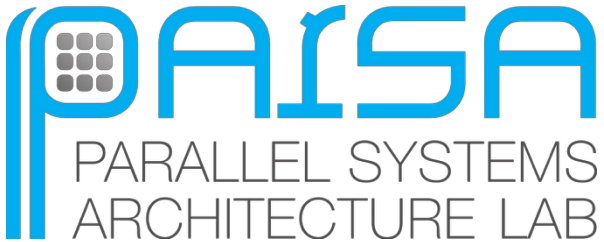
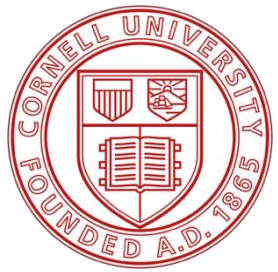
BACK TO THE FUTURE

The Return of Rigorous Full-System Timing Simulation

Babak Falsafi

With the collaboration of Mohammad Alian, Shanqing Lin, Ali Ansari,
Ayan Chakraborty, Pooria Poorsarvi Tehrani and Yuanlong Li



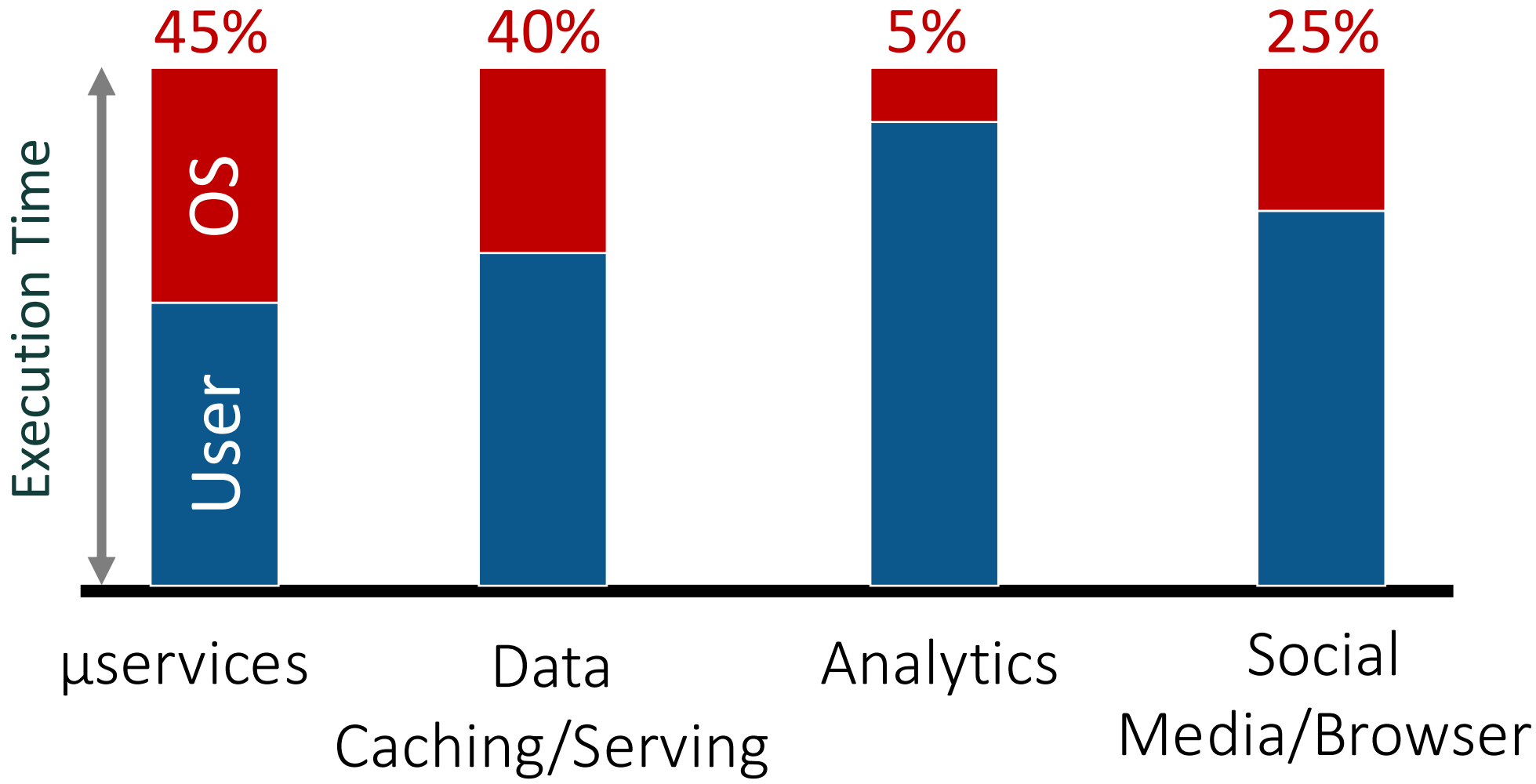


WHAT IS FULL-SYSTEM SIMULATION?

PARALLEL SYSTEMS ARCHITECTURE LAB

Full-system simulation is the emulation of an entire computer system—hardware, OS, and apps—so you can study how the whole stack behaves as if it were running on a real machine.

WHY FULL-SYSTEM SIMULATION?



WHY FULL-SYSTEM SIMULATION?

PARALLEL SYSTEMS ARCHITECTURE LAB

1. The rise of service-oriented, multi-tenant software
2. Heterogenous platforms have CPU and OS as a pillar
3. Agentic AI places a stronger emphasis on CPU and OS
4. Post-Moore platforms are about hardware/OS co-design

HISTORY OF FULL-SYSTEM SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB

SimOS

(MIPS, DEC Alpha)

HISTORY OF FULL-SYSTEM SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB

SimOS

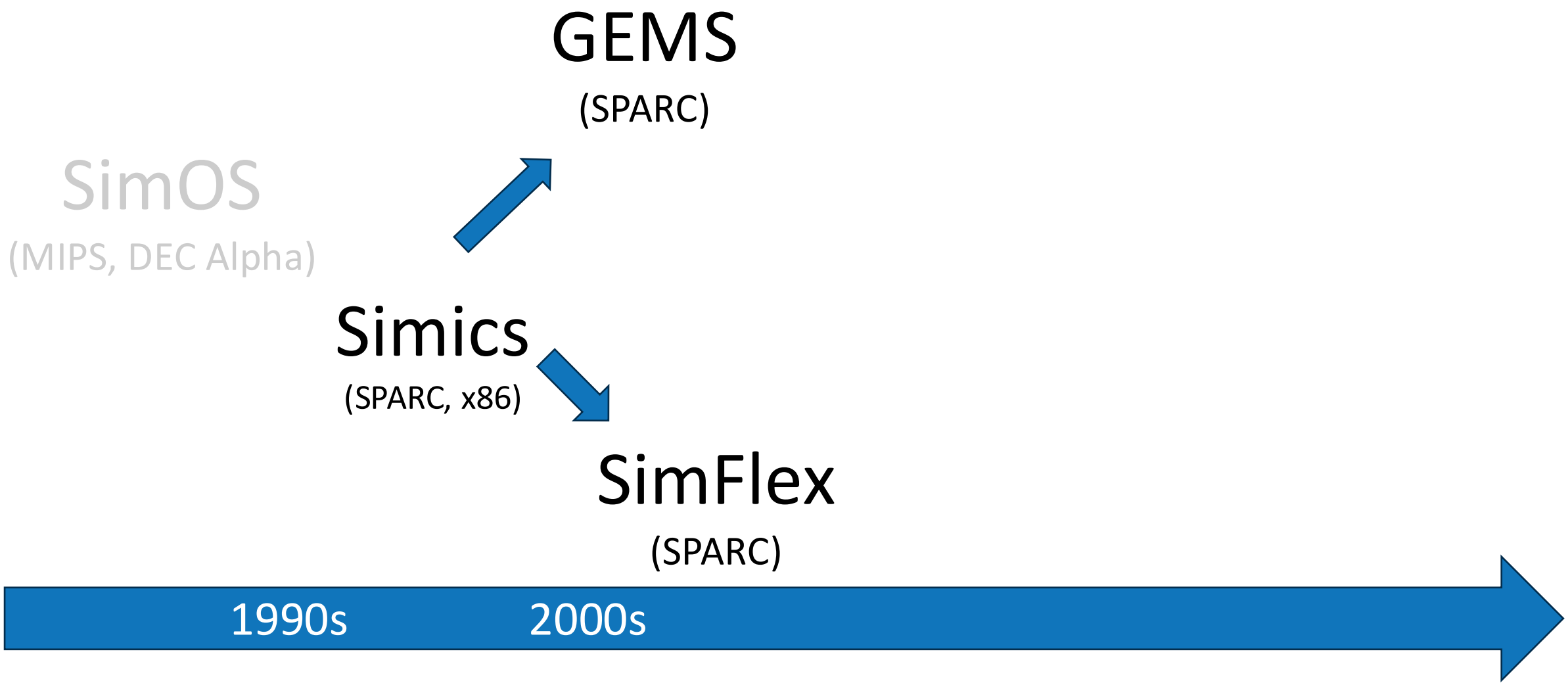
(MIPS, DEC Alpha)

Simics

(SPARC, x86)

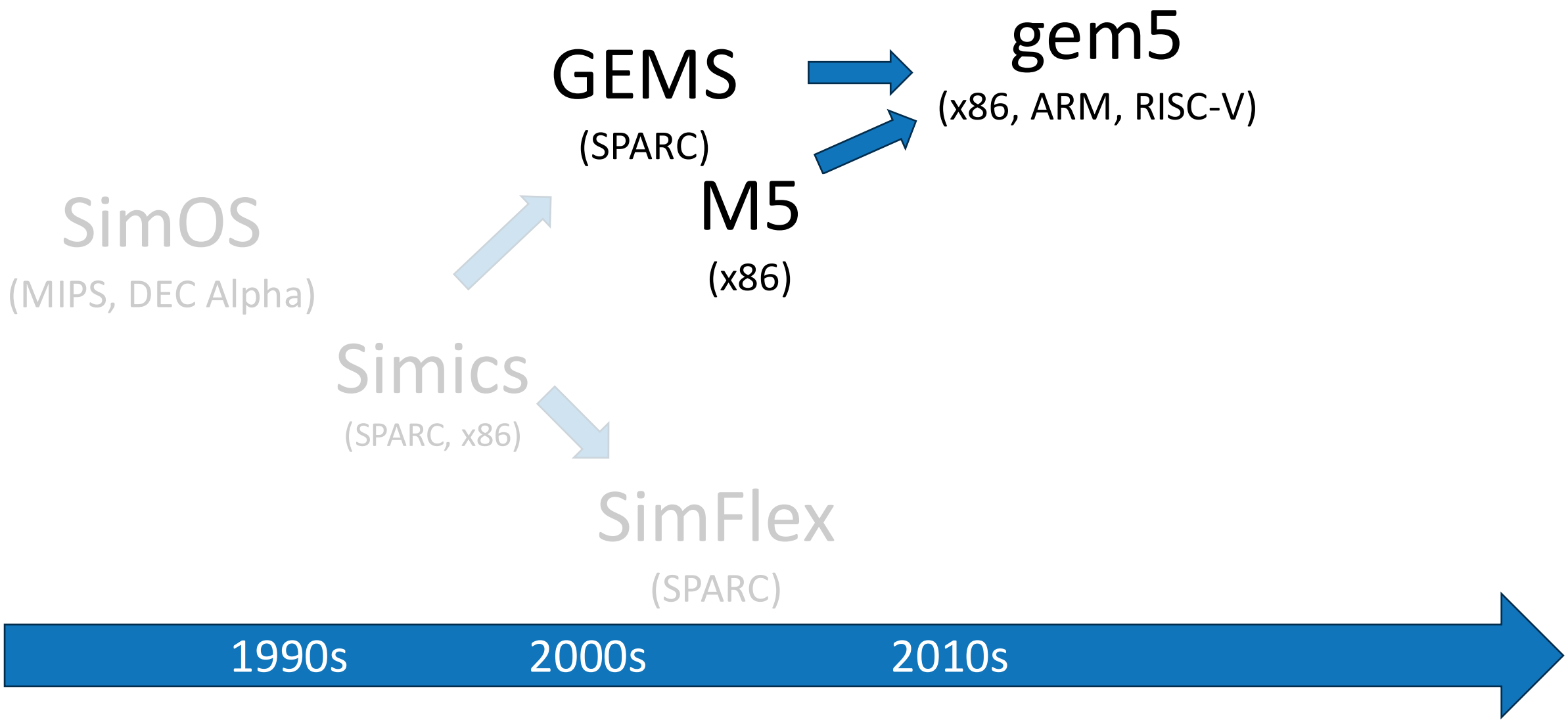
HISTORY OF FULL-SYSTEM SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB

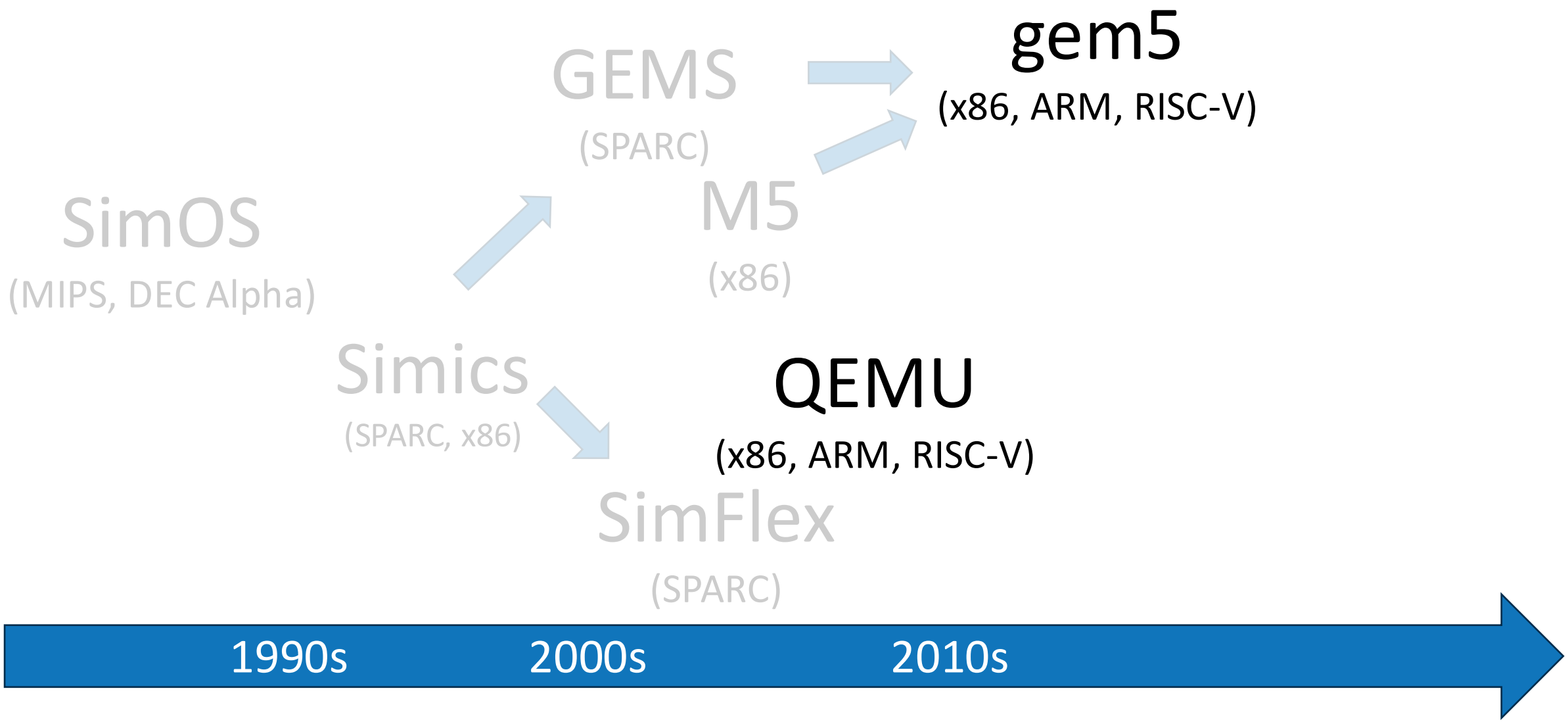


HISTORY OF FULL-SYSTEM SIMULATION

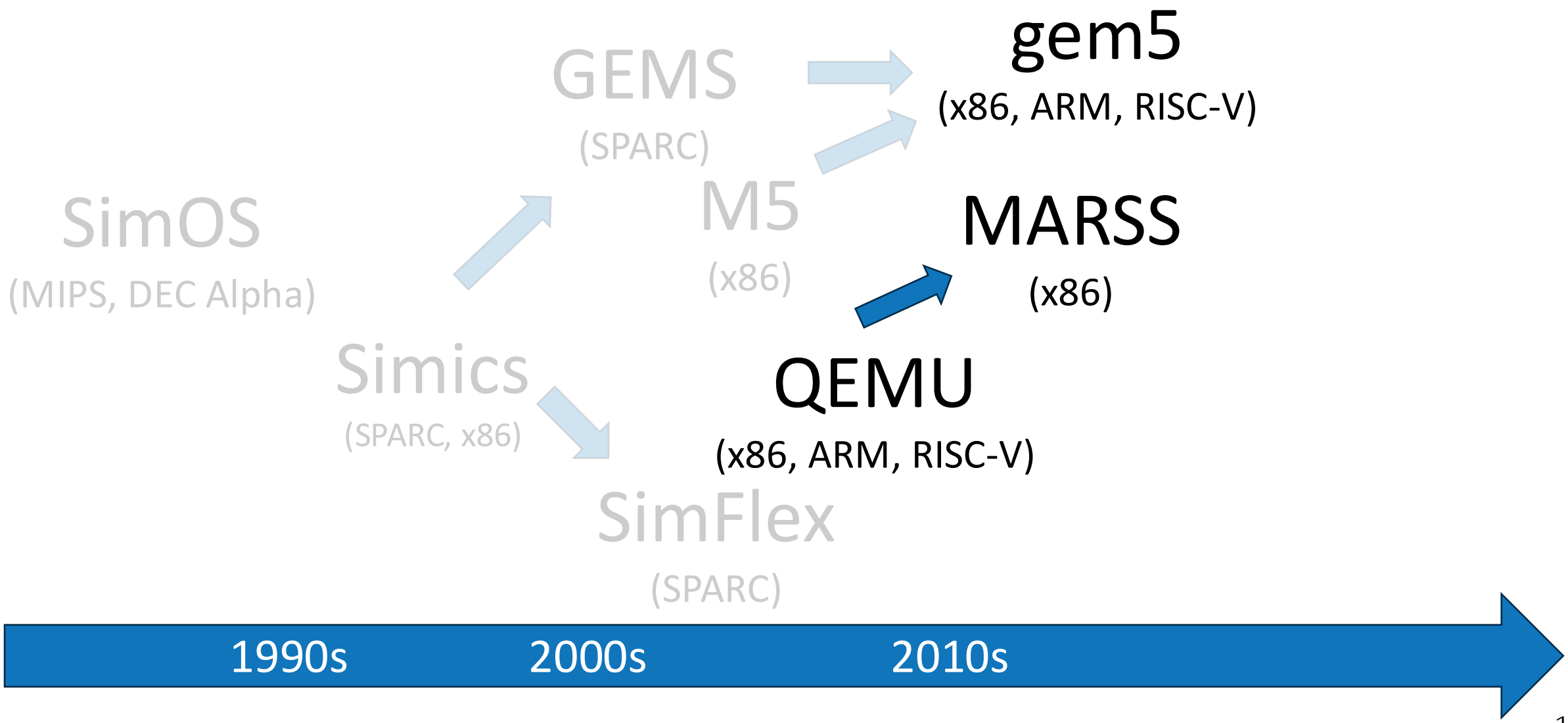
PARALLEL SYSTEMS ARCHITECTURE LAB



HISTORY OF FULL-SYSTEM SIMULATION

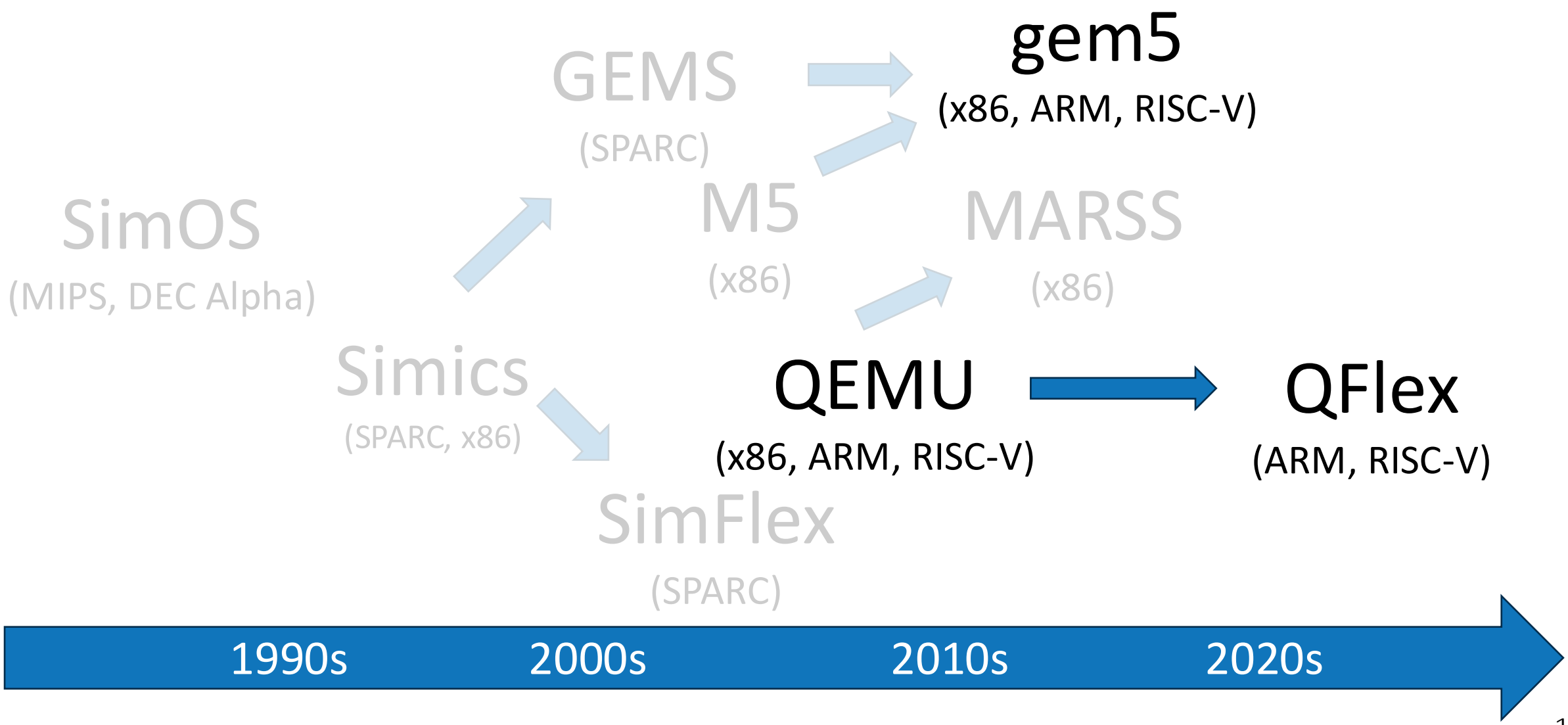


HISTORY OF FULL-SYSTEM SIMULATION



HISTORY OF FULL-SYSTEM SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB



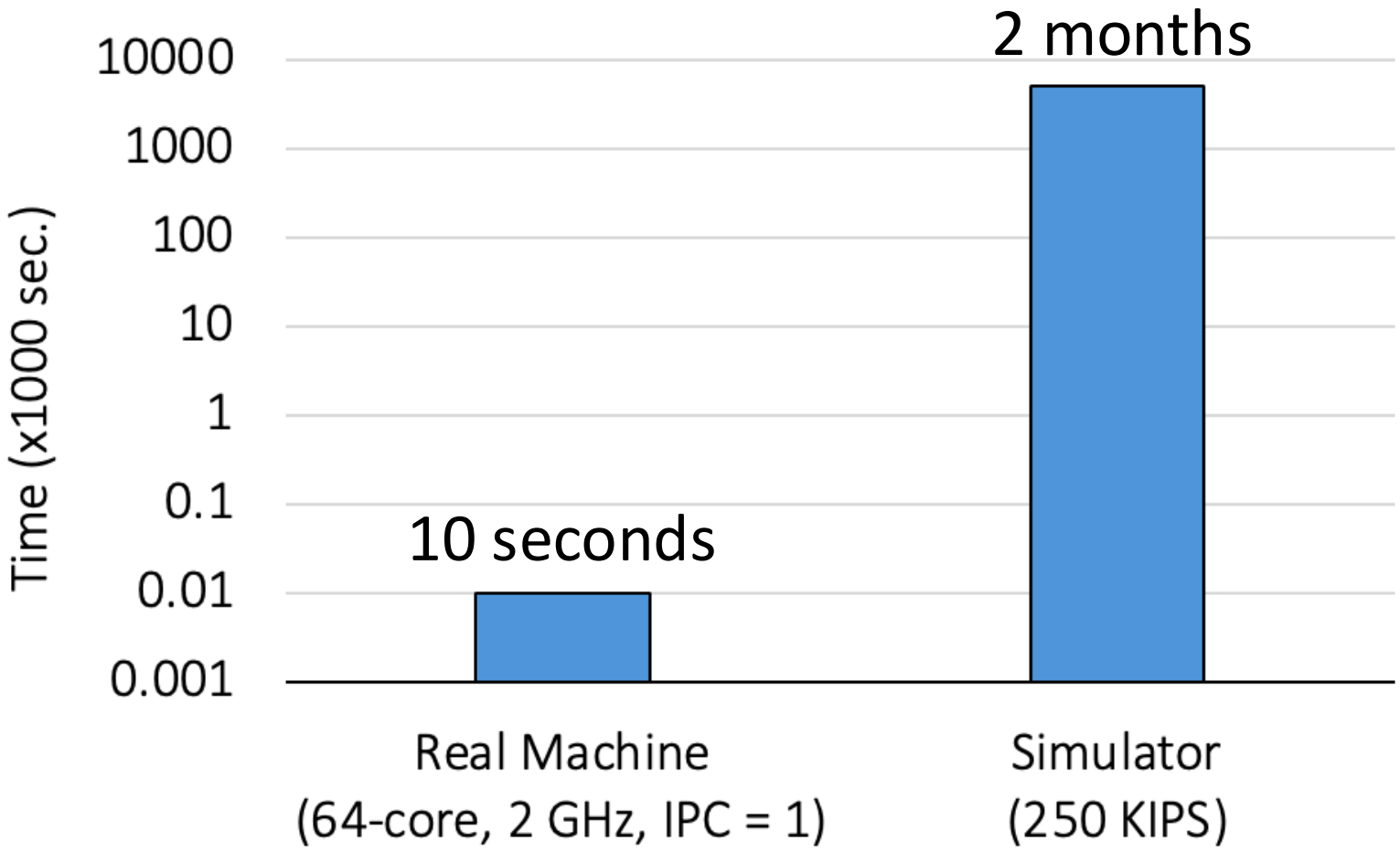
SIMULATION MODEL & SPEED

PARALLEL SYSTEMS ARCHITECTURE LAB

	Model	Speed (MIPS)
QEMU	ISA Emulation	> 400
QEMU + null instrumentation	Functional	85
QEMU + uArch	Functional	18
QEMU + uArch + OS	Functional	6
QEMU + timing	Timing	< 0.3

THE TIMING SIMULATION WALL

PARALLEL SYSTEMS ARCHITECTURE LAB



Simulating only a few seconds of timing takes several months

TALK ROADMAP

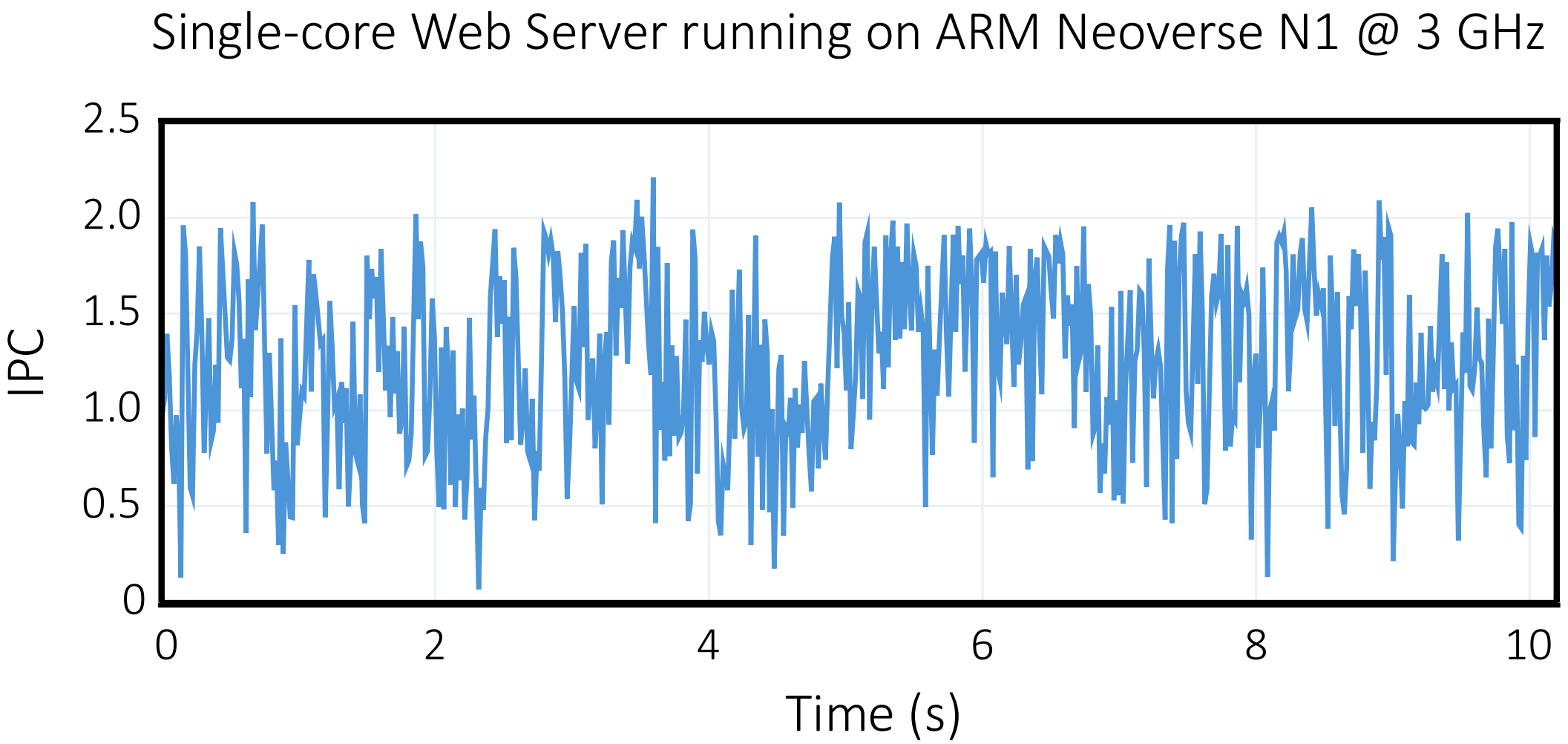
PARALLEL SYSTEMS ARCHITECTURE LAB

TALK ROADMAP

PARALLEL SYSTEMS ARCHITECTURE LAB

VARIABILITY IN PERFORMANCE

PARALLEL SYSTEMS ARCHITECTURE LAB



SOURCES OF PERFORMANCE VARIABILITY

PARALLEL SYSTEMS ARCHITECTURE LAB

Network
Resource contention
Background OS
Software/system hiccups

Wireless network
UI & graphics
Background OS
DVFS throttling

HOW MUCH TO MEASURE?

PARALLEL SYSTEMS ARCHITECTURE LAB

Identify the workload's minimum measurement window to contain variability in execution on real hardware.

