What not to do when simulating large workloads!

Maryam Babaie, Ayaz Akram, and Jason Lowe-Power

DArchR, Computer Science Department, UC-Davis

gem5 Workshop at ISCA 2023

Orlando, FL, USA

UCDAVIS
UNIVERSITY OF CALIFORNIA

Introduction

- The methodology refers to the settings of the experimental infrastructures.
  - Benchmark application
  - Simulation configuration
  - Evaluation metrics

Problem:

For *large-scale applications*, we must pick a portion that is representative and ensure that across different configurations the same portion will be compared.

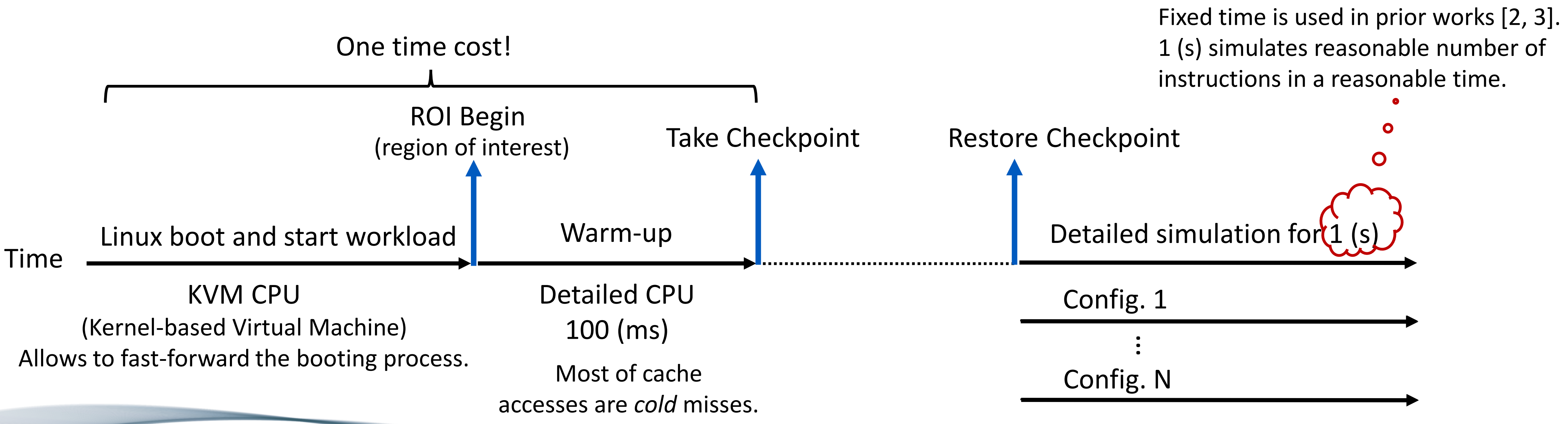
- **Properly measuring the amount of *actual work progress* at each run is vital!**

- This is a difficult task in large-scale applications [1].
  - Threads interfere with one another.
  - Long spin-loops

