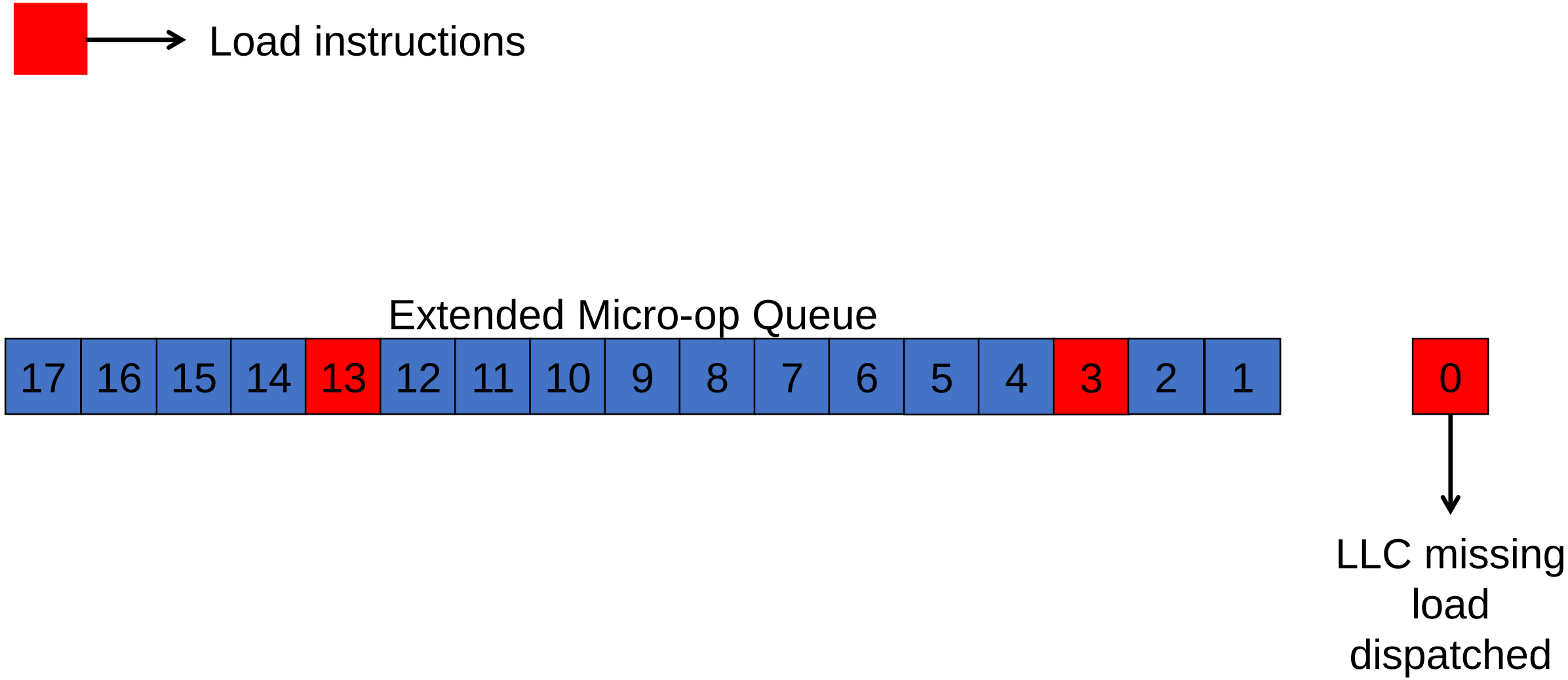
# Proactive Dispatch Halting



# Performance Bottlenecks



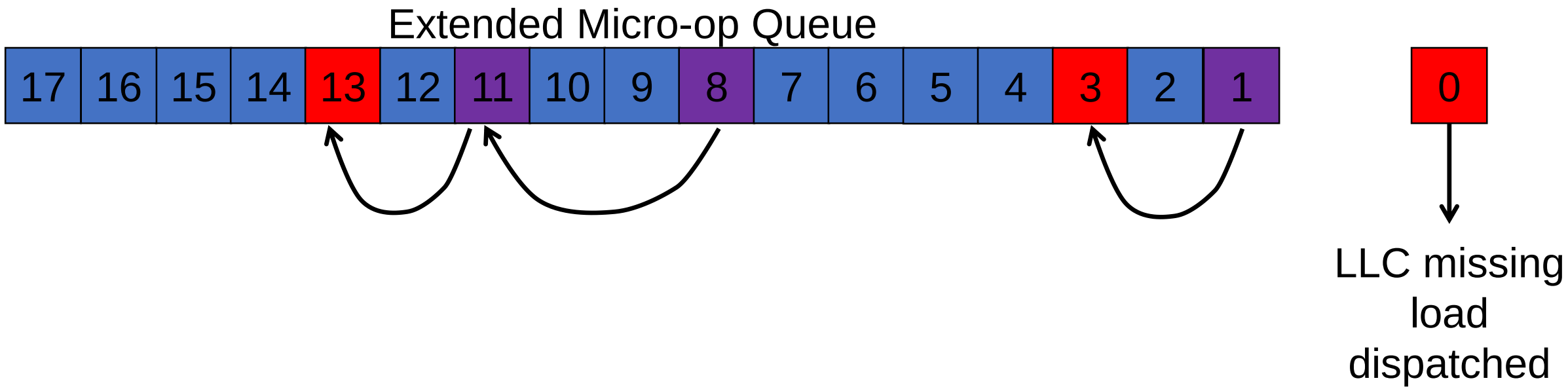
# Performance Bottlenecks



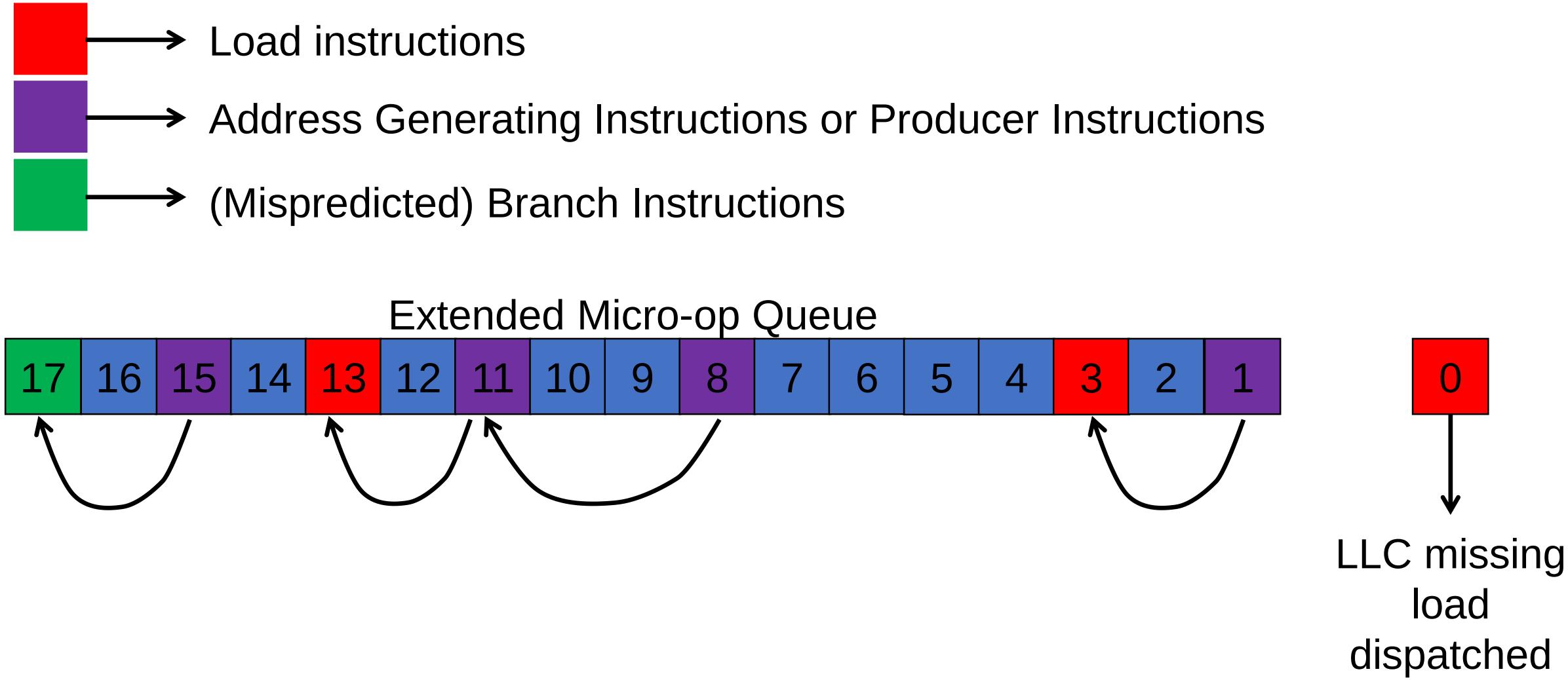
# Performance Bottlenecks

Load instructions

Address Generating Instructions or Producer Instructions



# Performance Bottlenecks



# Handling Performance Bottlenecks

# Handling Performance Bottlenecks

- Execute load/branch instructions and their producer instructions speculatively

# Handling Performance Bottlenecks

- Execute load/branch instructions and their producer instructions speculatively

- Find producer instructions iteratively

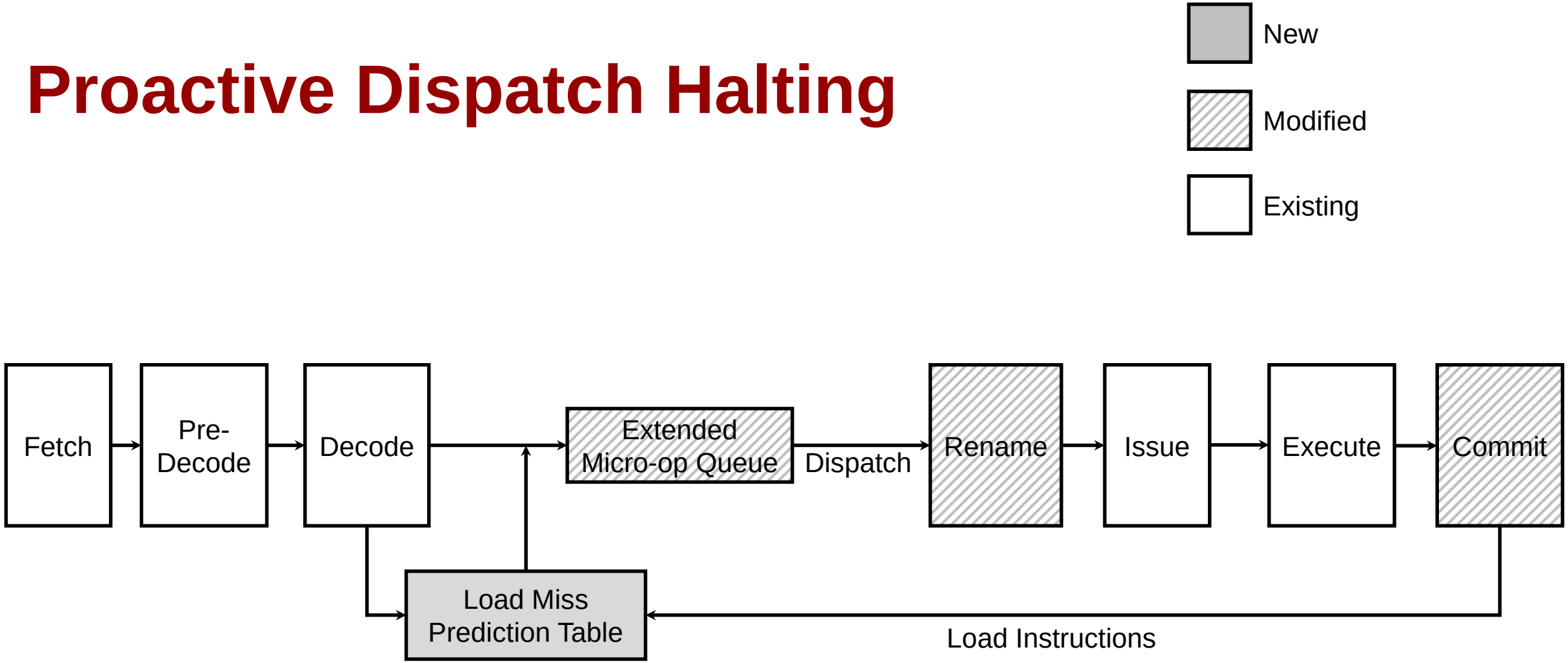
  -- Modify front-end Register Allocation Table (RAT)

# Handling Performance Bottlenecks

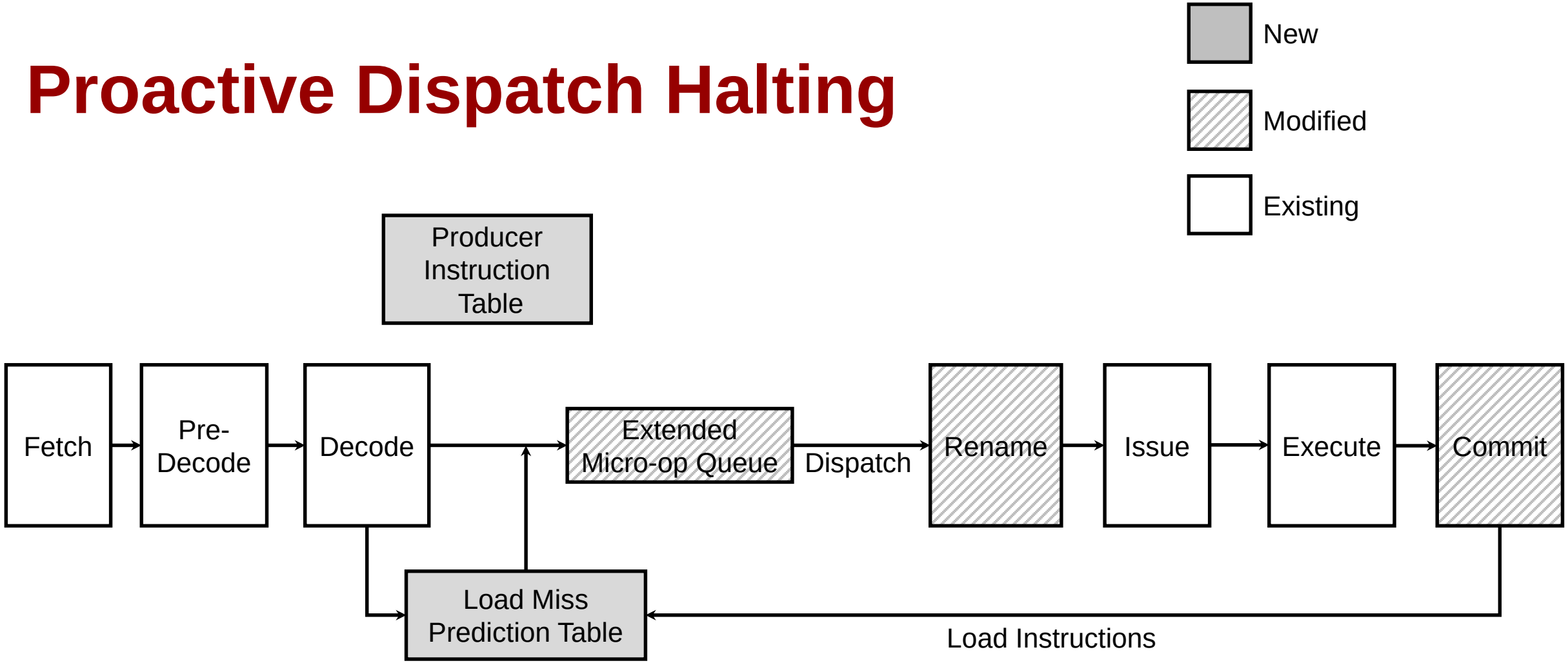
- Execute load/branch instructions and their producer instructions speculatively
- Find producer instructions iteratively

  -- Modify front-end Register Allocation Table (RAT)
- One additional RAT checkpoint

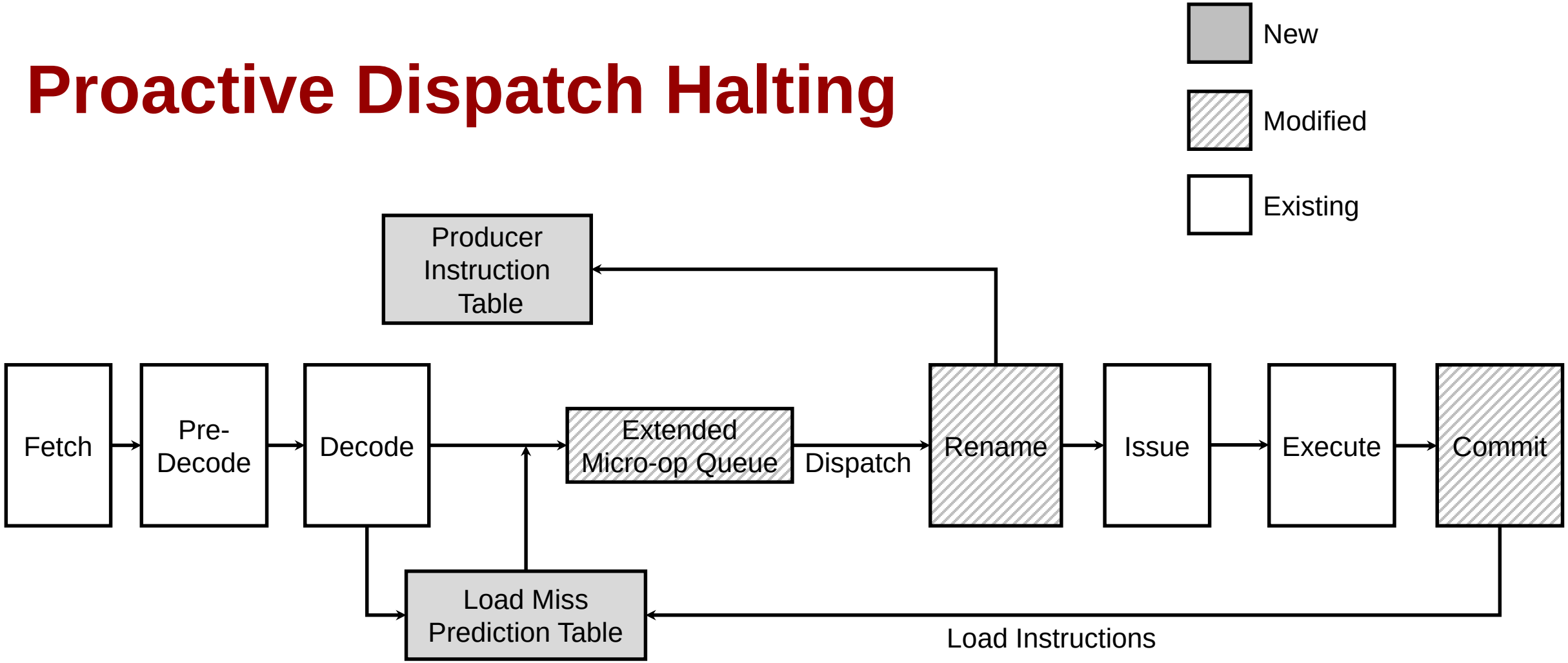
# Proactive Dispatch Halting



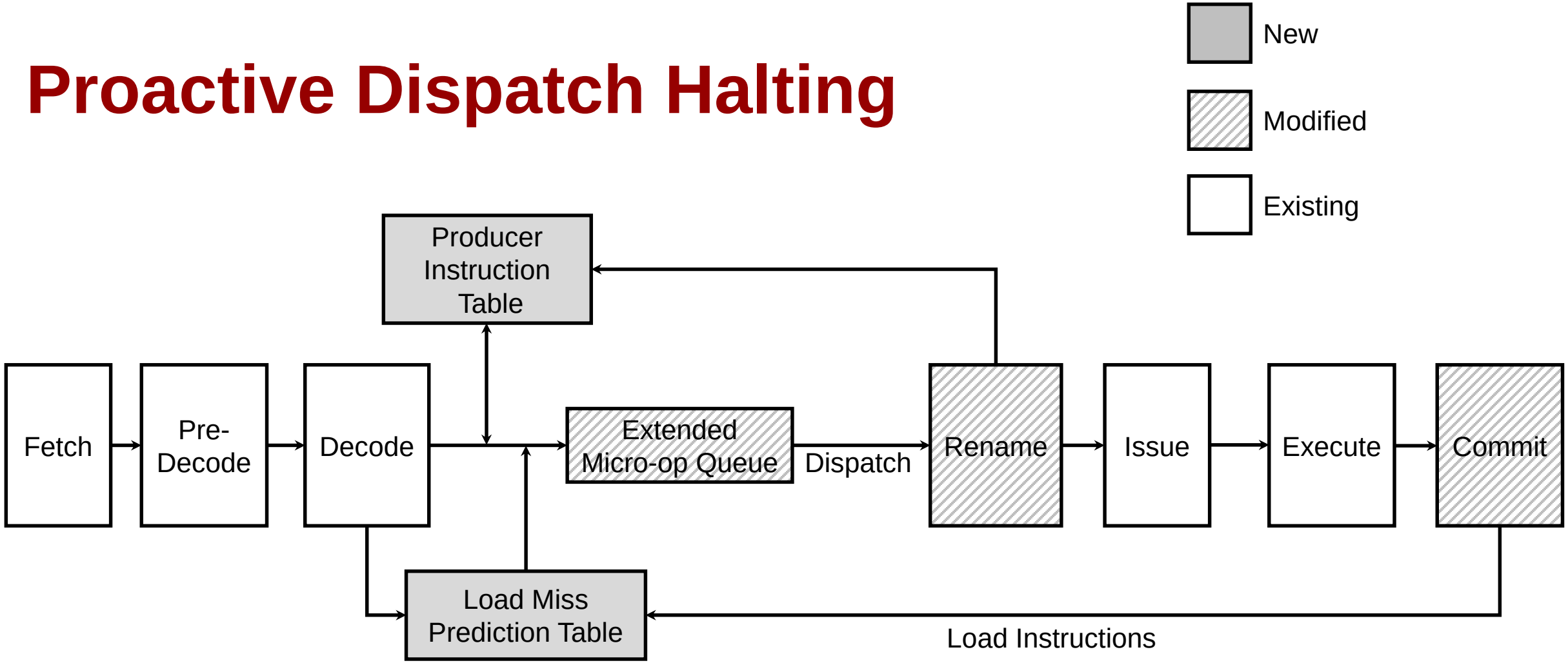
# Proactive Dispatch Halting



# Proactive Dispatch Halting



# Proactive Dispatch Halting



# Reactive Dispatch Halting

# Reactive Dispatch Halting

- Mark back-end as speculative after a load miss

# Reactive Dispatch Halting

- Mark back-end as speculative after a load miss
- EMQ works as a replay buffer

# Reactive Dispatch Halting

- Mark back-end as speculative after a load miss
- EMQ works as a replay buffer
- Upon resume