— Wenisch et. al., IEEE Micro'06 inspired by Alameldeen

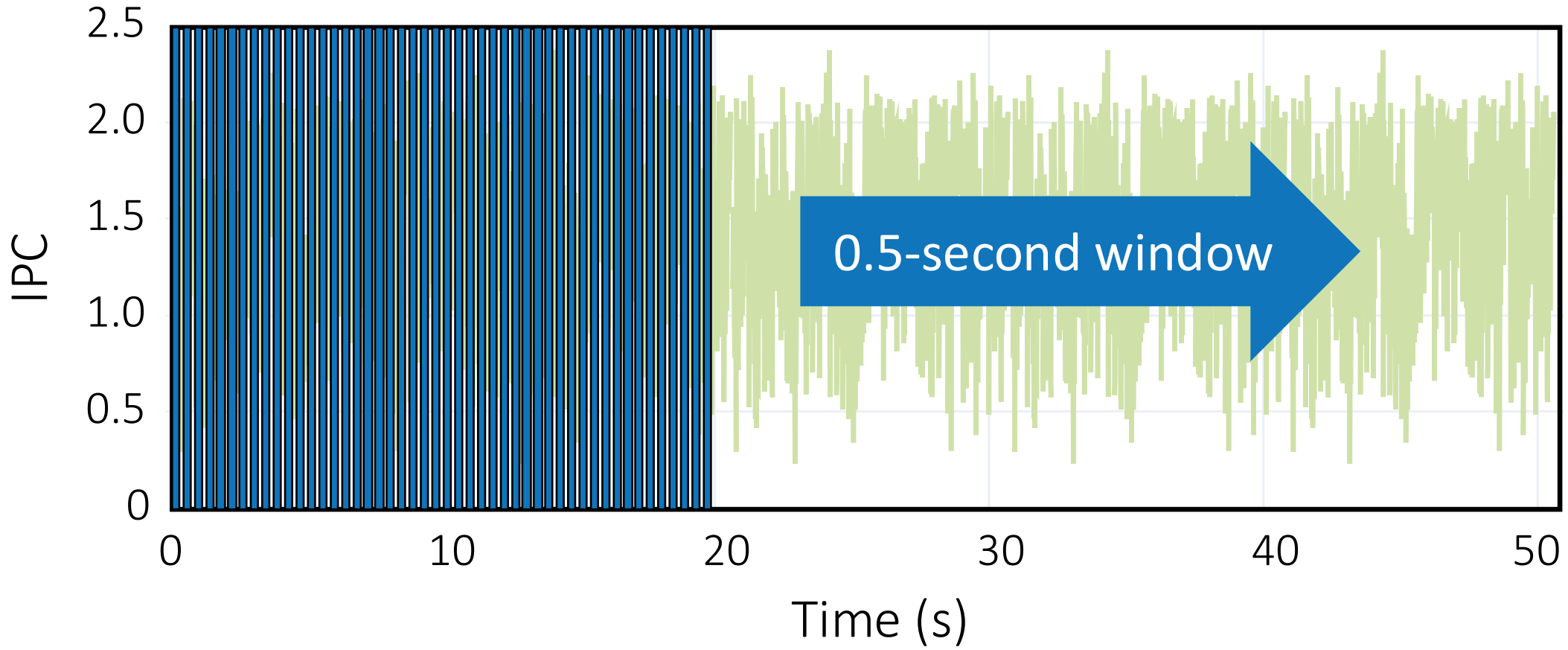
Assume a variability criterion: 5% variability in IPC with 95% confidence

Start with a minimum window (e.g., 0.5 seconds)

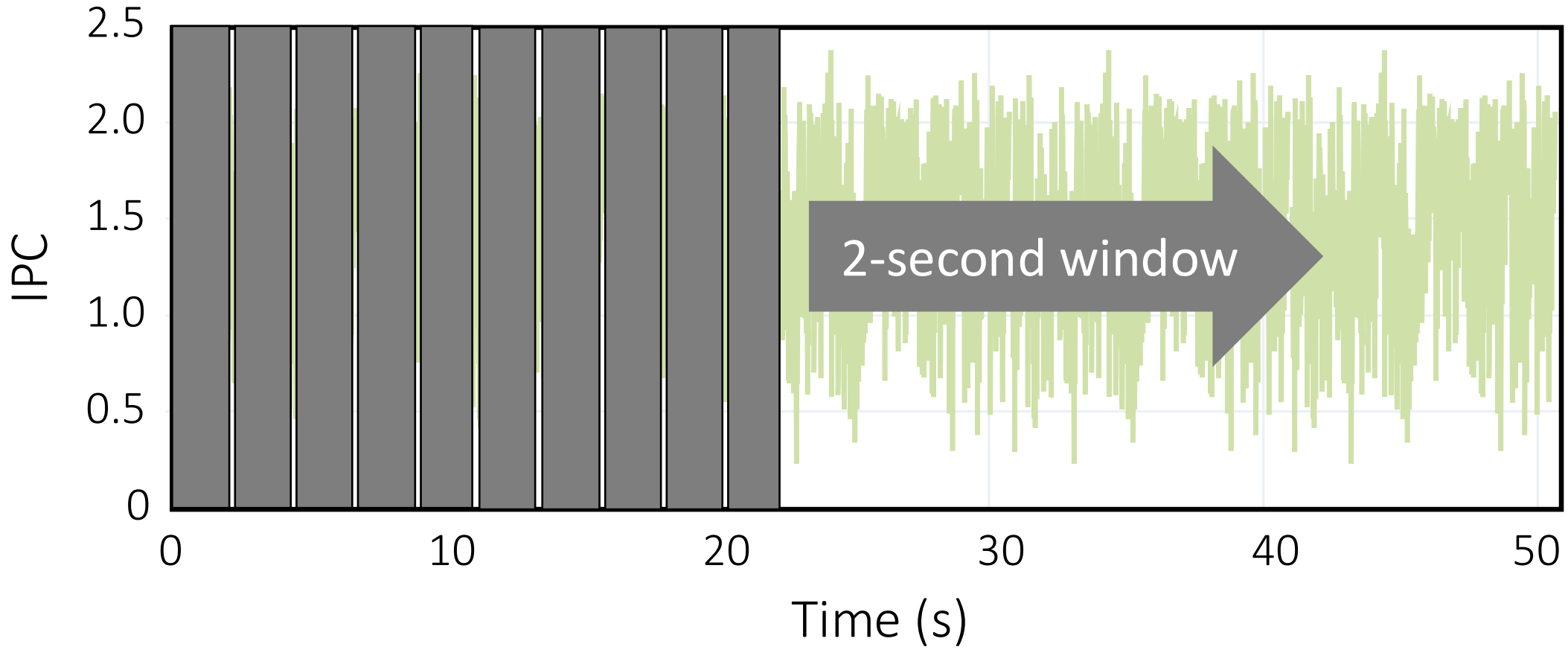
Repeat:

1. Measure the window 1000 times (or until the execution completes)
2. Test the IPC distribution? If variability contained, stop, if not double the window

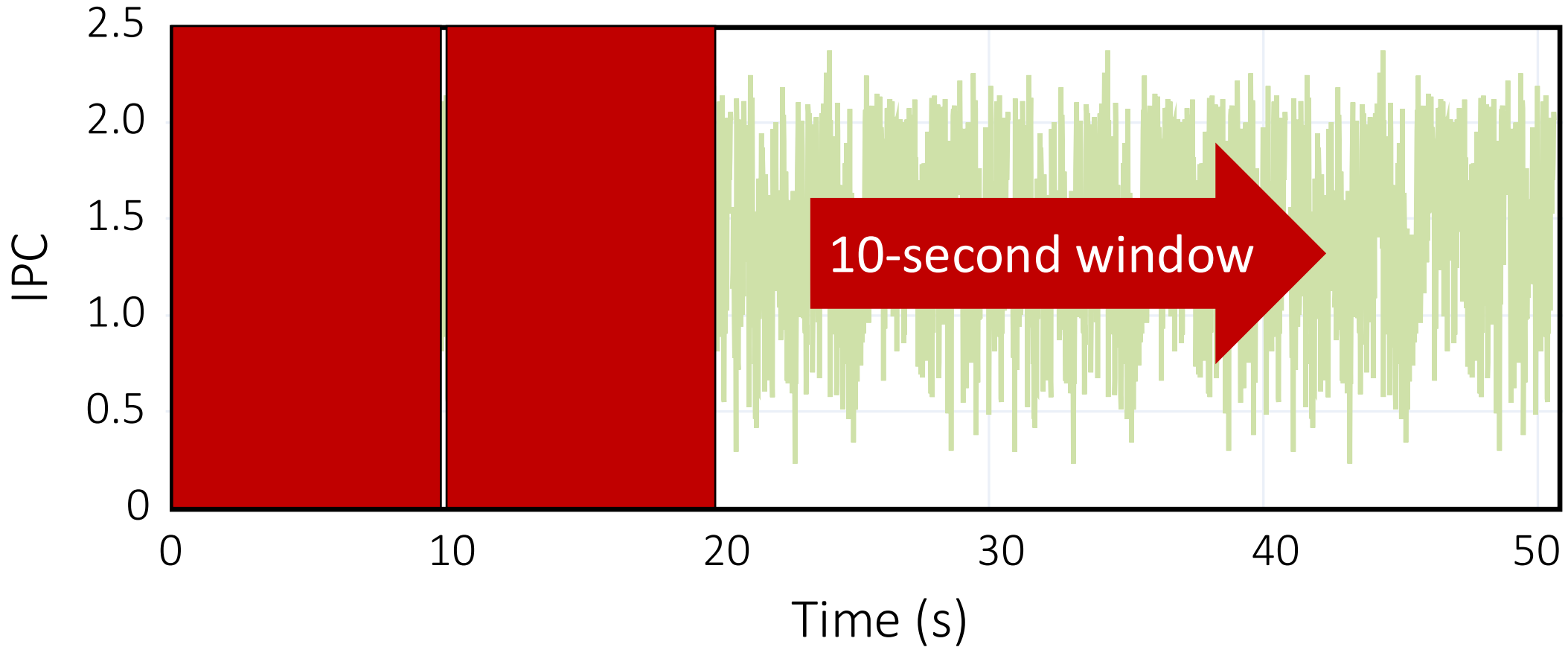
CONTAINING VARIABILITY



CONTAINING VARIABILITY



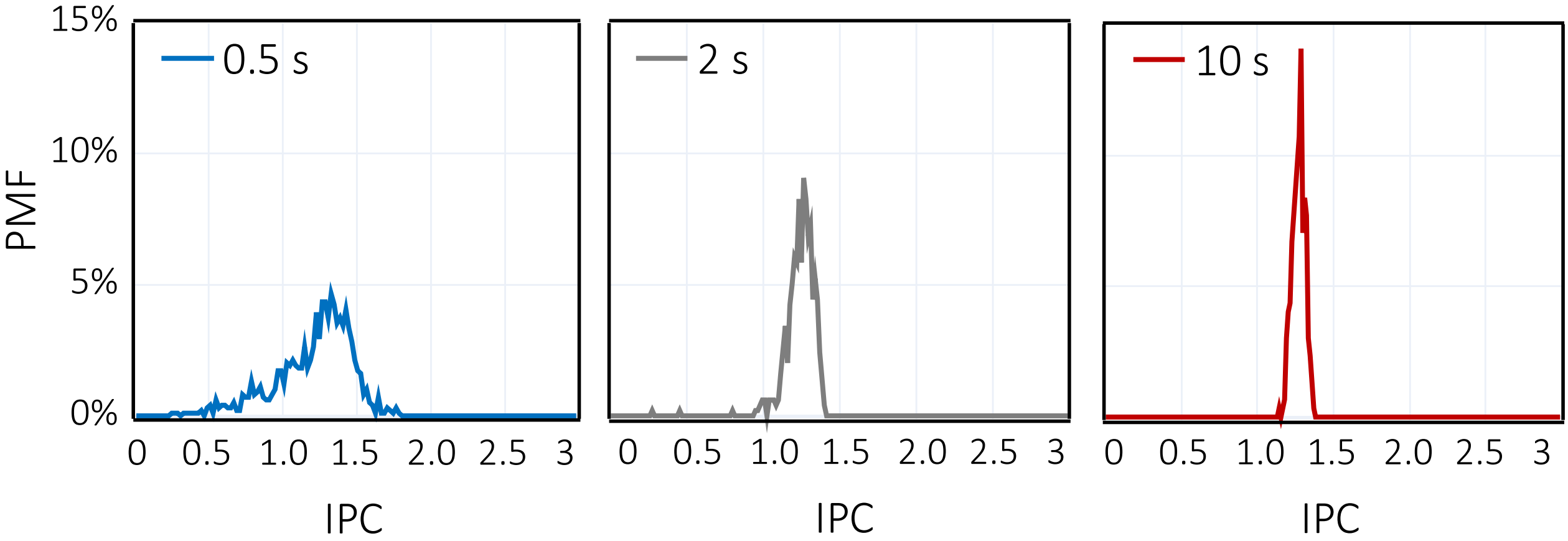
CONTAINING VARIABILITY



HOW MUCH TO MEASURE?

PARALLEL SYSTEMS ARCHITECTURE LAB

Single-core Web Server running on Neoverse N1 @ 3 GHz



MINIMUM MEASUREMENT

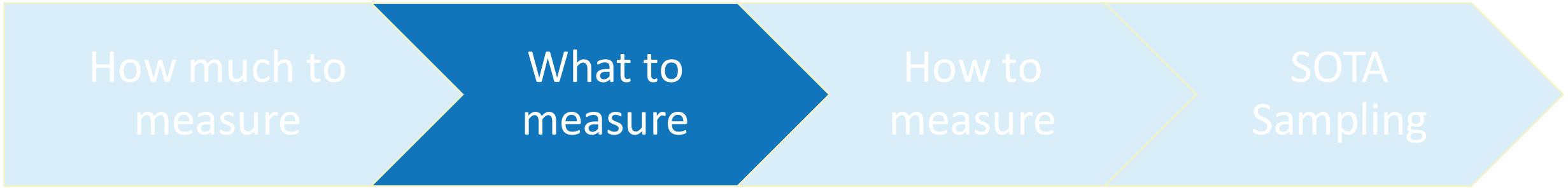
PARALLEL SYSTEMS ARCHITECTURE LAB

Execution measurement in seconds in server workloads to limit IPC variability to 5% with 95% confidence

CloudSuite	Data Analytics	60
	Data Caching	4
	Data Serving	20
	Graph Analytics	30
	In-Memory Analytics	10
	Media Streaming	4
	Web Search	5
	Web Serving	10
DCPerf	Django Bench	12
	FeedSim	10
	Video Transcode	10
DSB	Media Service	5

TALK ROADMAP

PARALLEL SYSTEMS ARCHITECTURE LAB



WHAT TO MEASURE

PARALLEL SYSTEMS ARCHITECTURE LAB

- Timing simulation counts cycles
- Computer architects often use IPC for throughput

✓ IPC works for a single core

- But, IPC doesn't work for multicore workloads

  [Alameldeen, IEEE Micro'06]

  ✗ threads spin but do not contribute to forward progress

  ✗ spinning threads result in high IPC

WHAT TO MEASURE: U-IPC

PARALLEL SYSTEMS ARCHITECTURE LAB

Modern workloads spin in the OS:

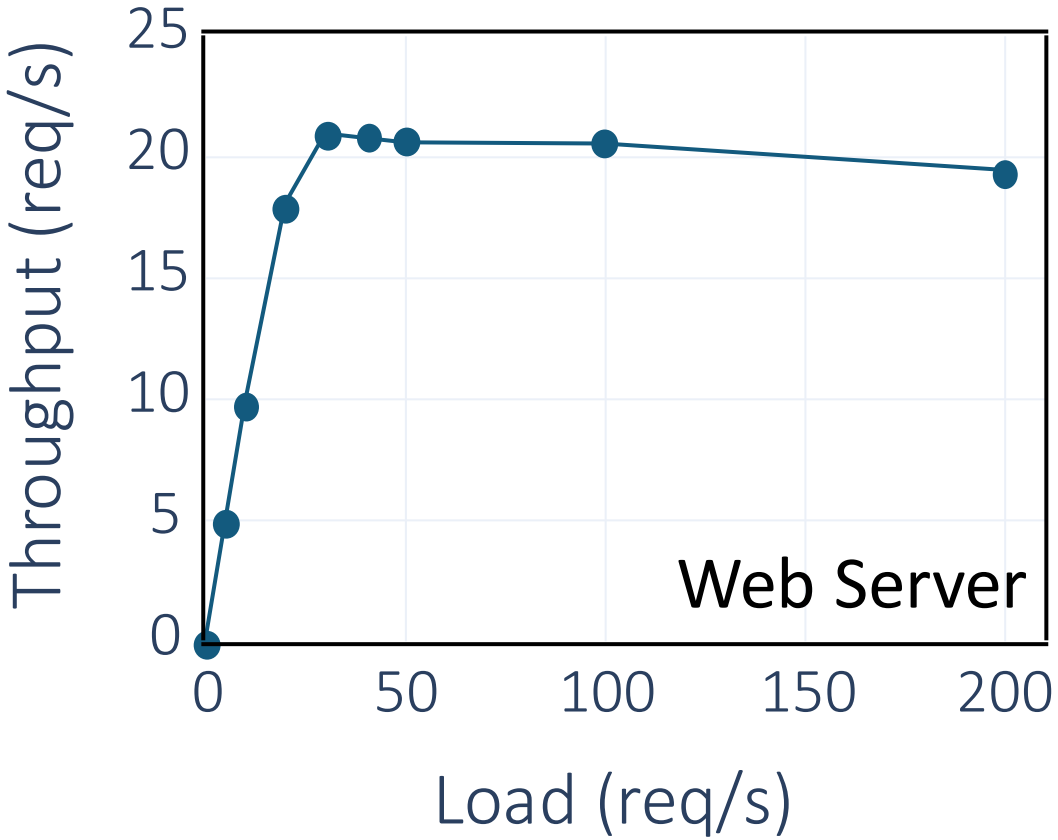
- OS needs to know about blocking events (I/O, locks)
- Switches blocked threads to maintain high core utilization

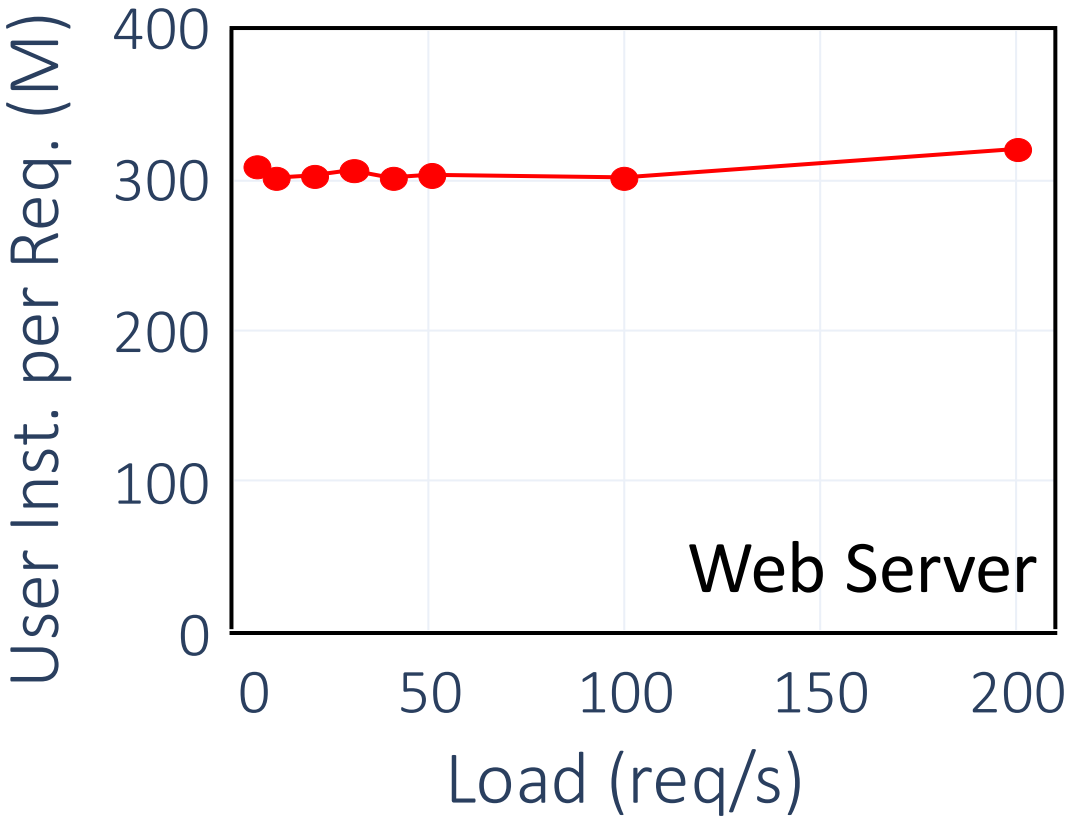
For a single core, IPC works!

For multicore, measure U-IPC [Wenisch, IEEE Micro'06]

$$\text{U-IPC} = \frac{\#\ of\ user\ instructions}{\#\ of\ cycles\ (including\ OS)}$$

WHAT TO MEASURE: U-IPC





- # of user instructions/request remains constant with load

→U-IPC measures forward progress

U-IPC CHALLENGES

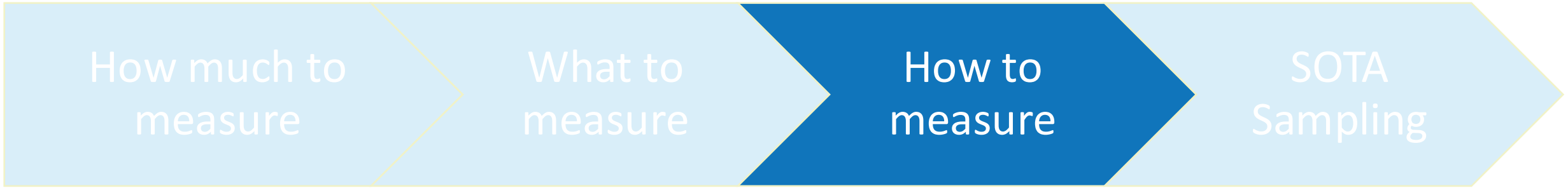
PARALLEL SYSTEMS ARCHITECTURE LAB

- Spinning only in the OS has to be validated per workload
  - Can be done on real hardware
  - Measure the number of instructions executed
  - Already validated for CloudSuite, DCPerf and DeathStarBench

- Any user-level spinning site must be detected and instrumented
  - E.g., the IX kernel implements a user-level network stack for Memcached
  - Count out the spinning from the total instruction count
  - Need a corrected (spin-free) IPC

TALK ROADMAP

PARALLEL SYSTEMS ARCHITECTURE LAB



COMMON MEASUREMENT TECHNIQUES

PARALLEL SYSTEMS ARCHITECTURE LAB

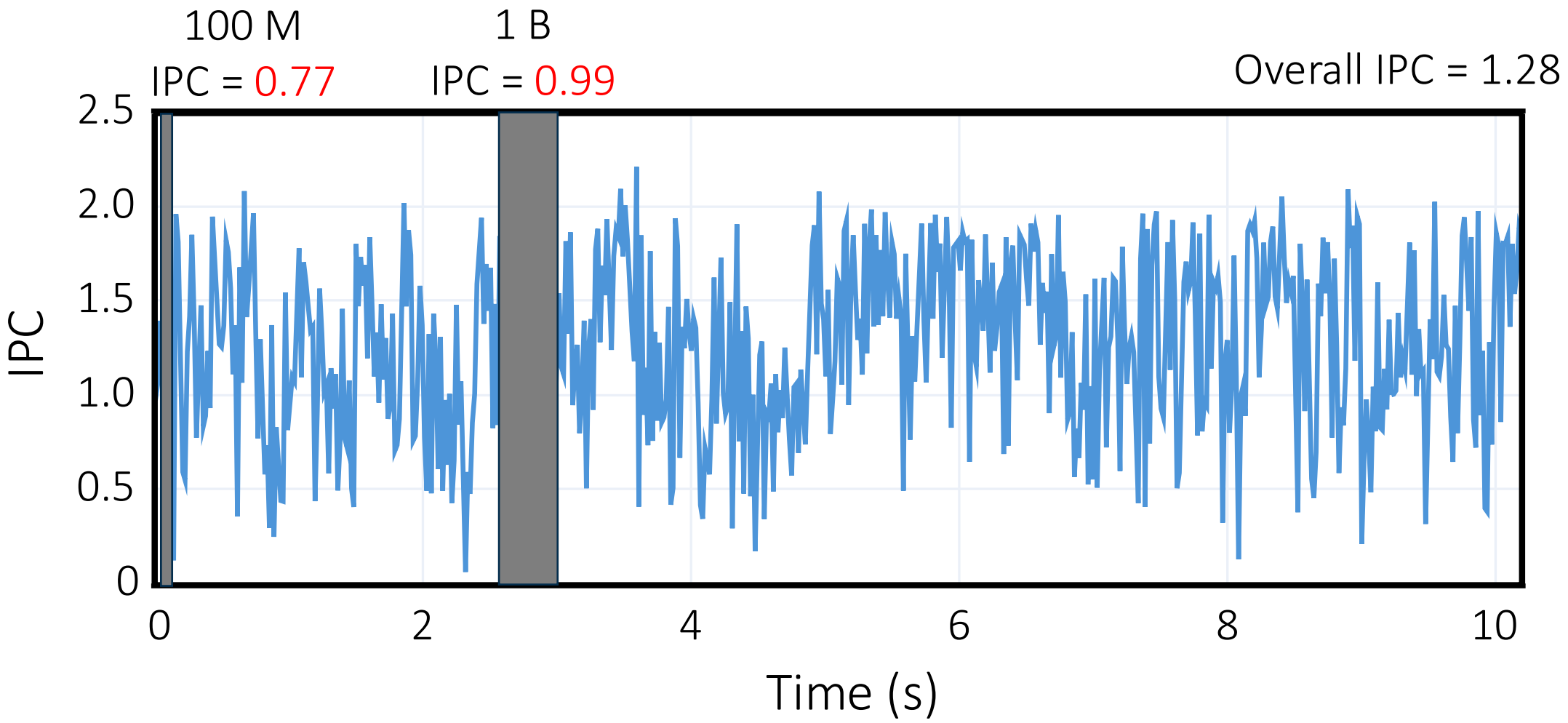
Abbreviated measurement:

1. Fixed window (e.g., 100M to 1B instructions) from the execution
   - Most common in academia
2. Basic-block vector clustering from the binary (SimPoint)
   - Common in academia and industry
   - User-level (or application) code
3. Statistical sampling (SMARTS)
   - Used in industry

WHAT'S WRONG WITH FIXED WINDOWS

PARALLEL SYSTEMS ARCHITECTURE LAB

Single-core Web Server running on ARM Neoverse N1 @ 3 GHz



SIMPOINT

PARISA
PARALLEL SYSTEMS ARCHITECTURE LAB

Extract windows of repetitively executing basic blocks in the workload using K-means clustering.

— Sherwood et. al., ASPLOS'02

- ✓ Abbreviates measurement by several orders of magnitude
- ✓ Higher accuracy relative to fixed windows
- ✓ Relatively simple to implement
- × Represents the binary not the execution (e.g., OS hiccups not included)
- × No statistical error bounds or confidence

SMARTS

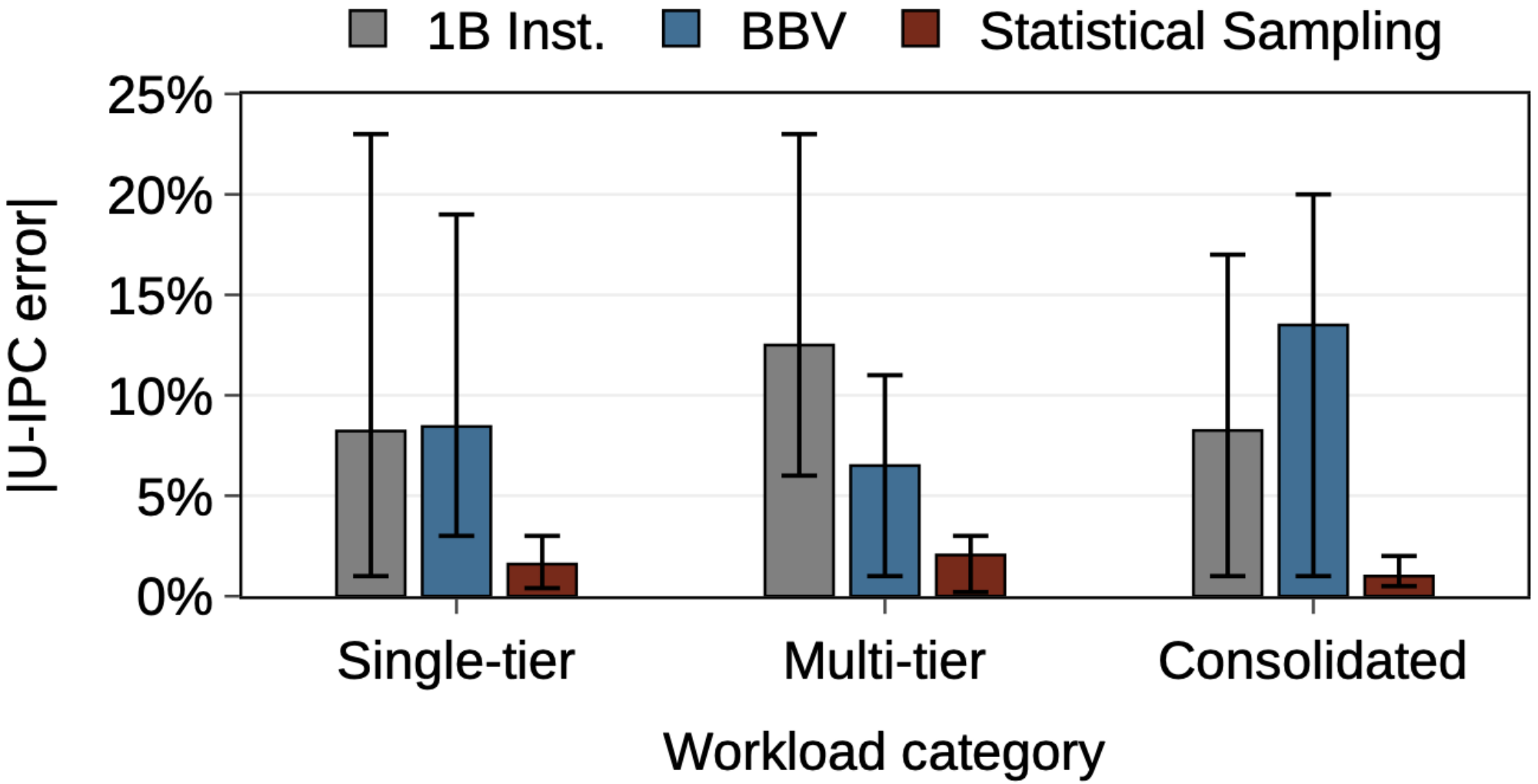
PARALLEL SYSTEMS ARCHITECTURE LAB

Extract a representative sample from a workload's execution using statistical sampling.