UCDAVIS

Example

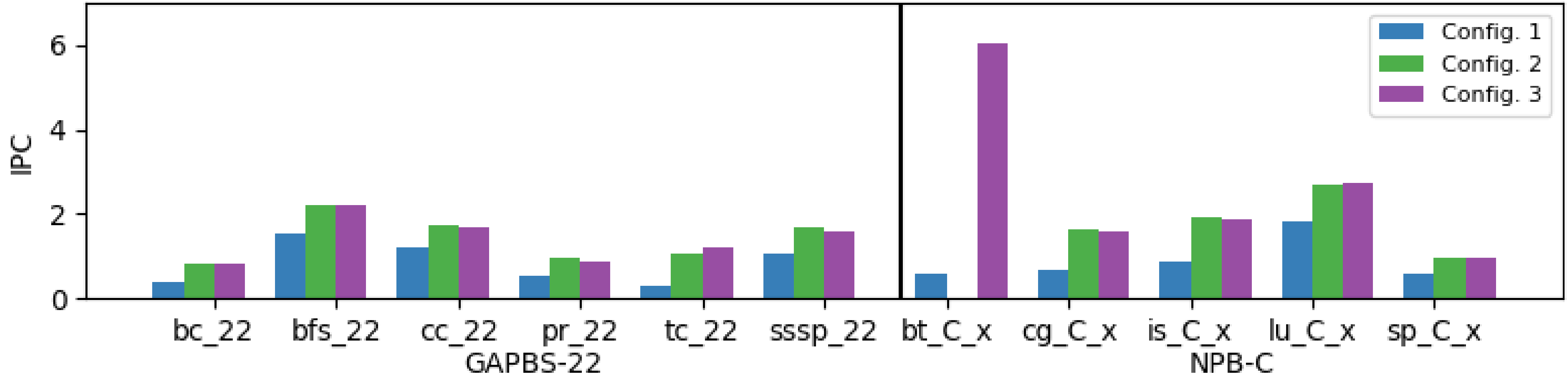
- We want to evaluate a proposed cache hierarchy.
- Benchmarks: a subset of GAPBS (input 22), NPB (class C) applications.
- Comparing 3 systems: Config.1, Config.2, Config.3
- Checkpoint: stores the architectural state of the system (e.g., the state of caches).
  - Each microarchitecture can restore it and will ensure that they all start with the *same* state.



UCDAVIS

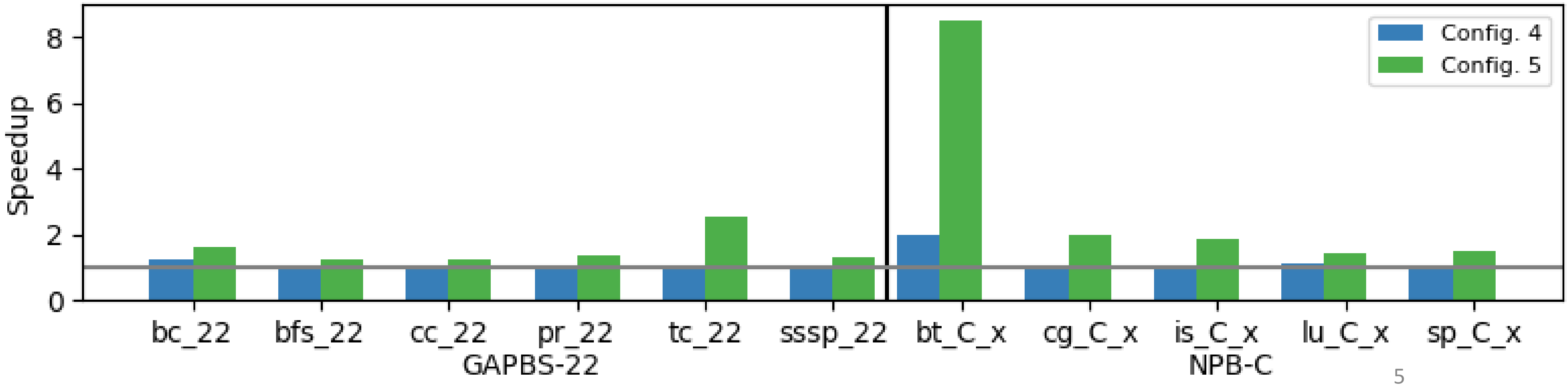
Experiment 1

- This is a fixed time simulation. So, we use IPC for evaluation.
- *bt* from NPB suite, in Config.2, never finished 1 (s) simulation within an expected time-frame! (busy in spin-loop)



Experiment 2

- Experiment 2: speedup of two new systems (Config. 4 and 5), compared to Config.1
- $Speedup_{\ Config.n} = \frac{IPC_{\ Config.n}}{IPC_{\ Config.1}}$
- *bt* in Config.5 is an outlier without any reason explained by Config.5.



UCDAVIS

What’s the issue?

- The detailed simulations are bounded by a fixed execution time.
- The Restore1(sec) part does not guarantee:
  - Maintaining the same program phase across different configurations for comparison fairness.
- Alameldeen et al. reported counting instructions as a metric to measure work progress and for performance comparison can lead to misleading conclusions [4].
  - They proposed a transaction time approach, instead.
  - Too long and complicated.