# Reactive Dispatch Halting

- Mark back-end as speculative after a load miss
- EMQ works as a replay buffer
- Upon resume
  - -- Copy retirement RAT to front-end RAT

Checkpointing RAT does not work

# Reactive Dispatch Halting

- Mark back-end as speculative after a load miss
- EMQ works as a replay buffer
- Upon resume
  - -- Copy retirement RAT to front-end RAT
  - -- Re-execute all instructions from the replay buffer or EMQ

Checkpointing RAT does not work

# Evaluation

OoO: Baseline out-of-order core

# Evaluation

OoO: Baseline out-of-order core

NO-SPEC: Proactive dispatch halting without speculative execution

# Evaluation

OoO: Baseline out-of-order core

NO-SPEC: Proactive dispatch halting without speculative execution

P-DH: Proactive dispatch halting

# Evaluation

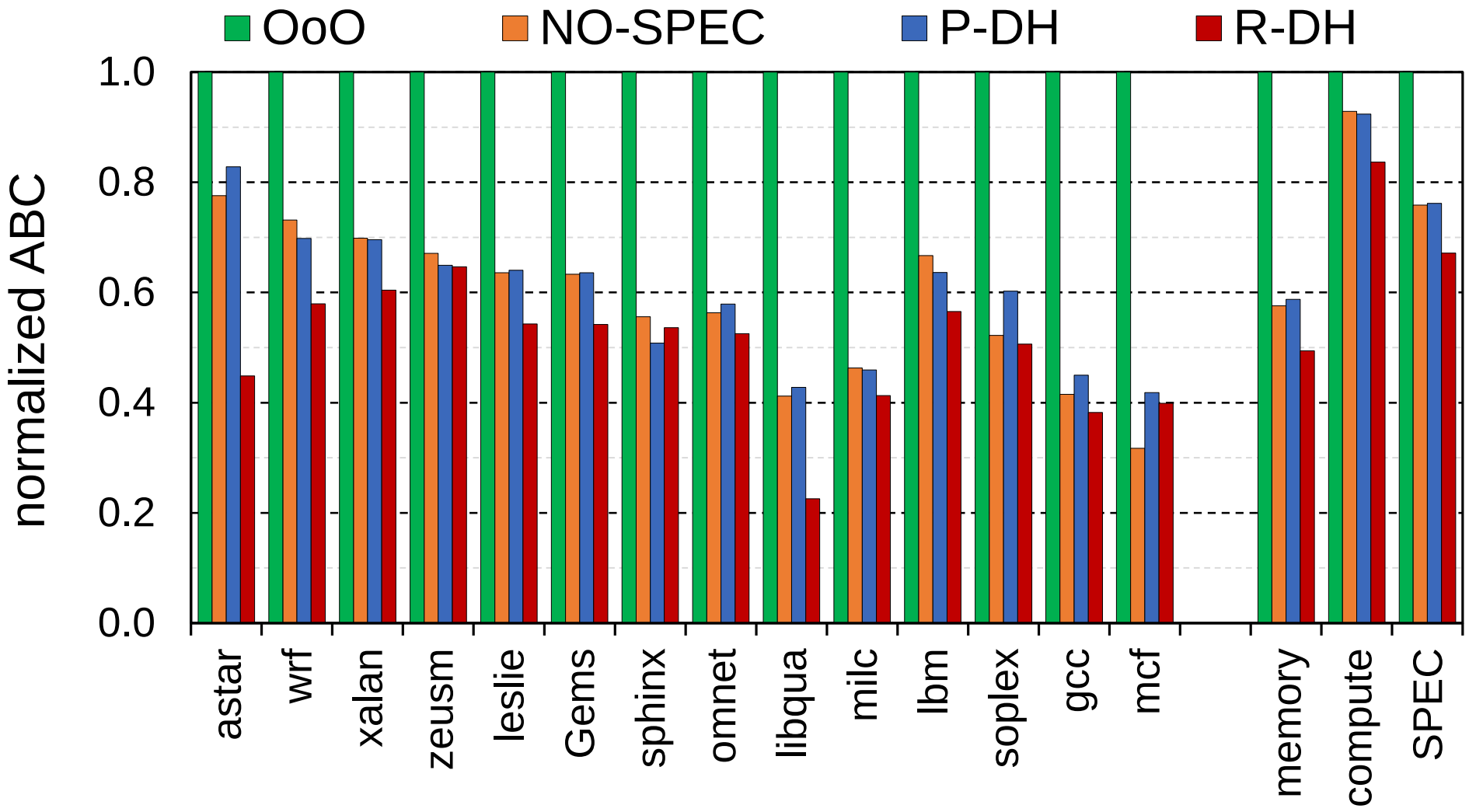
OoO: Baseline out-of-order core

NO-SPEC: Proactive dispatch halting without speculative execution

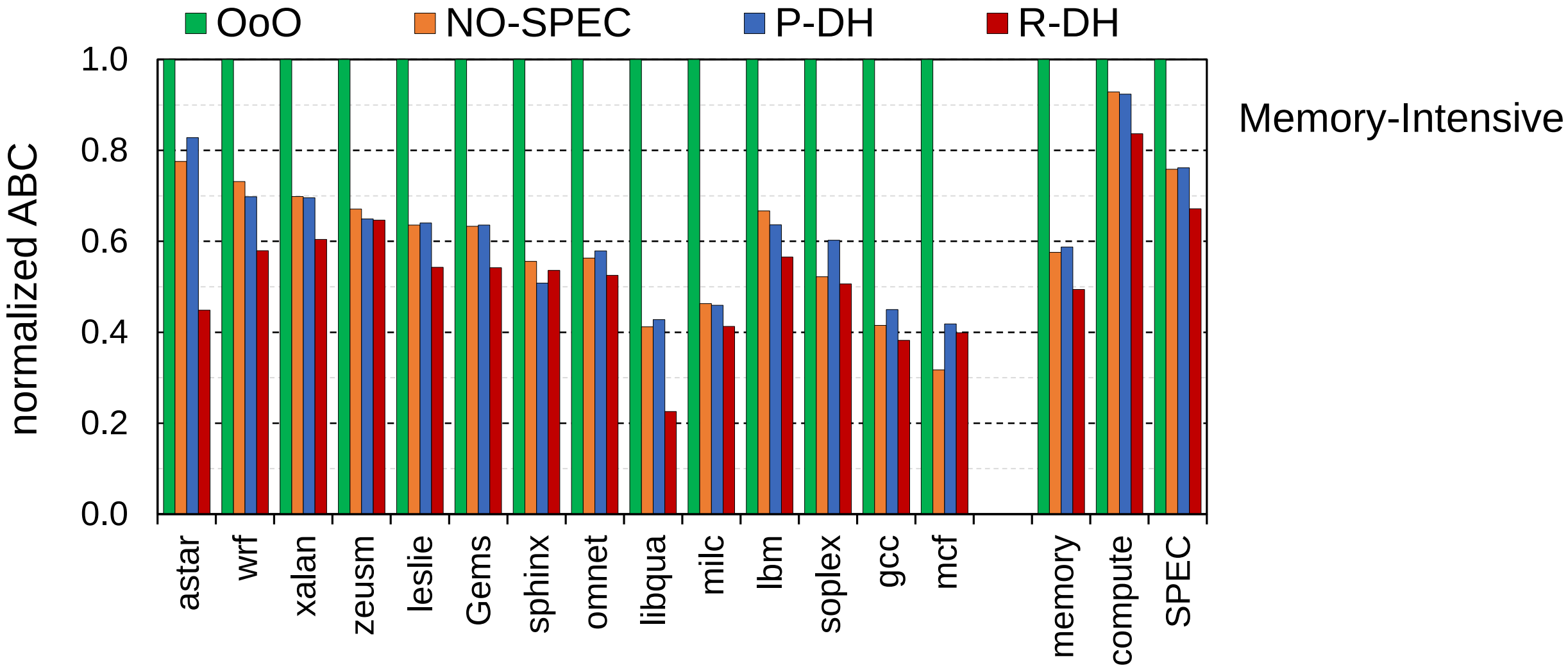
P-DH: Proactive dispatch halting

R-DH: Reactive dispatch halting

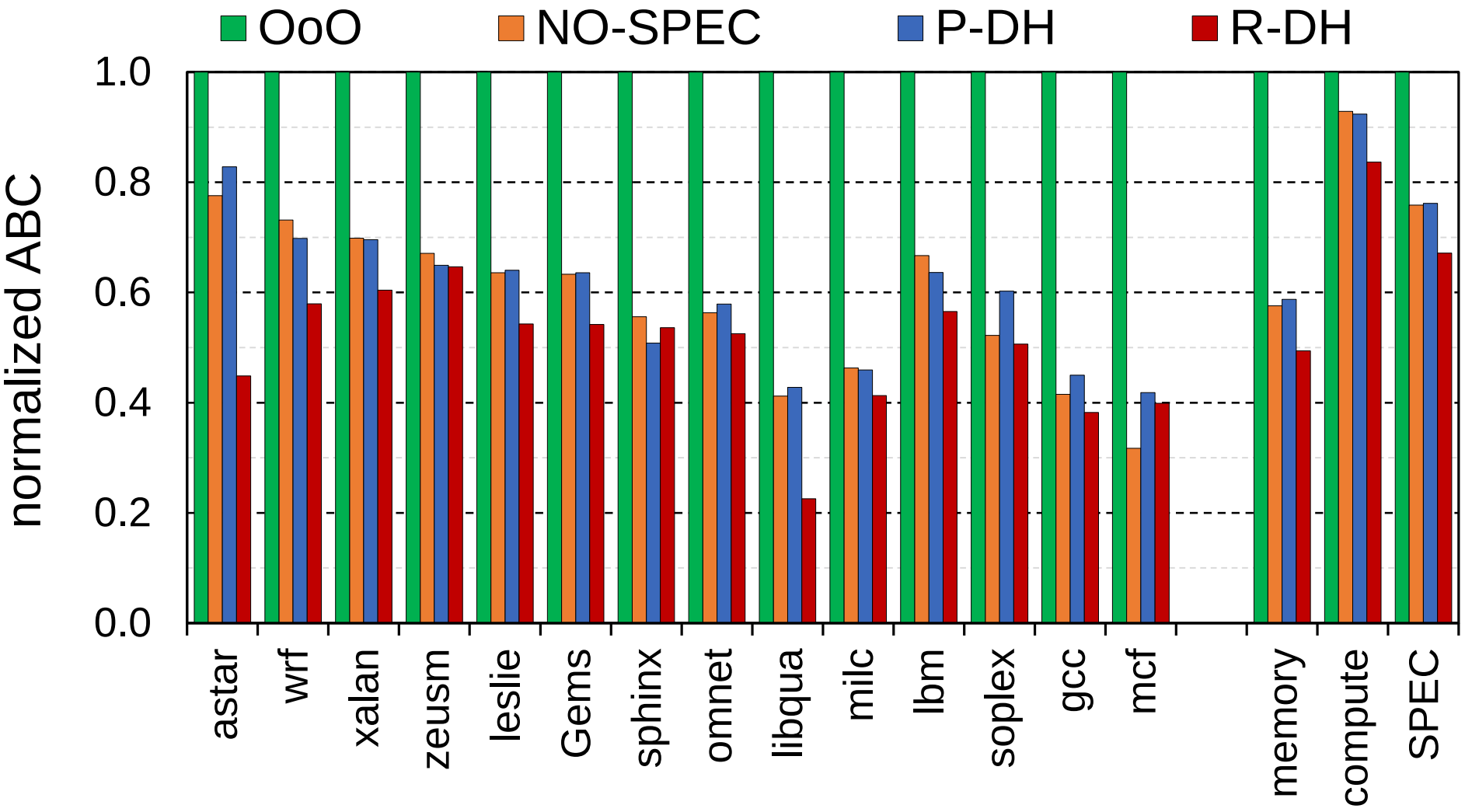
# Evaluation -- Reliability



# Evaluation -- Reliability



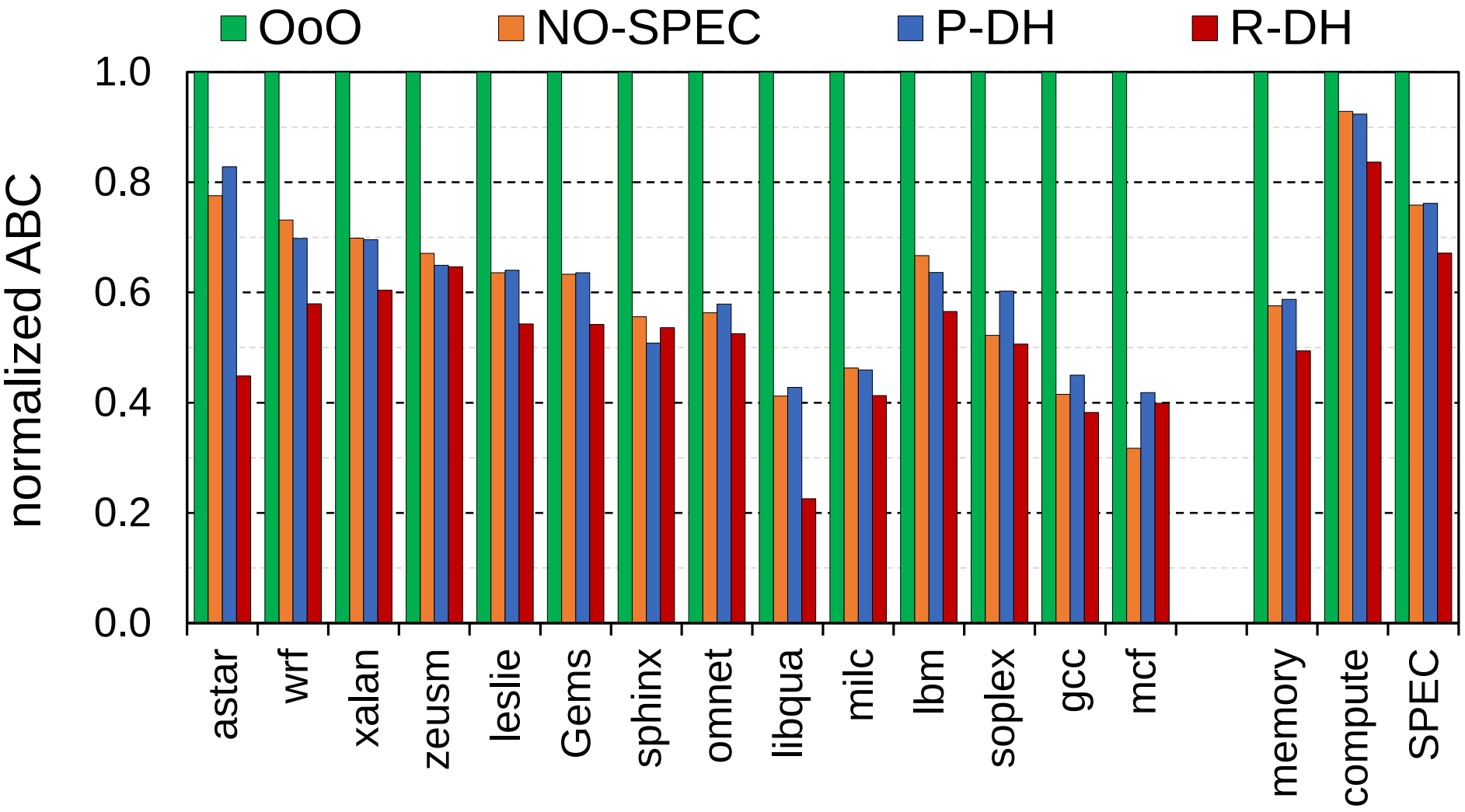
# Evaluation -- Reliability




Memory-Intensive

P-DH: 41.3%

# Evaluation -- Reliability

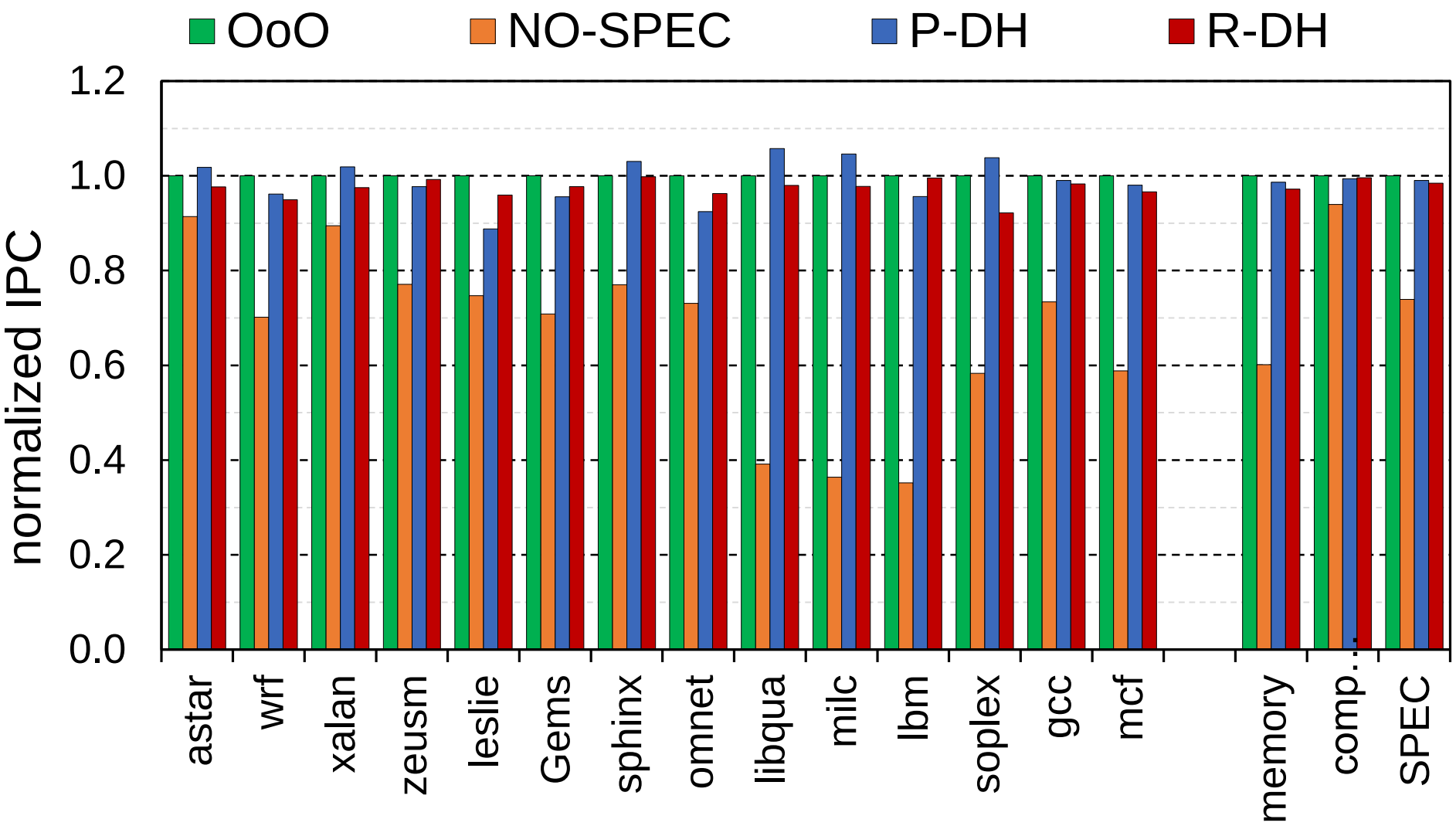



Memory-Intensive

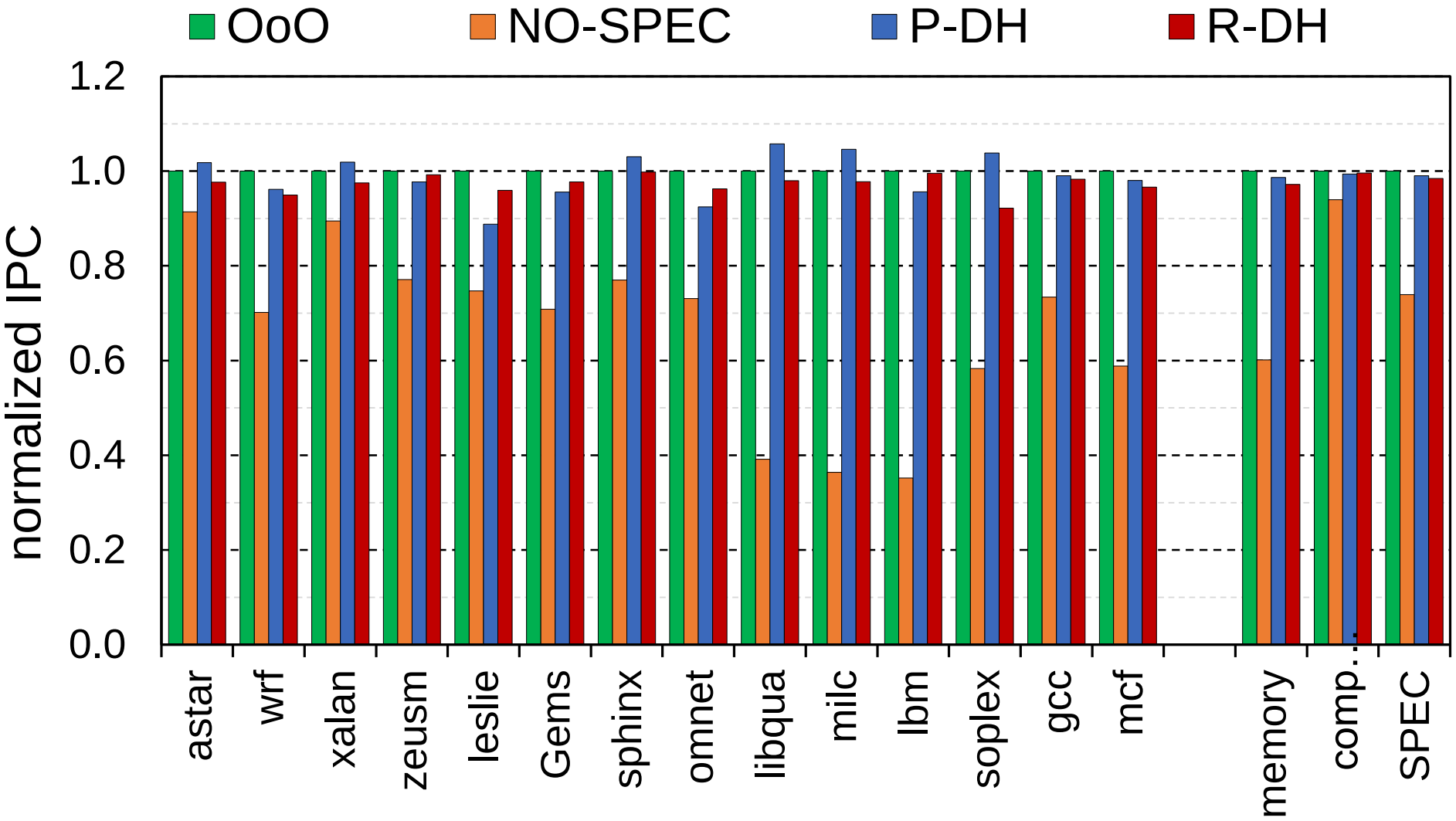
P-DH: 41.3%

R-DH: 50.6%

# Evaluation -- Performance



# Evaluation -- Performance



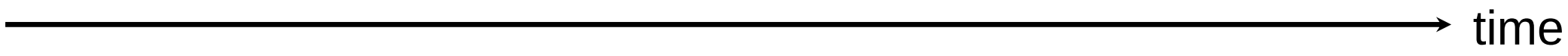

Dispatch halting does not degrade performance

# Contribution #3 [CAL 2019, HPCA 2020]

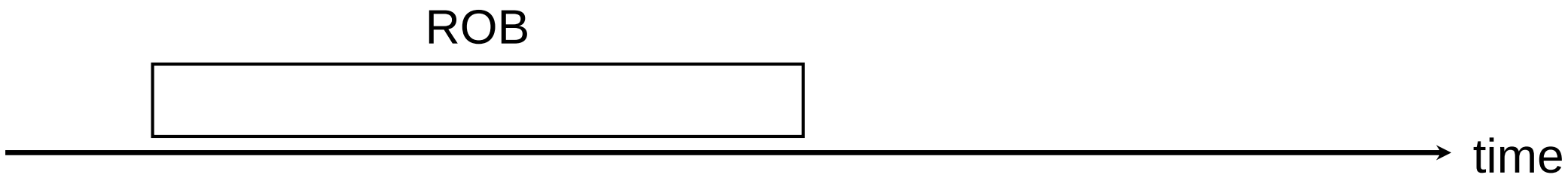
# Precise Runahead Execution to Improve Performance and Evaluate Reliability