— Wunderlich et. al., ISCA'03

- ✓ Abbreviates measurement by several orders of magnitude
- ✓ Error bounded with confidence
- ✓ Represents execution (not the binary)
- × Requires multiple synergistic simulation models
- × Simulation speed capped by higher-level models (e.g., cache simulation)

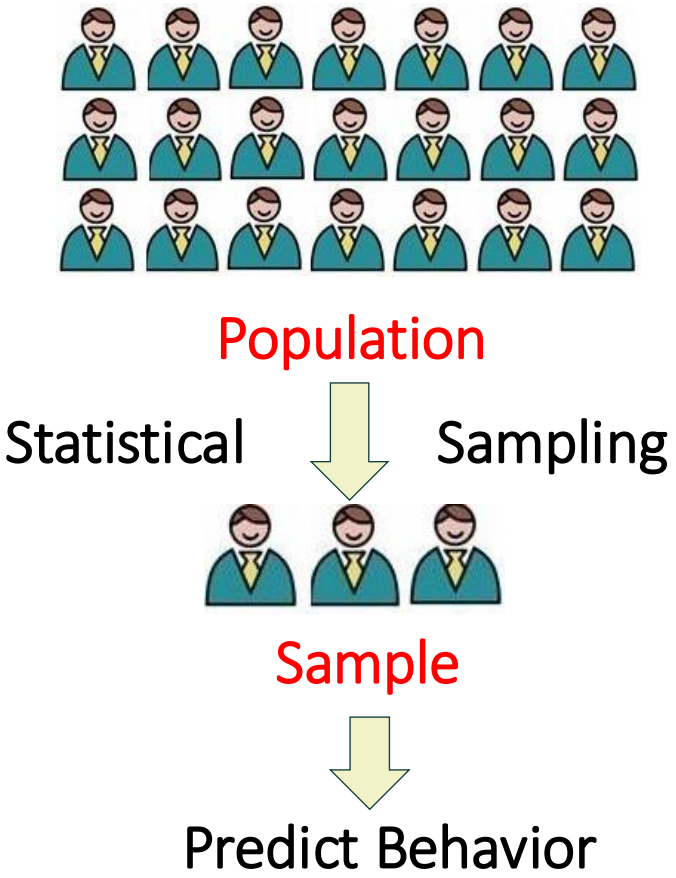
ACCURACY OF MEASUREMENT TECHNIQUES



HOW TO MEASURE? STATISTICAL SAMPLING

PARALLEL SYSTEMS ARCHITECTURE LAB

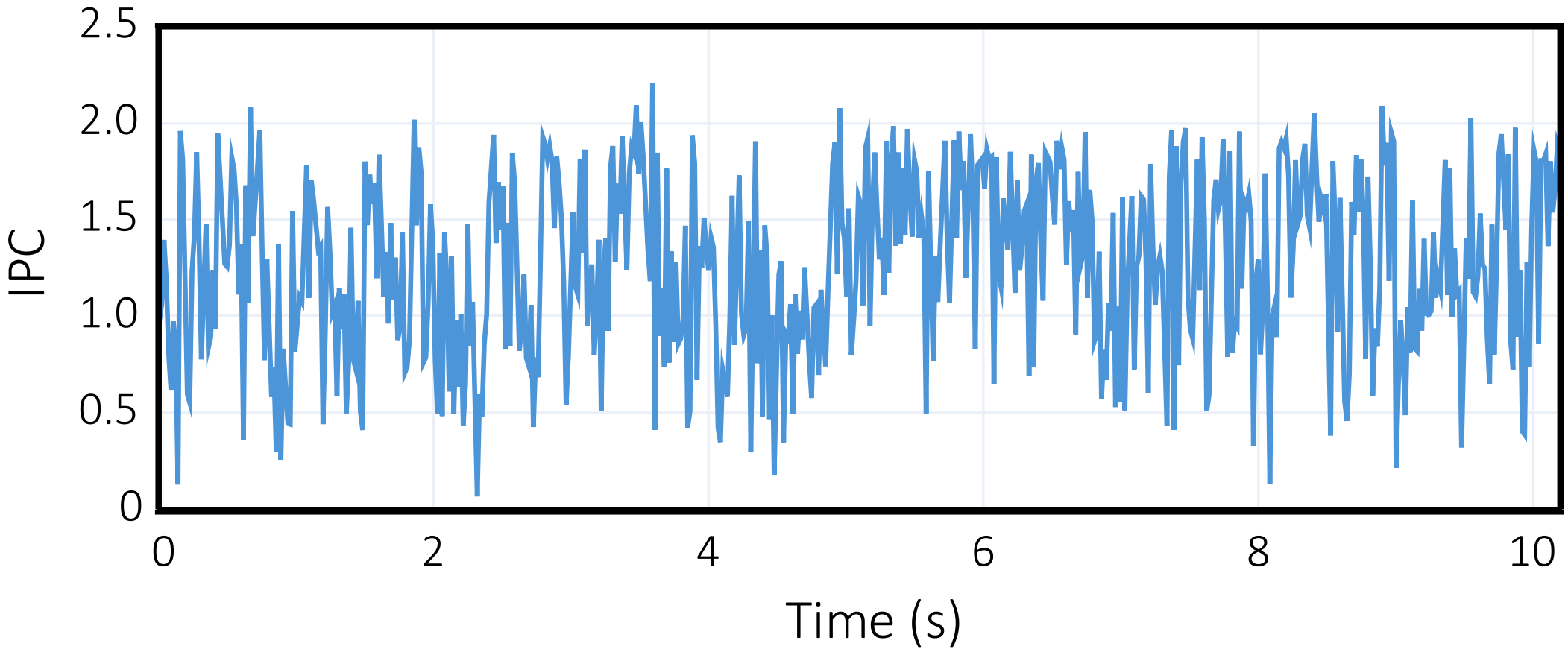
- Random selection of population
  - E.g., 3000 out of 300 million
- Predict the behavior based on the selected sample
- Features:
  - High accuracy
  - Simple
  - Strong mathematical foundation



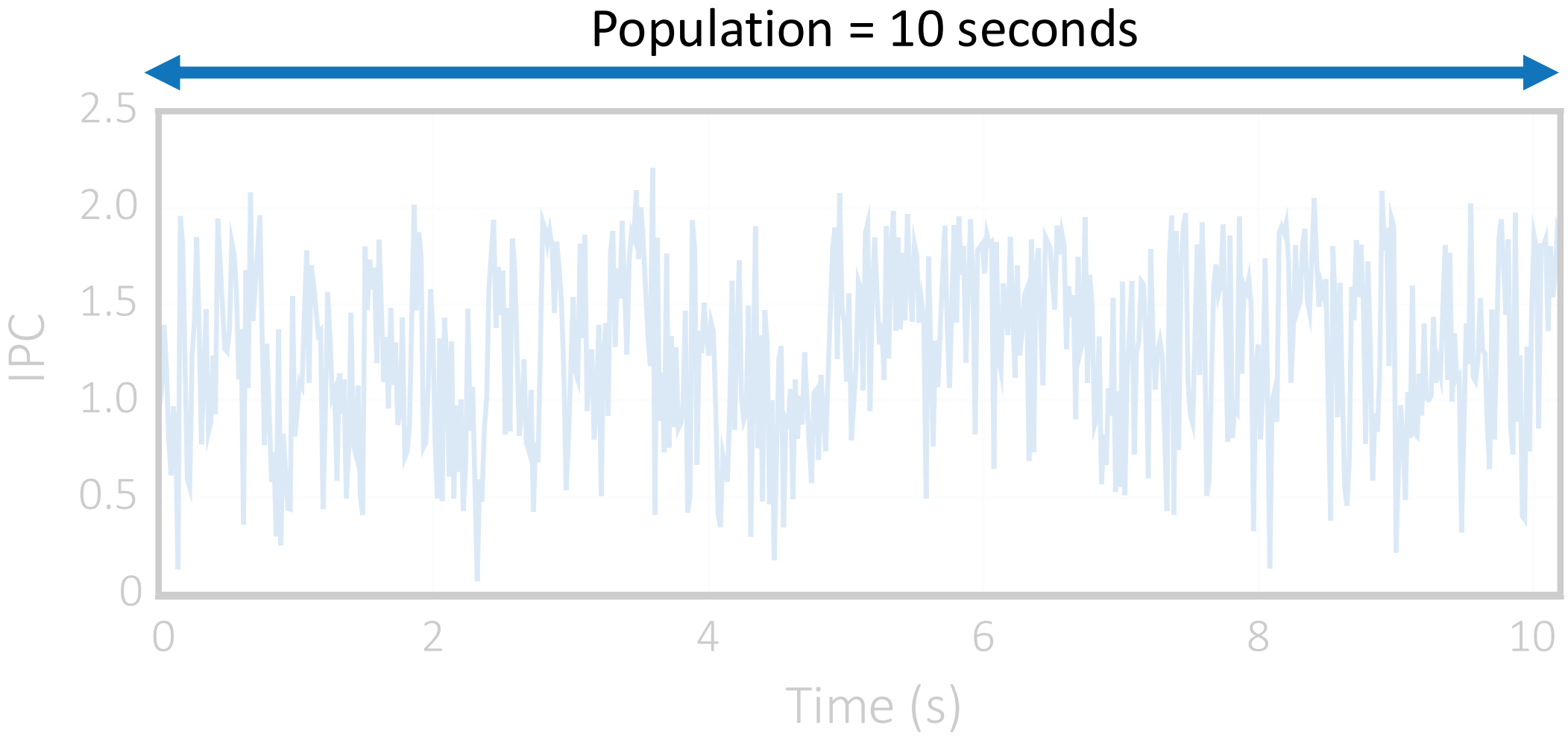
Power of a small part to predict behavior of a whole