“... we recommend measuring the time required to complete a fixed number of transactions (or requests) after a suitable warm-up time to eliminate cold-start effects. ...”

UCDAVIS

Solution

- NOT using a *fixed time* and counting instructions for comparison.
- ***LoopPoint:*** sampling technique for multi-threaded HPC applications with spin-loops [1].
  - Selects repeatable loop boundaries of a practical region size.
  - Records the most recent program counters (PC) within the region.
- Bound simulation by the most recent PC-count.
  - Provides a better mechanism to properly measure amount of work progress.

UCDAVIS

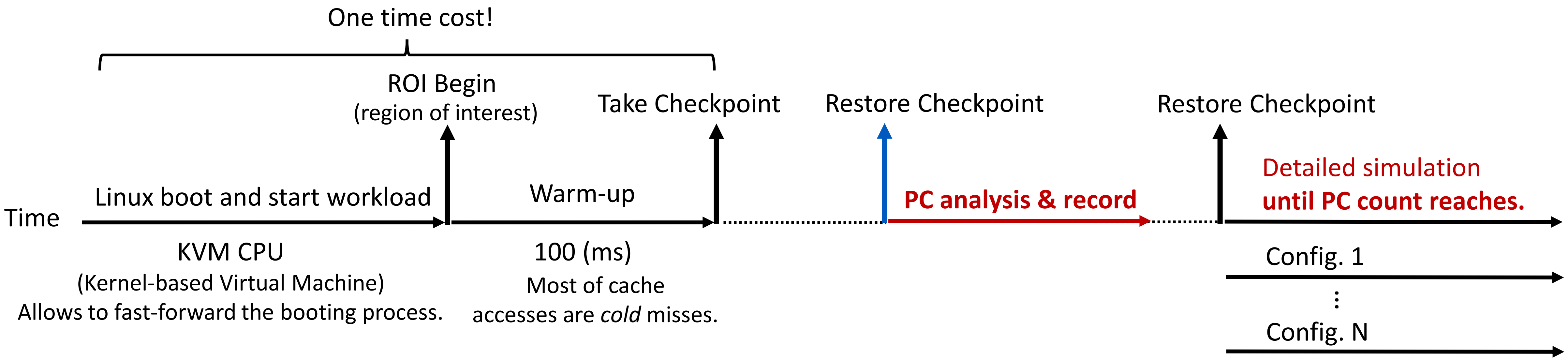
Fixing the Methodology

Then:

Fast forward kernel boot up → Warm-up → Checkpoint → Restore1(sec)

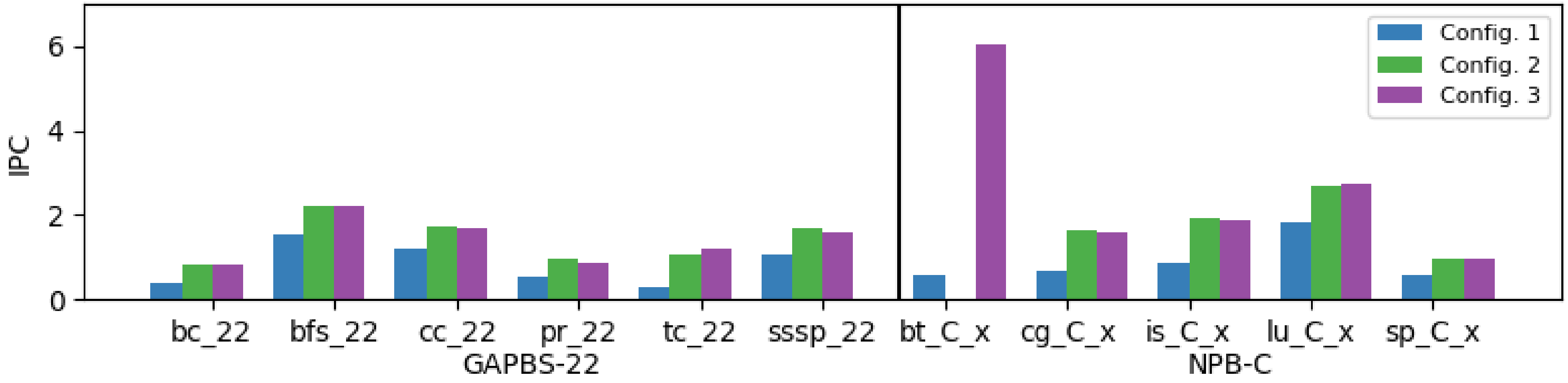
Now:

Fast forward kernel boot up → Warm-up → Checkpoint → *PC-analysis & record* → Restore until PC count reaches

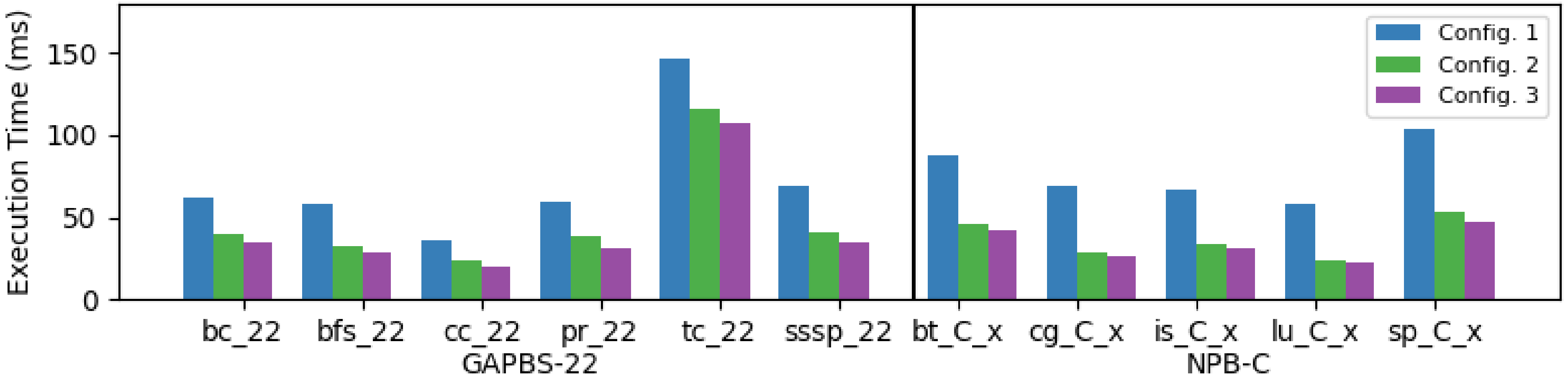


New Results

Before



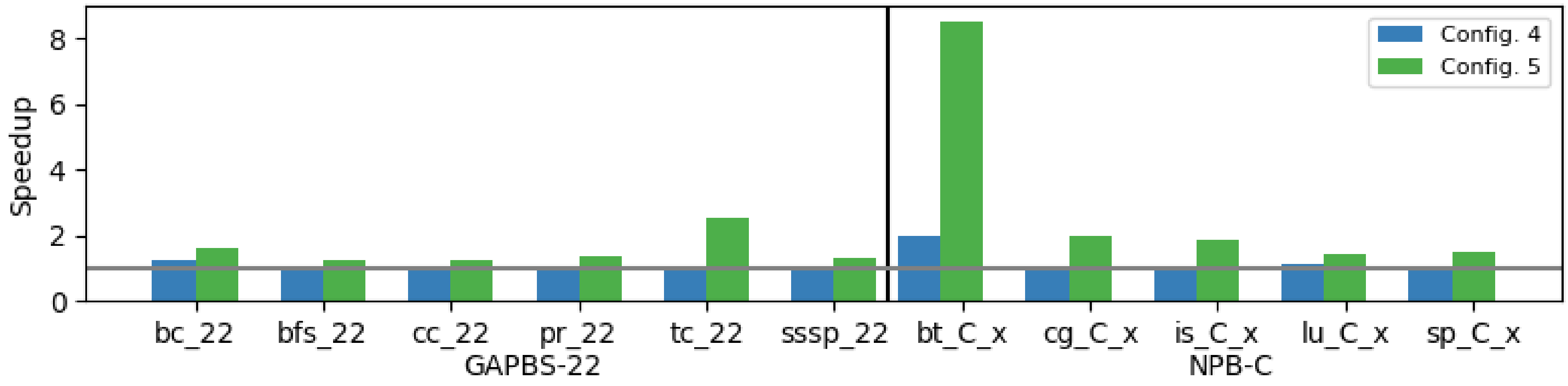
After



UCDAVIS

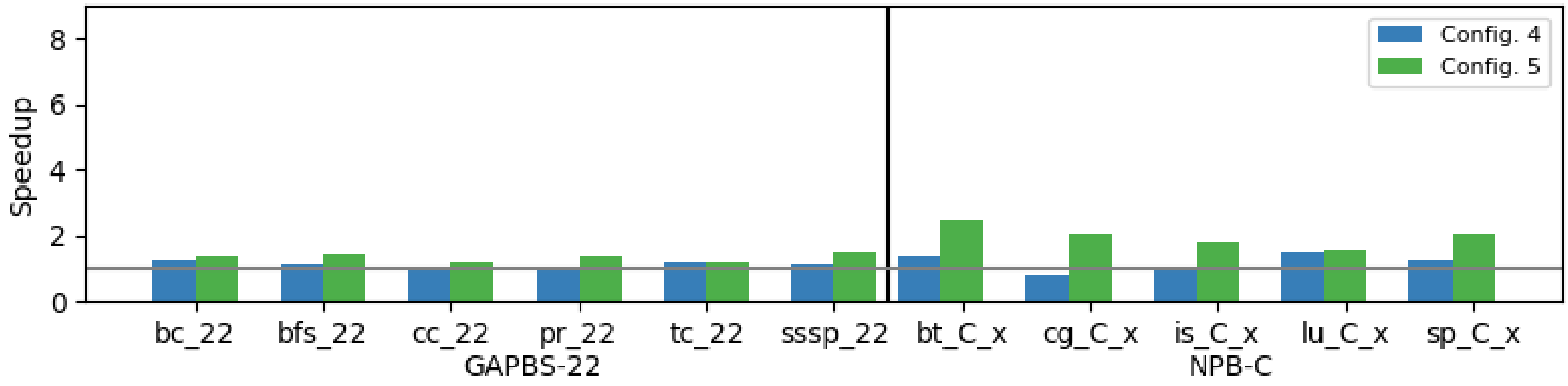
New Results

Before



$$Speedup = \frac{IPC_{\ Config.n}}{IPC_{\ Config.1}}$$

After



$$Speedup = \frac{time_{\ Config.1}}{time_{\ Config.n}}$$

UCDAVIS

Summary

- Properly measuring the amount of work progress in evaluation of large-scale application is vital.

- Using a fixed simulation-time approach can be misleading in these applications.

Fast forward kernel boot up → Warm-up → Checkpoint → Restore1(sec)

- Techniques like LoopPoint help in accurately tracking the amount of work progress for simulation of large-scale applications.

Fast forward kernel boot up → Warm-up → Checkpoint → *PC-analysis & record* → Restore until PC count reaches

UCDAVIS

Thank You!

Q & A

References

[1] Sabu, A., Patil, H., Heirman, W., & Carlson, T. E. (2022, April). LoopPoint: Checkpoint-driven sampled simulation for multi-threaded applications. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 604-618). IEEE.

[2] Jevdjic, D., Loh, G. H., Kaynak, C., & Falsafi, B. (2014, December). Unison cache: A scalable and effective die-stacked DRAM cache. In 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 25-37). IEEE.

[3] Jevdjic, D., Volos, S., & Falsafi, B. (2013). Die-stacked dram caches for servers: Hit ratio, latency, or bandwidth? have it all with footprint cache. ACM SIGARCH Computer Architecture News, 41(3), 404-415.

[4] Alameldeen, A. R., & Wood, D. A. (2006). IPC considered harmful for multiprocessor workloads. IEEE Micro, 26(4), 8-17.