# Full-ROB Stalls Degrade Performance

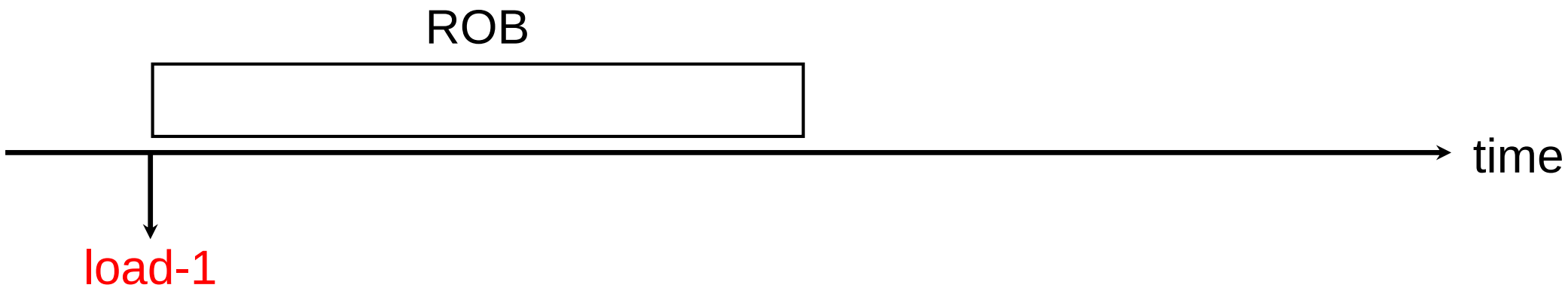
# Full-ROB Stalls Degrade Performance



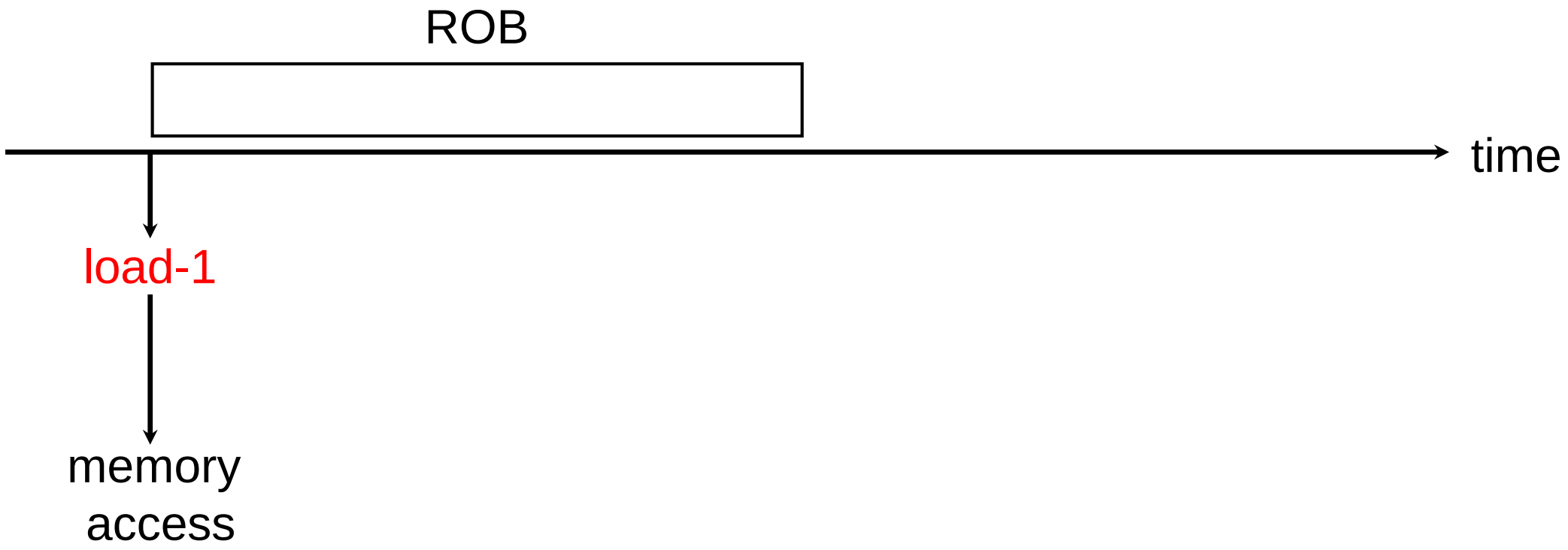
# Full-ROB Stalls Degrade Performance



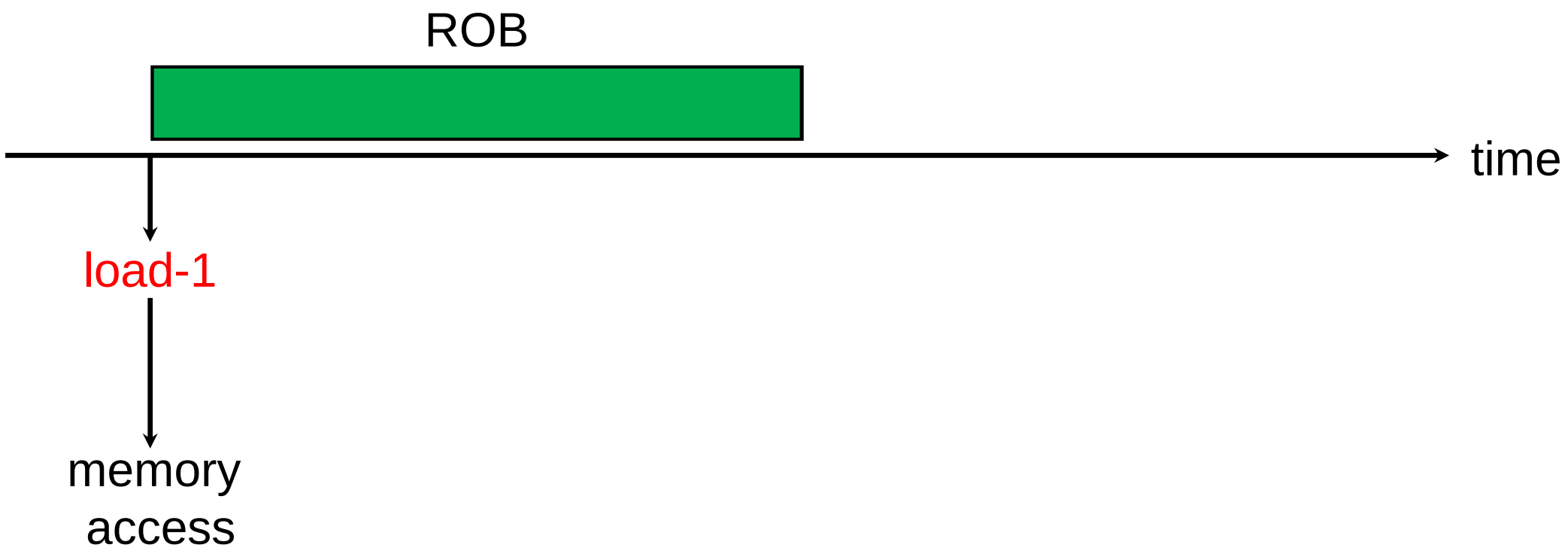
# Full-ROB Stalls Degrade Performance



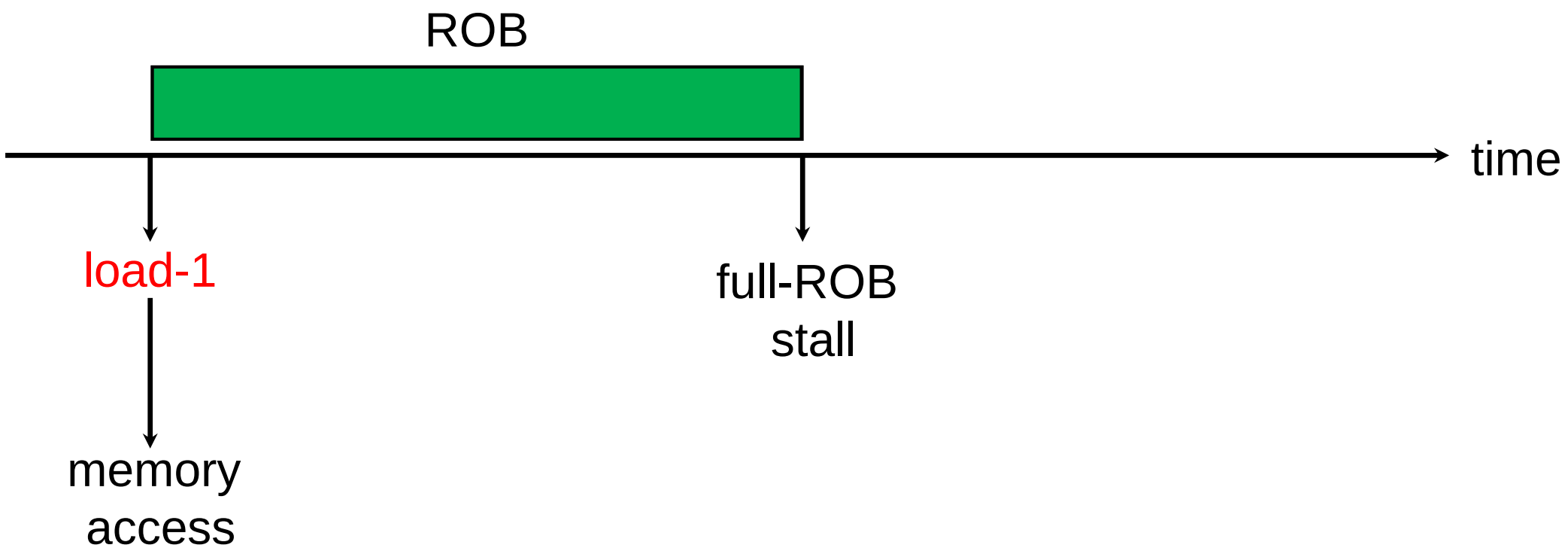
# Full-ROB Stalls Degrade Performance



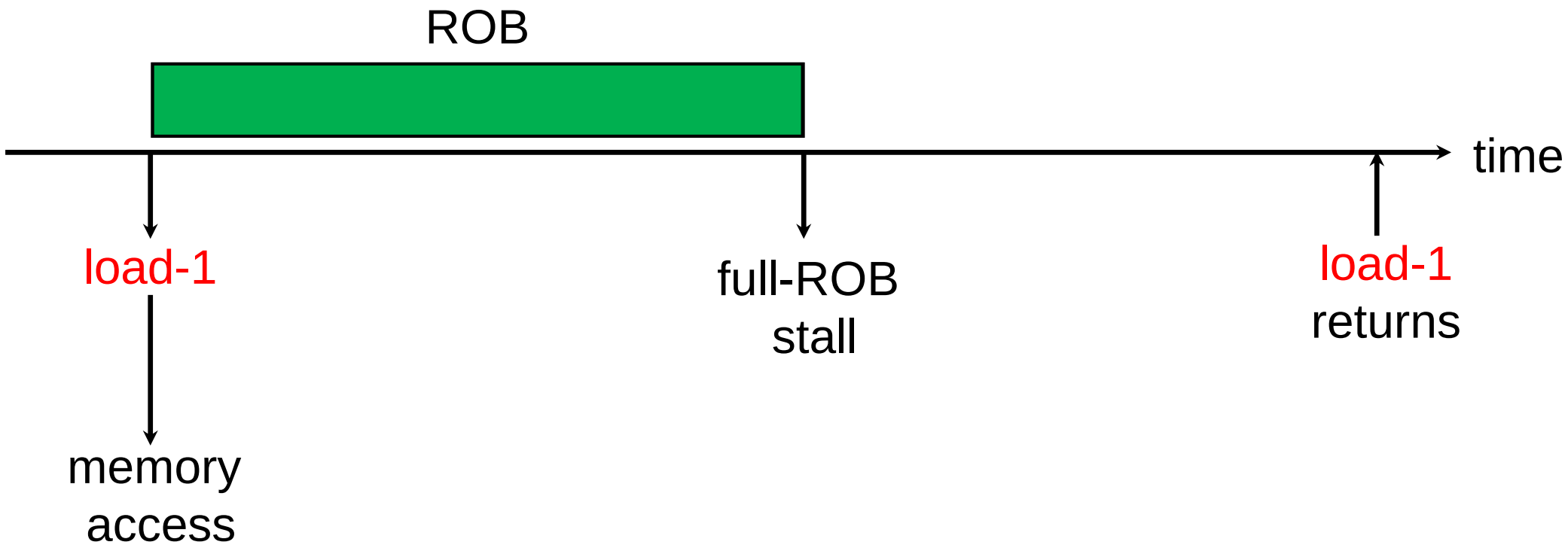
# Full-ROB Stalls Degrade Performance



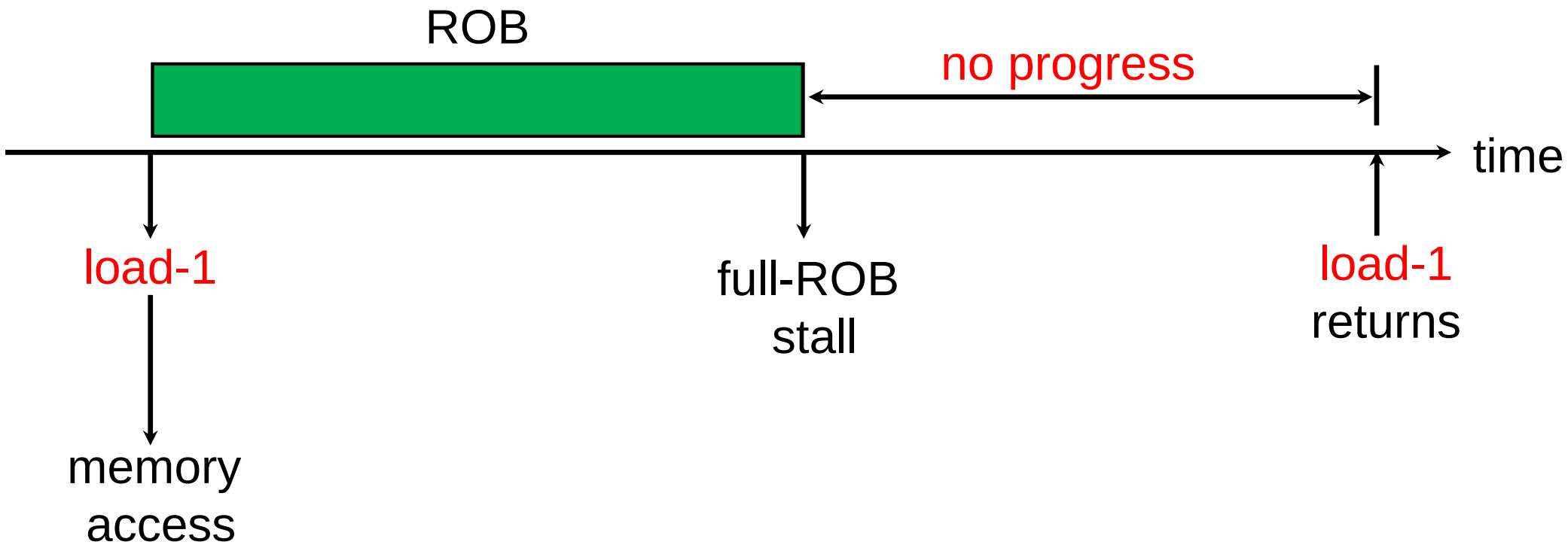
# Full-ROB Stalls Degrade Performance



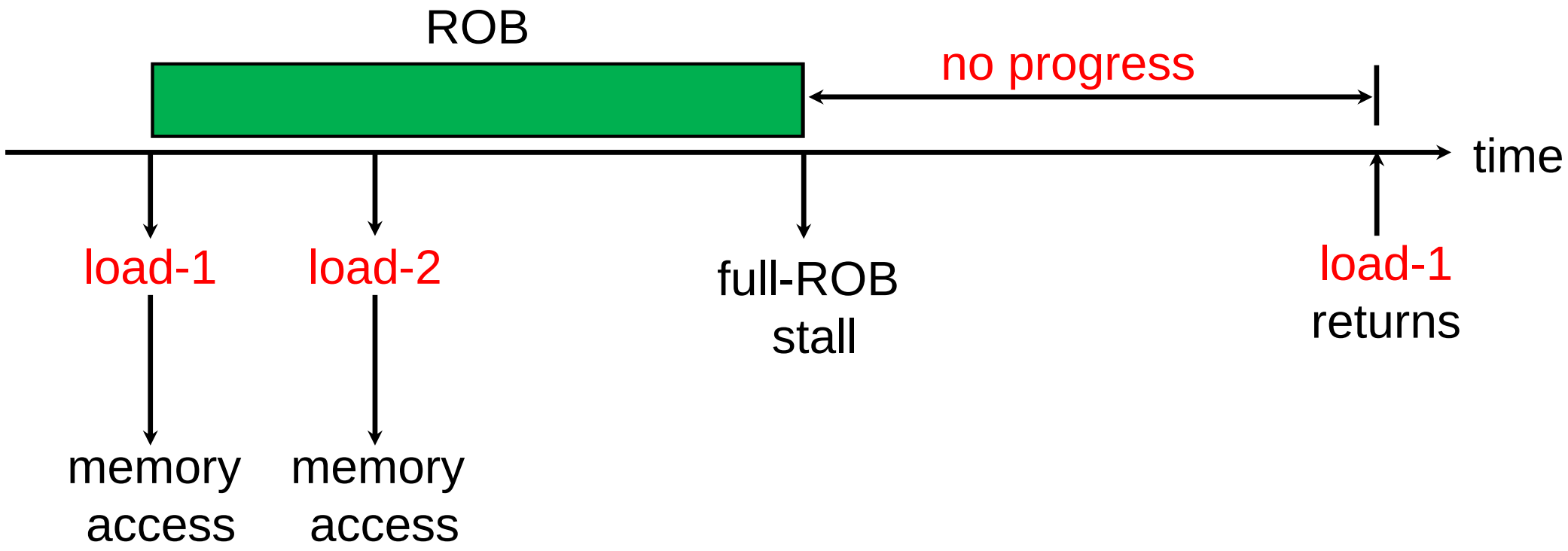
# Full-ROB Stalls Degrade Performance



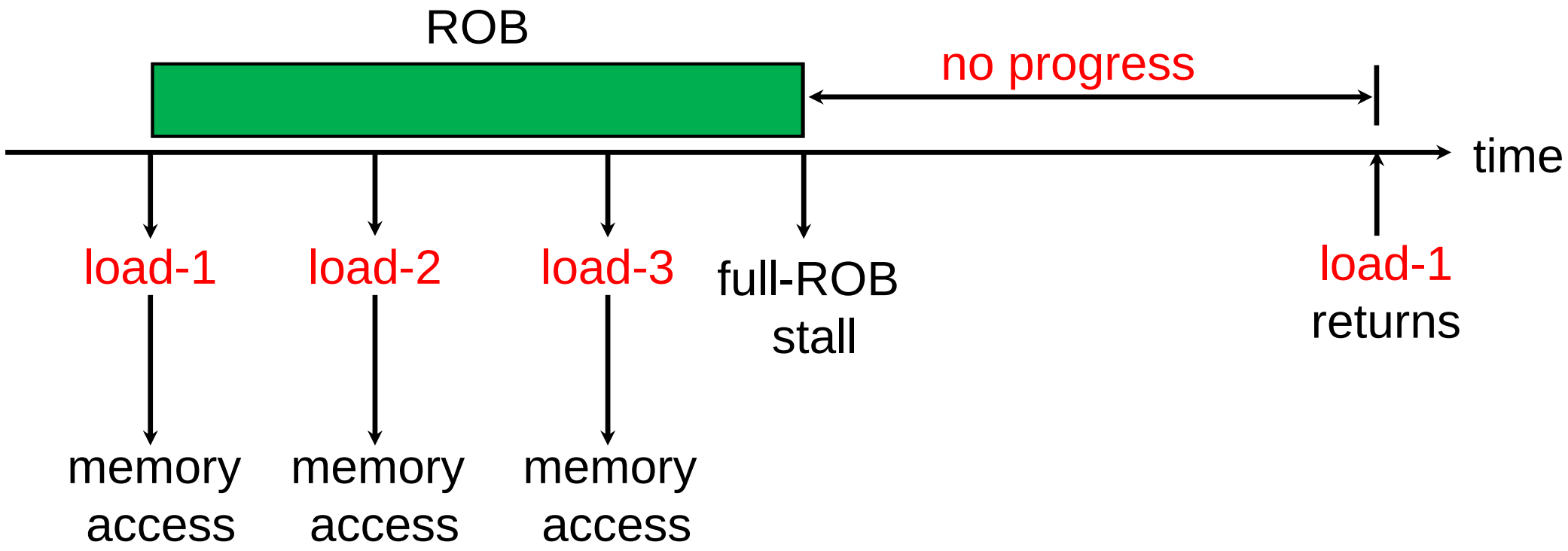
# Full-ROB Stalls Degrade Performance



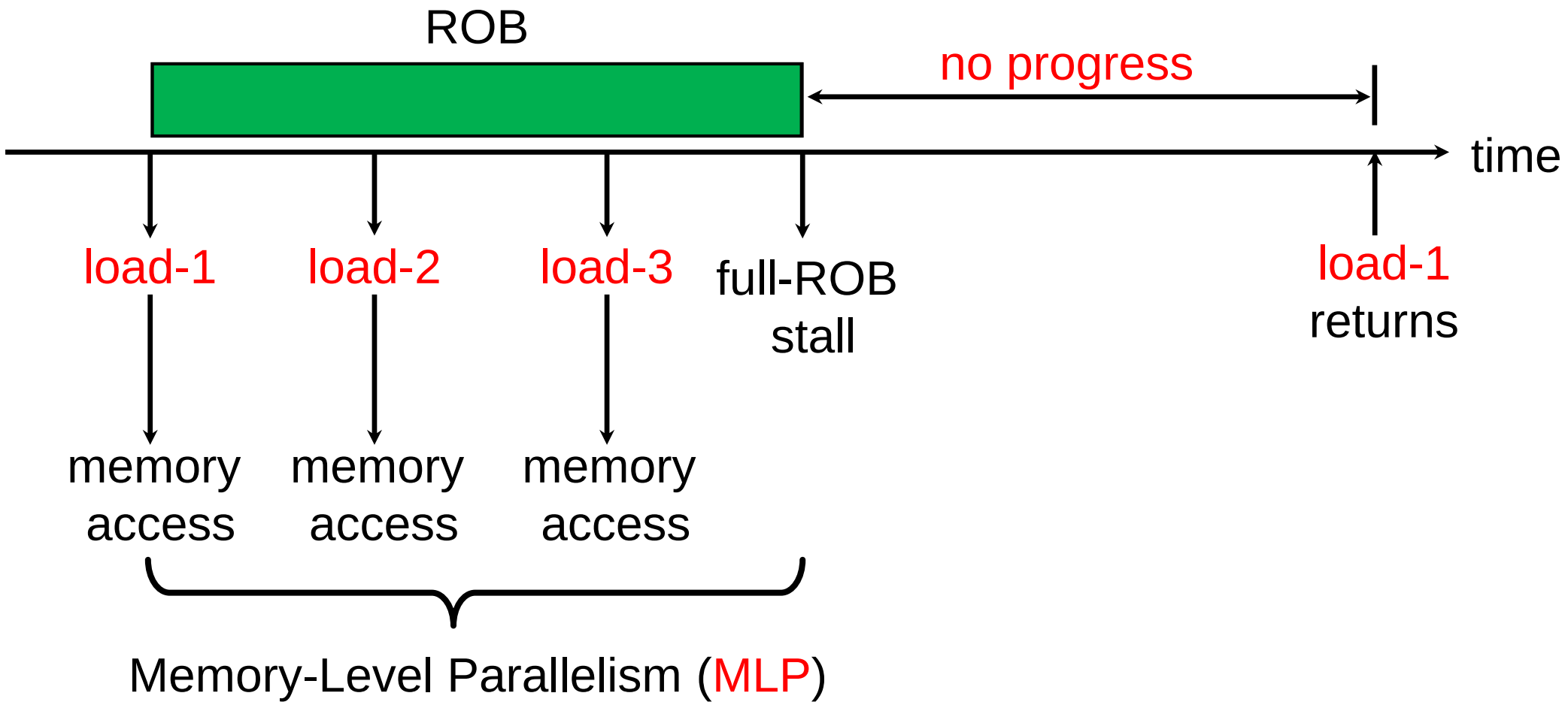