SAMPLING EXECUTION UNIFORMLY

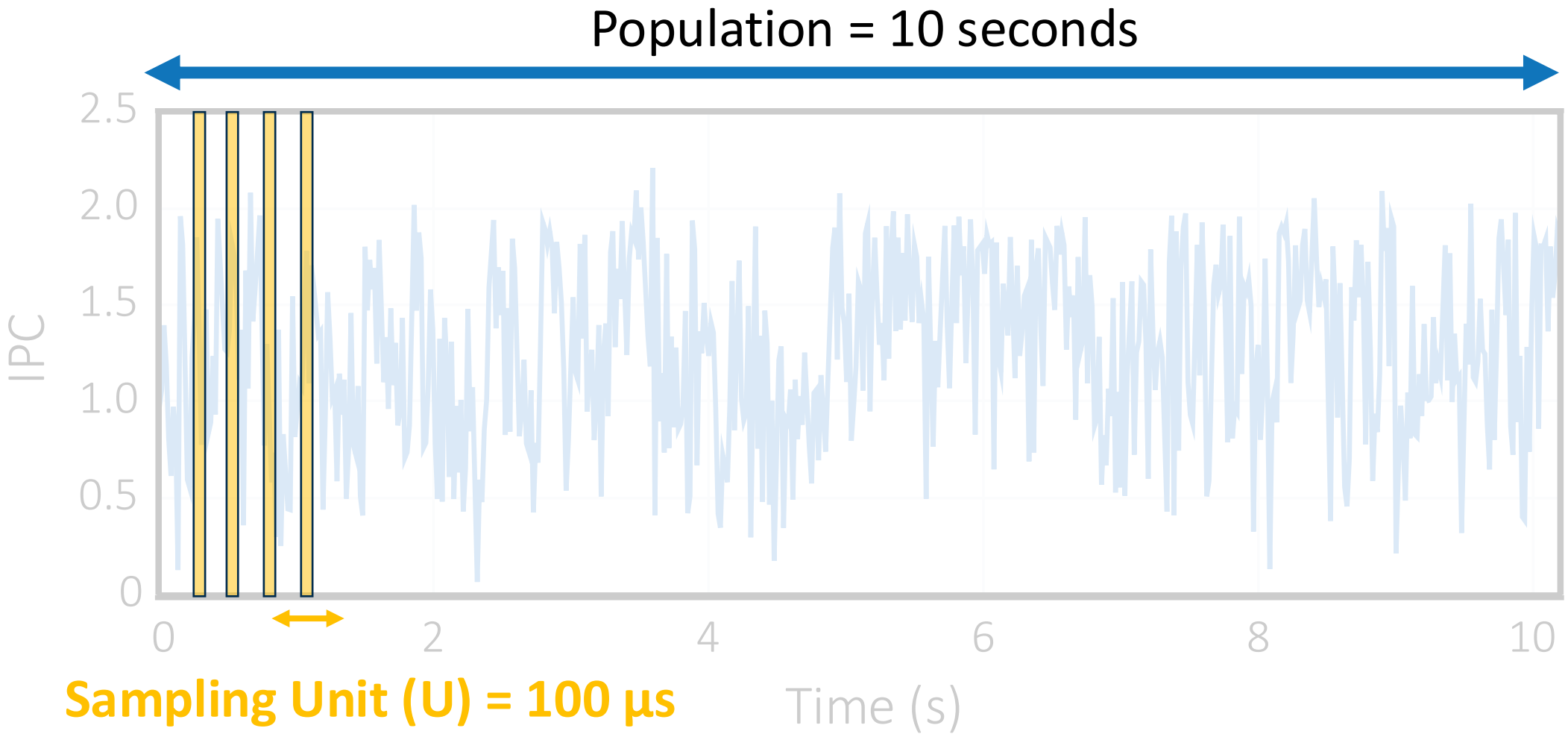
PARALLEL SYSTEMS ARCHITECTURE LAB



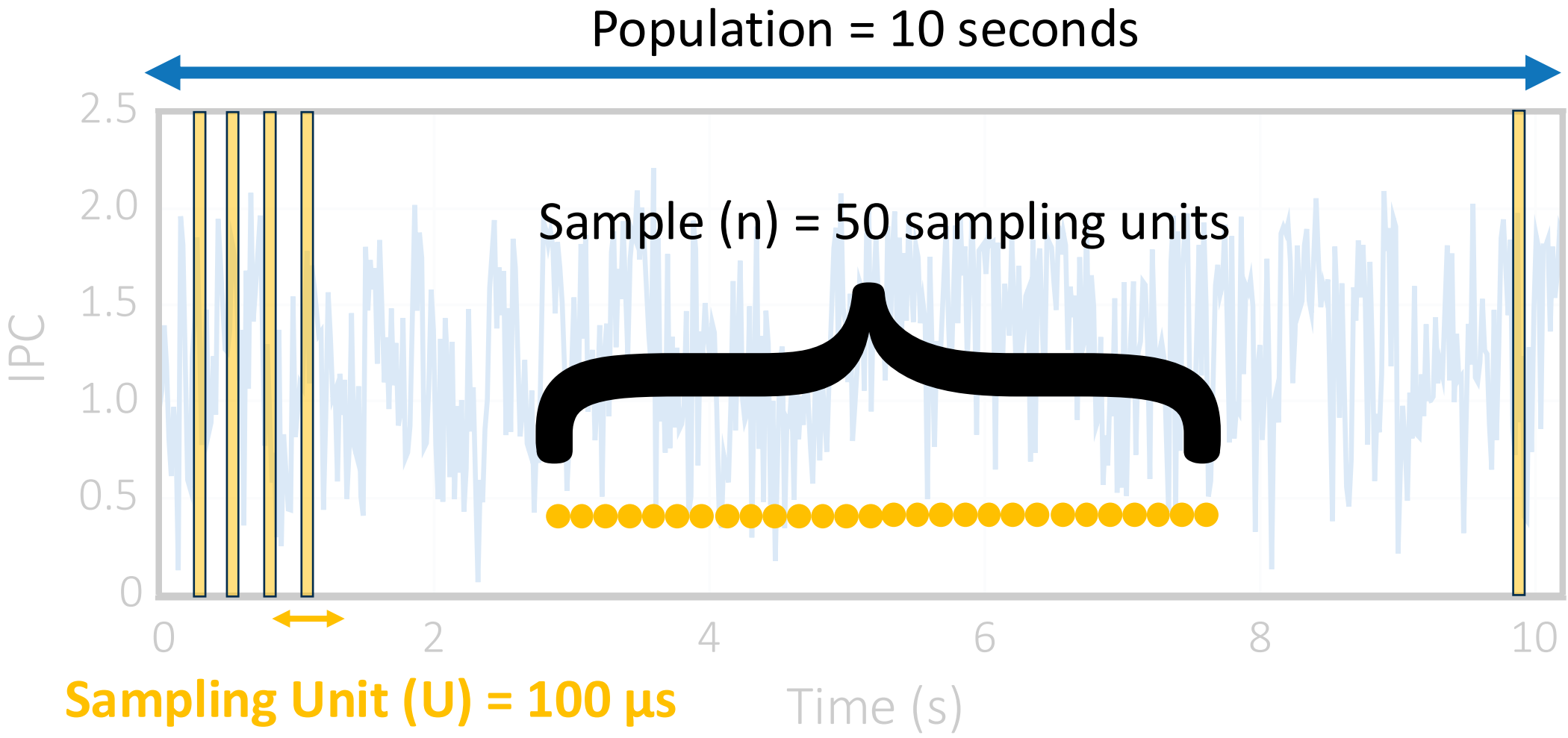
SAMPLING EXECUTION UNIFORMLY



SAMPLING EXECUTION UNIFORMLY



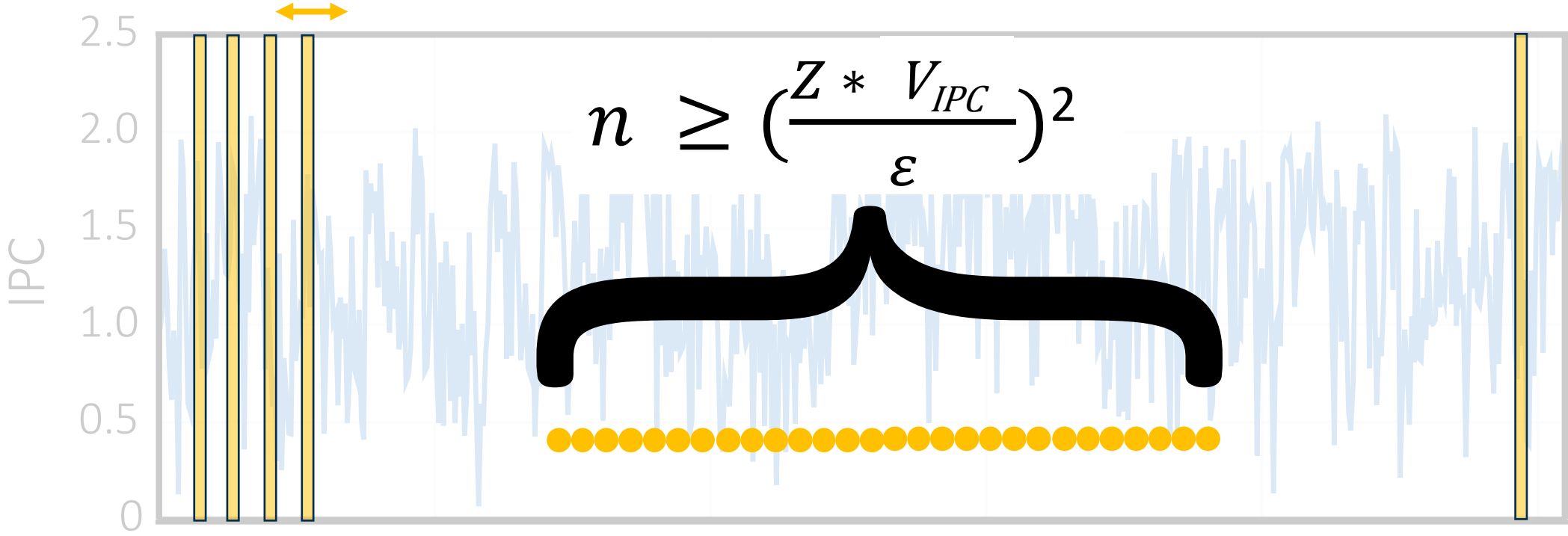
SAMPLING EXECUTION UNIFORMLY



SAMPLING PARAMETERS

Sampling Unit (U)

Total time in timing simulation = n * U



n = sample size

V_{IPC} = Coefficient of variation of IPC

Z = confidence level

ε = error bound

RELATIONSHIP BETWEEN n & U

Coefficient of variation

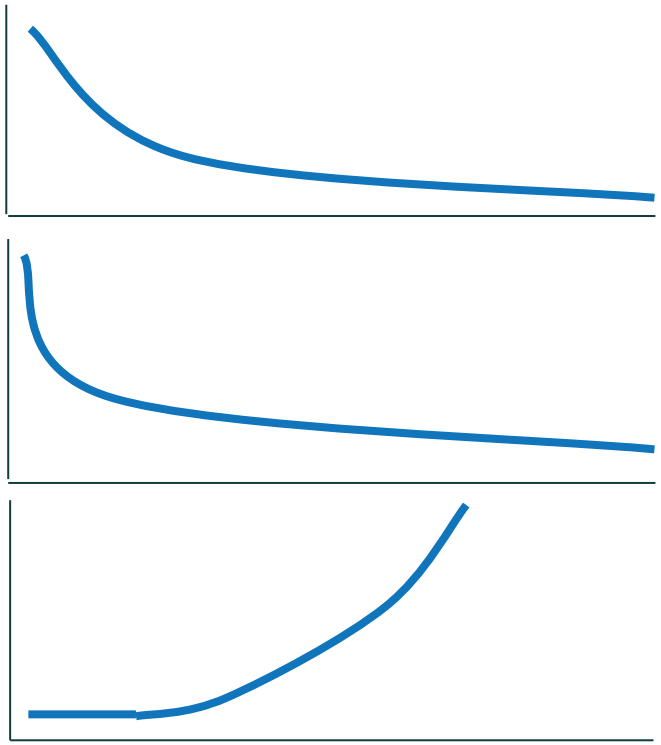
V_{IPC}

Required sample size

n

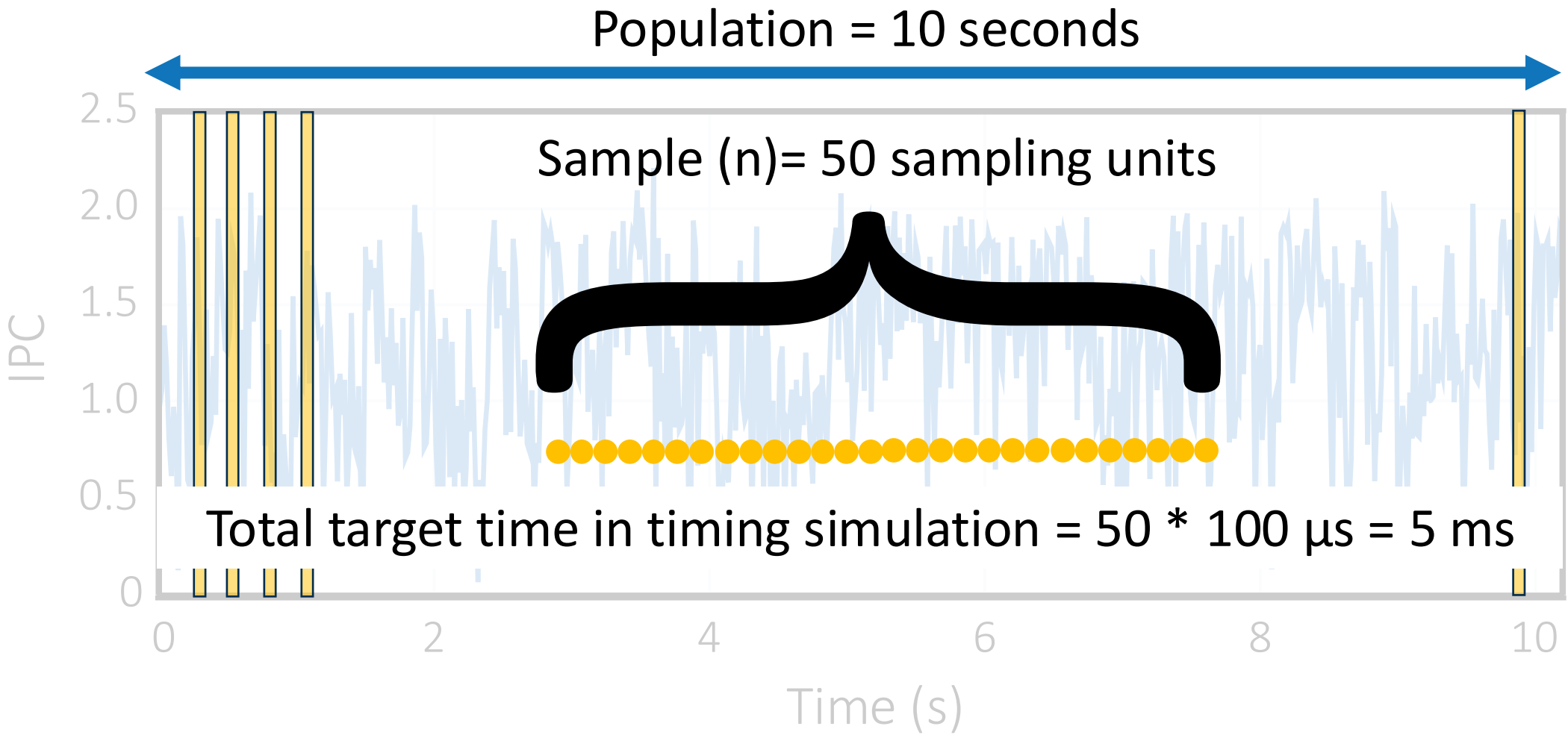
Total time in timing simulation

n * U



A large sample of small units minimizes total time

SAMPLING EXECUTION UNIFORMLY



TIMING SIMULATION TURNAROUND

PARALLEL SYSTEMS ARCHITECTURE LAB

- Assume a timing simulator at 50 KIPS

- For a 3 GHz target clock and an IPC of 1.28 means (our Web Server)
- A single-core timing simulation of 50 sampling units of 100 µs

→~ 35 seconds of simulation

- Assuming a 64-core target, it's just over ~ 37 minutes of simulation

→ Therefore, timing simulation is no longer a bottleneck!

SIMULATION SAMPLING: CHALLENGE

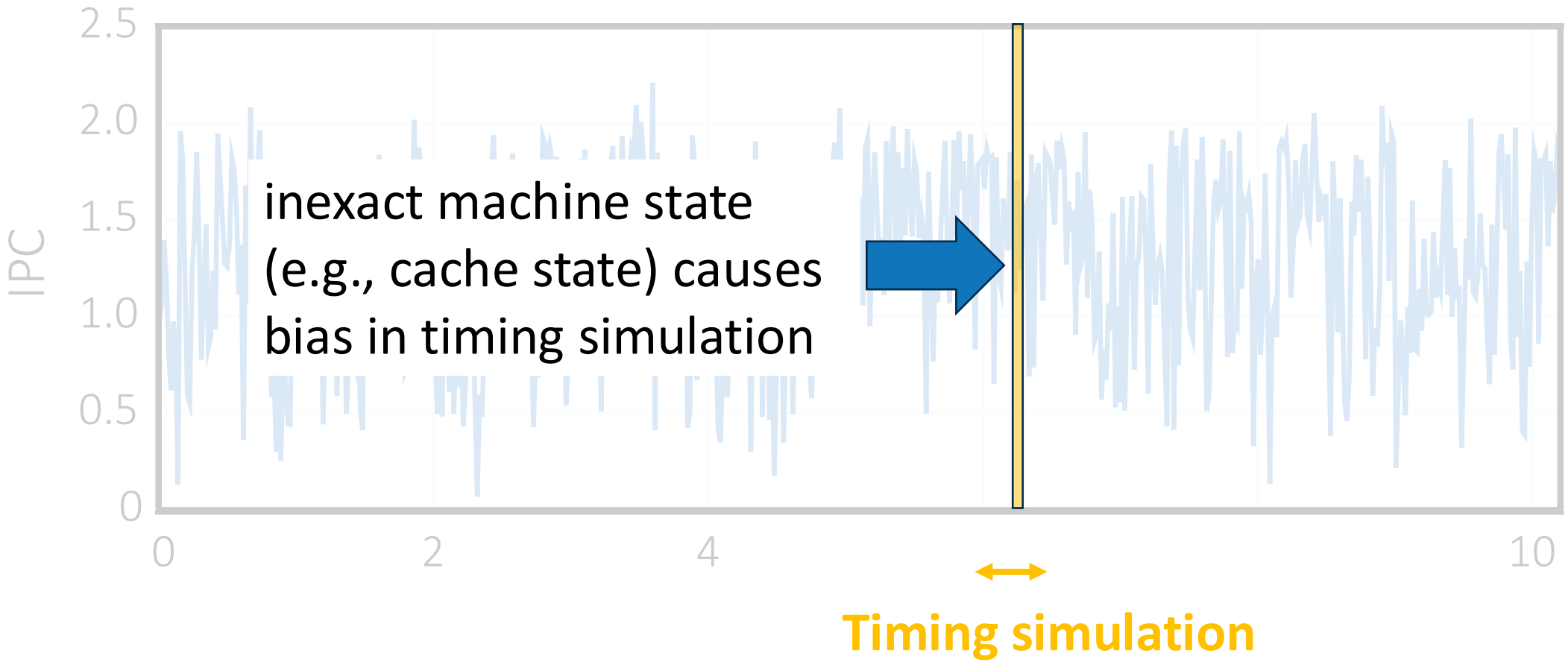
PARALLEL SYSTEMS ARCHITECTURE LAB

Simulation sampling works if machine state can be generated both accurately & fast prior to timing simulation.

— Wunderlich et. al., ISCA'03

- Architectural state (e.g., reg, memory) needed for execution
- uArch state (e.g., caches, TLBs, branch tables) needed for IPC
- Time-dependent activity (e.g., interrupts, communication among cores) needs to be modeled accurately

CHALLENGE #1: STATE ACCURACY



CHALLENGE #1: STATE ACCURACY

PARALLEL SYSTEMS ARCHITECTURE LAB

- Warm for fixed window prior to timing simulation
  - Works for uArch components with bounded warmup (e.g., buffers)
  - Impractical for components with unpredictable warmup (e.g., caches)
  - Bias in uArch leads to large error in timing simulation [Wunderlich, ISCA'03]

- Bias in state impacts time-dependent activity prior to a sampling unit
  - E.g., core interactions are a function how each advances in simulation

CHALLENGE #1: STATE ACCURACY

PARALLEL SYSTEMS ARCHITECTURE LAB

Type of uArch state

Short, predictable

- buffers in pipeline, cache hierarchy, NoC

Long, unpredictable

- caches, branch tables, prefetcher tables

Strategy

Fixed window warmup

- bounded worst-case analysis
- e.g., timing simulation of 200 µs

Keep state warm all the time

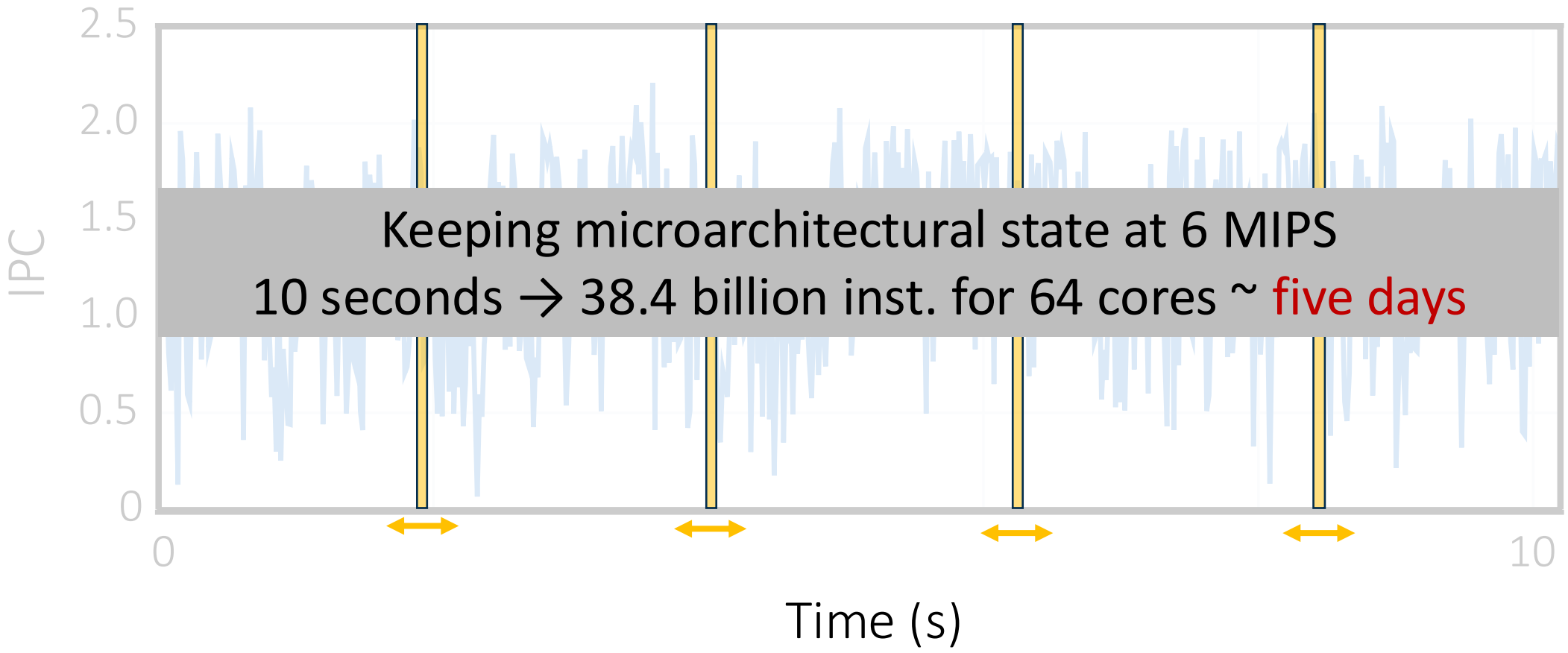
- e.g., functional simulation of uArch components
- slows down simulation!

CHALLENGE #2: SPEED

PARALLEL SYSTEMS ARCHITECTURE LAB

	Model	Speed (MIPS)
QEMU	ISA Emulation	> 400
QEMU + null instrumentation	Functional	85
QEMU + uArch	Functional	18
QEMU + uArch + OS	Functional	6
QEMU + timing	Timing	< 0.3

CHALLENGE #2: SPEED



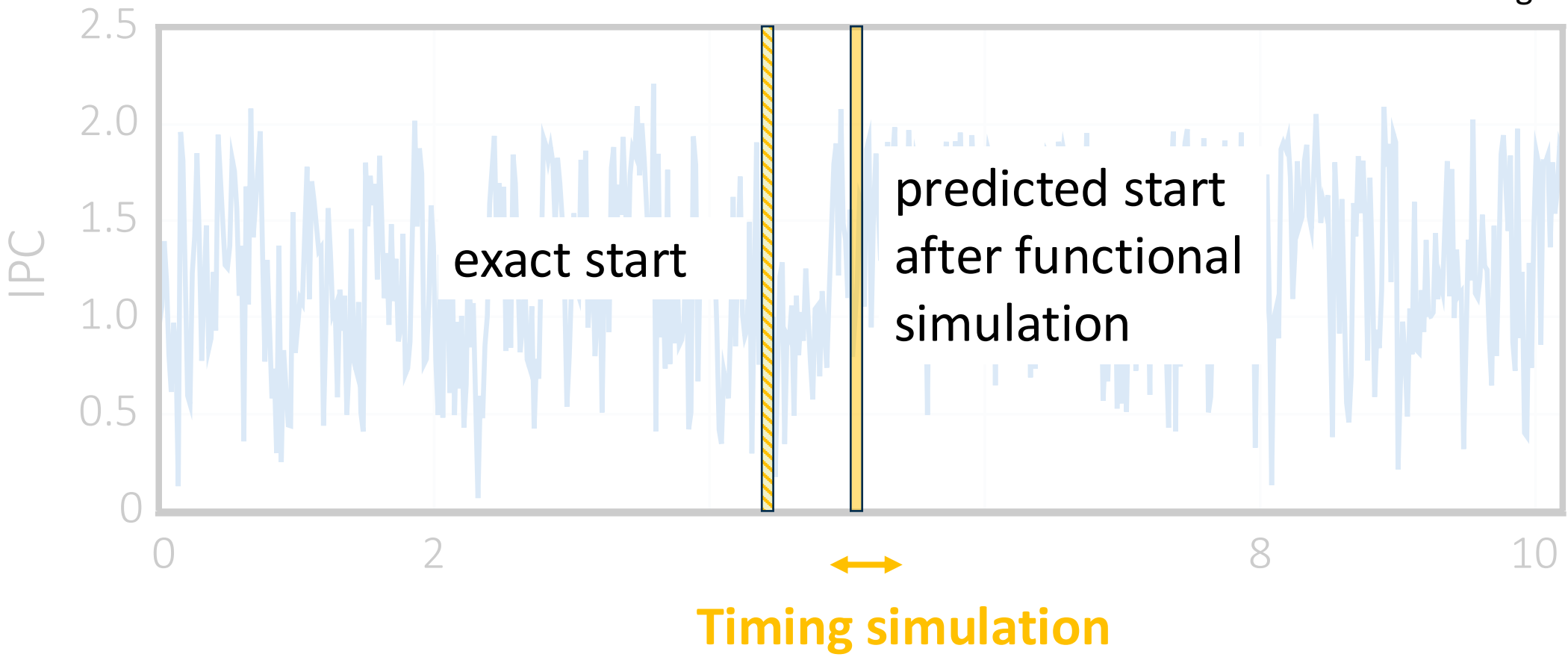
ACCELERATING FUNCTIONAL SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB

- For mobiles (e.g., with a few target cores) 6 MIPS may be enough
  - Can use sampling with QEMU and a sequential functional simulator
- For servers, can parallelize functional simulation on multicore hosts
  - Many prior user-level simulator examples: WWT, ESESC, Sniper, zSim
  - QFlex 3.0 has a parallel full-system functional simulator based on QEMU
  - Requires synchronizing simulation on multiple host cores

CHALLENGE #3: TIME ACCURACY

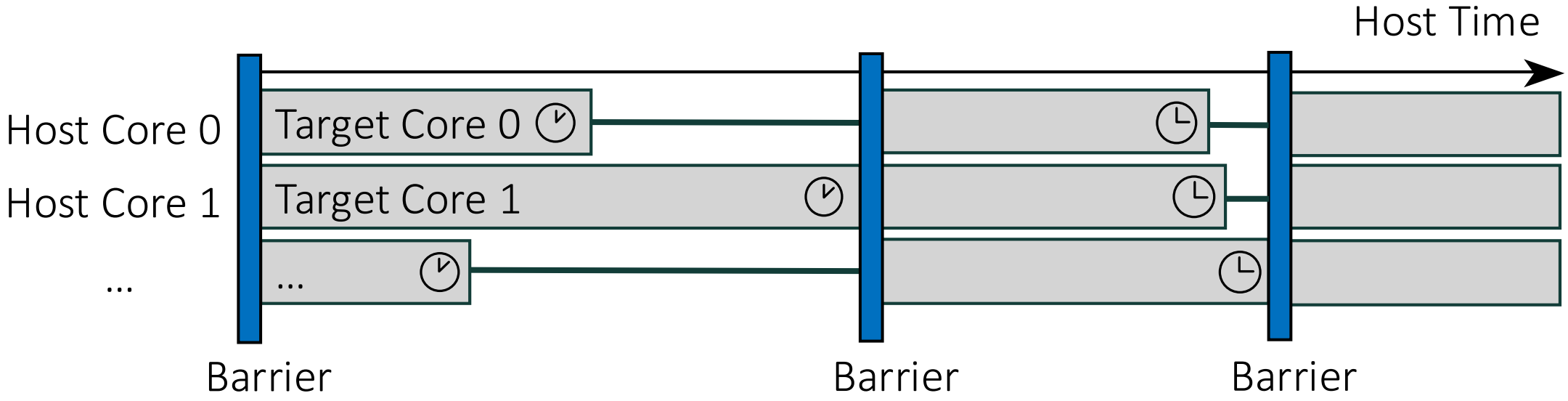
Bias in time impacts uniform sampling ($IPC_{functional}$!= IPC_{timing})



PARALLEL FUNCTIONAL SIMULATION

PARISA
PARALLEL SYSTEMS ARCHITECTURE LAB

- Periodically synchronize target cores through barriers [Fujimoto'90]
  - Parallel simulation quantum (PSQ) = period in target number of cycles
  - Functional simulation has no clock, only instruction counts
  - Need to model time per target core



MODELING TIME IN FUNCTIONAL SIMULATION

PARALLEL SYSTEMS ARCHITECTURE LAB

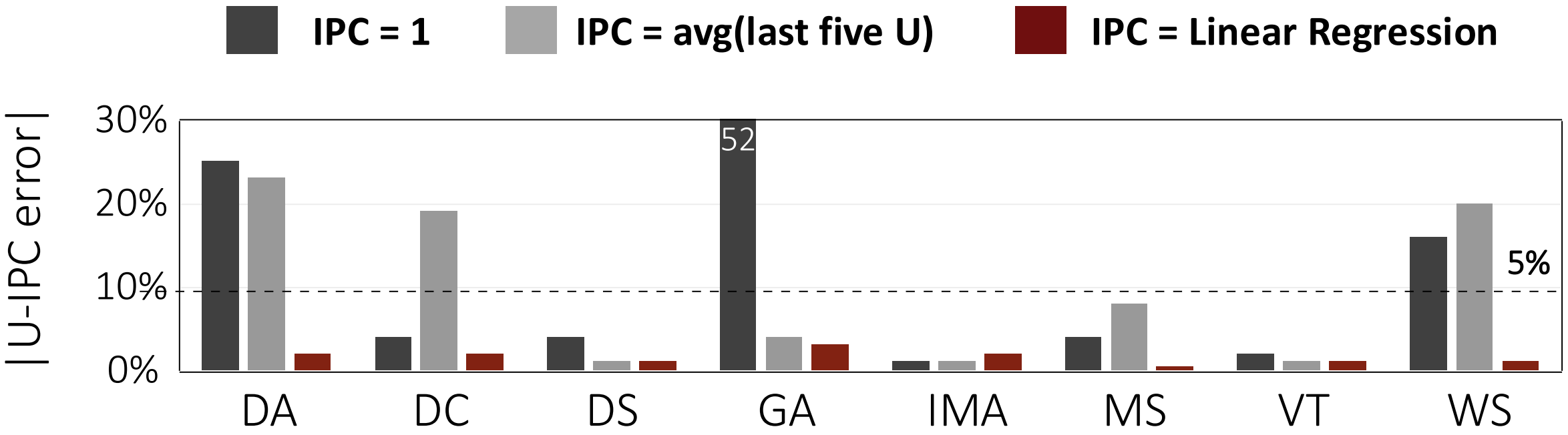
- Prior proposals for time-based sampling
  1. IPC = 1 used in SimFlex [Wenisch, IEEE Micro'06]
  2. IPC = measured IPC w/ timing simulation of prior sampling units used in ESESC [Ardestani, HPCA'13]

- Alternatively, IPC can be modeled analytically
  1. ideal CPI (width) + uArch front-/back-end stalls used in Sniper [Carlson, SC'11]
  2. interval simulation [Genbrugge, HPCA'10]
  3. linear regression model from a sample, Ben Lee's thesis [ASPLOS'06]

- Use per-core IPC to extract instruction counts
  - IPC includes all instructions (together with OS) for clock management

ERROR FROM BIAS IN TIME: SINGLE-TIER

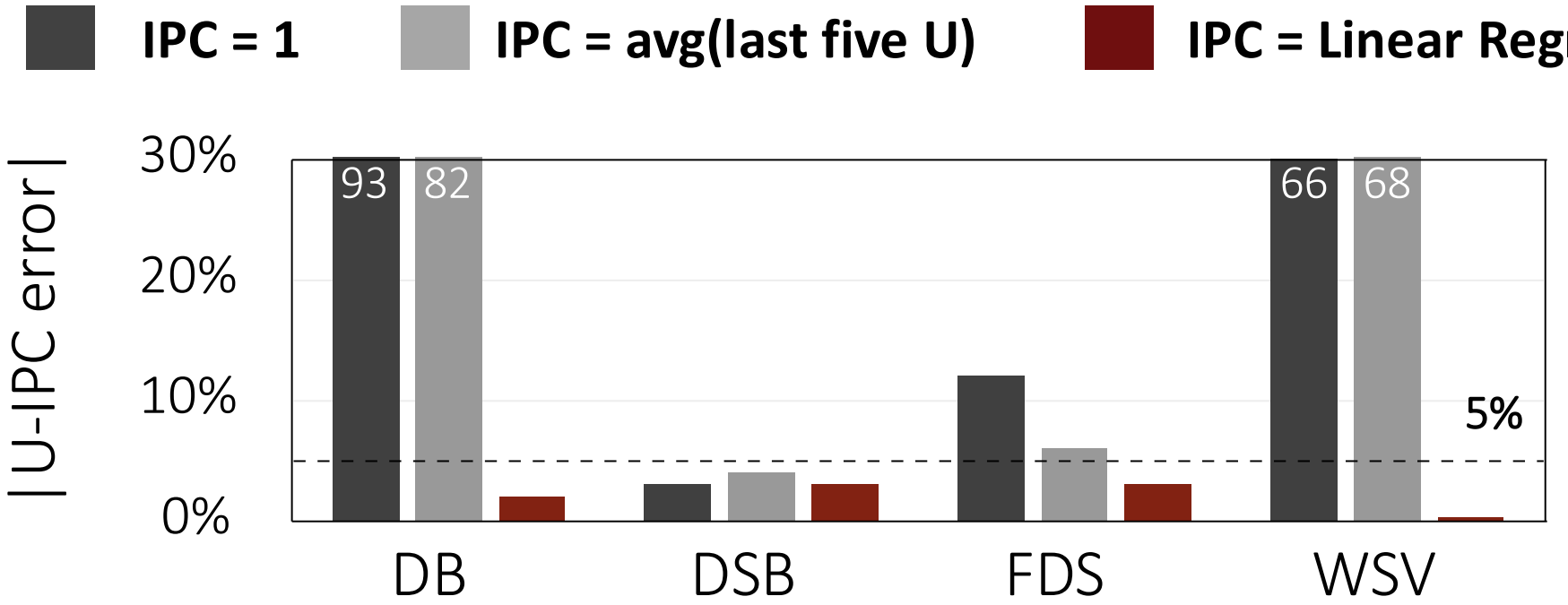
PARALLEL SYSTEMS ARCHITECTURE LAB



- Single-tier workloads from CloudSuite & DCPerf
- U-IPC error from samples generated from functional simulation with 3 models
- Compared to a full timing run of a two-core (ARM Neoverse N1) server
- First two models bias sampling unit towards faster windows of execution

ERROR FROM BIAS IN TIME: MULTI-TIER

PARALLEL SYSTEMS ARCHITECTURE LAB



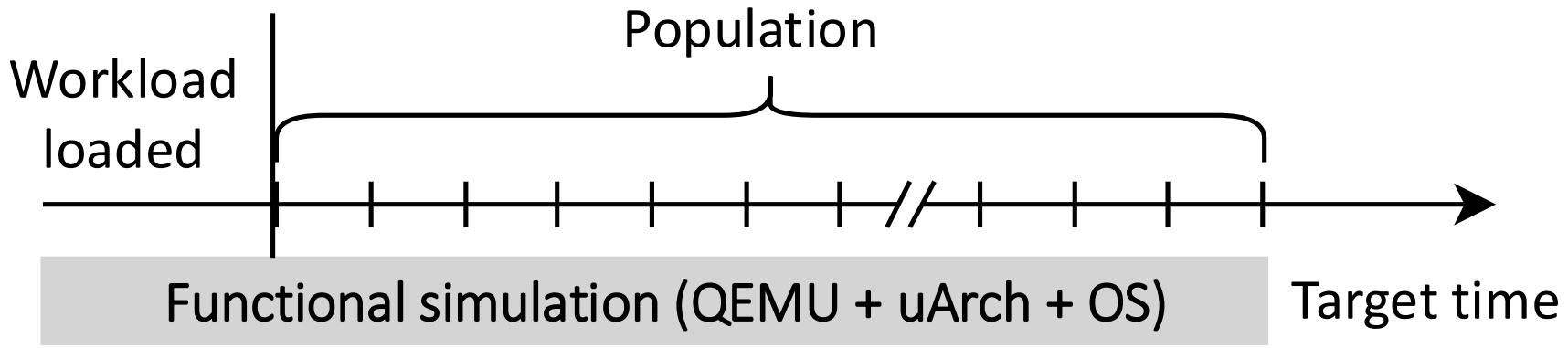
- Multi-tier workloads from CloudSuite, DCPerf, DeathStarBench
- These workloads are highly impacted by the bias in relative speed among tiers
- Prior work has not studied multi-tier workloads

TALK ROADMAP

PARALLEL SYSTEMS ARCHITECTURE LAB

How much to measure → What to measure → How to measure → **SOTA Sampling**

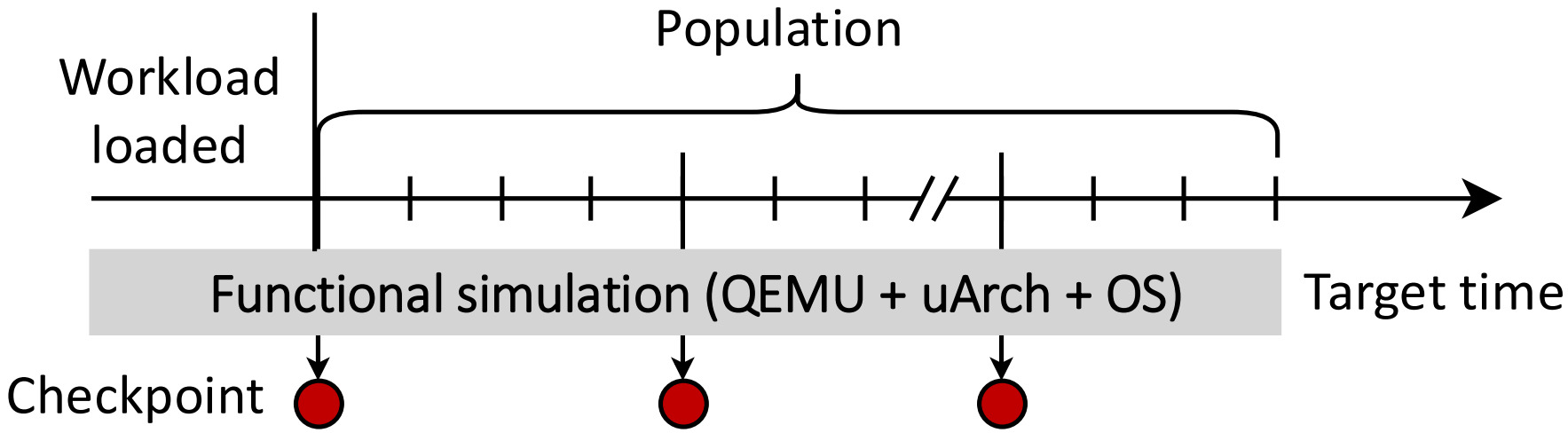
SOTA SAMPLING WITH QFlex 3.0



Run the workload for the duration of population (e.g., 5-60 seconds) using functional simulation

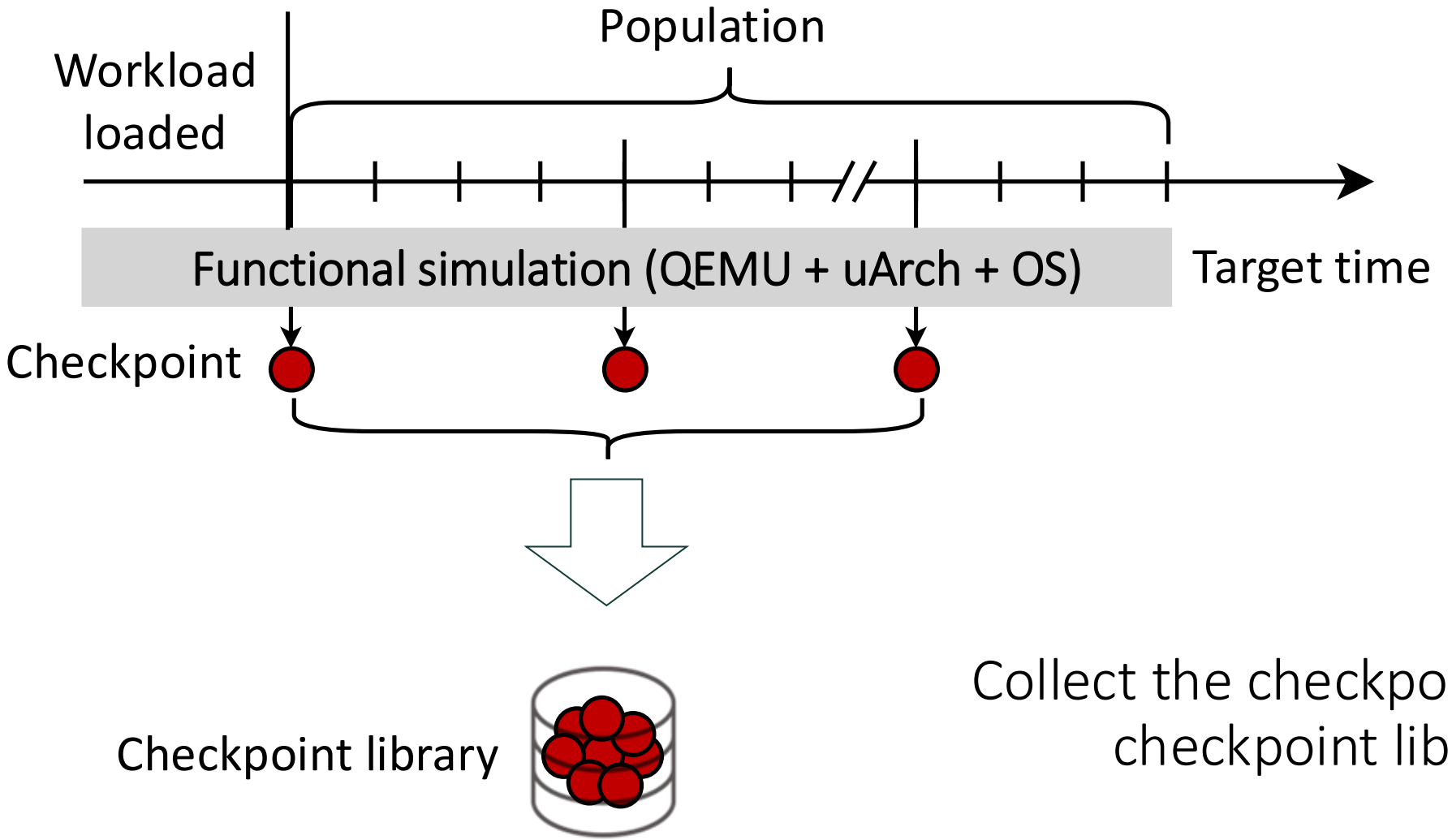
Generate architectural & microarchitectural state for a sample

DRAWING A SAMPLE



For a given sample size (e.g., 500) dump incremental checkpoints at uniform intervals

COLLECTING CHECKPOINTS

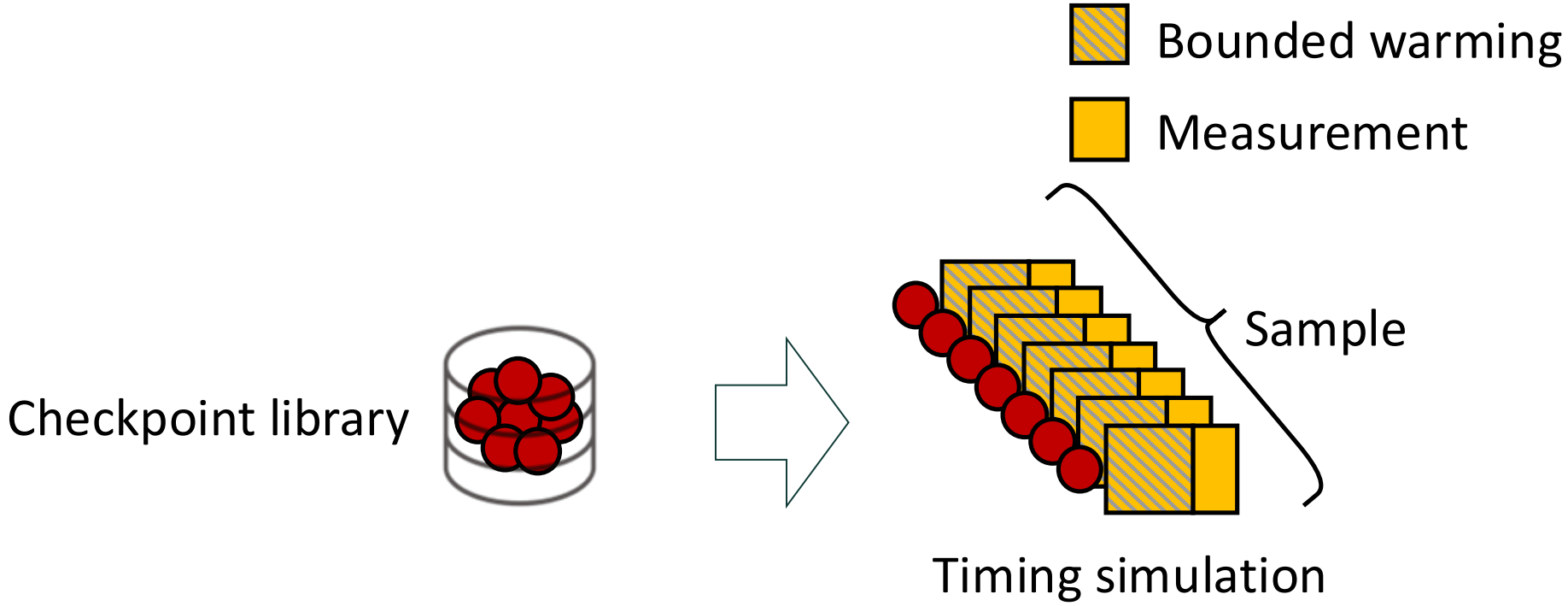


Collect the checkpoints into a checkpoint library

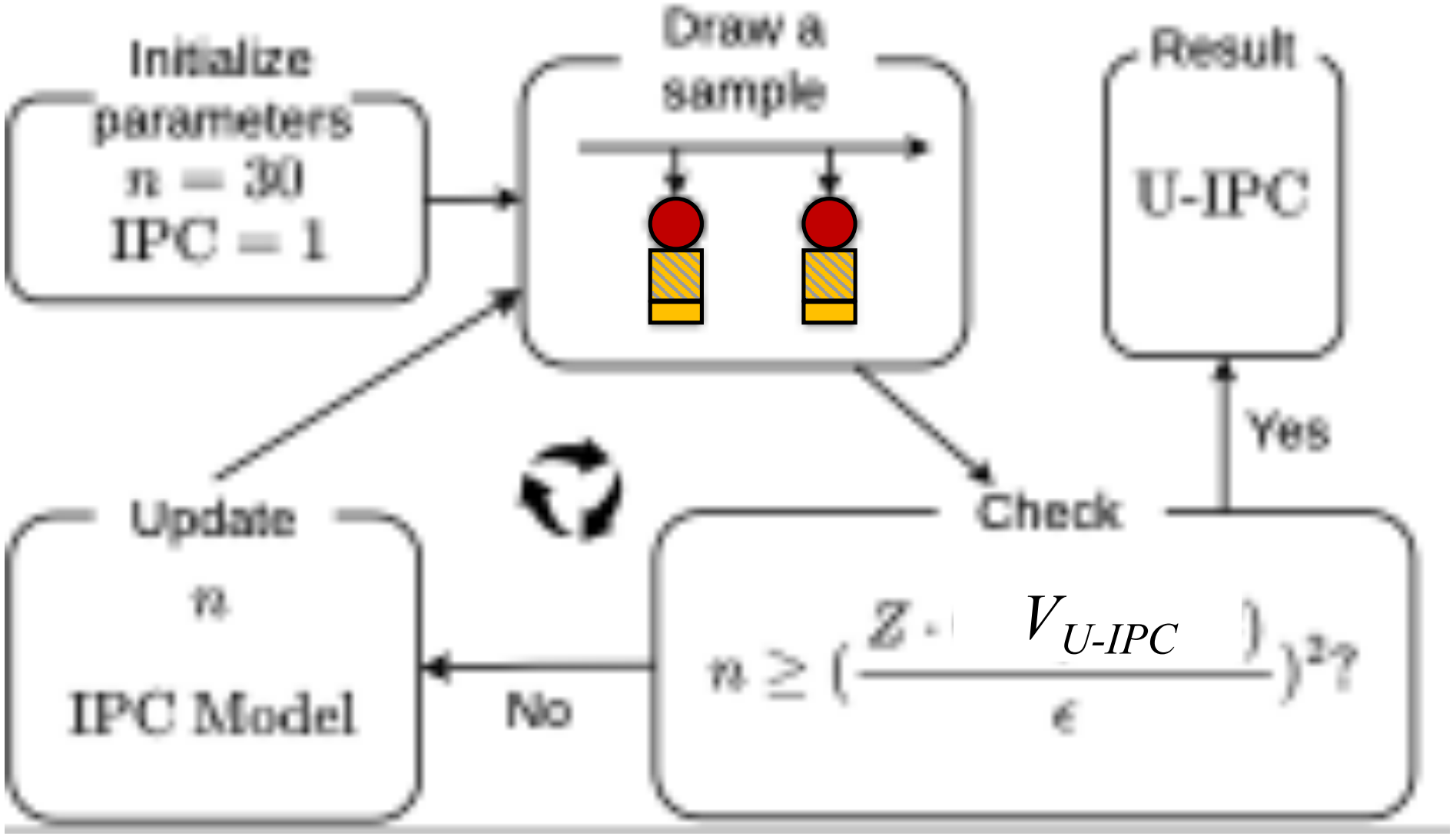
TIMING SIMULATON

Simulate the sample with timing simulation

Timing simulation runs are independent, can be batched



END-TO-END SAMPLING FRAMEWORK



SAMPLING USING STATISTICAL TOOLS

PARALLEL SYSTEMS ARCHITECTURE LAB

1. Evaluating bias in a sample
   - Full timing runs possible for a few target cores (e.g., up to 8) lasting weeks
   - For more target cores, sequential functional simulation can be used as a reference
   - Various tools for bias in distribution: Wasserstein distance, KL divergence, K-S test,…

2. Interactive sampling [Wenisch, IEEE Micro'06]
   - Tune error and confidence bounds
   - Shuffle checkpoint library
   - Draw a random subset of checkpoints

3. Matched-pair comparison [Ekman, ISPASS'05]

OPEN PROBLEMS

PARALLEL SYSTEMS ARCHITECTURE LAB

- Limitations of statistical sampling
  - Extreme metrics (e.g., max temperature) [Ardestani, HPCA'13]
  - Coarse-grain metrics (e.g., request latency)
  - SLO (tail latency) requires hundreds of seconds of measurement
- Multi-node simulation
  - Simics has multi-node support, QEMU/M5 don't
  - Dist-gem5 [Alian, ISPASS'17]
- Interoperability
  - How to draw a sample with QFlex and run the sample with any timing simulator?
  - QPoints [Godala, gem5 workshop ISCA'23]

CONCLUSIONS: BACK TO THE FUTURE

PARALLEL SYSTEMS ARCHITECTURE LAB

- Need to measure a few to tens of seconds of execution

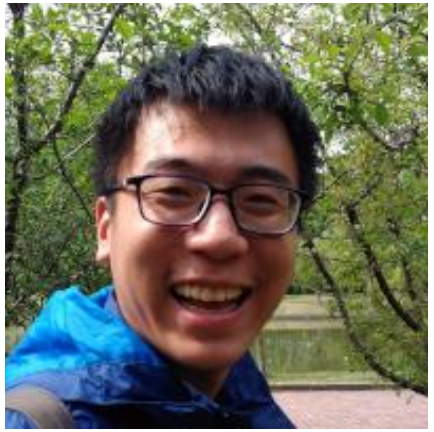
- Current practices to measure performance are error-prone
  - Need full-system simulation
  - Fixed windows do not represent the workload or execution
  - Techniques representing the workload (e.g., SimPoint) do not represent the execution
  - Need not just an error bound but also confidence

- Statistical sampling is 20-year-old promising approach for full-system timing
  - It's a lot of work (workloads, metrics, statistics/math, tools)
  - Great place to pick up, need to address open problems

OUR TEAM

PARALLEL SYSTEMS ARCHITECTURE LAB

Mohammad

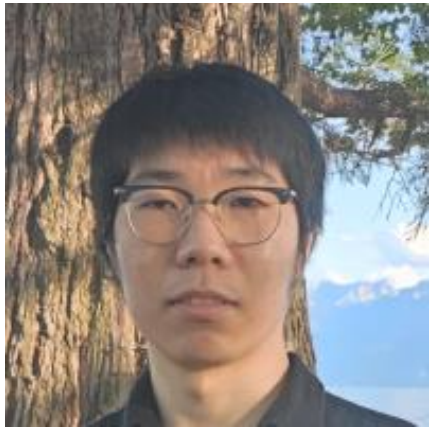


Shanqing

Ali

Ayan

Pooria



Yuanlong

Thank You!

PARALLEL SYSTEMS ARCHITECTURE LAB

For more information, please visit us at

qflex.epfl.ch

EPFL