# Full-ROB Stalls Degrade Performance



# Full-ROB Stalls Degrade Performance



# Full-ROB Stalls Degrade Performance



# Full-ROB Stalls Degrade Performance

# Full-ROB Stalls Degrade Performance

**Larger ROB**

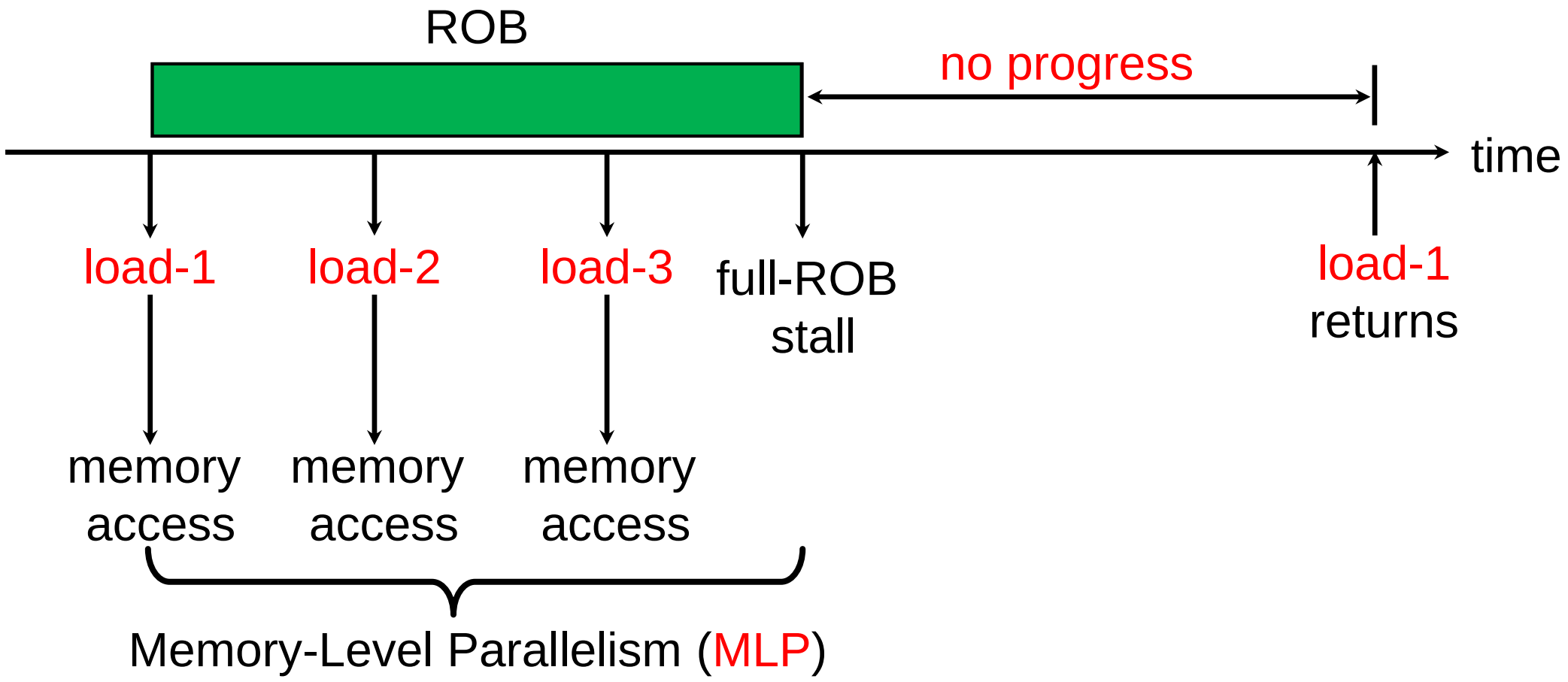
# Full-ROB Stalls Degrade Performance

**Larger ROB** ⇒ **Increased MLP**

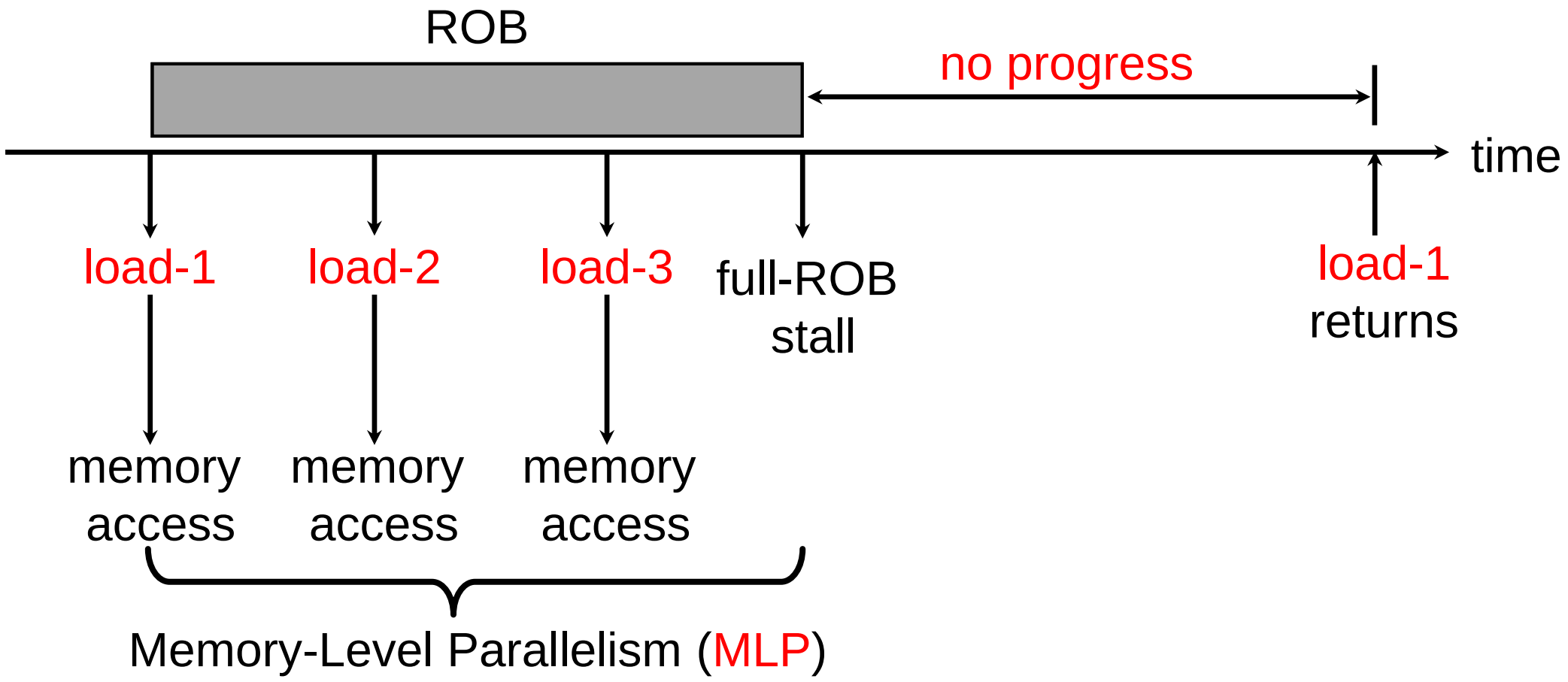
# Full-ROB Stalls Degrade Performance

**Larger ROB** ⇒ **Increased MLP** ⇒ **Better performance**

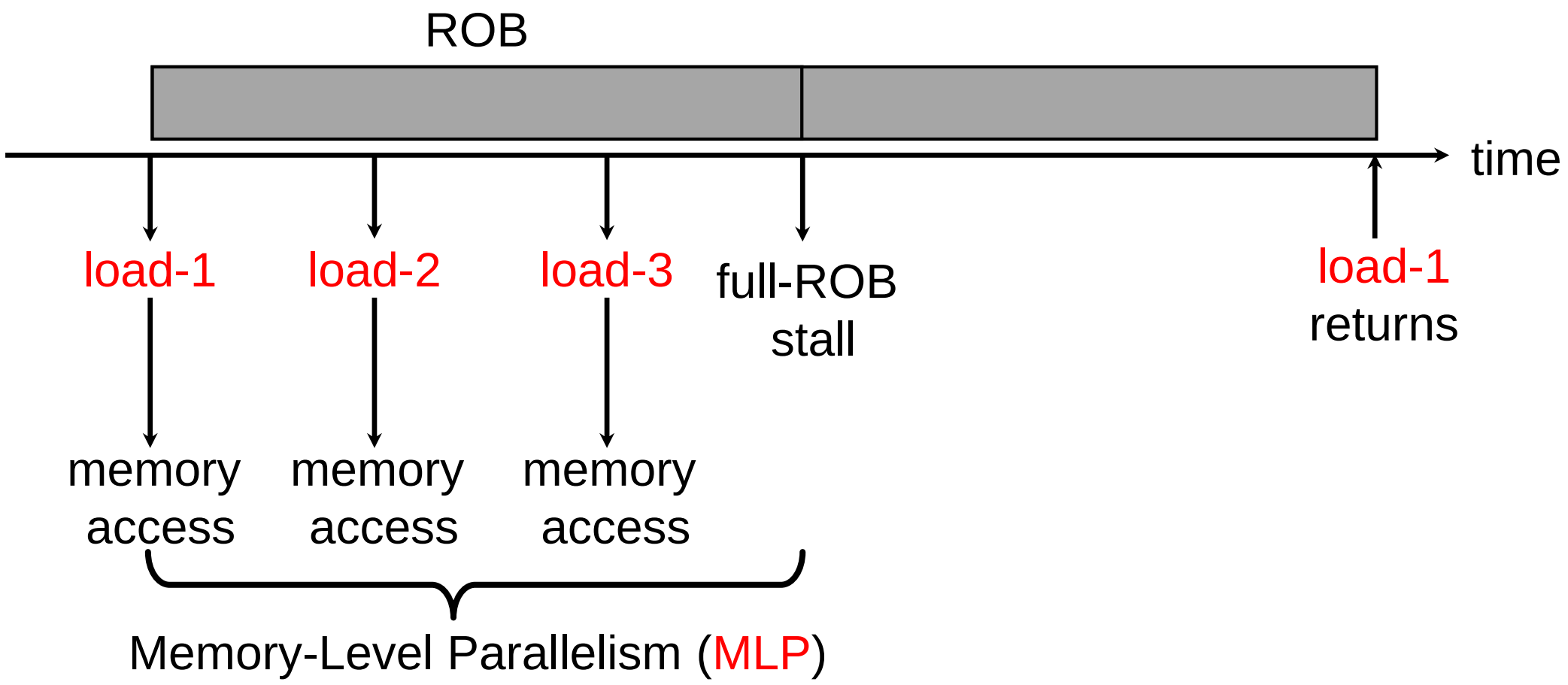
# Runahead Execution Prefetches under a Full-ROB Stall



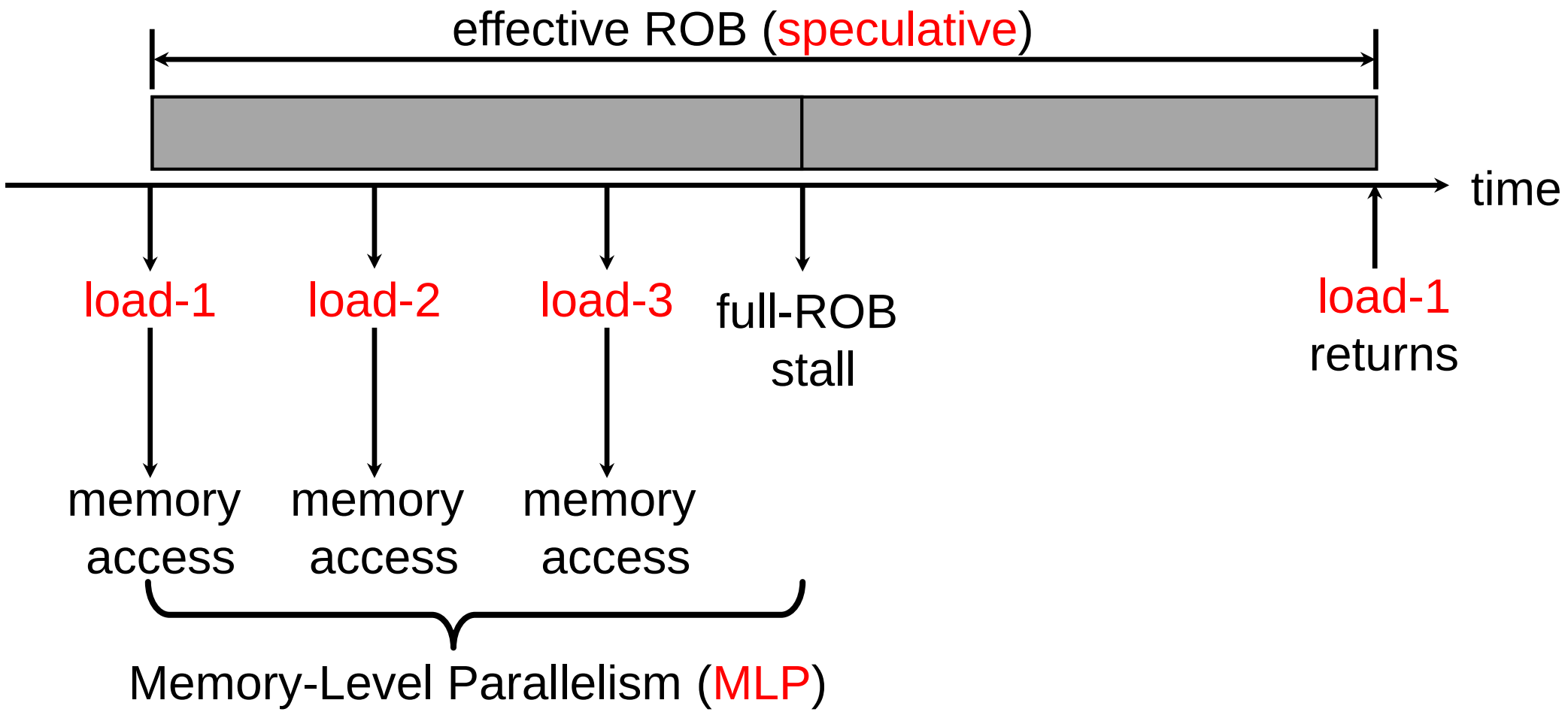
# Runahead Execution Prefetches under a Full-ROB Stall



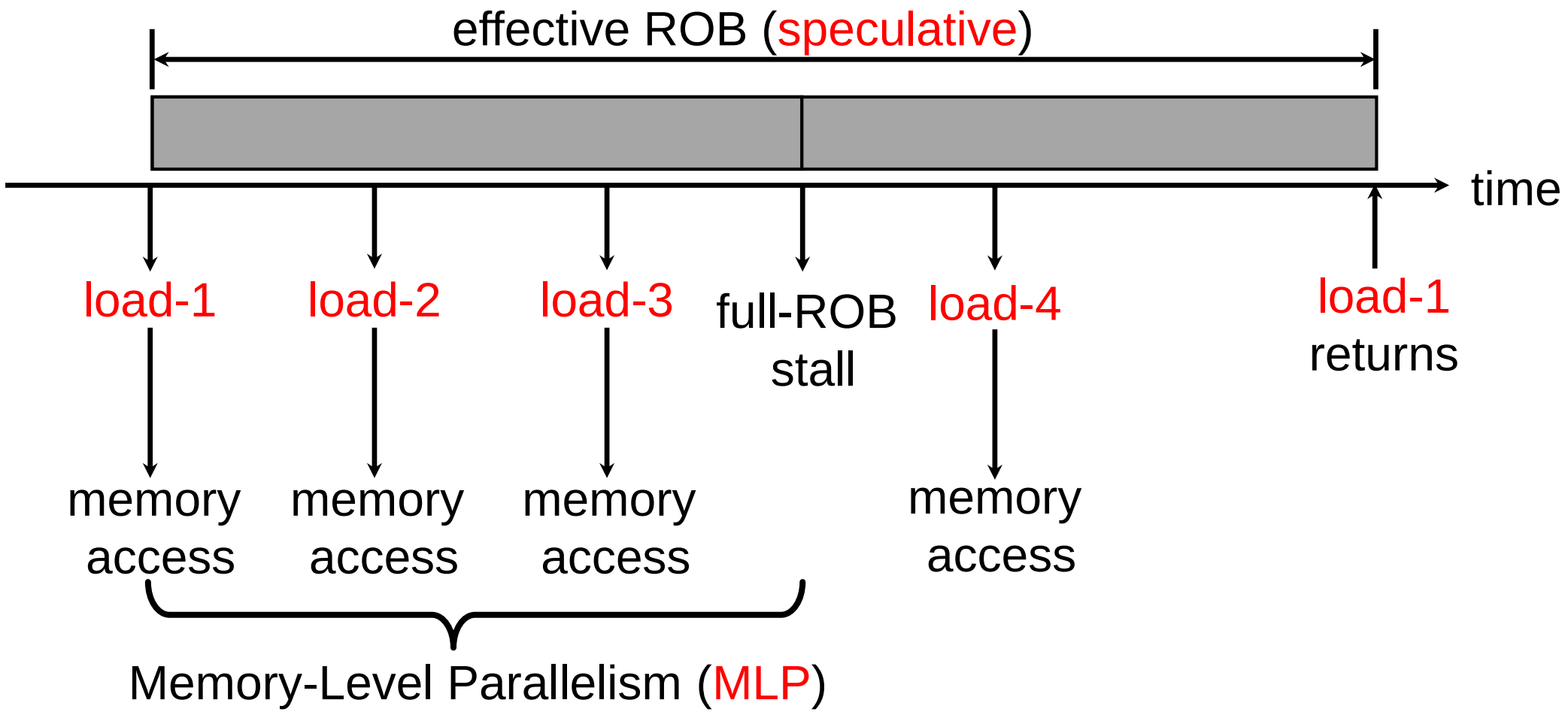
# Runahead Execution Prefetches under a Full-ROB Stall



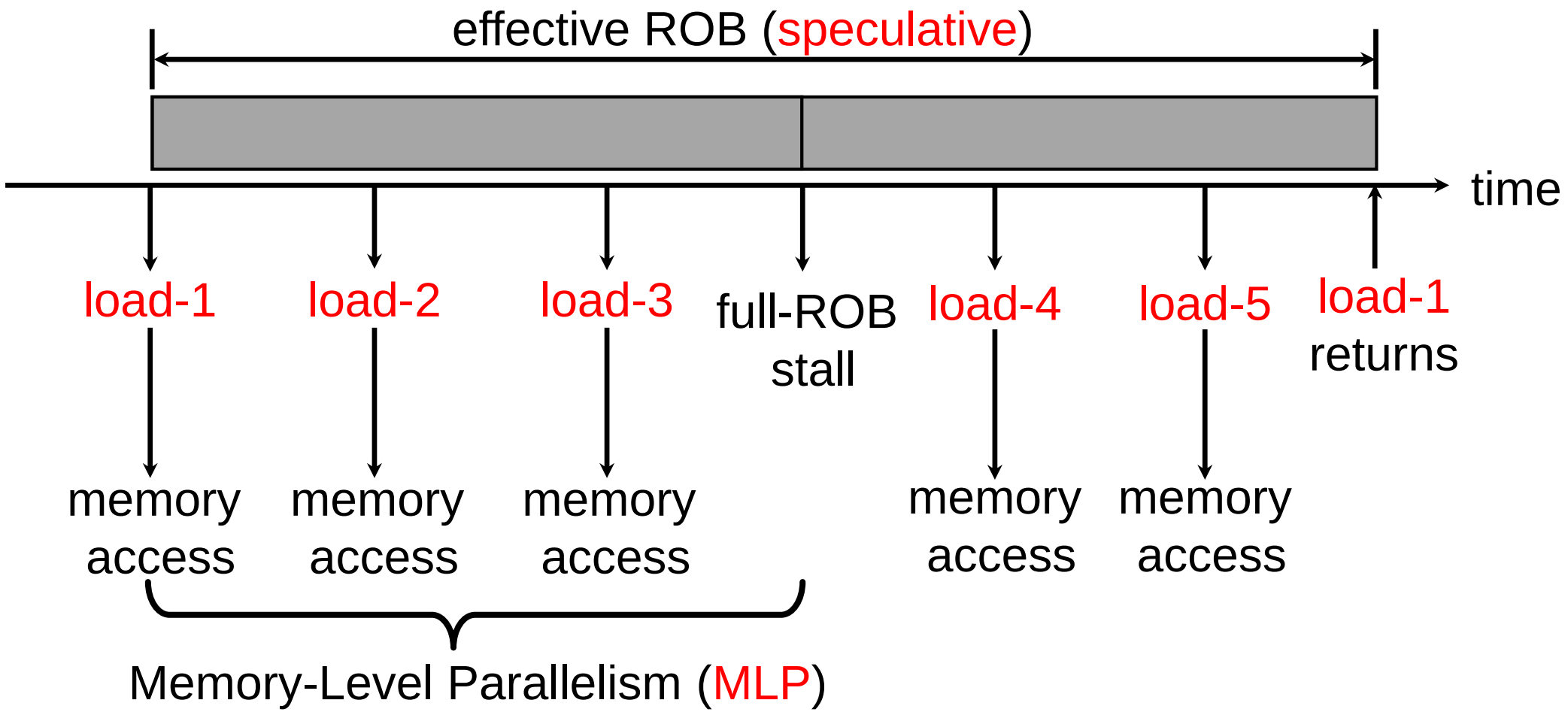
# Runahead Execution Prefetches under a Full-ROB Stall



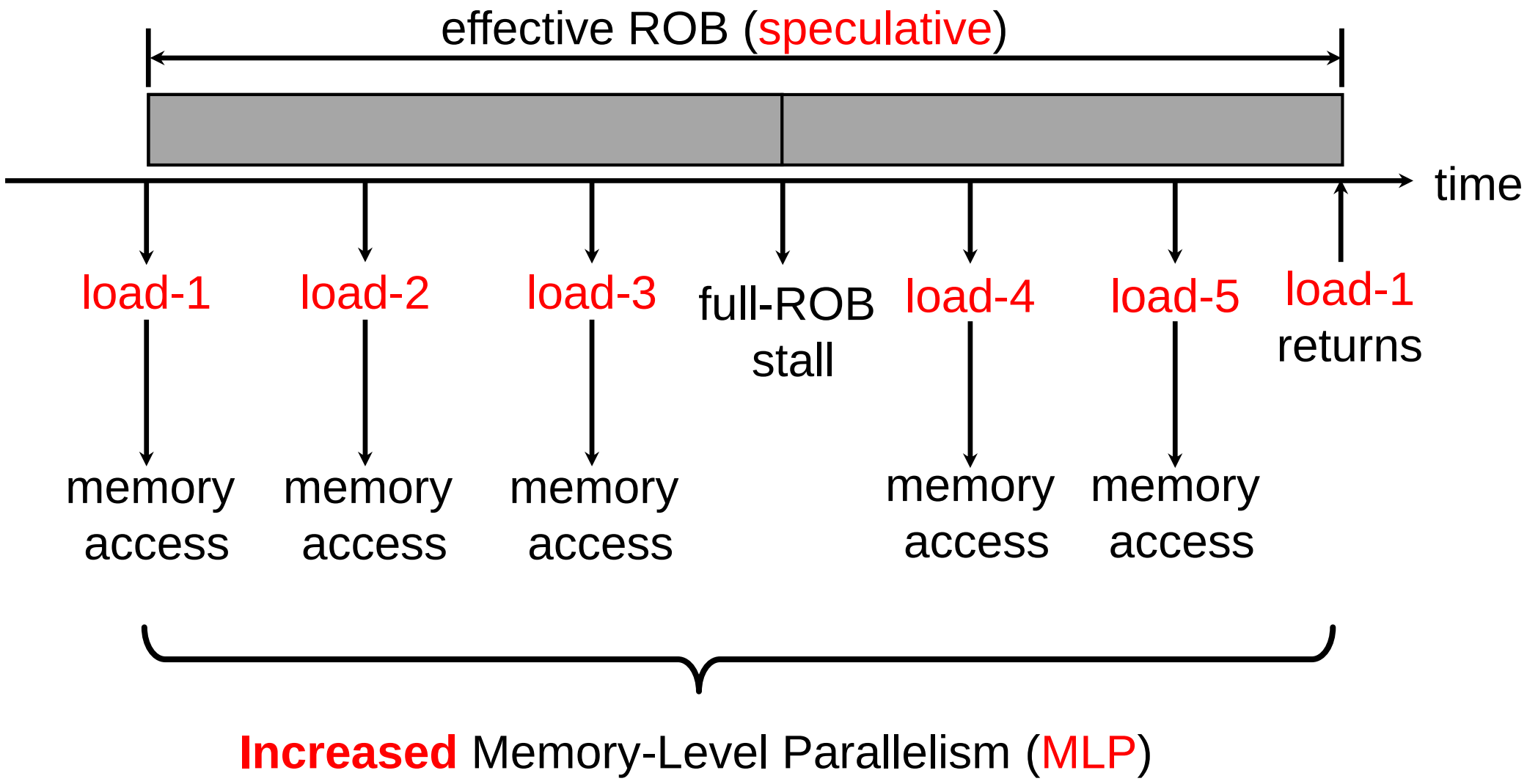
# Runahead Execution Prefetches under a Full-ROB Stall



# Runahead Execution Prefetches under a Full-ROB Stall



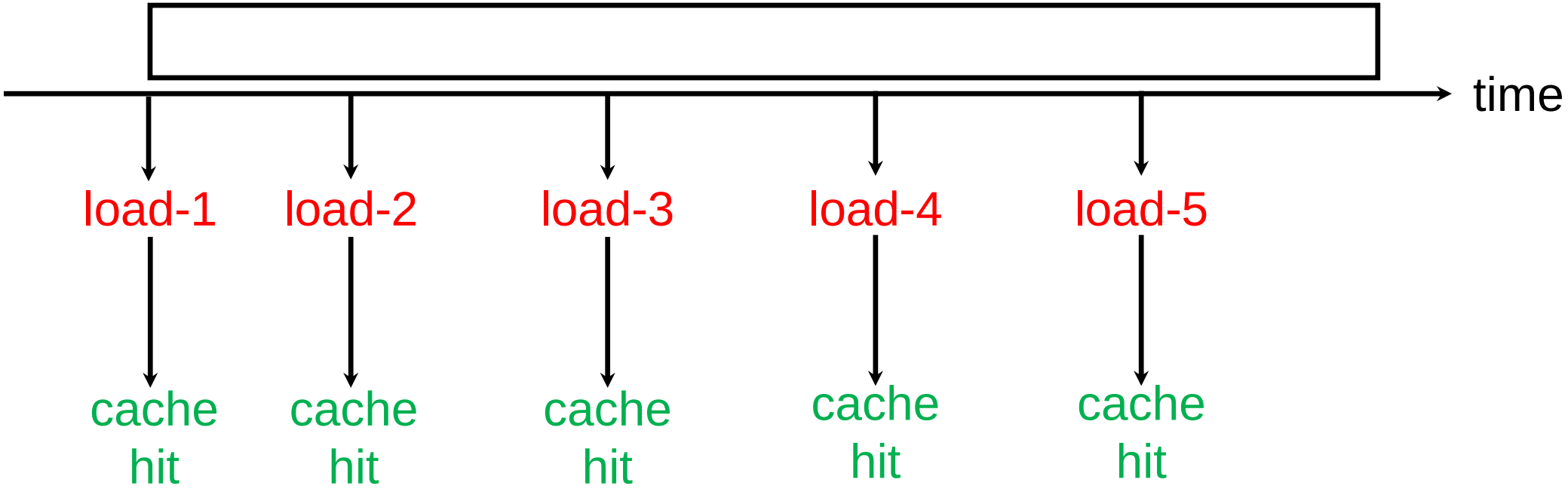
# Runahead Execution Prefetches under a Full-ROB Stall



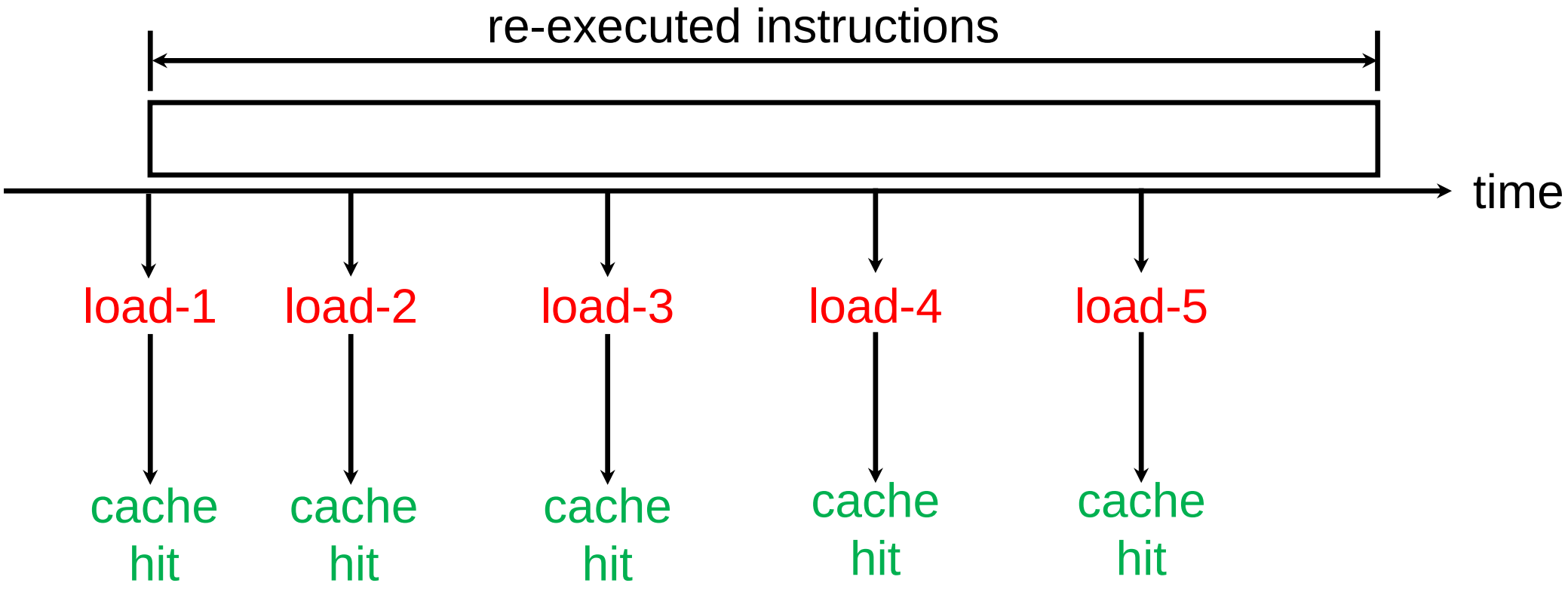
# Runahead Execution Re-Executes Instructions

# Runahead Execution Re-Executes Instructions

# Runahead Execution Re-Executes Instructions



# Runahead Execution Re-Executes Instructions



# Runahead Buffer Finds the Most Critical Slice of Instructions

# Runahead Buffer Finds the Most Critical Slice of Instructions

- Finds the stalling slice of instructions in ROB

# Runahead Buffer Finds the Most Critical Slice of Instructions

- Finds the stalling slice of instructions in ROB
- Executes only this slice after a full-ROB stall

# Runahead Buffer Finds the Most Critical Slice of Instructions

- Finds the stalling slice of instructions in ROB
- Executes only this slice after a full-ROB stall
- Clock-gates the front-end for saving power

# Runahead Buffer Finds the Most Critical Slice of Instructions

- Finds the stalling slice of instructions in ROB
- Executes only this slice after a full-ROB stall
- Clock-gates the front-end for saving power

⇩

**Better energy-efficiency and similar performance**

# Runahead Techniques Relative to OoO Core

# Runahead Techniques Relative to OoO Core

| Runahead execution | Runahead buffer |
| --- | --- |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | High ↑ | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | High ↑ | |
| Energy Efficiency | | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | High ↑ | |
| Energy Efficiency | Low ↓ | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | High ↑ | |
| Energy Efficiency | Low ↓ | Same | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | |
| Performance | High ↑ | High ↑ | |
| Energy Efficiency | Low ↓ | Same | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | |
| Energy Efficiency | Low ↓ | Same | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |

# Precise Runahead Exection: Three Key Insights

# Precise Runahead Exection: Three Key Insights

1. No need to flush the ROB

# Precise Runahead Exection: Three Key Insights

1. No need to flush the ROB

2. Slices execute quickly

# Precise Runahead Exection: Three Key Insights

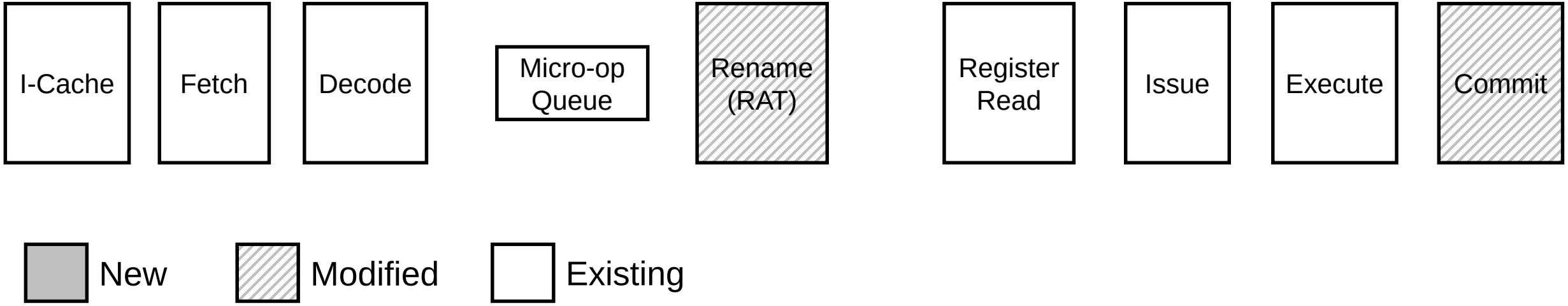
1. No need to flush the ROB

2. Slices execute quickly

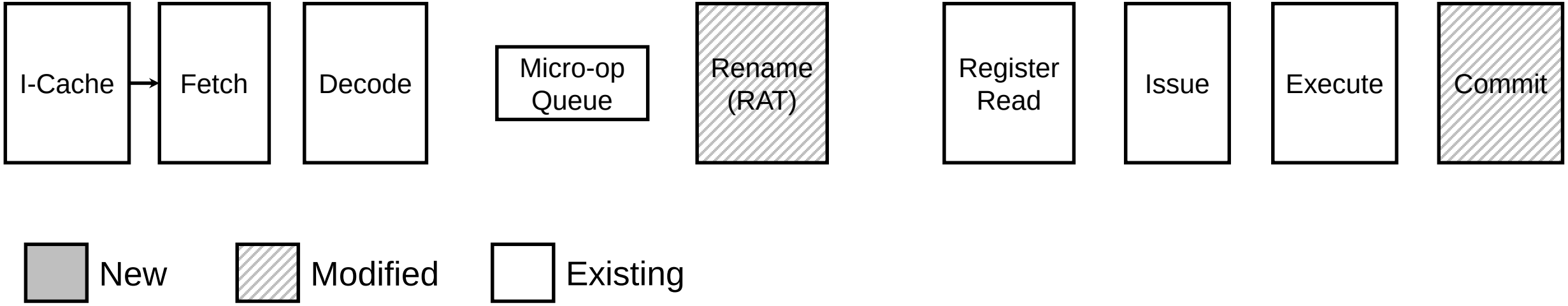
3. Only physical registers must be recycled

# Precise Runahead Execution

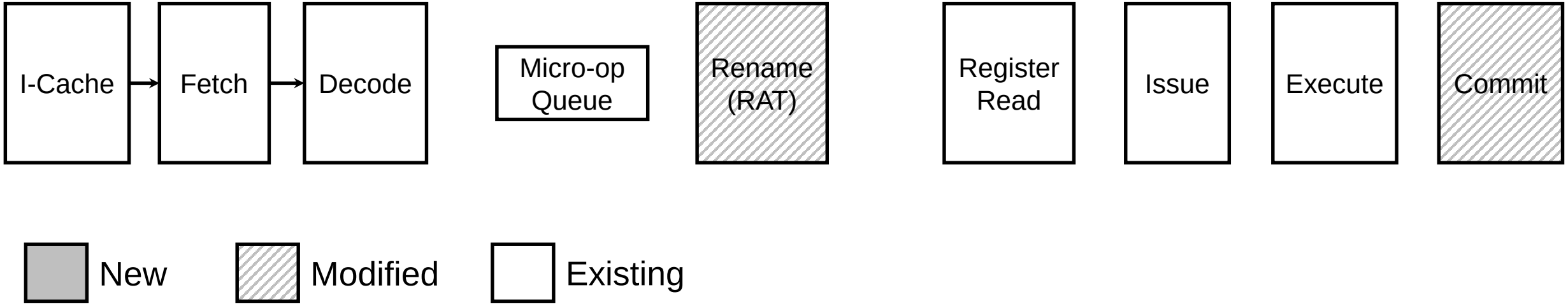
# Precise Runahead Execution



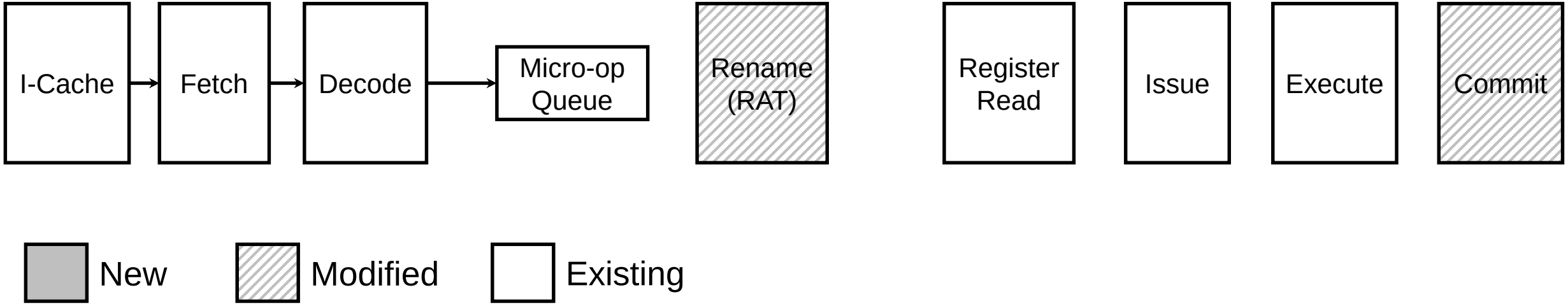
# Precise Runahead Execution



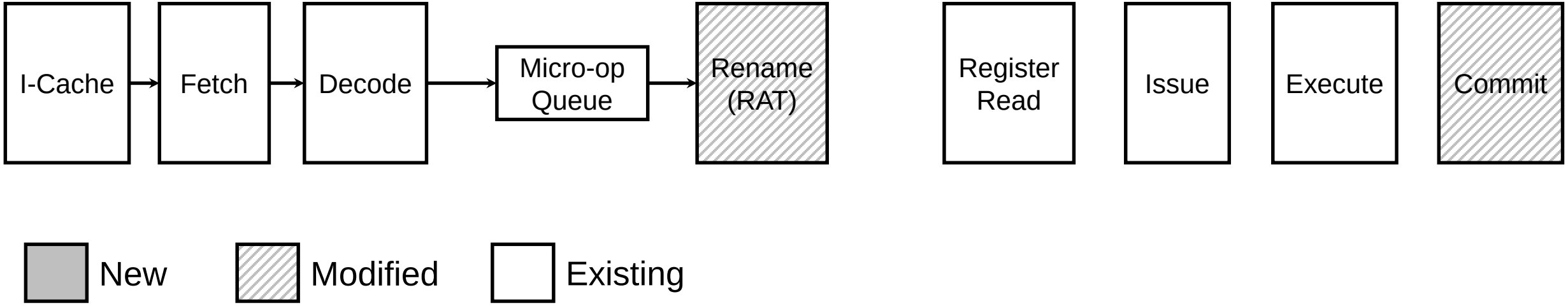
# Precise Runahead Execution



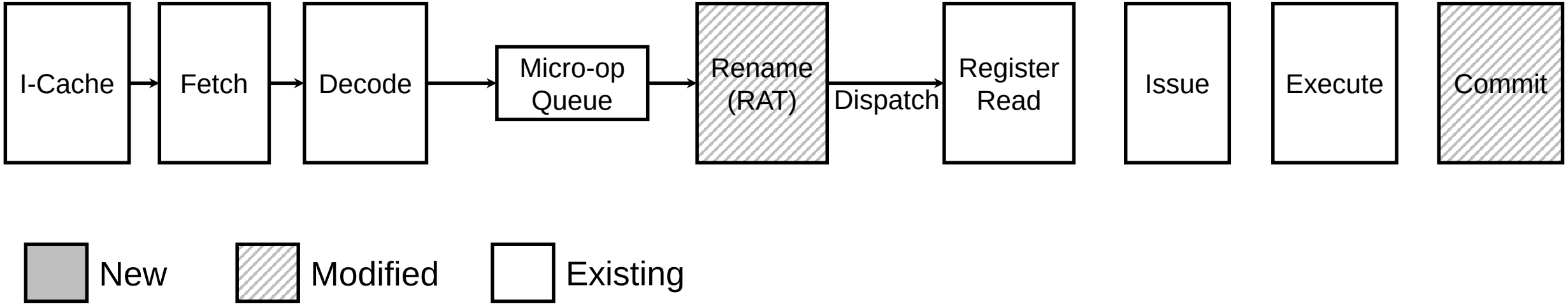
# Precise Runahead Execution



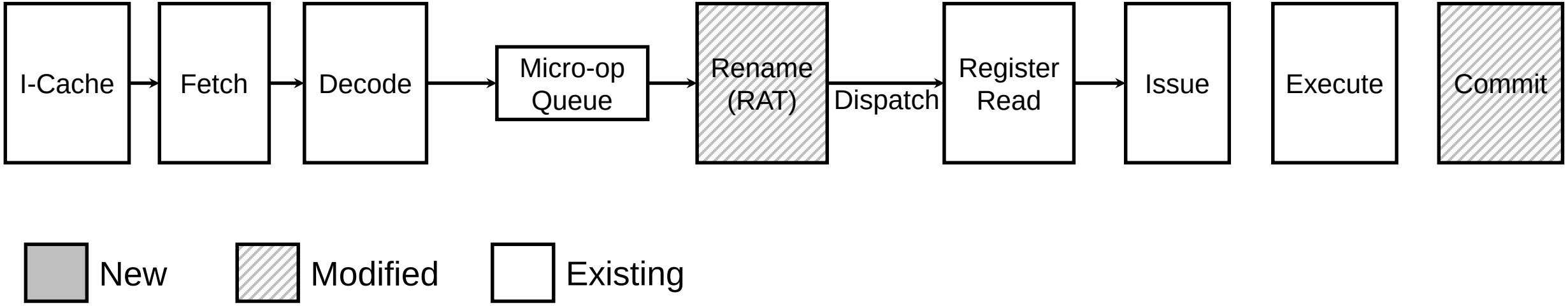
# Precise Runahead Execution



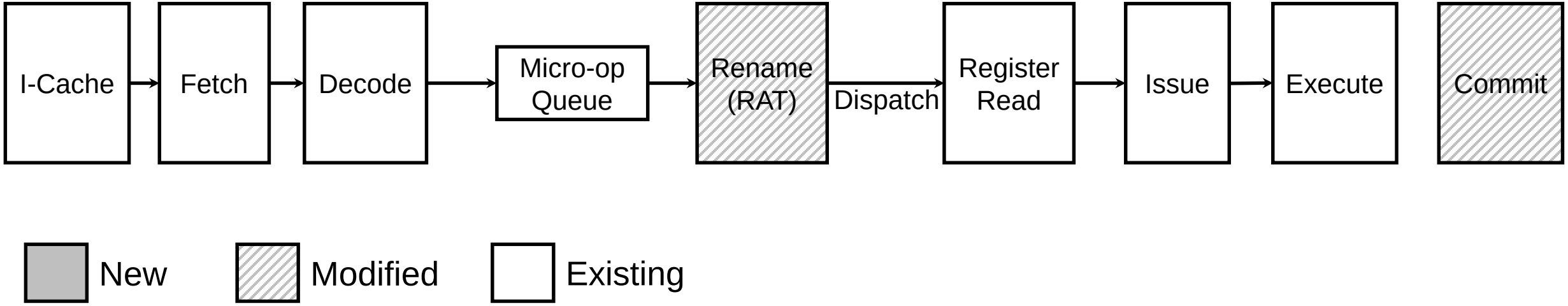
# Precise Runahead Execution



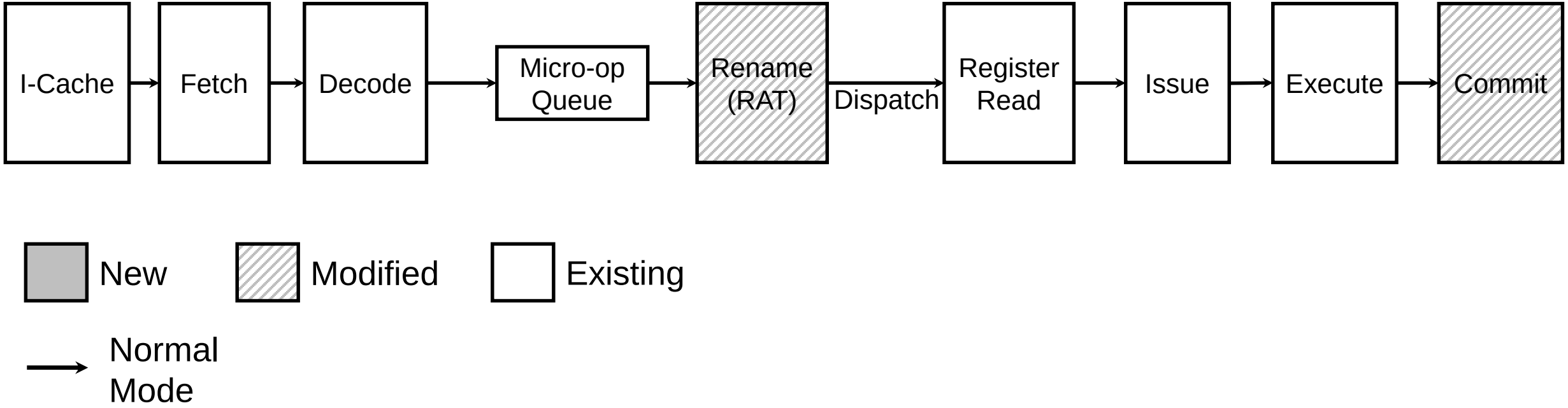
# Precise Runahead Execution



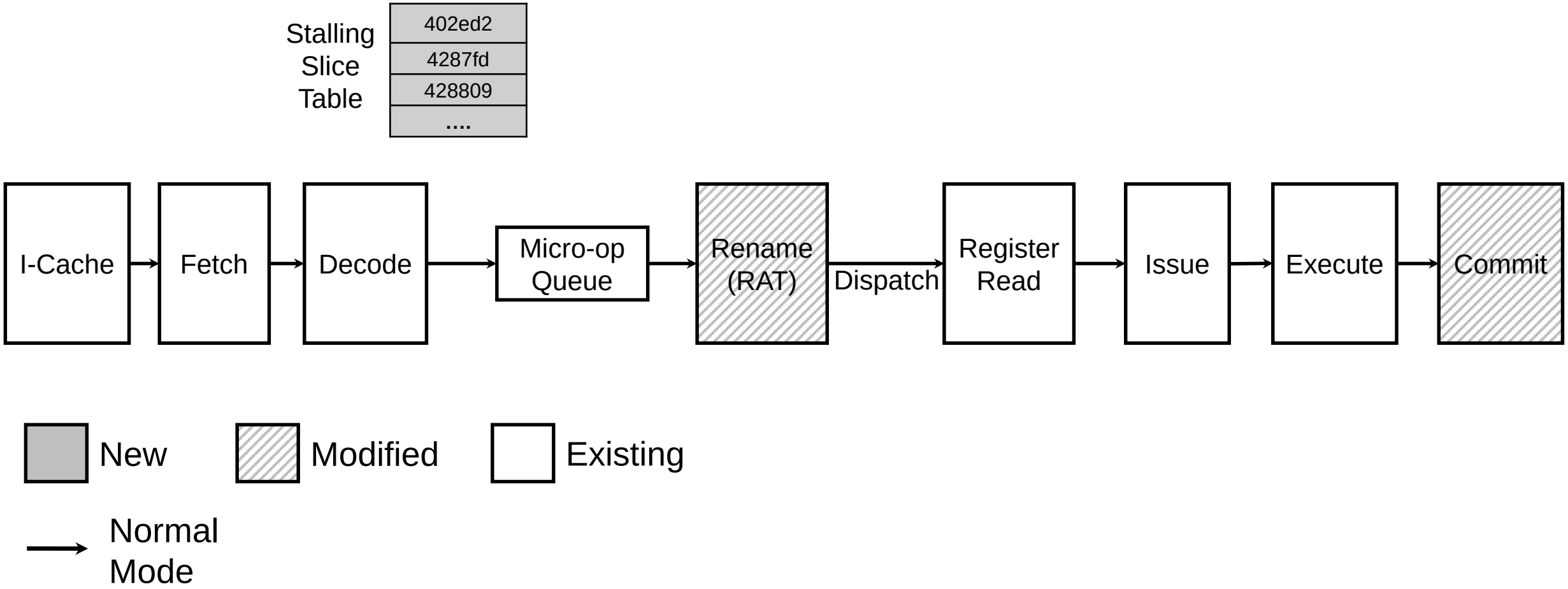
# Precise Runahead Execution



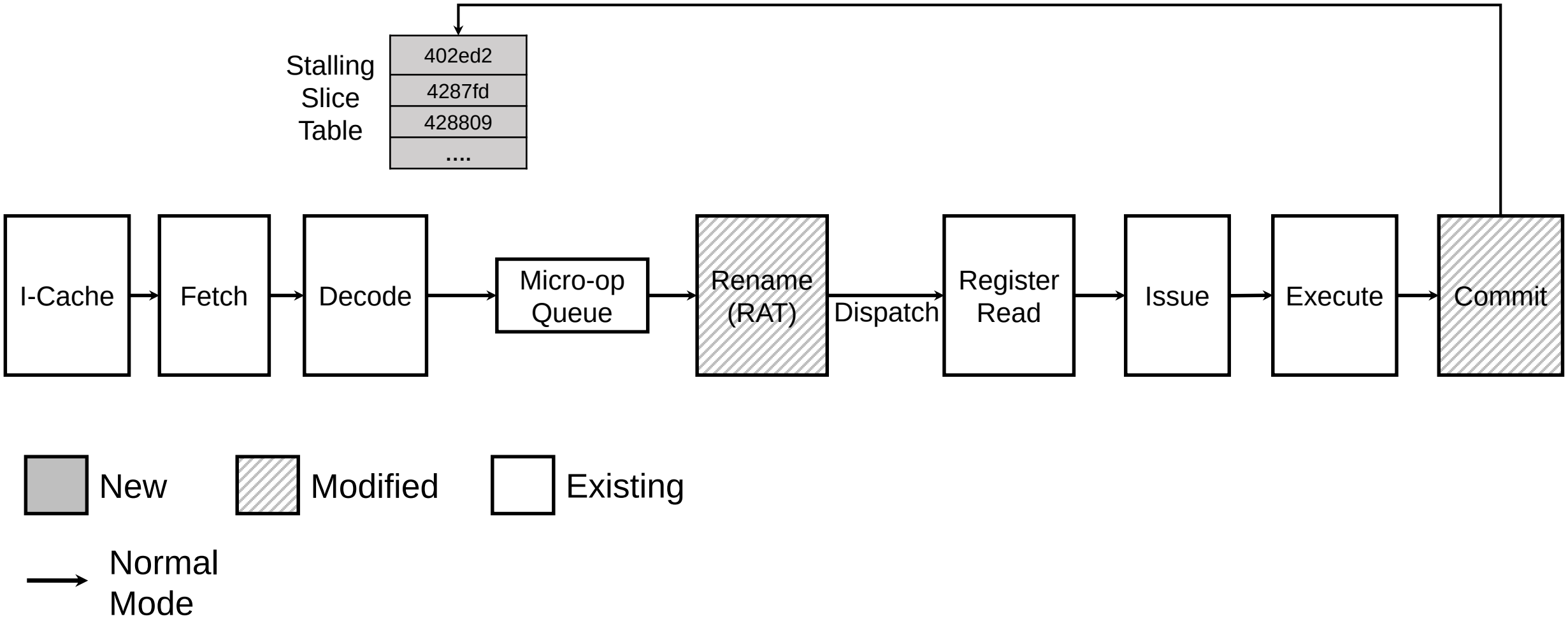
# Precise Runahead Execution



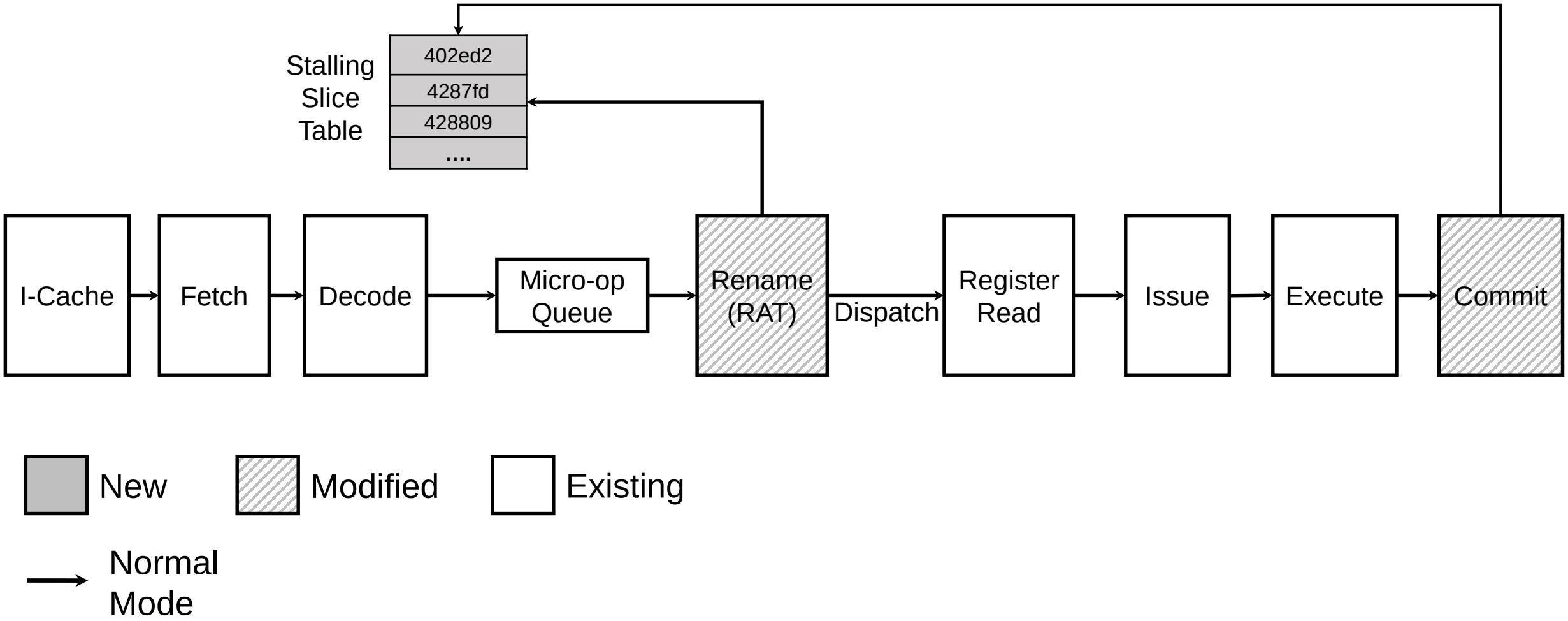
# Precise Runahead Execution



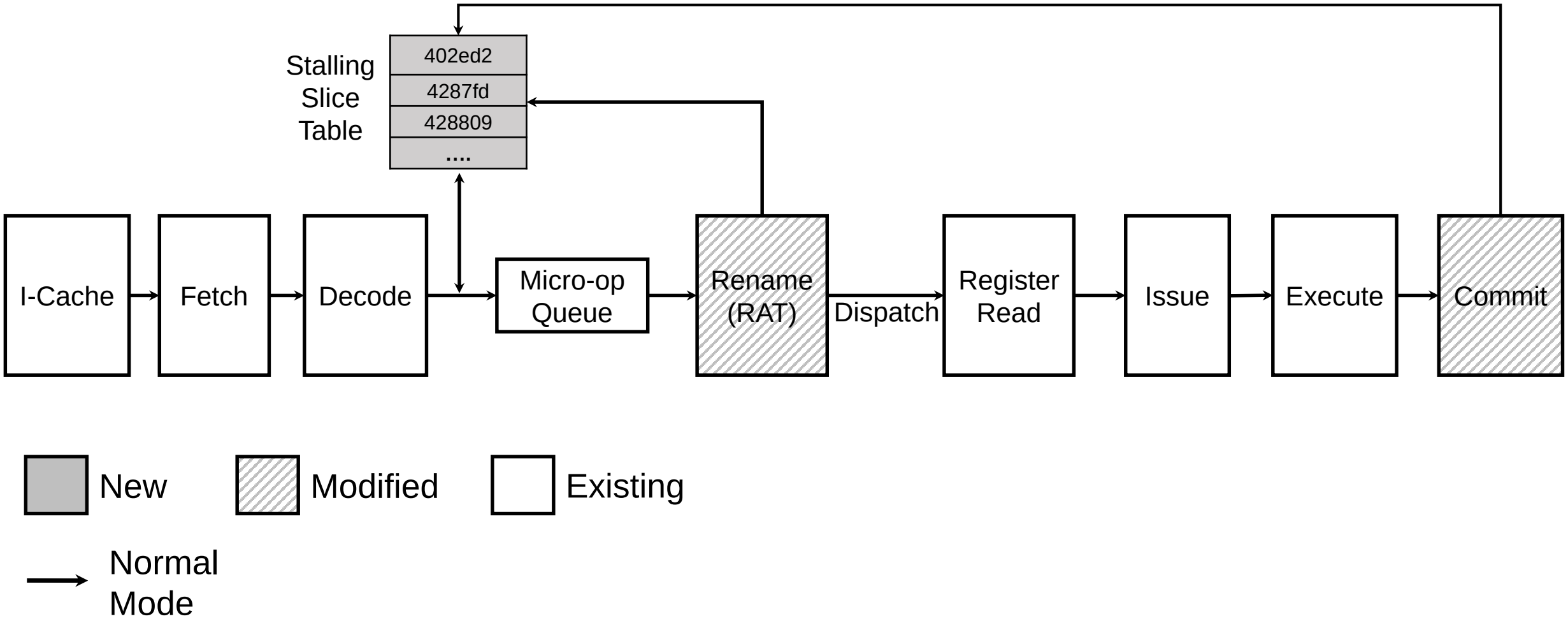
# Precise Runahead Execution



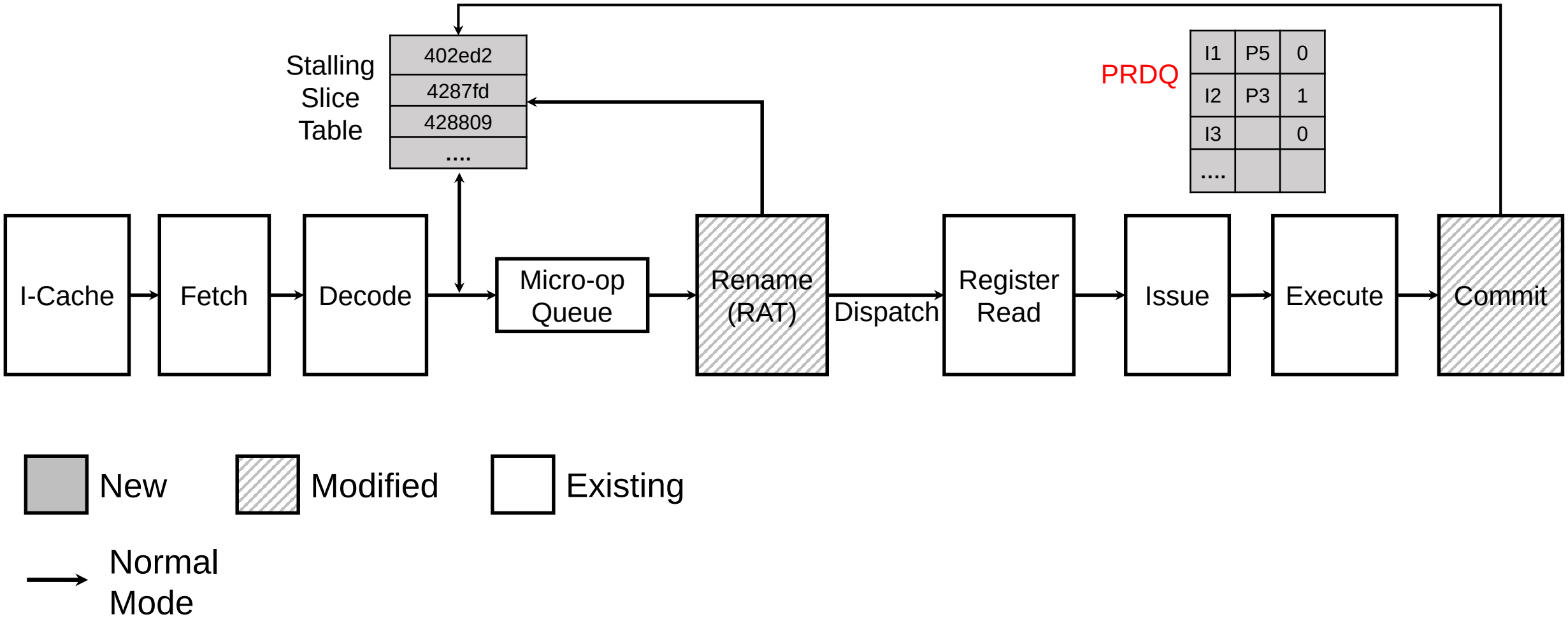
# Precise Runahead Execution



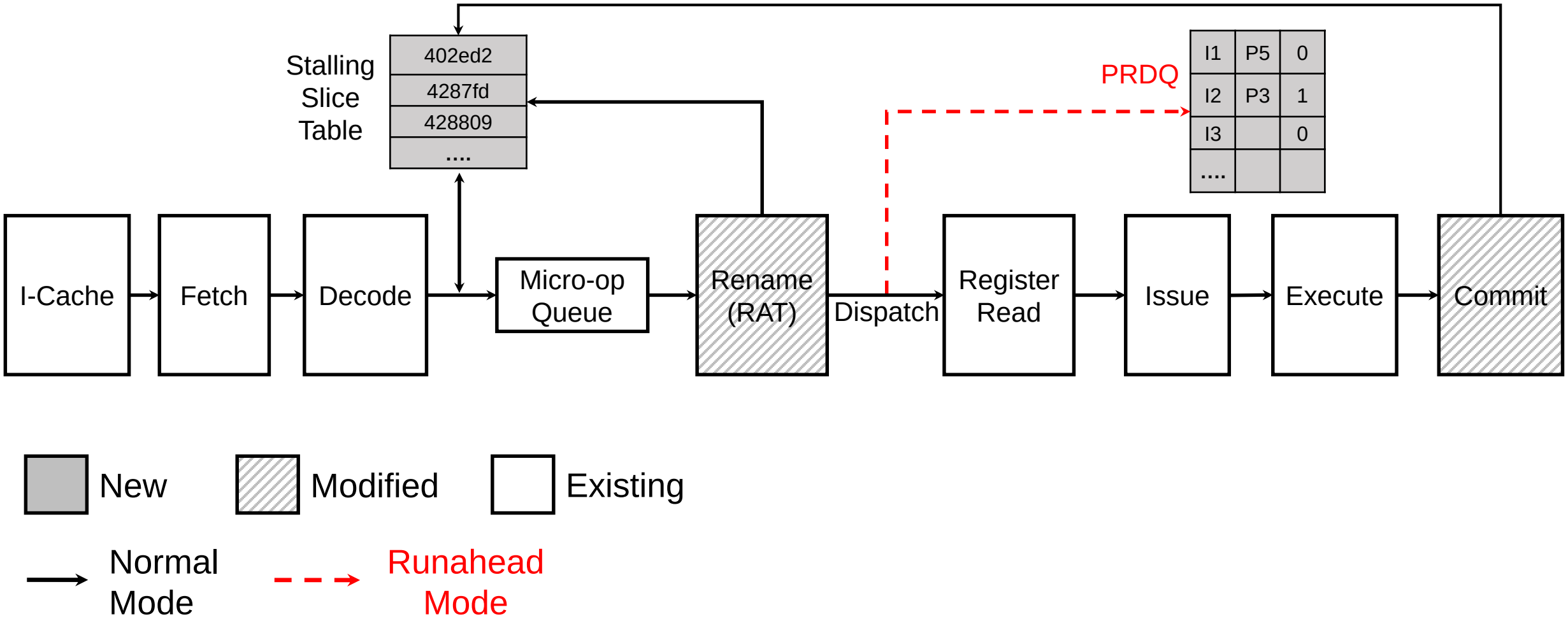
# Precise Runahead Execution



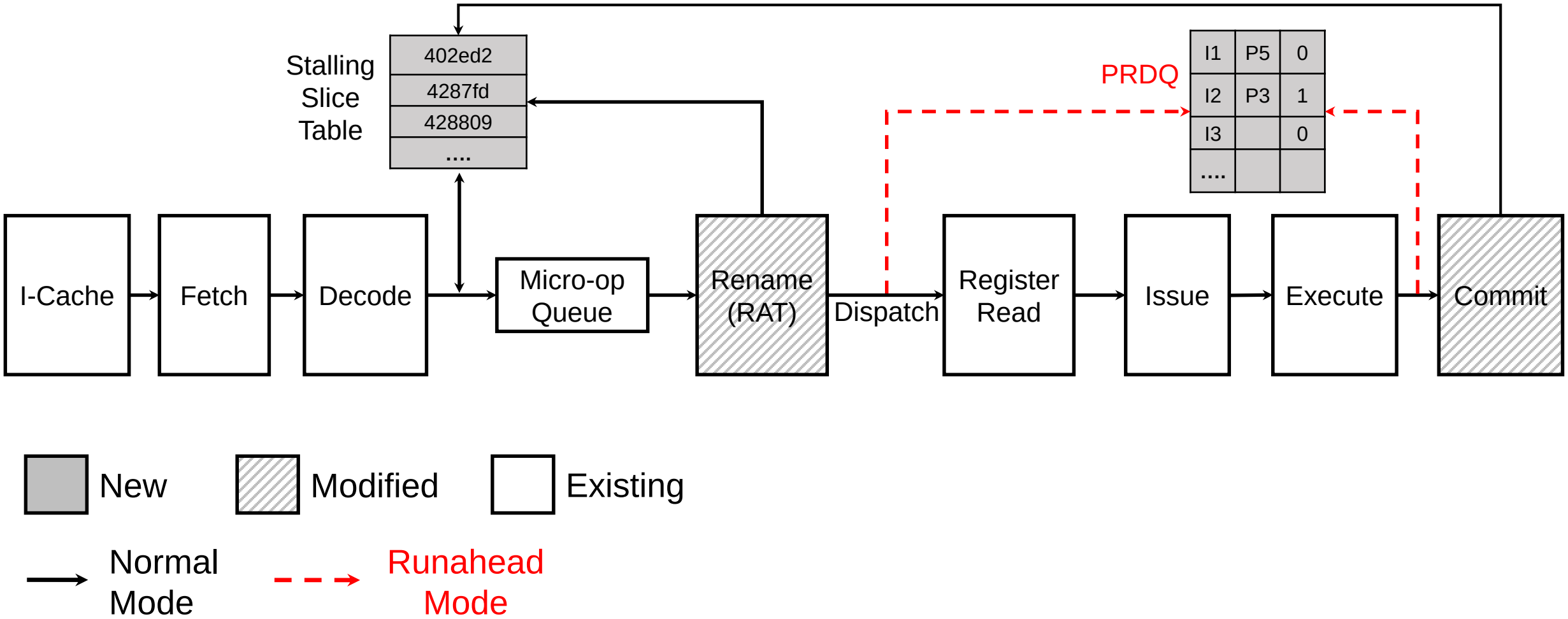
# Precise Runahead Execution



# Precise Runahead Execution



# Precise Runahead Execution



# Evaluation

**OoO:** Baseline out-of-order core

# Evaluation

**OoO:** Baseline out-of-order core

**RA:** Runahead execution*
-- No short runahead intervals

*[Mutlu et al. ISCA'05]

# Evaluation

**OoO:** Baseline out-of-order core

**RA:** Runahead execution*

-- No short runahead intervals

-- No overlapping intervals

*[Mutlu et al. ISCA'05]

# Evaluation

**OoO:** Baseline out-of-order core

**RA:** Runahead execution*

-- No short runahead intervals

-- No overlapping intervals

**RA-buffer:** Runahead buffer**

*[Mutlu et al. ISCA'05] **[Hashemi et al. MICRO'15]

# Evaluation

**OoO:** Baseline out-of-order core

**RA:** Runahead execution*

-- No short runahead intervals

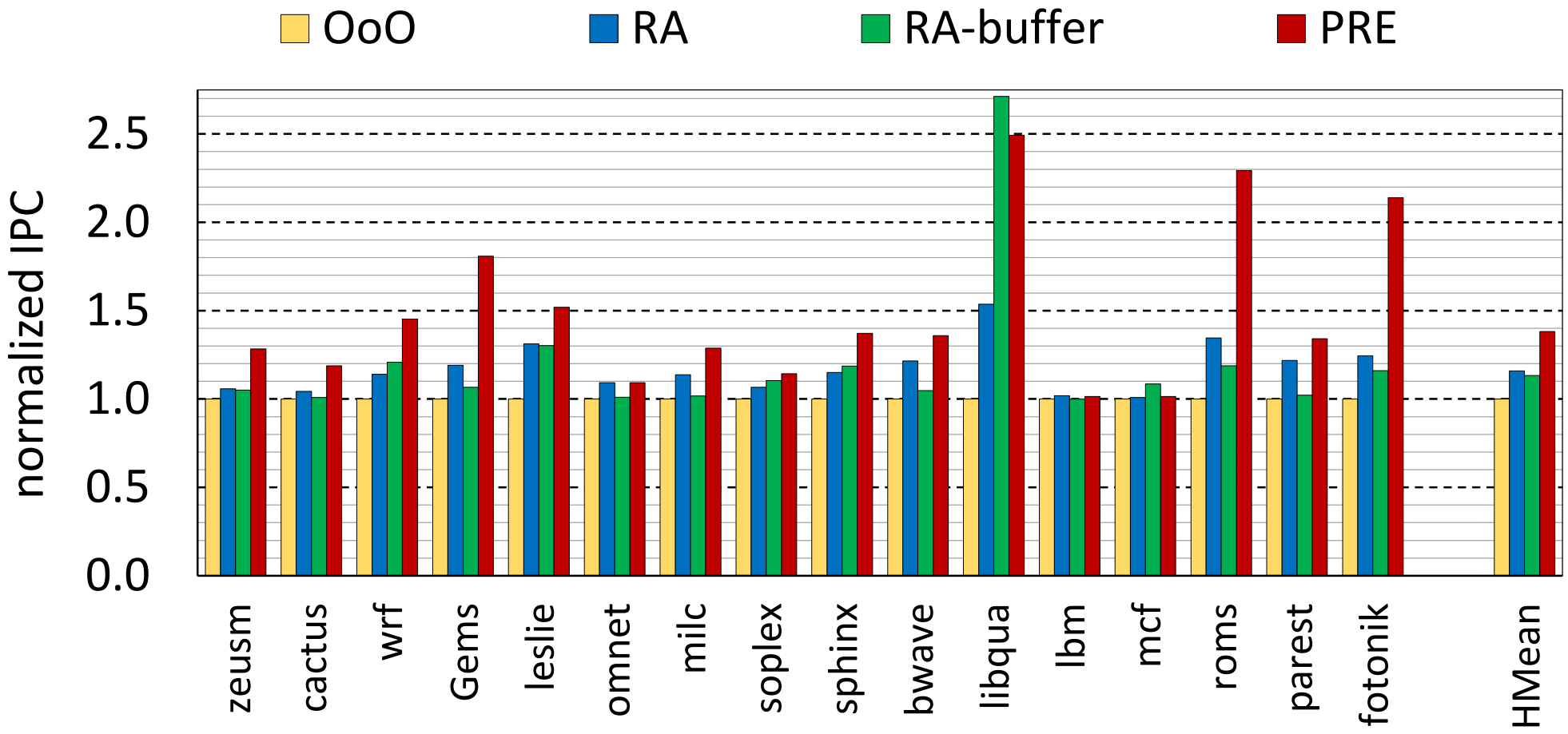
-- No overlapping intervals

**RA-buffer:** Runahead buffer**

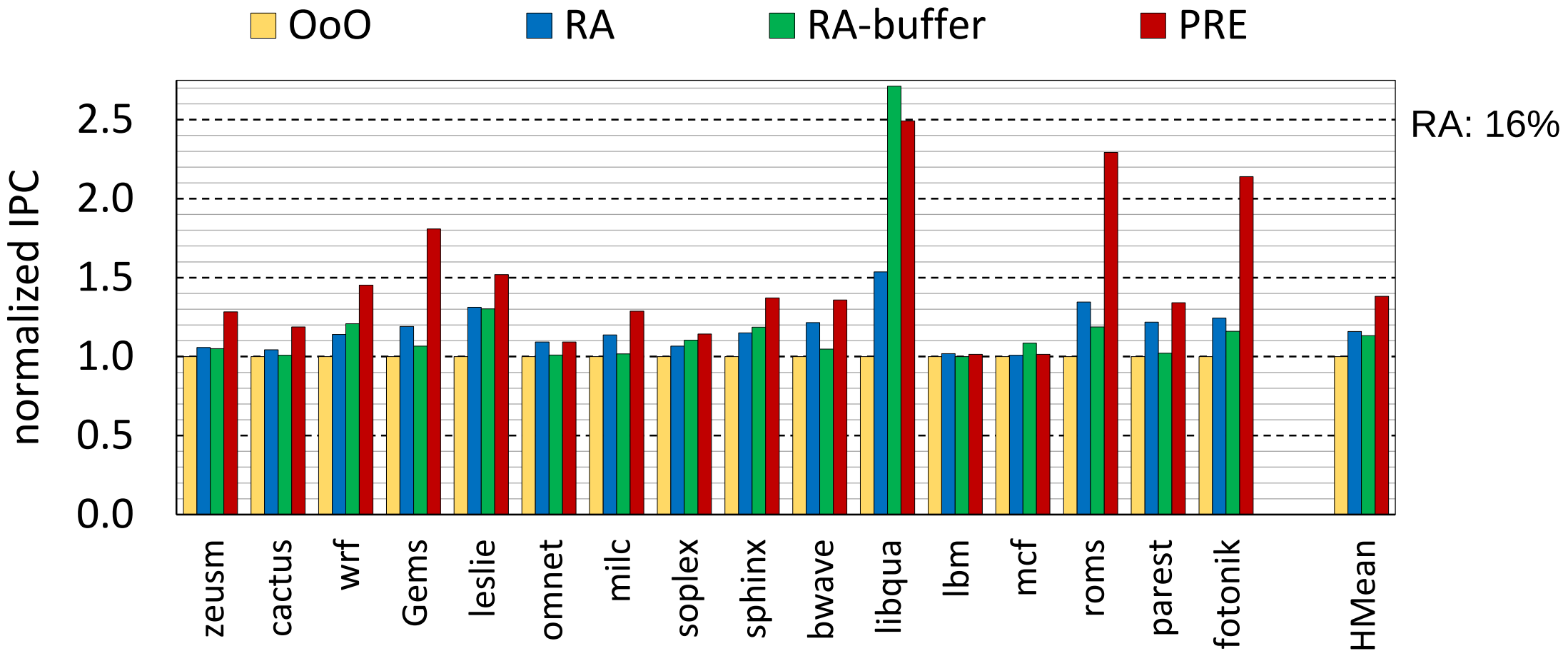
**PRE:** Precise runahead execution***

*[Mutlu et al. ISCA'05] **[Hashemi et al. MICRO'15] ***[Naithani et al. HPCA'20]

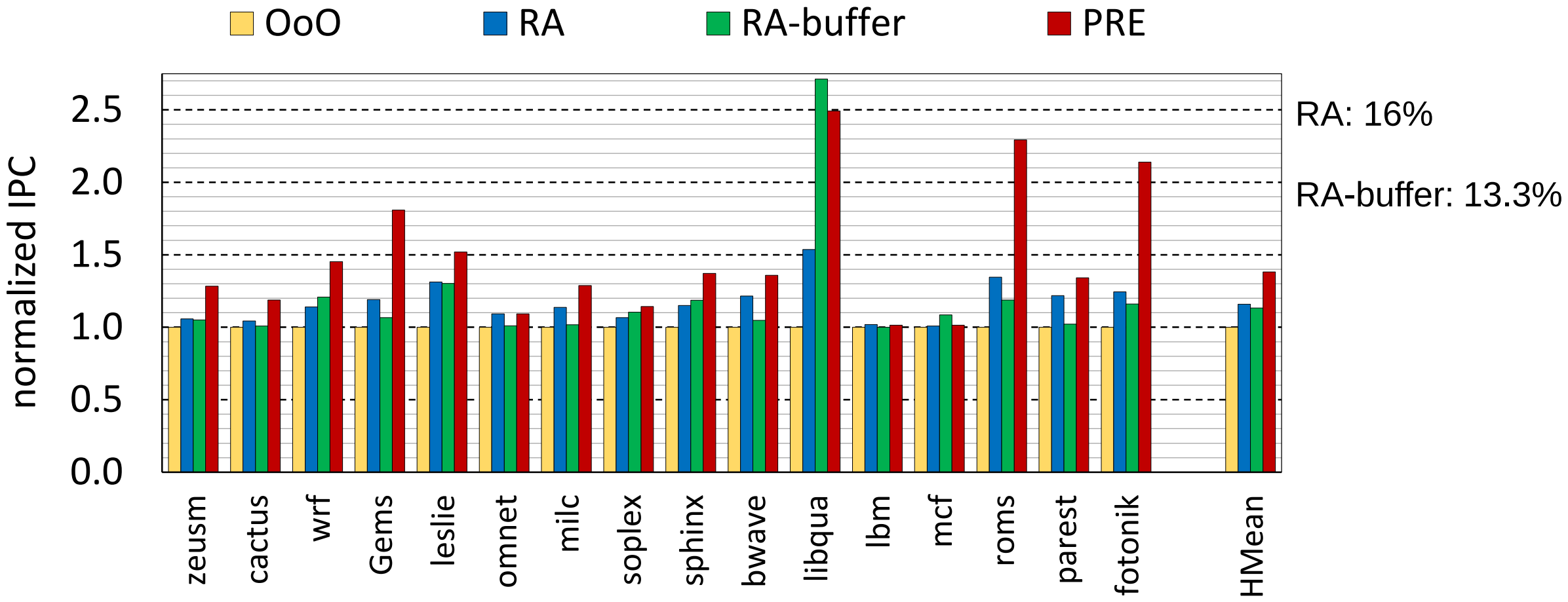
# Evaluation -- Performance



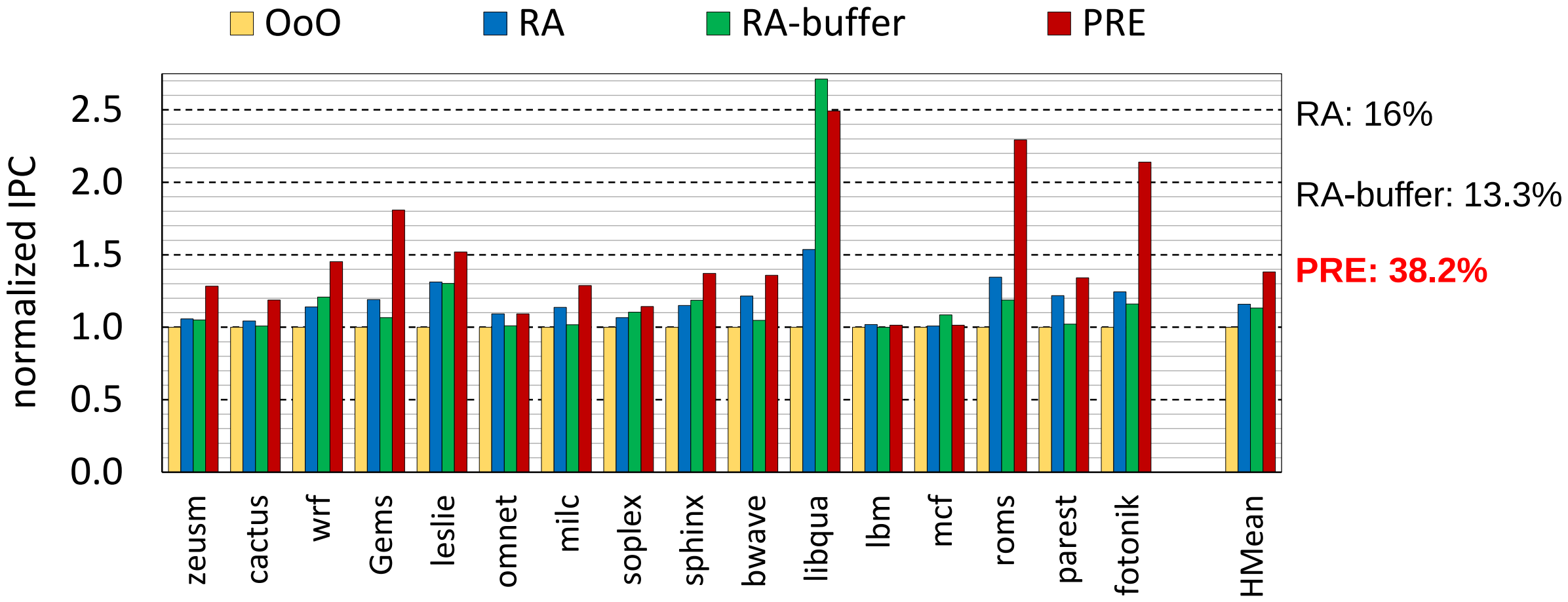
# Evaluation -- Performance



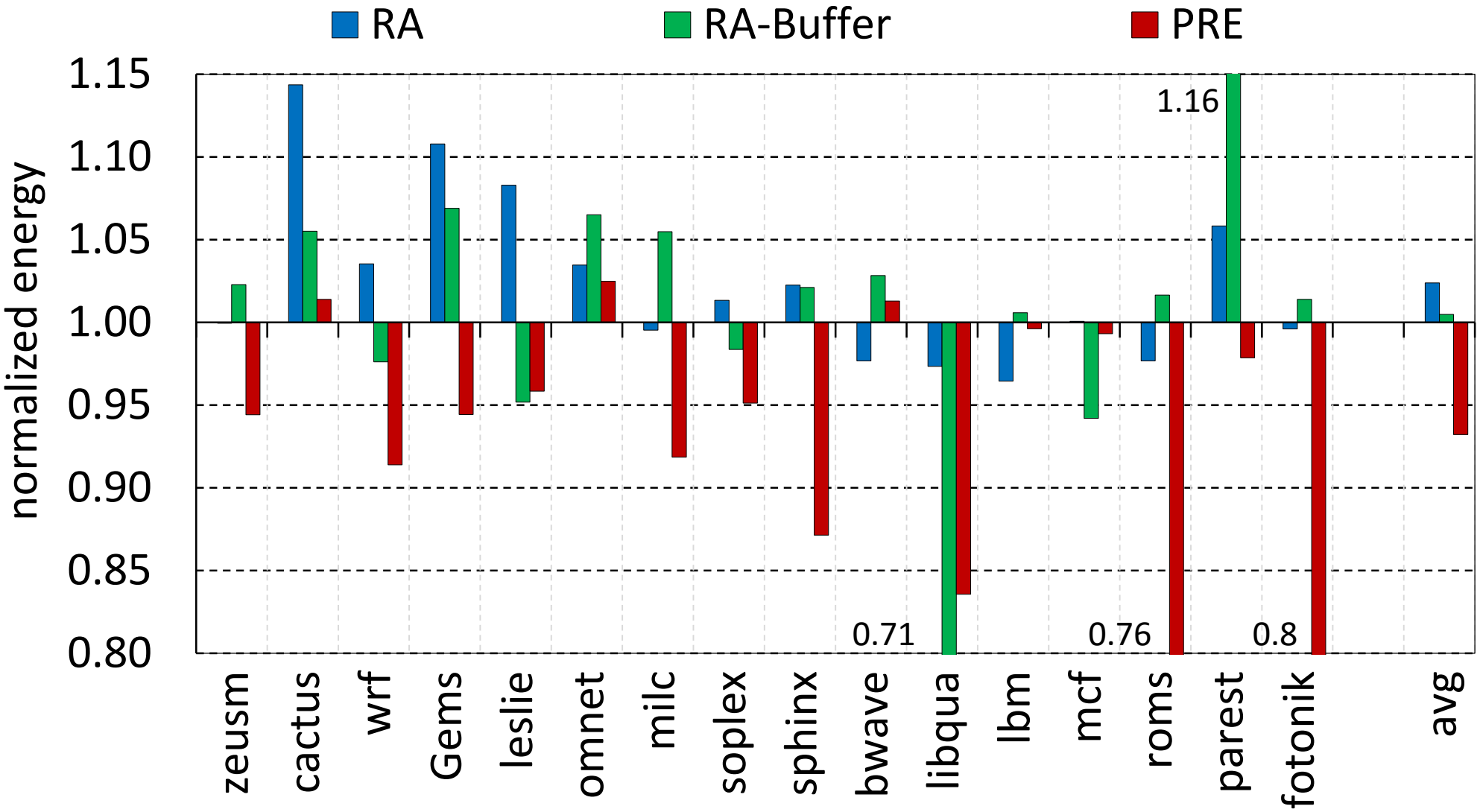
# Evaluation -- Performance



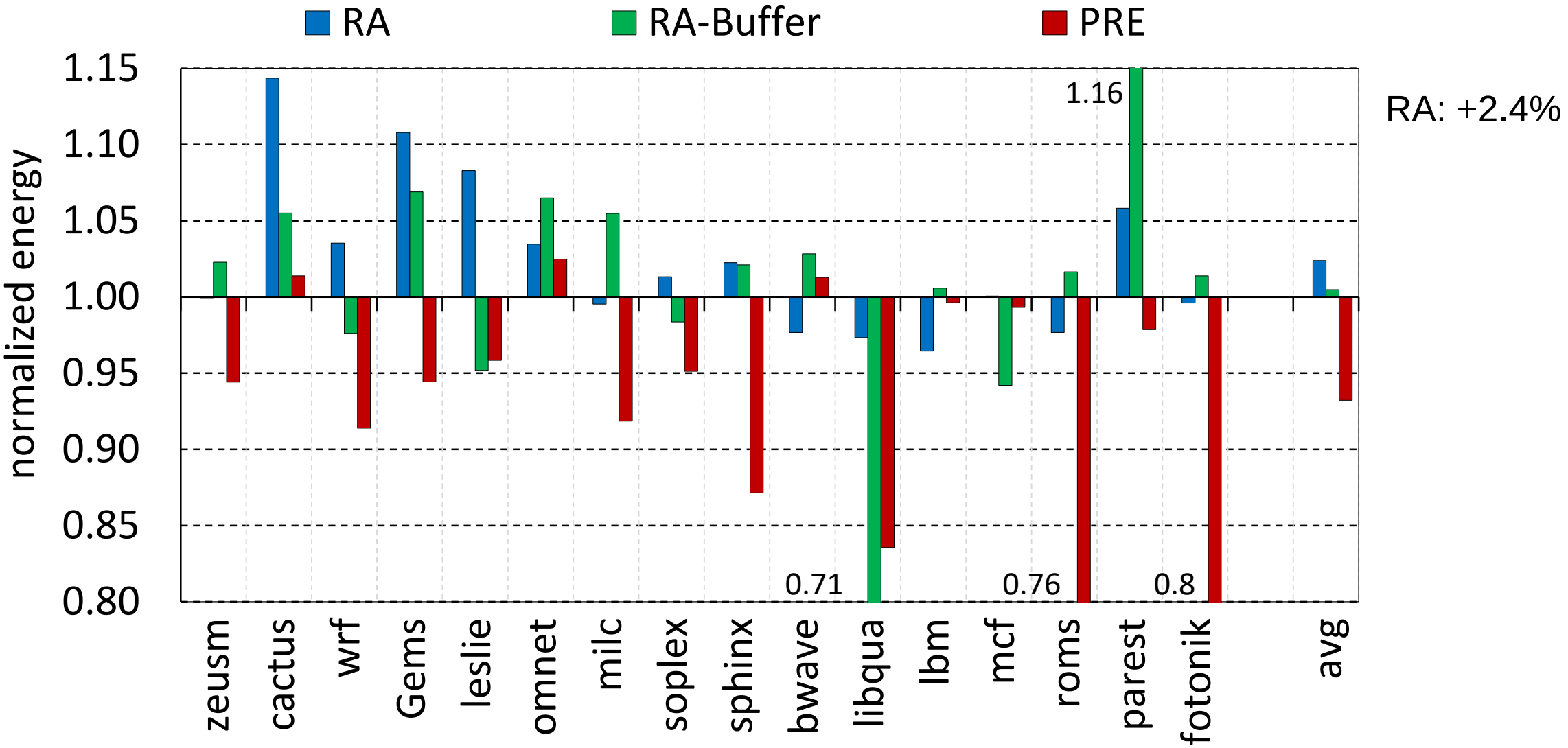
# Evaluation -- Performance



# Evaluation -- Energy

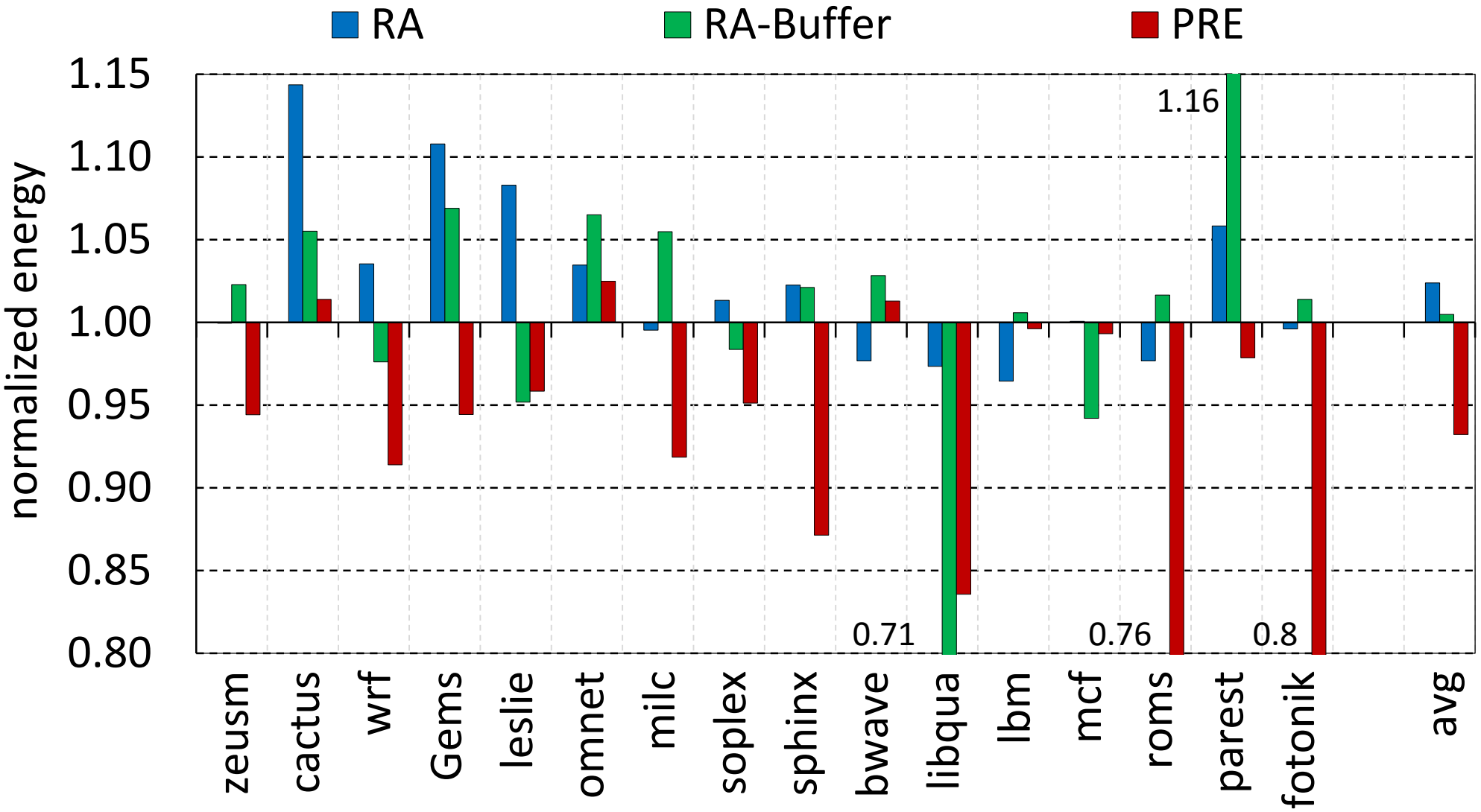


# Evaluation -- Energy




RA: +2.4%

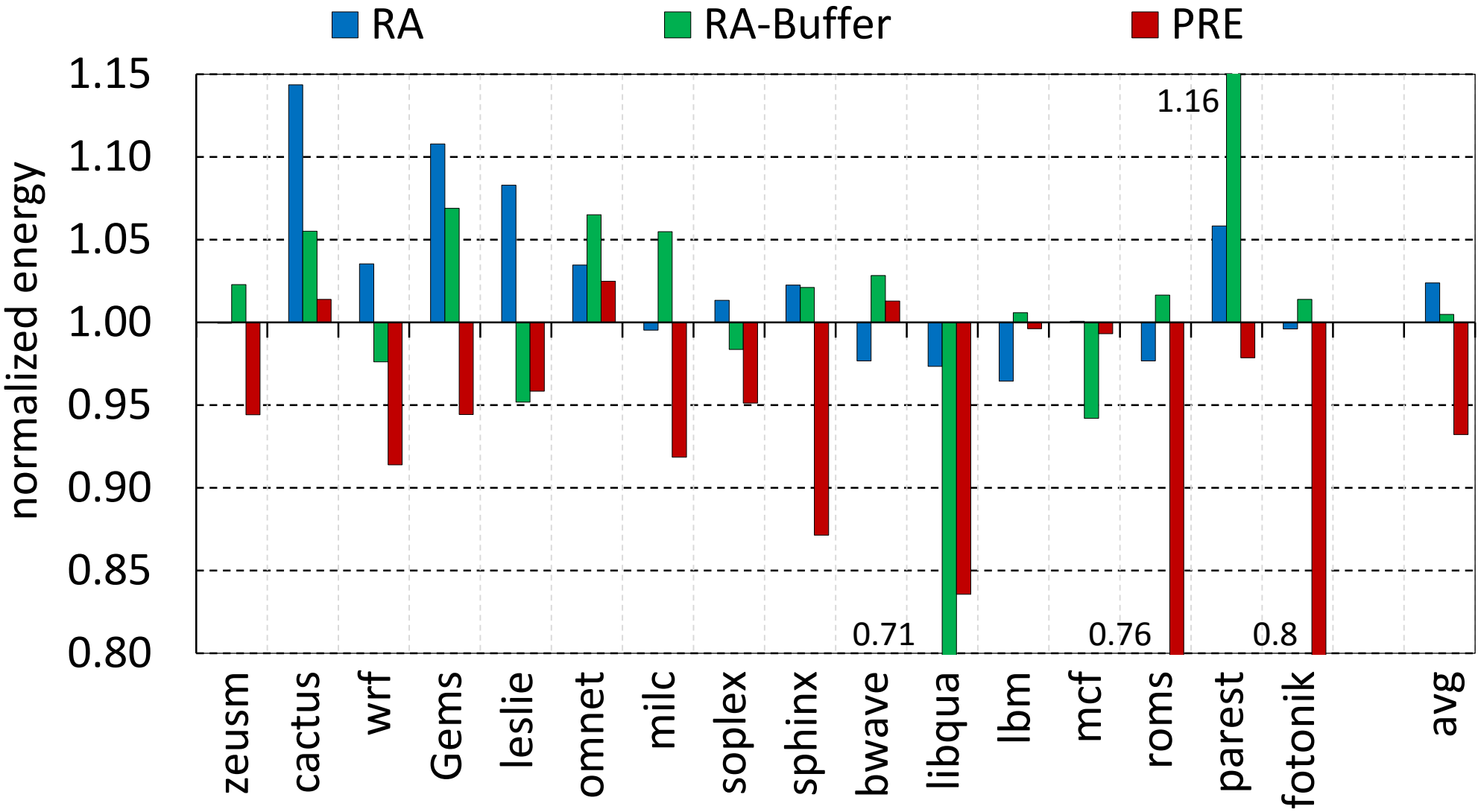
# Evaluation -- Energy




RA: +2.4%

RA-buffer: Same

# Evaluation -- Energy

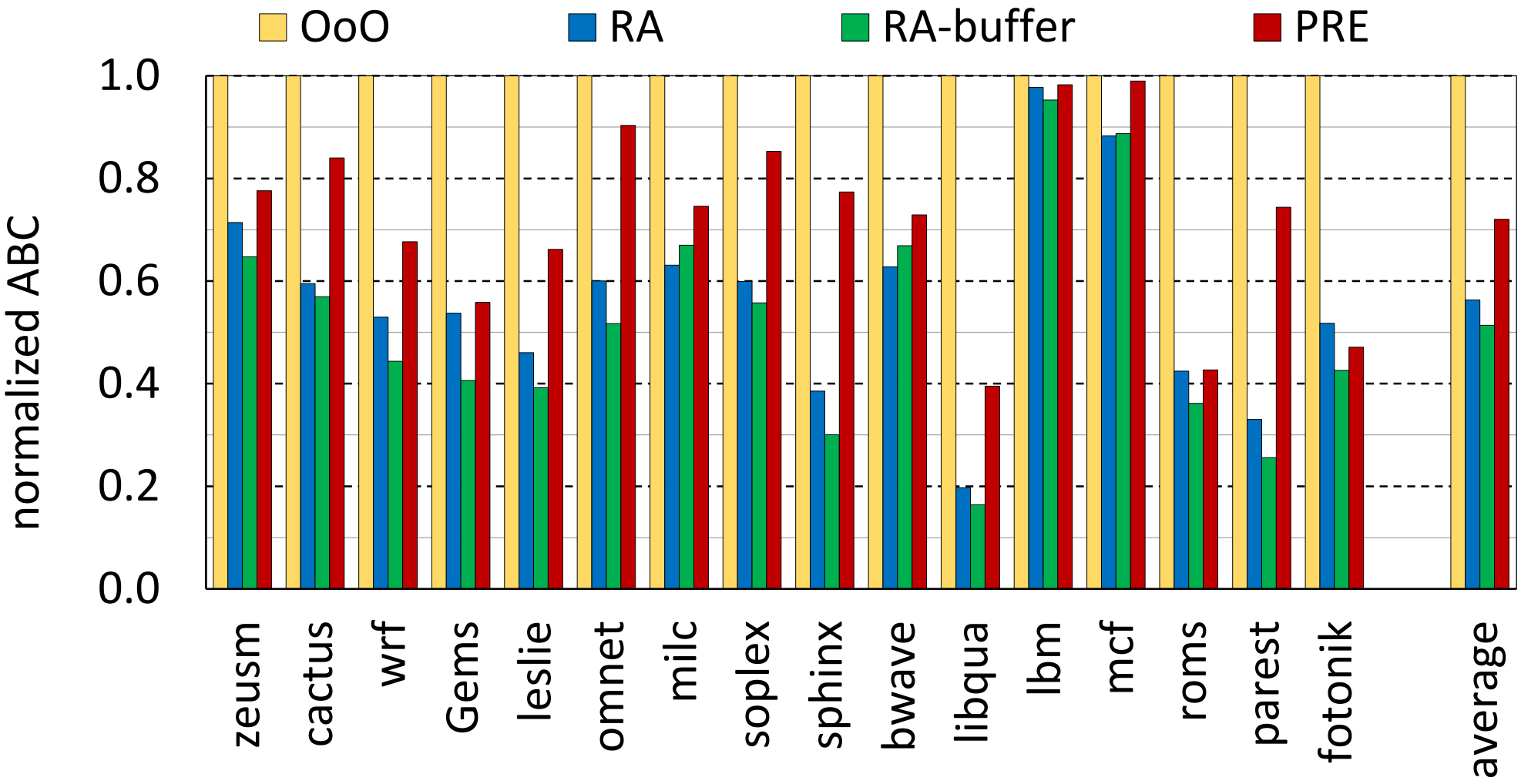



RA: +2.4%

RA-buffer: Same

**PRE: −6.8%**

# Evaluation -- Reliability




RA: 43.7%

RA-buffer: 48.6%

**PRE: 28%**

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |
| Reliability | | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |
| Reliability | High ↑ | | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |
| Reliability | High ↑ | High ↑ | |

# Runahead Techniques Relative to OoO Core

| | Runahead execution | Runahead buffer | **Precise runahead** |
|---|---|---|---|
| Flush ROB | ✓ | ✓ | ✗ |
| Execute all future instructions | ✓ | Only one slice | All slices |
| Performance | High ↑ | High ↑ | Very high ↑↑ |
| Energy Efficiency | Low ↓ | Same | High ↑ |
| Reliability | High ↑ | High ↑ | Medium ↑ |

# Summary of Novel Contributions

# Summary of Novel Contributions

1. Reliability-aware scheduling for multiprogram workloads

   Exploits the difference in vulnerability between core types to improve reliability

# Summary of Novel Contributions

1. Reliability-aware scheduling for multiprogram workloads

   Exploits the difference in vulnerability between core types to improve reliability

2. Dispatch halting for single-threaded workloads

   Exploits invulnerable speculation to improve reliability under memory accesses

# Summary of Novel Contributions

1. Reliability-aware scheduling for multiprogram workloads

   Exploits the difference in vulnerability between core types to improve reliability

2. Dispatch halting for single-threaded workloads

   Exploits invulnerable speculation to improve reliability under memory accesses

3. Precise runahead execution for single-threaded workloads

   Executes only useful future instructions after a full-ROB stall without flushing the ROB

# Improving Soft Error Reliability in Modern Processors

**Ajeya Naithani**

**Advisor: Prof. Lieven Eeckhout**

**Doctoral Thesis Defense**

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE