

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis



# Agenda

Time (Eastern)	Speaker	Topic
13.20 to 13.30	Trevor E. Carlson	Overview of the tutorial
13.30 to 14.20	Akanksha Chaudhari	Performance analysis, simulation, sampling
14.20 to 15.20	Harish Patil	Using tools: Pin, PinPlay, SDE, ELFies
15.20 to 15.40		Break
15.40 to 16.20	Alen Sabu	Multi-threaded sampling and LoopPoint
16.20 to 17.00	Changxi Liu	Sniper and LoopPoint demo
17.00 to 17.40	Zhantong Qiu	Using LoopPoint with gem5

# Agenda

Time (Eastern)	Speaker	Topic
13.20 to 13.30	Trevor E. Carlson	Overview of the tutorial
13.30 to 14.20	Akanksha Chaudhari	Performance analysis, simulation, sampling
14.20 to 15.20	Harish Patil	Using tools: Pin, PinPlay, SDE, ELFies
15.20 to 15.40		Break
15.40 to 16.20	Alen Sabu	Multi-threaded sampling and LoopPoint
16.20 to 17.00	Changxi Liu	Sniper and LoopPoint demo
17.00 to 17.40	Zhantong Qiu	Using LoopPoint with gem5

# Performance Analysis, Simulation, Sampling

- Speaker: Akanksha Chaudhari
  - Research Assistant, National University of Singapore
- Topics Covered
  - Architectural exploration and evaluation
  - Simulation as a tool for performance estimation
  - Methods for fast estimation using simulation
  - State-of-the-art single-threaded sampled simulation techniques



# Using Tools: Pin, PinPlay, SDE, ELFies

- Speaker: Harish Patil
  - Principal Engineer, Intel Corporation
- Topics Covered
  - Binary instrumentation using Pin or writing Pintools
  - PinPlay kit and PinPlay-enabled tools
  - SDE build kit for microarchitecture emulation
  - Checkpointing threaded applications using PinPlay, SDE
  - Detailed discussion on ELFies including its generation and usage



# Multi-threaded Sampling and LoopPoint

- Speaker: Alen Sabu
  - PhD Candidate, National University of Singapore
- Topics Covered
  - Sampled simulation of multi-threaded applications
  - Existing methodologies and their drawbacks
  - Detailed discussion on LoopPoint methodology
  - Experimental results of LoopPoint



# Sniper and LoopPoint Demo

- Speaker: Changxi Liu
  - PhD Candidate, National University of Singapore
- Topics Covered
  - Overview of Sniper simulator
  - High-level structure of LoopPoint code
  - Demo on how to use LoopPoint tools
  - Sampling custom workloads using LoopPoint



# Using LoopPoint with gem5

- Speaker: Zhantong Qiu
  - Undergraduate student, University of California, Davis
- Topics Covered
  - Overview of gem5 simulator
  - Structure of LoopPoint integration code
  - Demo on simulating LoopPoint regions on gem5
  - Running ELFies on gem5



# Agenda

Time (Eastern)	Speaker	Topic
13.20 to 13.30	Trevor E. Carlson	Overview of the tutorial
13.30 to 14.20	Akanksha Chaudhari	Performance analysis, simulation, sampling
14.20 to 15.20	Harish Patil	Using tools: Pin, PinPlay, SDE, ELFies
15.20 to 15.40		Break
15.40 to 16.20	Alen Sabu	Multi-threaded sampling and LoopPoint
16.20 to 17.00	Changxi Liu	Sniper and LoopPoint demo
17.00 to 17.40	Zhantong Qiu	Using LoopPoint with gem5

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis

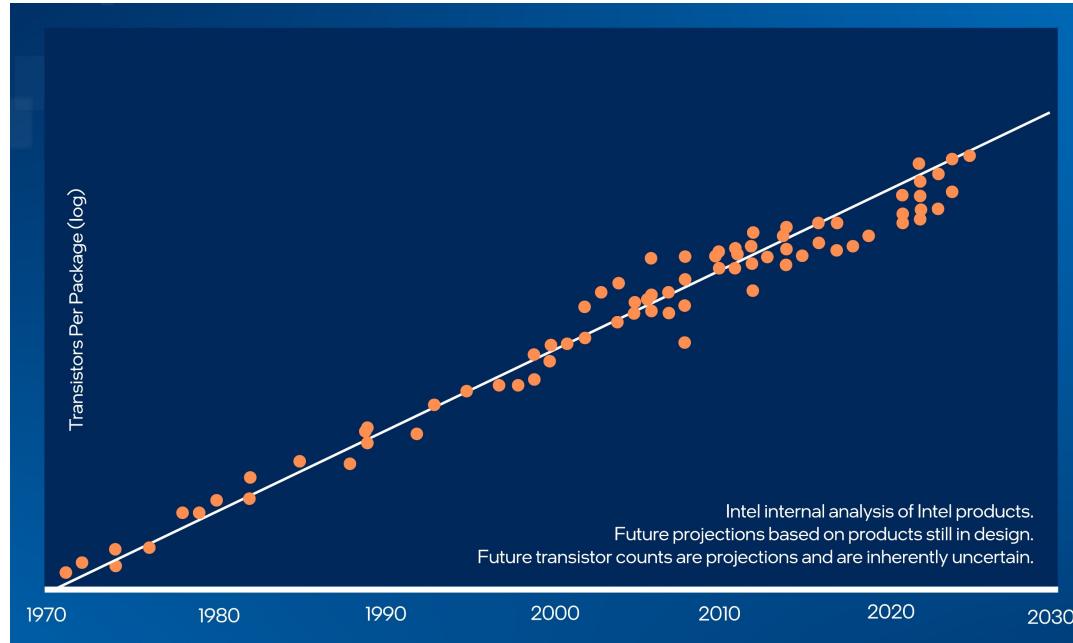


Session 1

# Performance Analysis, Simulation, Sampling

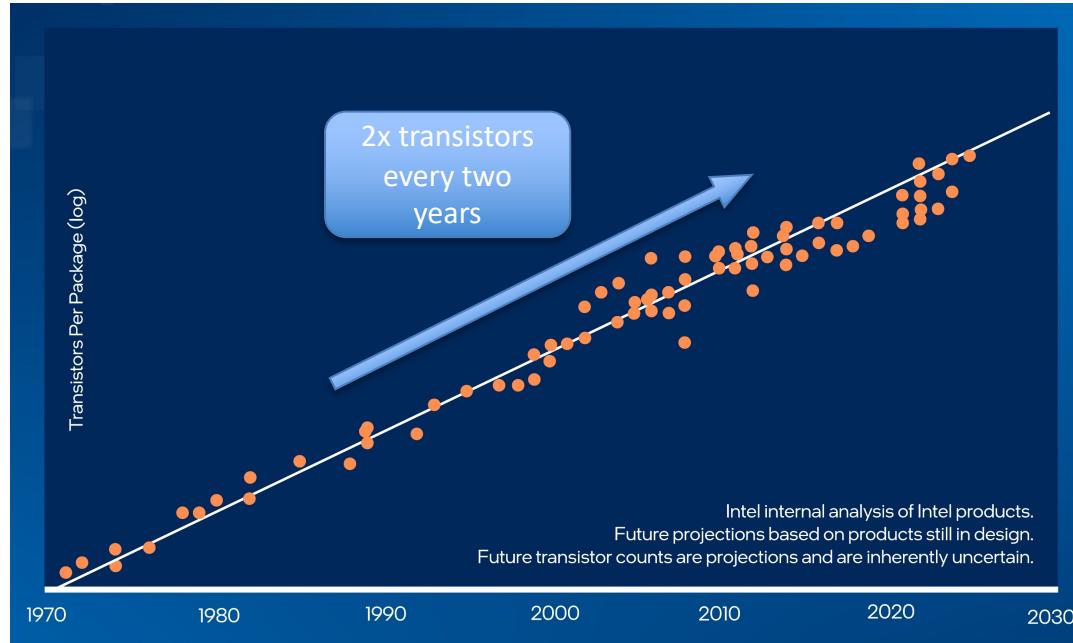
AKANKSHA CHAUDHARI, RESEARCH ASSISTANT  
NATIONAL UNIVERSITY OF SINGAPORE

# Architectural Trends in Processor Design



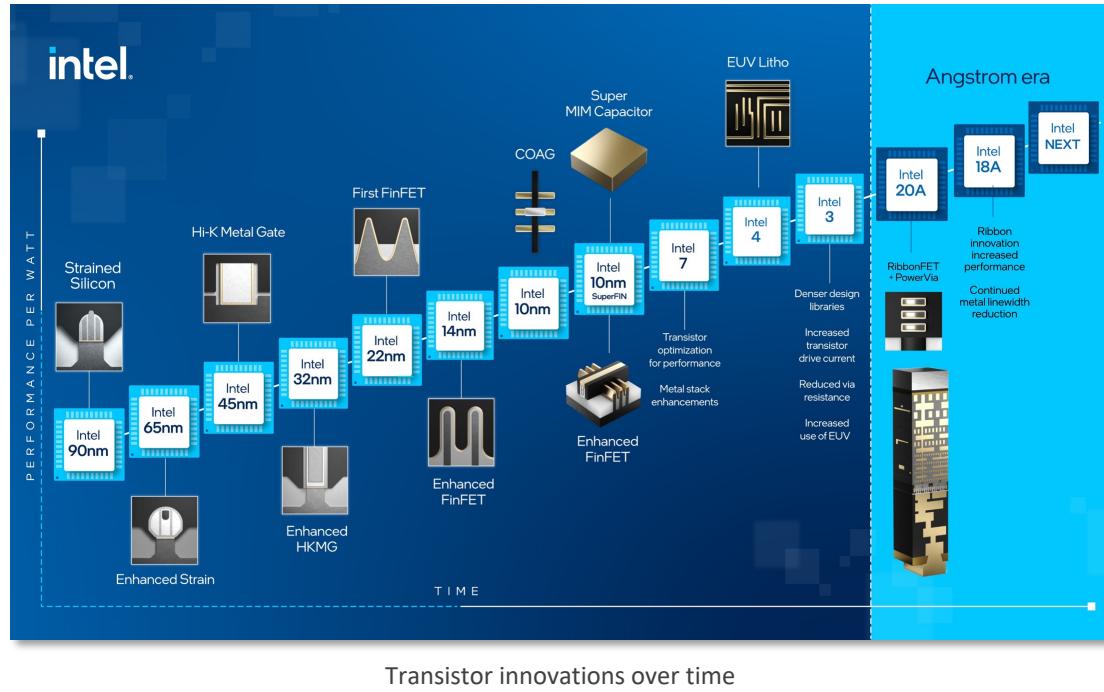
Moore Law number of transistor per device: past, present, future [Intel]

# Architectural Trends in Processor Design

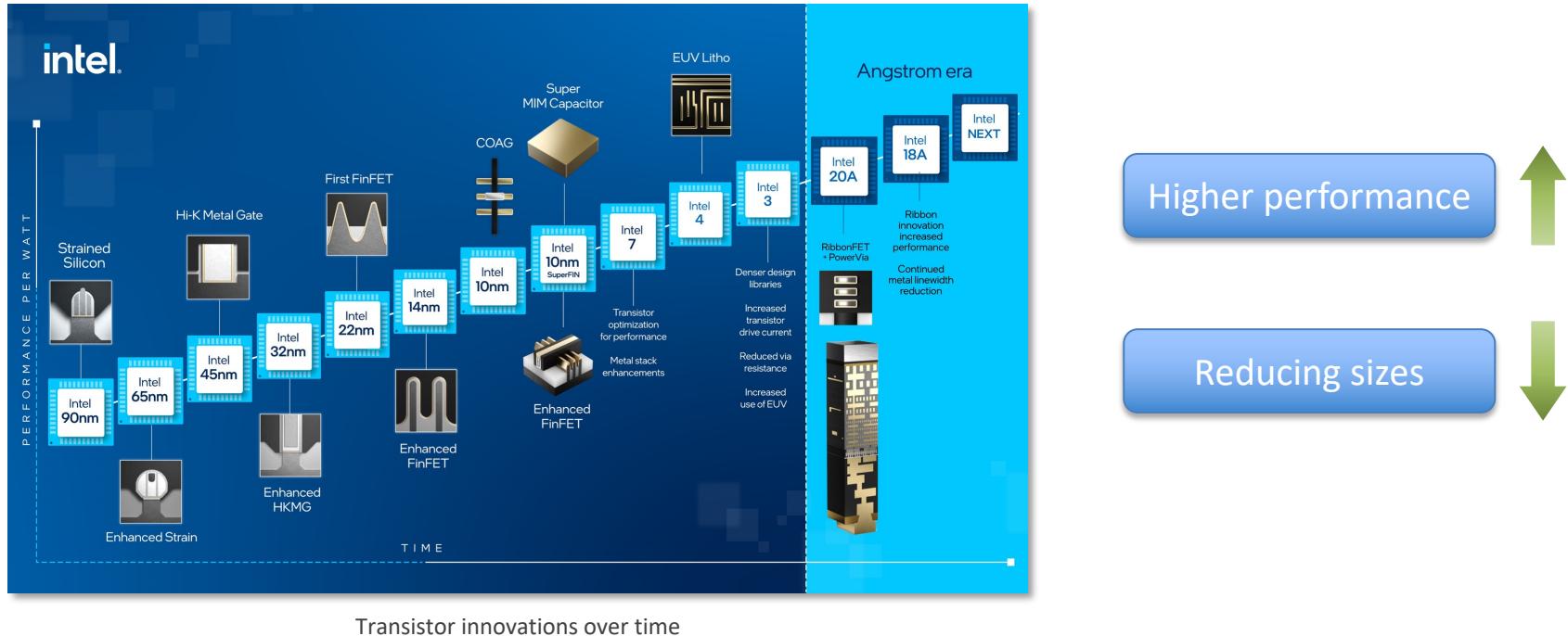


Moore Law number of transistor per device: past, present, future [Intel]

# Architectural Trends in Processor Design



# Architectural Trends in Processor Design

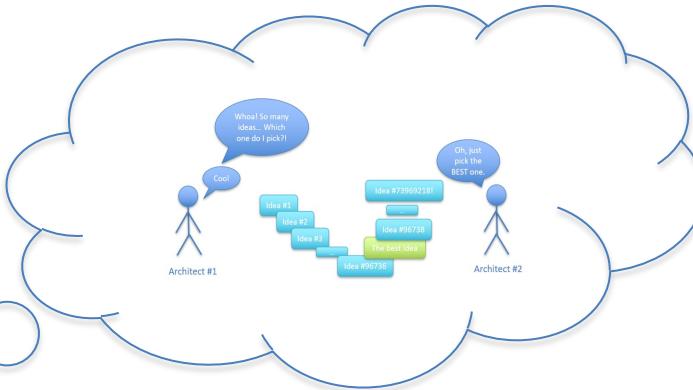
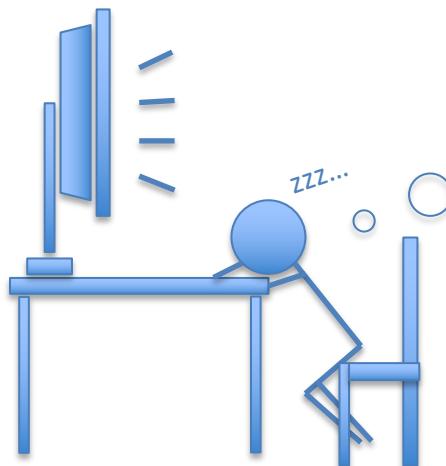


# Exploration and Evaluation of New Ideas



# Exploration and Evaluation of New Ideas

The Architect IRL

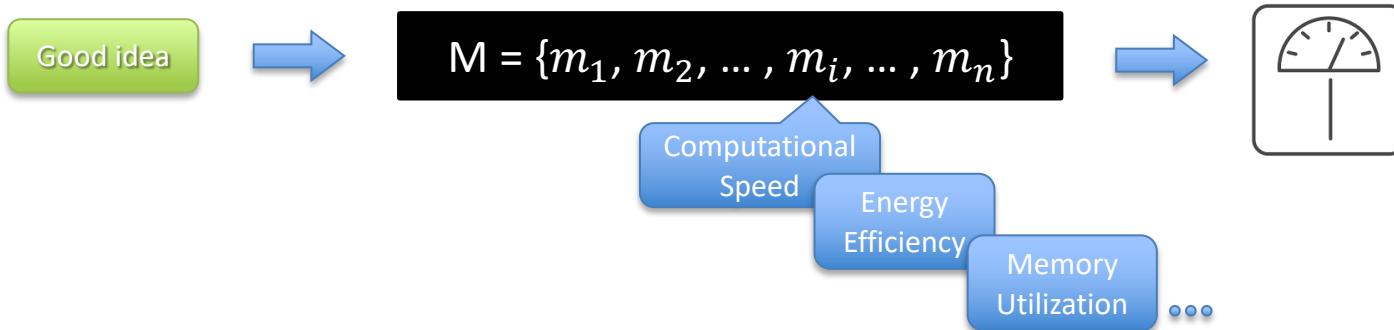


The Important Question:

So how do we then **explore new ideas quickly** and **evaluate them accurately** to find the **BEST** idea?

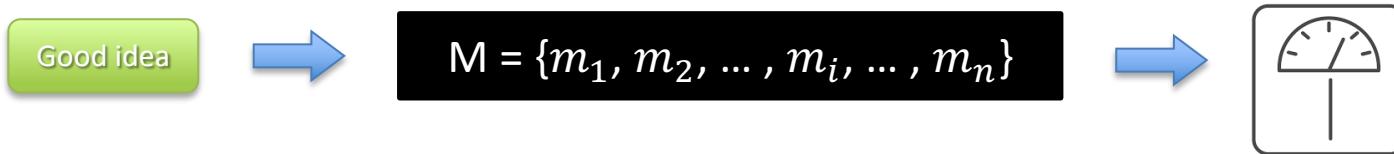
# Exploration and Evaluation of New Ideas

- A “good” idea optimizes a finite set of performance metrics:



# Exploration and Evaluation of New Ideas

- A “good” idea optimizes a finite set of performance metrics:



How to assess whether a given idea  
improves the target metrics?

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modeling
- Simulation
- Prototyping
- Actual implementation

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- **Theoretical proof**
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- **Simulation**
- Prototyping
- Actual implementation

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- **Prototyping**
- Actual implementation

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- **Actual implementation**

# Exploration and Evaluation of New Ideas

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

Fairly complex for  
modern architectures

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

Evaluating  
different workload  
profiles is difficult

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

Evaluating different workload profiles is difficult



Worst-case estimates can be misleading

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

Expensive!

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
  - Analytical modelling
  - Simulation
  - Prototyping
  - Actual implementation
- 
- A red rounded rectangle surrounds the first four items in the list. An arrow points from the word 'Prototyping' to a red callout box containing the text '(Relatively) less expensive'.

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
  - Analytical modelling
  - Simulation
  - Prototyping
  - Actual implementation
- 
- ```
graph LR; A[Prototyping] --> B["(Relatively) less expensive"]; A --> C["BUT limited by the capability of its components"]
```

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

Not feasible for  
exploration of large  
design spaces!

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- Simulation
- Prototyping
- Actual implementation

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
  - Analytical modelling
  - **Simulation**
  - Prototyping
  - Actual implementation
- Allows for varying degrees of abstractions and accuracy

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
  - Analytical modelling
  - **Simulation**
  - Prototyping
  - Actual implementation
- Feasible exploration of  
large design spaces

# An “Evaluation” of the Evaluation Methods

Different evaluation methods:

- Theoretical proof
- Analytical modelling
- **Simulation**
- Prototyping
- Actual implementation



Most feasible way to explore and evaluate a large and complex design space in terms of time, cost and efficiency!



# Simulation: An Overview

- How does simulation work?
  - Mimics the key functional and/or timing behavior of a system to reflect its performance in terms of the target metrics.

# Simulation: An Overview

- How does simulation work?
  - Mimics the key functional and/or timing behavior of a system to reflect its performance in terms of the target metrics.
- How does this help us?
  - Enables fast exploration of design space (to discover the next big idea!).
  - Allows verification, debugging, and optimization of existing systems.
  - Also enables evaluation and understanding of non-existent systems.

# Simulation: An Overview

- How does simulation work?
  - Mimics the key functional and/or timing behavior of a system to reflect its performance in terms of the target metrics.
- How does this help us?
  - Enables fast exploration of design space (to discover the next big idea!).
  - Allows verification, debugging, and optimization of existing systems.
  - Also enables evaluation and understanding of non-existent systems.
- Caution: A simulator is only as good as the person who uses it.

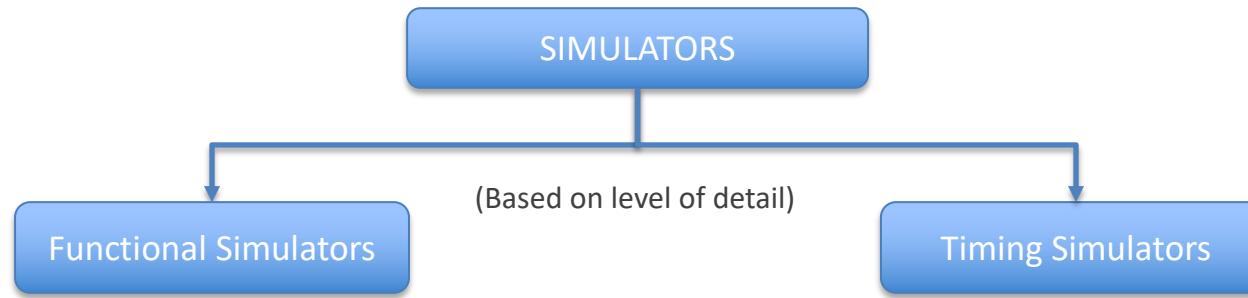
# Simulation: An Overview

- An ideal simulation technique:
  - High speed → For faster exploration.
  - High flexibility → For wider exploration.
  - High accuracy/low simulation error → For accurate evaluation.
- Practical simulation techniques involve trade-offs:
  - Speed vs. accuracy
  - Accuracy vs. flexibility
  - Flexibility vs. speed

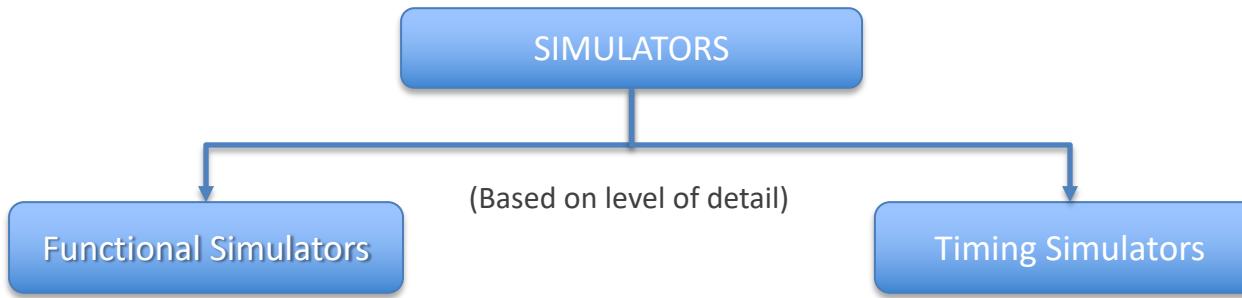
# Simulation: An Overview

- An ideal simulation technique:
  - High speed → For faster exploration.
  - High flexibility → For wider exploration.
  - High accuracy/low simulation error → For accurate evaluation.
- Practical simulation techniques involve trade-offs:
  - Speed vs. accuracy
  - Accuracy vs. flexibility
  - Flexibility vs. speed

# Different Simulation Techniques

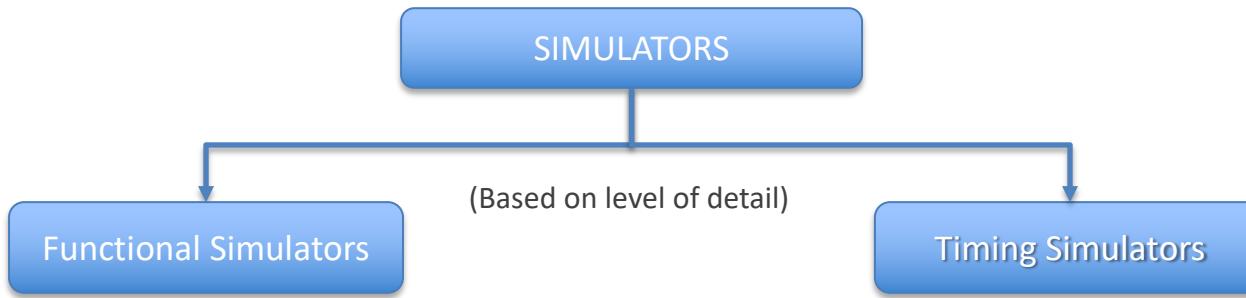


# Different Simulation Techniques



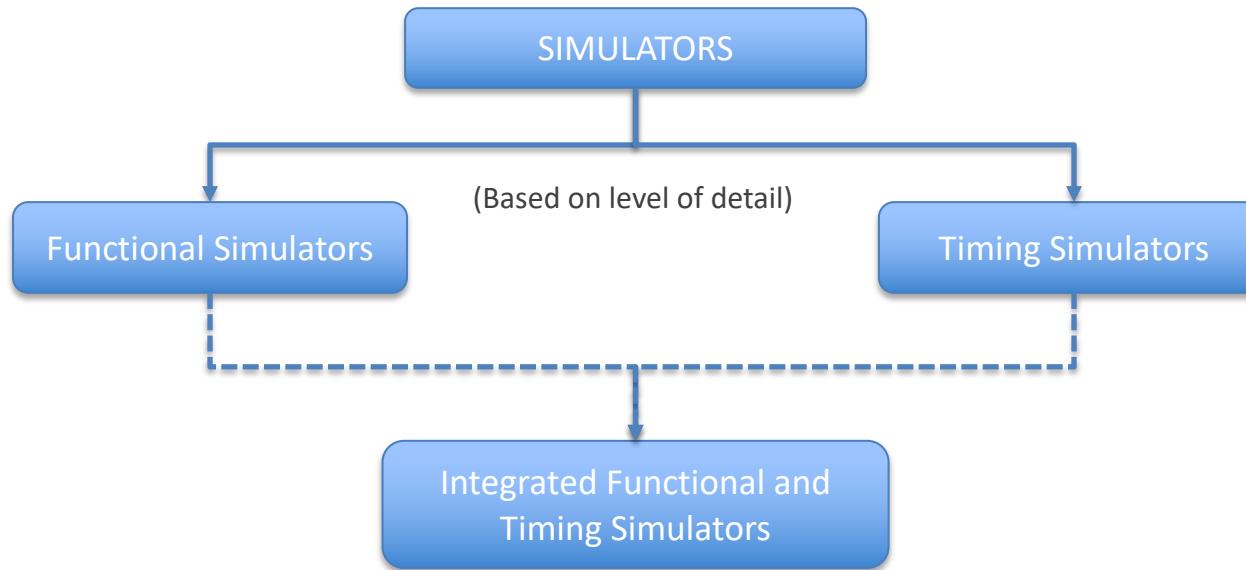
- Implement only architectural details and achieve same functionality as the modeled architecture.
- Tracks architectural stats (memory access locality, instruction count/mix).
- Faster, but cannot track detailed microarchitectural parameters.

# Different Simulation Techniques

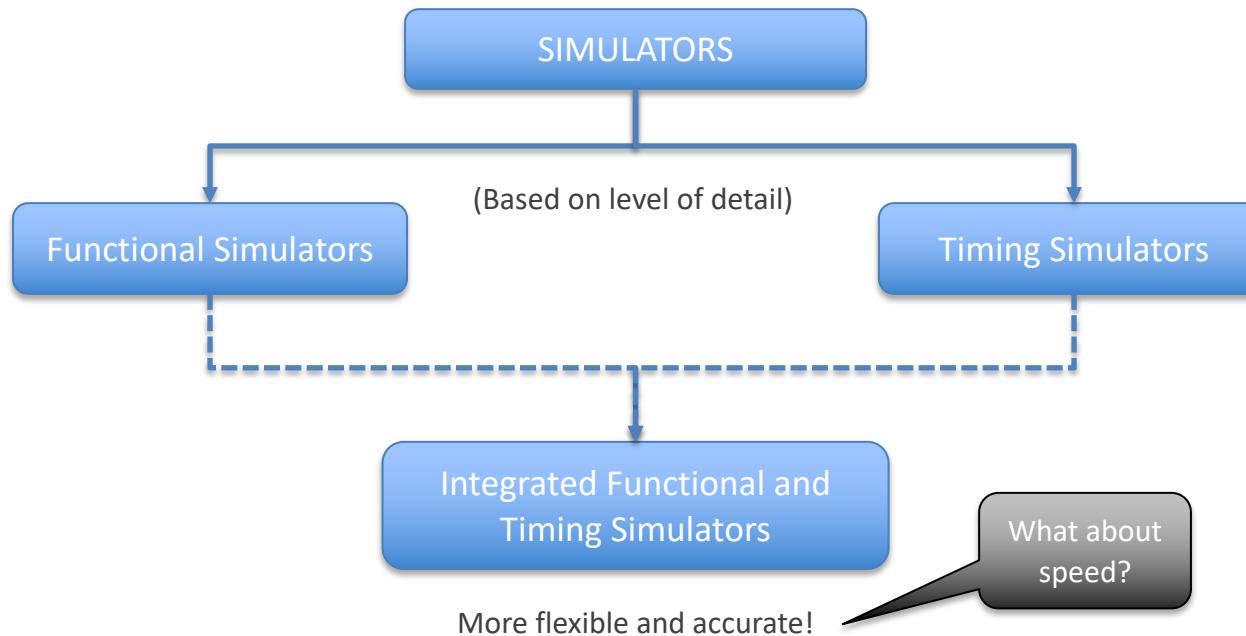


- Implement the microarchitecture.
- Produces detailed microarchitectural stats (IPC, runtime, memory performance).
- Do not have to emulate the functionality of the modeled architecture.

# Different Simulation Techniques



# Different Simulation Techniques



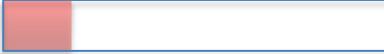
# Techniques to Simulate Faster

- Partial simulation and extrapolation
  - Simulating the first 1 billion instructions in detail.

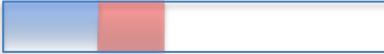


Detailed simulation

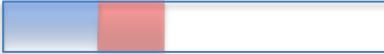
# Techniques to Simulate Faster

- Partial simulation and extrapolation
  - Simulating the first 1 billion instructions in detail.  Detailed simulation
  - Fast-forwarding to skip the initialization phase and then simulating in detail.  Fast-forwarding using Functional simulation

# Techniques to Simulate Faster

- Partial simulation and extrapolation
  - Simulating the first 1 billion instructions in detail. Detailed simulation
  - Fast-forwarding to skip the initialization phase and then simulating in detail. Fast-forwarding using Functional simulation
  - Fast-forwarding, warming up  $\mu$ -architectural state, and then simulating in detail. Warming up the microarchitectural state

# Techniques to Simulate Faster

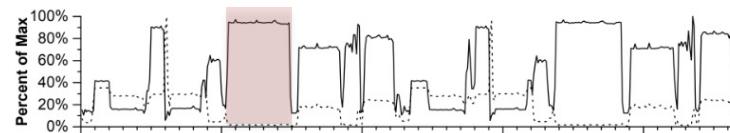
- Partial simulation and extrapolation
  - Simulating the first 1 billion instructions in detail.
  - Fast-forwarding to skip the initialization phase and then simulating in detail.
  - Fast-forwarding, warming up  $\mu$ -architectural state, and then simulating in detail.
- Workload reduction: simulating for reduced input sets or loop counts.

# Techniques to Simulate Faster

- Partial simulation and extrapolation
  - Simulating the first 1 billion instructions in detail.  Detailed simulation
  - Fast-forwarding to skip the initialization phase and then simulating in detail.  Fast-forwarding using Functional simulation
  - Fast-forwarding, warming up  $\mu$ -architectural state, and then simulating in detail.  Warming up the microarchitectural state
- Workload reduction: simulating for reduced input sets or loop counts.

# Techniques to Simulate Faster

- Problems with these techniques:
  - Partial simulation and extrapolation
    - Fails to capture global variations in program behavior and performance.



- Workload reduction
  - Benchmark behavior may vary significantly across different input sizes.
  - Simulation with reduced input sets or loop counts does not reflect the actual performance.

# Sampled Simulation to the Rescue!

- Sampling enables the simulation of selective **representative** regions.
  - Subset of regions within a program execution that represent the behavior of the entire application when extrapolated.
- Selecting representative regions
  - Targeted sampling (like in SimPoint)  

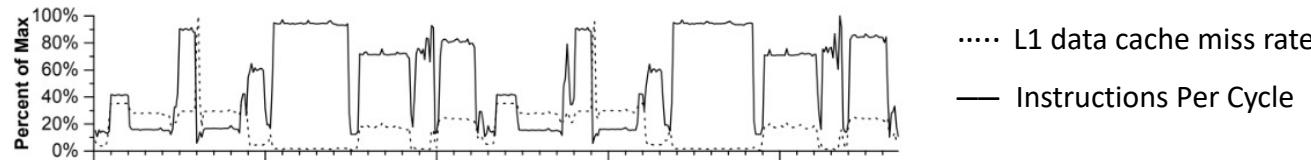
  - Statistical sampling (like in SMARTS)  


 (Full) program execution

 Representative regions

# Sampled Simulation Techniques: SimPoint

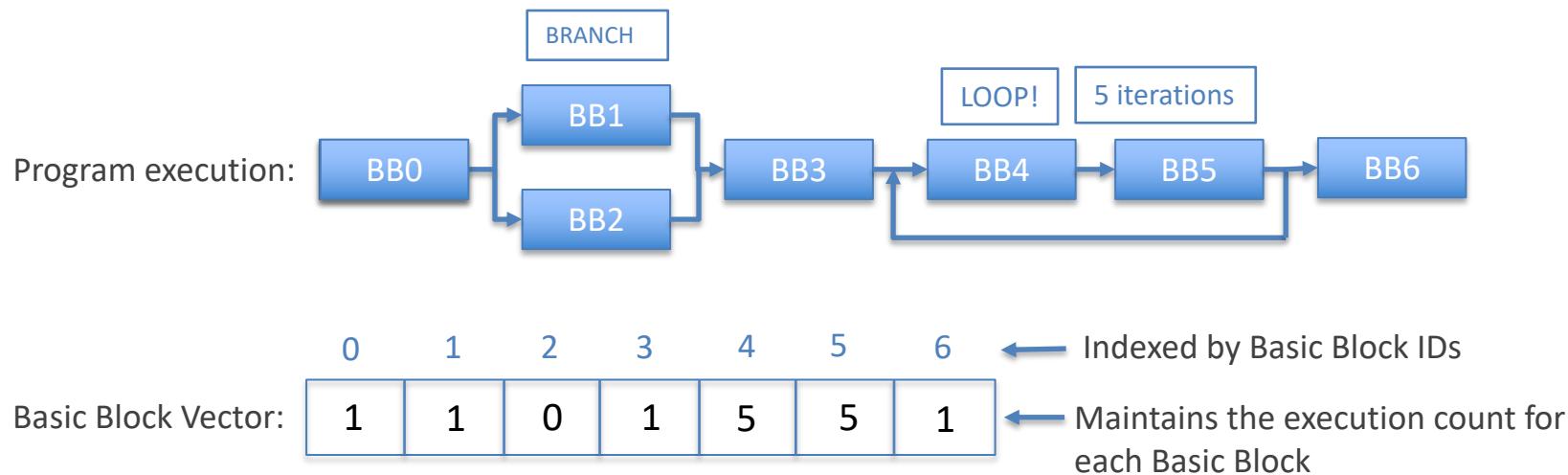
- Large-scale program behaviors vary significantly over their run times.



- Main goal: To automatically and efficiently analyze program behavior over the different phases of execution.
- SimPoint uses Basic Block Vectors (BBV) as a hardware-independent metric for characterizing the program behavior in different phases.

# Sampled Simulation Techniques: SimPoint

Basic Block Vector (BBV) is a single-dimensional array that maintains a count of how many times each basic block was executed in each interval



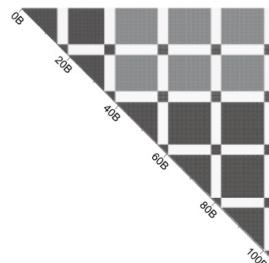
# Sampled Simulation Techniques: SimPoint

- Basic Block Similarity: Measured using Euclidean or Manhattan Distances.

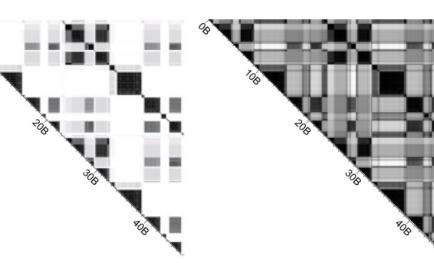
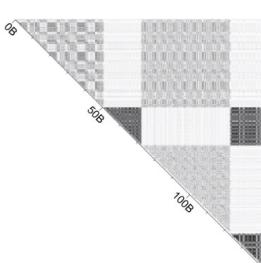
$$EuclideanDist(a, b) = \sqrt{\sum_{i=1}^D (a_i - b_i)^2}$$

$$ManhattanDist(a, b) = \sum_{i=1}^D |a_i - b_i|$$

- Depicted by Basic Block Similarity Matrices.



Using Manhattan distances



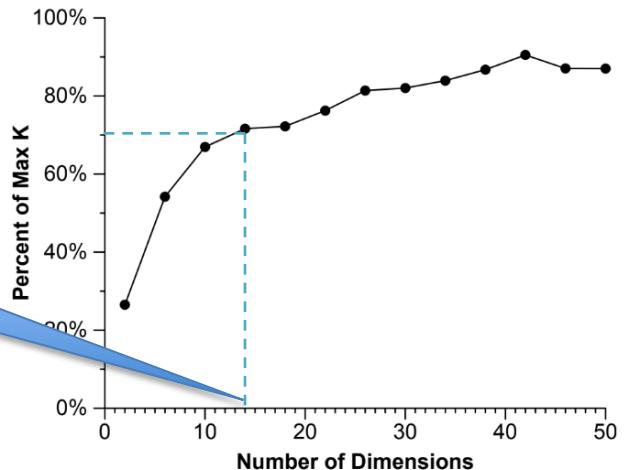
Using Euclidian distances

- Diagonal of the matrix → program execution
- Point (x, y) gives similarity index
- ↑ darkness → ↑ similarity

# Sampled Simulation Techniques: SimPoint

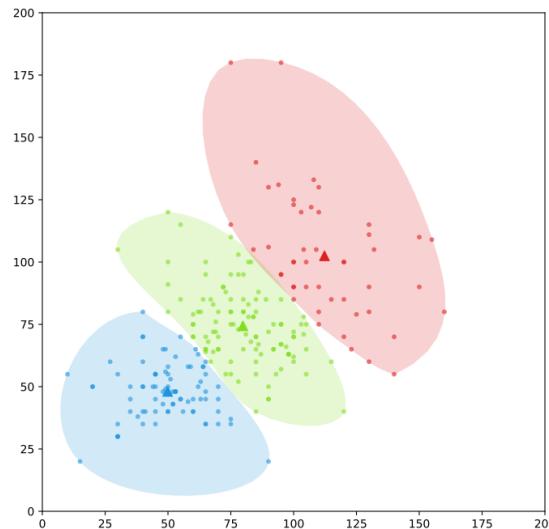
- The BBVs obtained from the profiling step have a very large number of dimensions!
- “Curse of dimensionality”:
  - Hard to cluster the data.
  - Clustering time increases significantly.

Solution: Reduce number of dimensions  
using Random Linear Projections



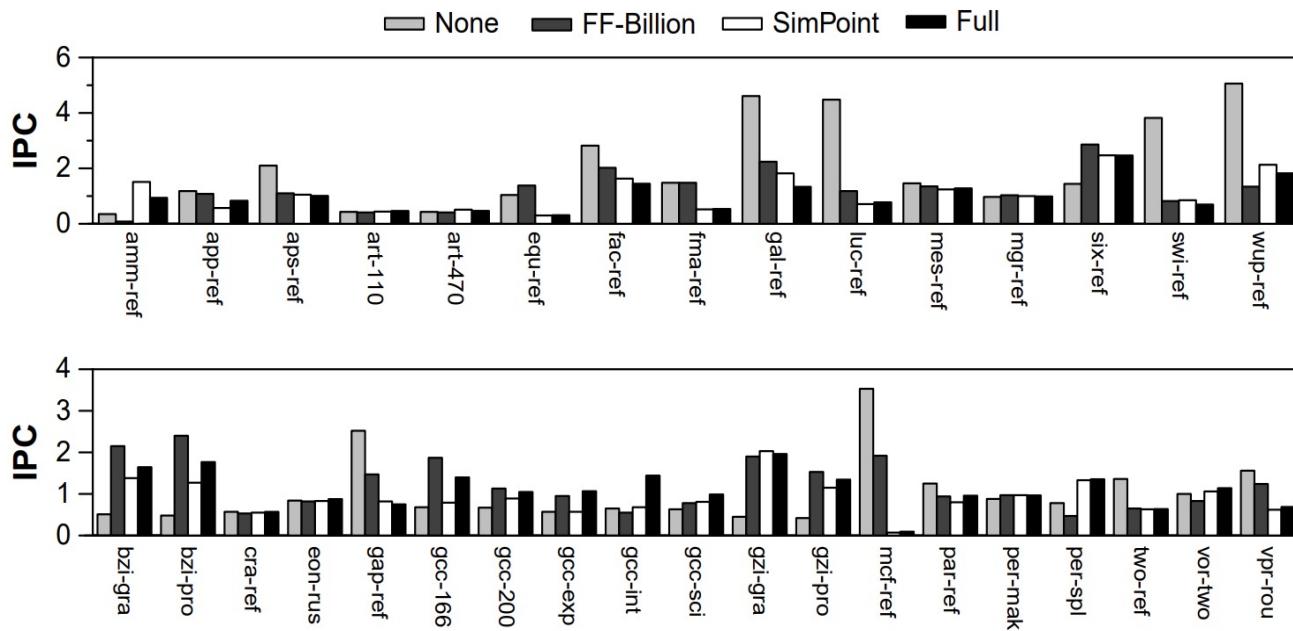
# Sampled Simulation Techniques: SimPoint

K-means clustering:



- Representative region → single simulation point
  - BBV with the lowest distance from the centroid of all cluster centers.
- Representative regions → multiple simulation points
  - For each cluster, choose the BBV that is closest to the centroid of the cluster.

# Sampled Simulation Techniques: SimPoint

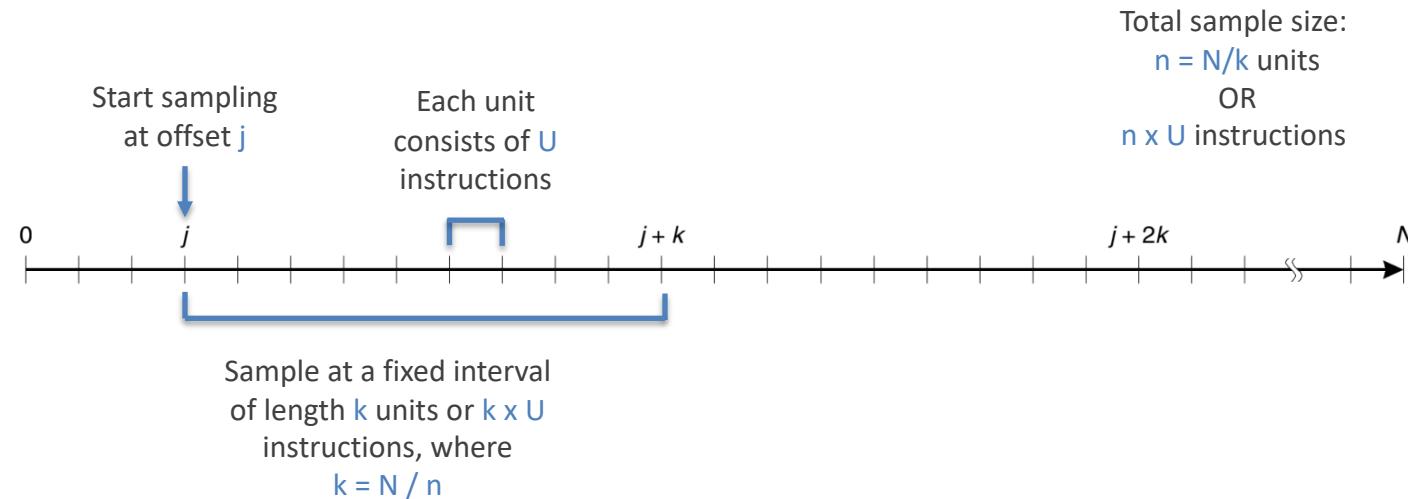


# Sampled Simulation Techniques: SMARTS

- Main idea behind SMARTS:
  - Using systematic statistical sampling:
    - To identify a minimal representative sample from the population for simulation.
    - To establish a confidence level for the error on sample estimates.
  - Simulating using two modes :
    - Detailed simulation of sampled instructions.
    - Functional simulation of remaining instructions.

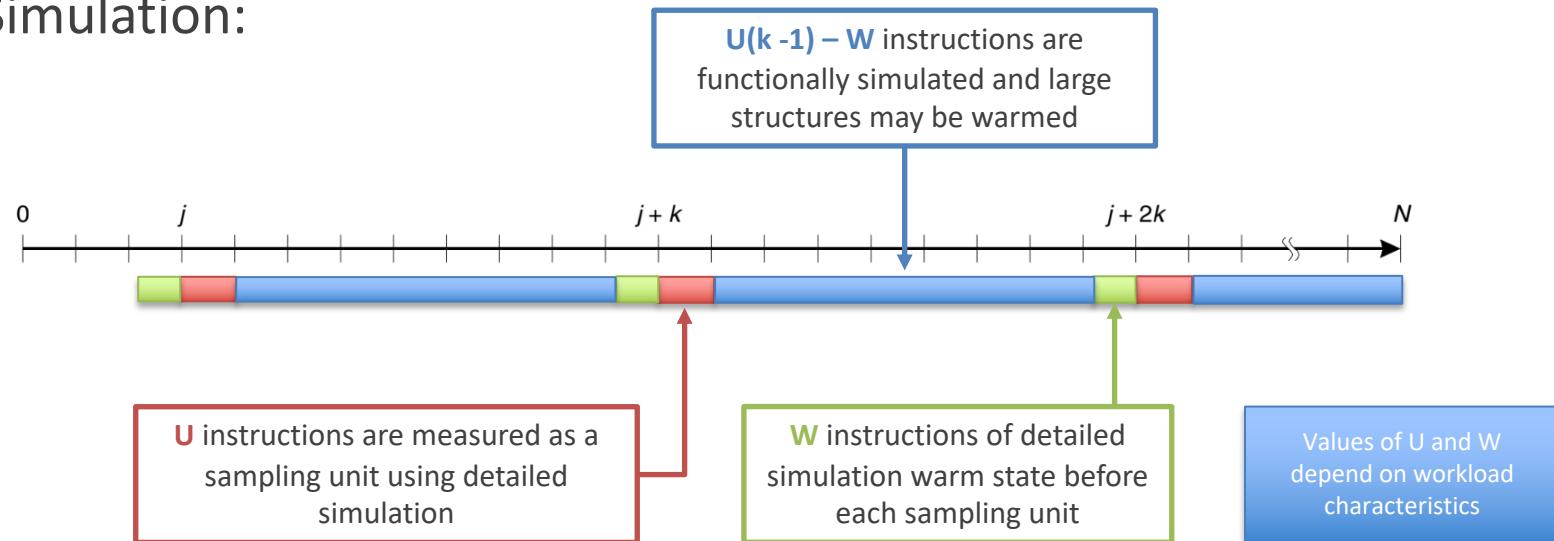
# Sampled Simulation Techniques: SMARTS

- SMARTS uses Systematic Sampling:



# Sampled Simulation Techniques: SMARTS

- Simulation:



# Sampled Simulation Techniques: SMARTS

- Evaluation results:
    - Average error:
      - 0.64% for CPI
      - 0.59% for EPI
    - Speedup over full-stream simulation:
      - 35x for 8-way out-of-order processors
      - 60x for 16-way out-of-order processors
- } By simulating fewer than 50 million instructions in detail per benchmark.

# Agenda

| Time (Eastern) | Speaker            | Topic                                      |
|----------------|--------------------|--------------------------------------------|
| 13.20 to 13.30 | Trevor E. Carlson  | Overview of the tutorial                   |
| 13.30 to 14.20 | Akanksha Chaudhari | Performance analysis, simulation, sampling |
| 14.20 to 15.20 | Harish Patil       | Using tools: Pin, PinPlay, SDE, ELFies     |
| 15.20 to 15.40 |                    | Break                                      |
| 15.40 to 16.20 | Alen Sabu          | Multi-threaded sampling and LoopPoint      |
| 16.20 to 17.00 | Changxi Liu        | Sniper and LoopPoint demo                  |
| 17.00 to 17.40 | Zhantong Qiu       | Using LoopPoint with gem5                  |

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis

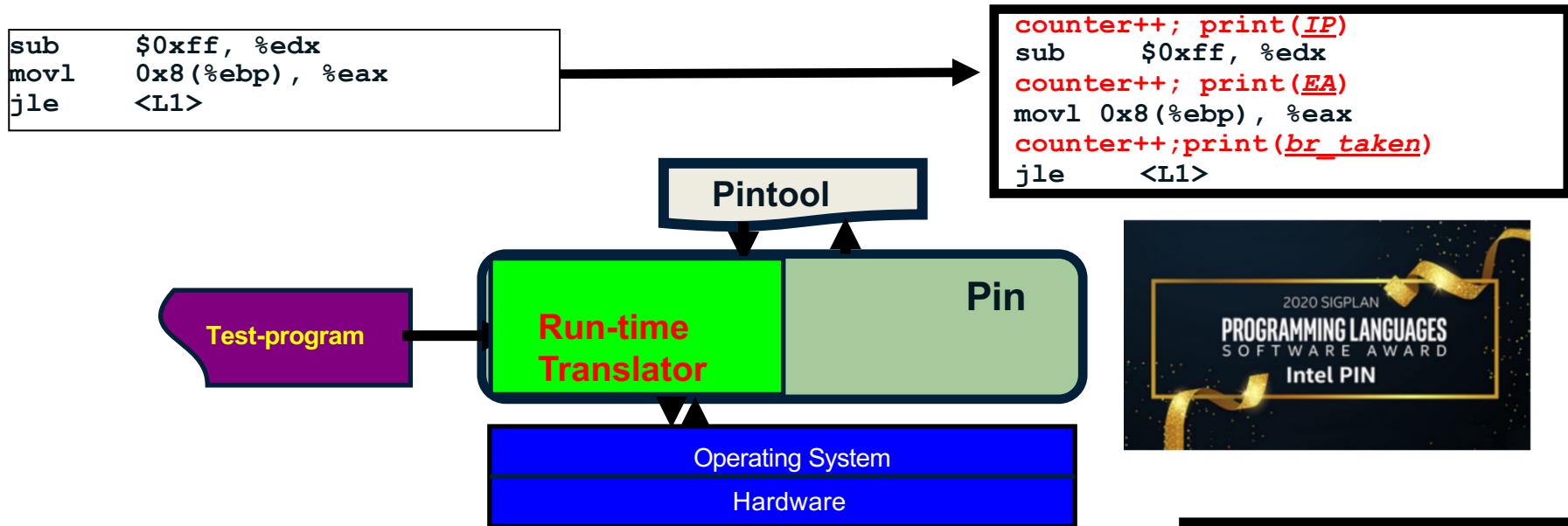


## Session 2

# Using Tools: Pin, PinPlay, SDE, ELFies

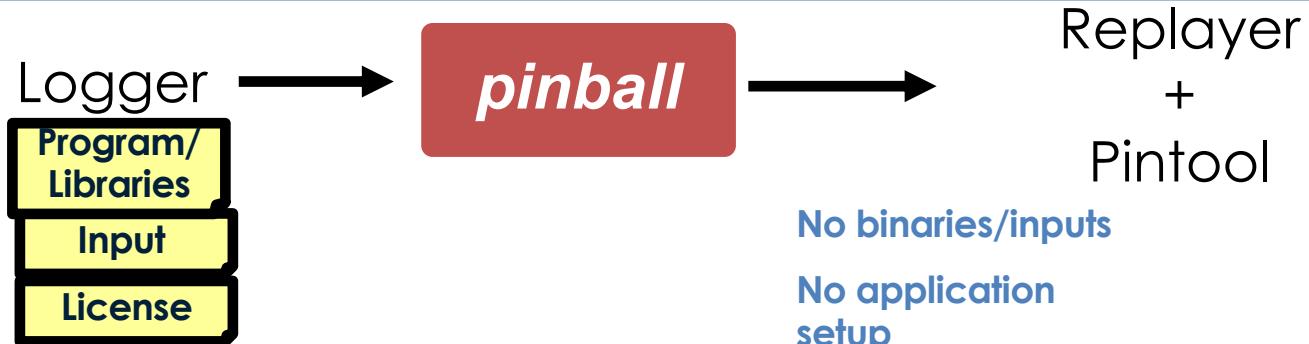
HARISH PATIL, PRINCIPAL ENGINEER (DEVELOPMENT TOOLS SOFTWARE)  
INTEL CORPORATION

# Pin: A Tool for Writing Program Analysis Tools



Normal output +  
*Analysis output*

# *PinPlay*: Software-based User-level Capture and Replay



**Platforms** : Linux, Windows, MacOS

No binaries/inputs

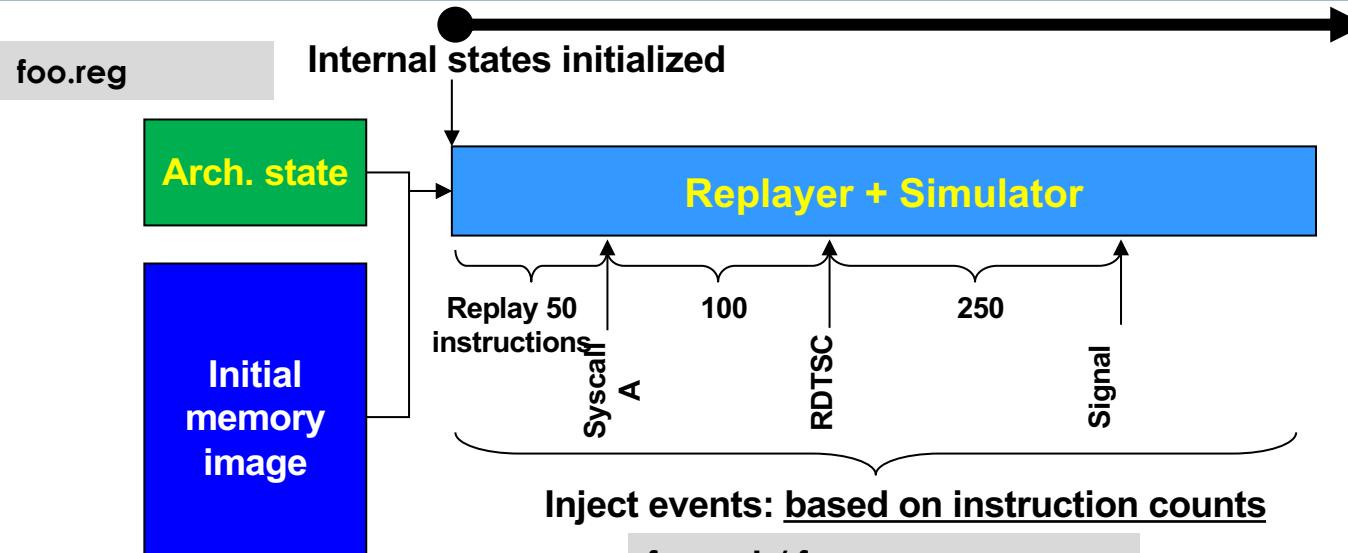
**Upside** : It works! Large OpenMP / MPI programs, Oracle

No application setup

**Downside** : High run-time overhead: ~100-200X for capture →  
Cannot be turned on all the time

No license  
checking

# *Pinball (single-threaded):* Initial memory/register + injections



**foo.text**

- **System calls** : skipped by injecting next rip/ memory changed
- **CPUID, RDTSC** : affected registers injected
- **Signals/Callbacks** : New register state injected

# *Pinball (multi-threaded):* Pinball (single-threaded) + Thread-dependencies

foo.reg (per-thread)



foo.text

Application Memory (common)

foo.reg (per-thread)

Event injection works only if same behavior  
(same instruction counts) is guaranteed  
during replay

foo.sel (per-thread)

[T1] 2 T2 2  
[T1] 3 T2 3

[T2] 5 T4 1

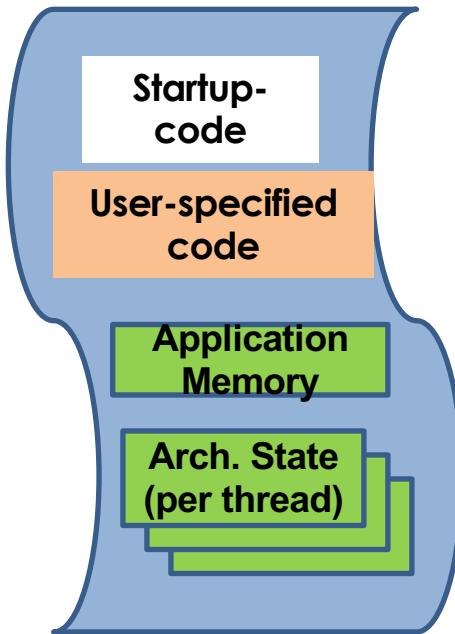
Thread T2 cannot execute instruction  
5 until T4 executes instruction 1

foo.race (per-thread)

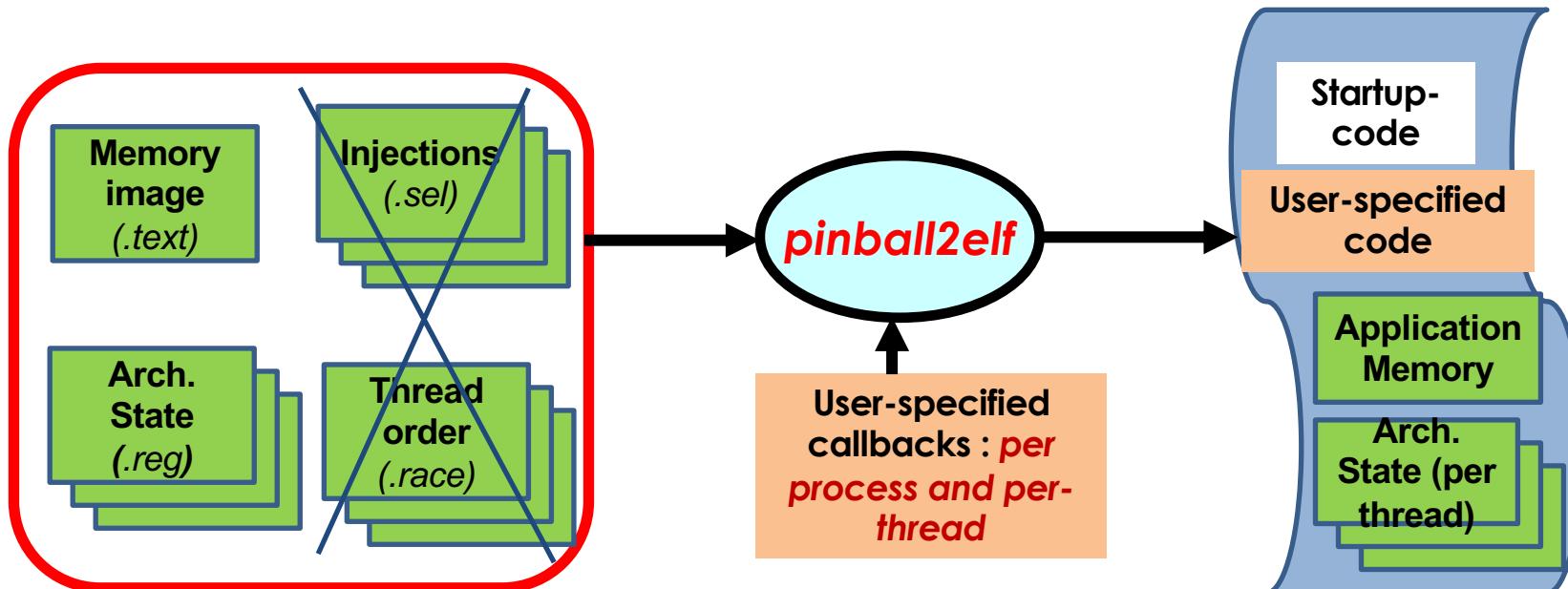
Thread T1 cannot execute instruction 2  
until T2 executes instruction 2

# *ELFie : An Executable Application Checkpoint*

- **Checkpoint**: Memory + Registers
- **Application** : Only program state captured -- no OS or simulator states
- **Executable** : In the **Executable Linkage Format** commonly used on Linux



# *pinball2elf*: Pinball converter to ELF



# Getting started with *pinball2elf*

**Prerequisite:** '*perf*' installed on your Linux box (*perf stat /bin/ls* should work)

- Clone pinball2elf repository: *git clone https://github.com/intel/pinball2elf.git*
- *cd pinball2elf/src*
- *make all*
- *cd .../examples/ST*
- *./testST.sh*

Running ..../scripts//pinball2elf.basic.sh pinball.st/log\_0

..

Running ..../scripts//pinball2elf.perf.sh pinball.st/log\_0 st  
export ELFIE\_PERFLIST=0:0,0:1,1:1

...

hw\_cpu\_cycles:47272 hw\_instructions:4951 sw\_task\_clock:224943

*Tested : Ubuntu 20.04.4 LTS : gcc/g++ 7.5.0 and 9.4.0  
and Ubuntu 18.04.6 LTS: gcc/g++ 7.5.0*

# *ELFie types: basic, sim, perf*

|                            | <b><i>basic</i></b>                               | <b><i>sim</i></b>                                                                              | <b><i>perf</i></b>                                                                                                                                              |
|----------------------------|---------------------------------------------------|------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| How to create              | <code>scripts/pinball2elf.basic.sh pinball</code> | <code>scripts/pinball2elf.sim.sh pinball</code>                                                | <code>scripts/pinball2elf.perf.sh pinball perf.out</code>                                                                                                       |
| Exits gracefully?          | NO, either hangs or dumps core                    | NO, either hangs or dumps core<br>Simulator handles exit                                       | YES, when retired instruction count reaches pinball icount                                                                                                      |
| Environment variables used | NONE                                              | ELFIE_VERBOSE=0/1<br>ELFIE_COREBASE=X<br>Set affinity : thread 0 → core X, thread 1 → core x+1 | "ELFIE_WARMUP" to decide whether to use warmup<br>"ELFIE_PCCONT" to decide how to end warmup/simulation regions<br>ELFIE_PERFLIST, enables performance counting |

# Example: *ELFIE\_PERFLIST* with a *perf ELFie*

*ELFIE\_PERFLIST*, enables performance counting

( based on /usr/include/linux/perf\_event.h  
perf type: 0 --> HW 1 --> SW  
HW counter: 0 --> PERF\_COUNT\_HW\_CPU\_CYCLES  
HW counter: 1 --> PERF\_COUNT\_HW\_CPU\_INSTRUCTIONS  
SW counter: 0 --> PERF\_COUNT\_SW\_CPU\_CLOCK  
... <see perf\_event.h:'enum perf\_hw\_ids' and 'enum

perf\_sw\_ids')

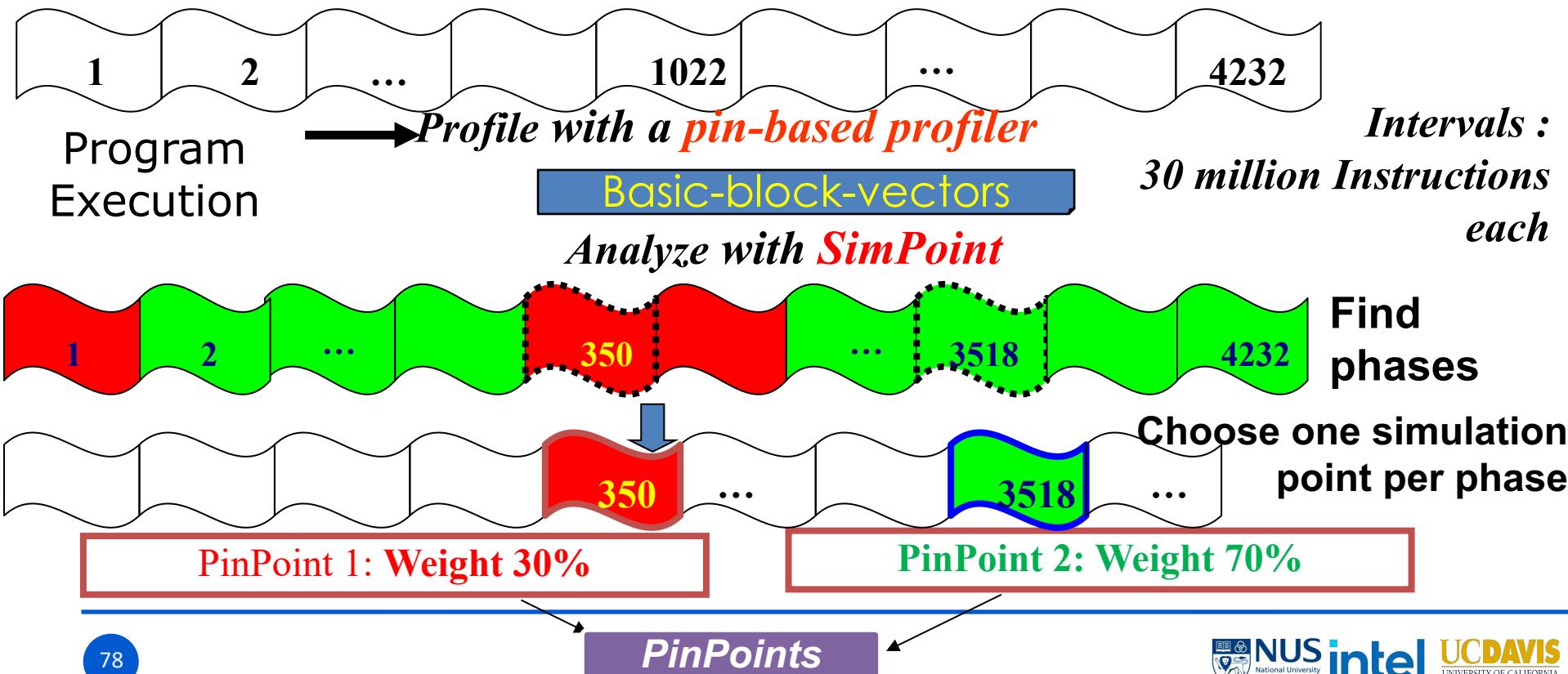
```
% cd examples/MT  
% ../../scripts/pinball2elf.perf.sh pinball.mt/log_0 perf.out  
% setenv ELFIE_PERFLIST "0:0,0:1,1:1"
```

```
% pinball.mt/log_0.perf.elfie
```

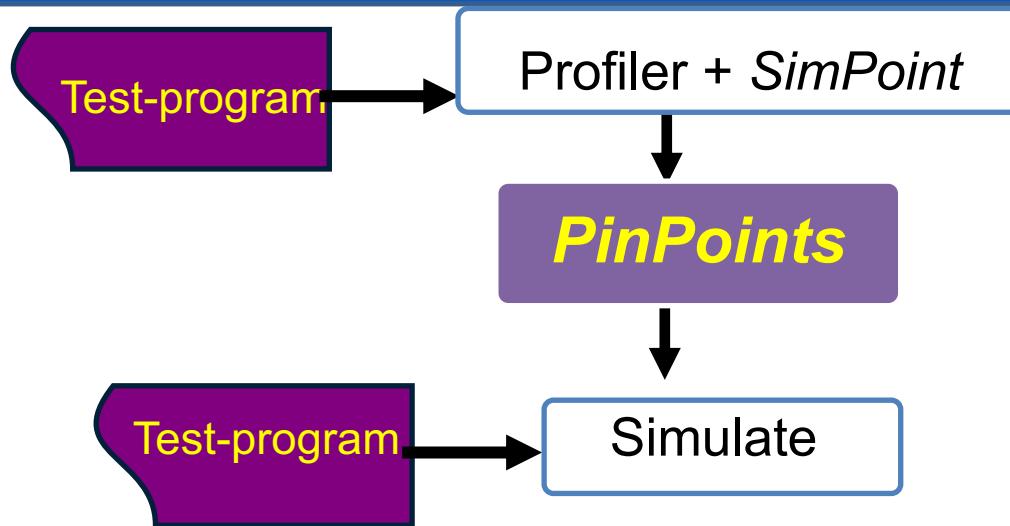
```
└── perf.out.0.perf.txt  
└── perf.out.1.perf.txt  
└── perf.out.2.perf.txt
```

```
ROI start: TSC 48051110586217756  
Thread start: TSC 48051110623843452  
-----  
Simulation end: TSC 48051110625045322  
Sim-end-i-count 3436  
hw_cpu_cycles:36148 hw_instructions:3476  
sw_task_clock:141901  
-----  
Thread end: TSC 48051110625366502  
ROI end: TSC 48051110625959364  
hw_cpu_cycles:40097 hw_instructions:4455  
sw_task_clock:188637
```

# *PinPoints* == *Pin + SimPoint*



# *PinPoints* : The repeatability challenge

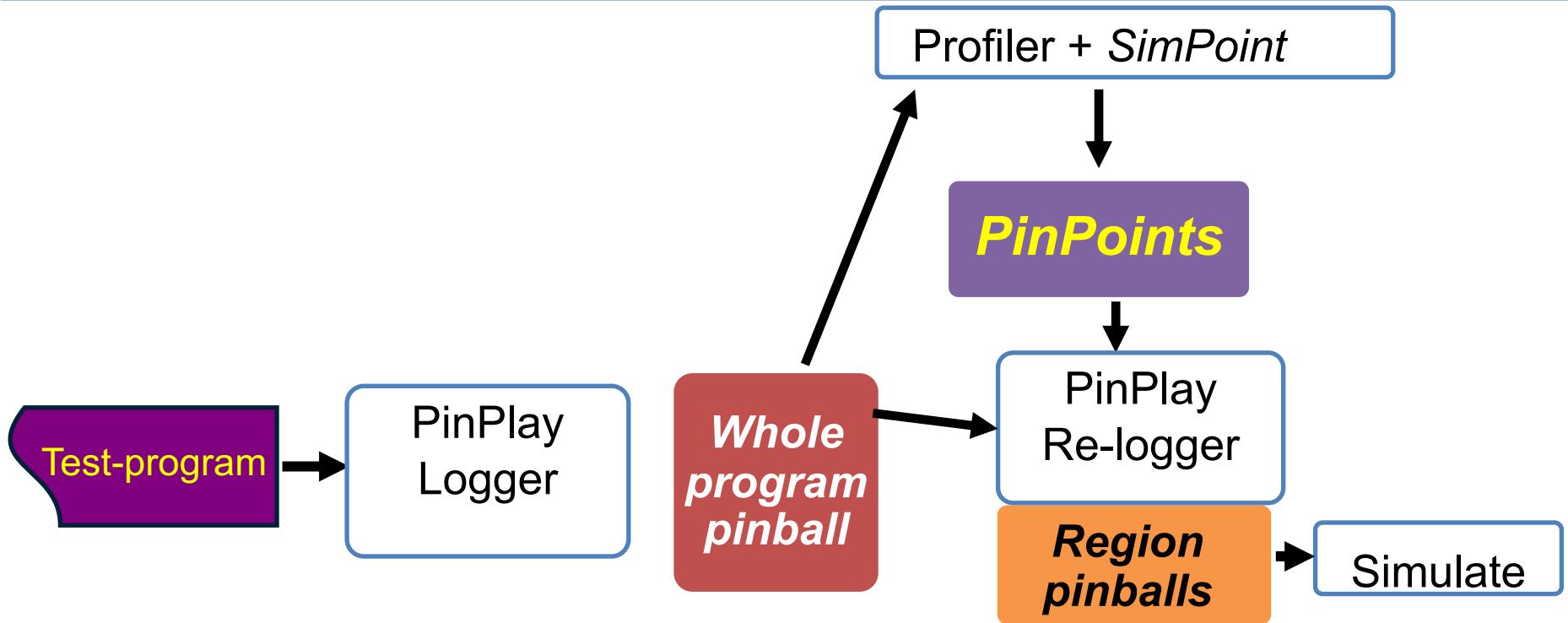


**Problem:** Two runs are not exactly same → PinPoints missed (PC marker based)

[ “*PinPoints out of order*” “*PinPoint End seen before Start*” ]

Found this for 25/54 SPEC2006 runs!

# *PinPlay* provides repeatability



# Single-threaded *PinPoints* → SPEC2006/2017 pinballs publicly available

1. University of California (San Diego), Intel Corporation, and Ghent University  
<https://www.spec.org/cpu2006/research/simpoint.html>
2. University of Texas at Austin  
<https://www.spec.org/cpu2017/research/simpoint.html>
3. Northwestern University  
[Public Release and Validation of SPEC CPU2017 PinPoints](#)

# Simulation of multi-threaded Programs: The non-determinism challenge

- Runs across different configurations are non-deterministic [Alameldeen'03]
  - Locks are acquired in different order
  - Unprotected shared-memory accesses
- One can't compare two runs/simulations of the same benchmark directly
  - Change in micro-architecture present/simulated or execution path taken?*

1. Alameldeen'03 [Variability in Architectural Simulations of Multi-threaded Workloads](#) (HPCA2003)

# Dealing with non-determinism

1. Run multiple simulations for each studied configuration [Alameldeen'03]
  - Needs random perturbation for each run
  - Average behavior per configuration
  - Cost: multiple runs
- 2. Force deterministic behavior so that one run in each configuration is performed [Pereira'08 @ Intel ]
  - Same execution paths
  - Cost: loss in fidelity, thread behavior tied to tracing machine
- 3. Simulate the same “amount of work” [Alameldeen'06] : *LoopPoint* approach
  - A. Pereira'08: [Reproducible Simulation of Multi-Threaded Workloads for Architecture Design Exploration, International Symposium on Workload Characterization \(IISWC'08\)](#)
  - B. Alameldeen'06 [IPC Considered Harmful for Multi-processors Workloads \(IEEE-Micro-2006\)](#)

# **LoopPoint**: Key idea 1: Filtering Synchronization Code during profiling

Why: Profiling should look only at ‘real work’

What: Skip profiling of synchronization code

How?

- Automatically with Loop Analysis: Very hard

“Spin Detection Hardware for Improved Management of Multithreaded Systems”  
Transactions on Parallel and Distributed Systems, 2006

- **Look for loops that do not update architectural state**
- Was implemented in Sniper(Pin-2) but many OpenMP spin loops maintain stats hence do update architecture state

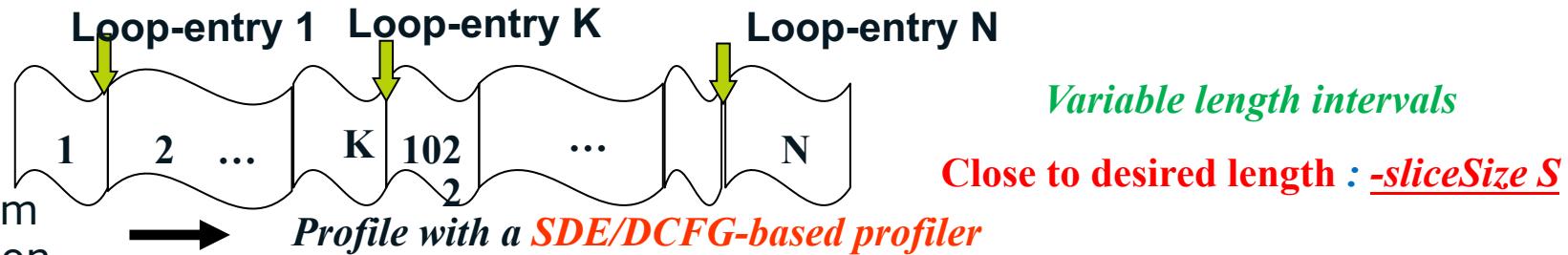
✓ **Heuristic**

- Filter synchronization library code: e.g. libiomp5.so, libpthread.so

# **LoopPoint**: Key idea 2: Loops as ‘Units of work’

**Why:** Property of program/binary : independent of architecture

## Profiling



- Global counting of loop-entries
- Region start/stop : only in the main image
  - Stop when ‘desired global instruction count’ (SliceSize) is reached
  - Do not count instructions in synchronization library

# *DCFG* Generation with *PinPlay*

## Dynamic Control-Flow Graph (DCFG)

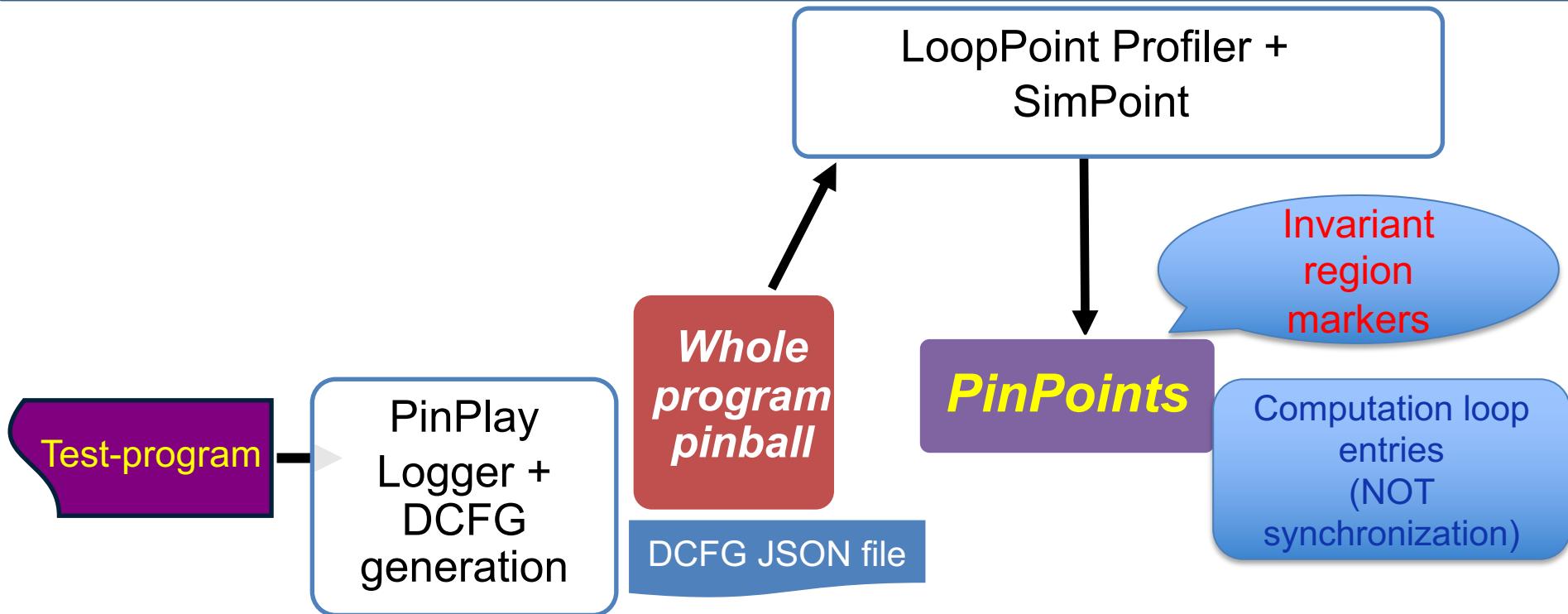
Directed graph extracted for a specific execution:

Nodes → basic blocks

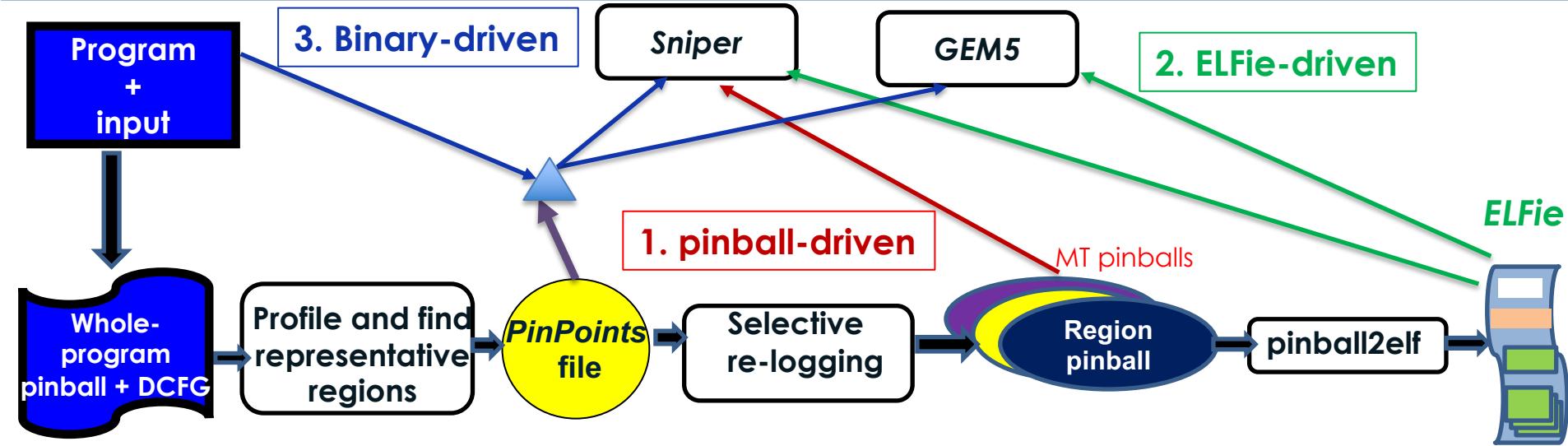
Edges → control-flow : augmented with per-thread execution counts



# *PinPlay + DCFG : Stronger Repeatability*



# *LoopPoint*: Simulation alternatives



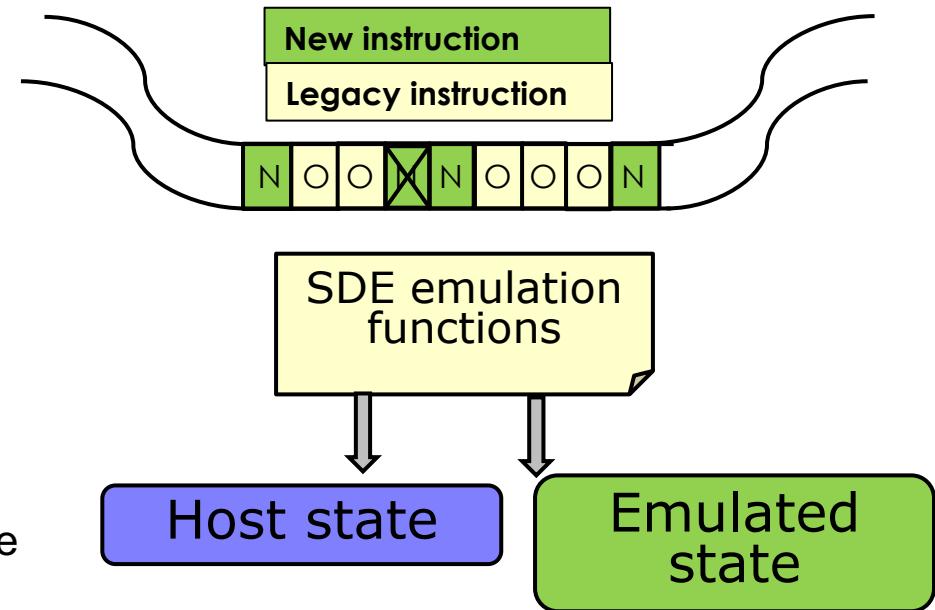
**Requirement:** Execution invariant region specification  
(PC+count for compute loop entries)

# Intel Software Development Emulator (*Intel SDE*)

- The Intel® Software Development Emulator is a **functional user-level (ring 3) emulator** for x86 (32b and 64b) new instructions built upon Pin and XED (X86 encoder/decoder)
- **Goal:** New instruction/register emulation between the time when they are designed and when the hardware is available.
- Used for compiler development, architecture and workload analysis, and tracing for architecture simulators
- No special compilation required
- Supported on Windows/Linux/Mac OS
- Runs only in user space (ring 3)

# How SDE Works

- Based on Pin (<http://pintool.intel.com>) and XED decoder/encoder (<https://github.com/intelxed/xed>)
- Instrument new instructions
  - Add call to emulation routine
  - Delete original instruction
- Emulation routine:
  - Update native state with emulated state



# Using *SDE* for *PinPoints* and *LoopPoint*

Prerequisites:

1. SDE build kit (version 9.0 or higher) from Intel

<http://www.intel.com/software/sde>

2. pinplay-tools from Intel

<https://github.com/intel/pinplay-tools>

3. SimPoint sources from UCSD

<https://cseweb.ucsd.edu/~calder/simpoint/>

4. Pinball2elf sources from Intel

<http://pinelfie.org> → <https://github.com/intel/pinball2elf>

# Getting ready for *LoopPoint* ...

1. Expand SDE build-kit : *setenv SDE\_BUILD\_KIT <path to SDE kit>*
2. *cp -r pinplay-tools/pinplay-scripts \$ SDE\_BUILD\_KIT*
3. Build simpoint (see pinplay-tools/pinplay-scripts/README.simpoint)
  - *cp <path>/SimPoint.3.2/bin/simpoint \$ SDE\_BUILD\_KIT/pinplay-scripts/PinPointsHome/Linux/bin/*
4. Build global looppoint tools
  - *setenv PINBALL2ELF <path to pinball2lef repo>*
  - *cd pinplay-tools/GlobalLoopPoint*
  - *./sde-build-GlobalLoopPoint.sh*

# **SDE kit expanded for LoopPoint**

**sde-external-9.0.0-2021-11-07-lin**

```
└── ...
   └── intel64
       ├── sde-global-event-icounter.so
       └── sde-global-looppoint.so
   └── ...
   └── pinplay-scripts
       PinPointsHome/
           └── Linux
               └── bin
                   └── LICENSE.simpoint
                       └── simpoint
```

# Running *LoopPoint* for an OpenMP program

- `cd pinplay-tools/dotproduct-omp # see README there`
- `make # builds dotproduct-omp → base.exe`
- `./sde-run.looppoint.global_looppoint.concat.filter.flowcontrol.sh`

`~/pinplay-tools/dotproduct-omp`

  └── `dotproduct.1_282016.Data`

  └── `dotproduct.1_282016.pp`

  └── `whole_program.1`

bbv files (\*.bb), PinPoints  
file (\*.csv, \*.CSV)

Region pinballs

Whole-program pinball + DCFG

# Summary: Simulation of Multi-threaded Programs: Tools & Methodologies

Where to simulate?

**SDE + LoopPoint**  
Compute-loop iterations as  
'Unit of work'

How to simulate?

1. Pinball-driven
2. ELFie-driven
3. Binary-driven

Are the regions representative?

1. Simulation (Sniper) -based  
2. ELFie-based / Binary+ROI/Perf (*not covered*)  
Whole-program performance vs  
Region-predicted performance

# Agenda

| Time (Eastern) | Speaker            | Topic                                      |
|----------------|--------------------|--------------------------------------------|
| 13.20 to 13.30 | Trevor E. Carlson  | Overview of the tutorial                   |
| 13.30 to 14.20 | Akanksha Chaudhari | Performance analysis, simulation, sampling |
| 14.20 to 15.20 | Harish Patil       | Using tools: Pin, PinPlay, SDE, ELFies     |
| 15.20 to 15.40 |                    | <b>Break</b>                               |
| 15.40 to 16.20 | Alen Sabu          | Multi-threaded sampling and LoopPoint      |
| 16.20 to 17.00 | Changxi Liu        | Sniper and LoopPoint demo                  |
| 17.00 to 17.40 | Zhantong Qiu       | Using LoopPoint with gem5                  |

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis



Session 3

# Multi-threaded Sampling and LoopPoint

ALEN SABU, PHD CANDIDATE

NATIONAL UNIVERSITY OF SINGAPORE

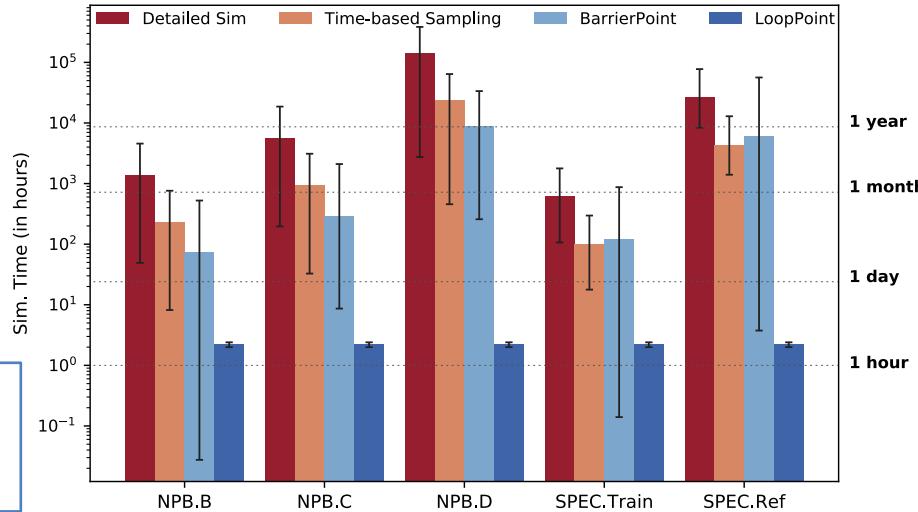
# Simulation in the Post-Dennard Era

Microarchitectural simulation is slow

Solution: Simulate  
regions of interest

# Simulation in the Post-Dennard Era

Solution: Simulate regions of interest



Benchmarks with 8 threads,  
static schedule, passive wait-  
policy, simulated at 100 KIPS.

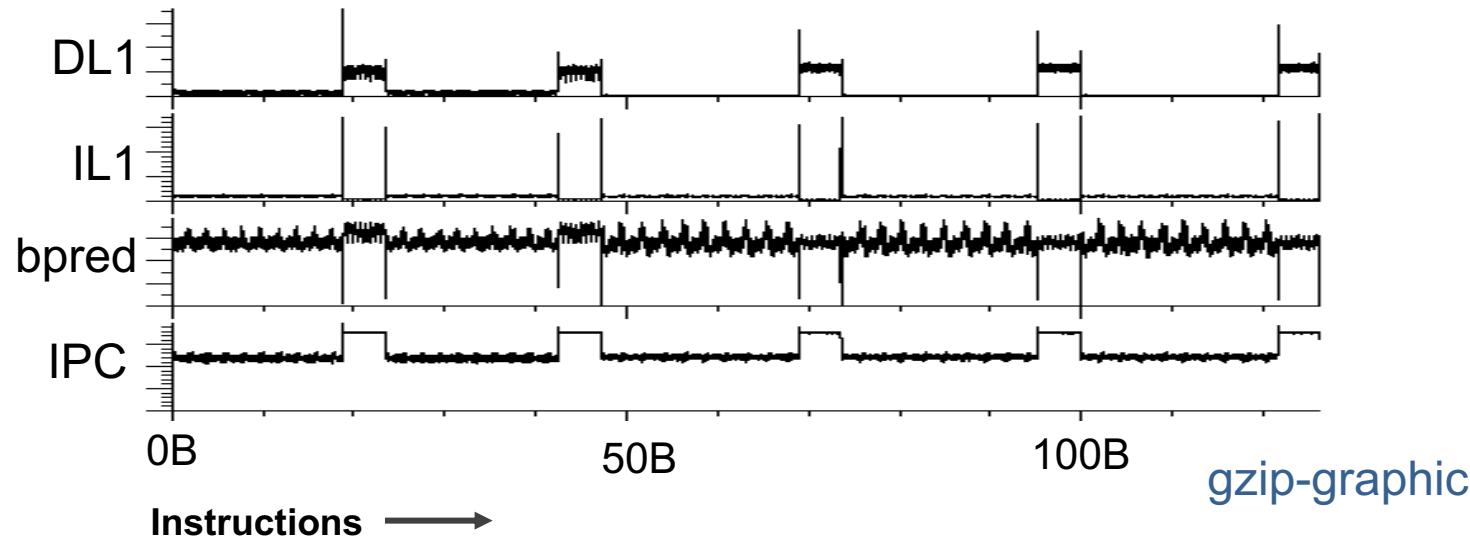
Microarchitectural simulation is slow

# Selection of Regions of Interest

**Program executions are structured as phases**

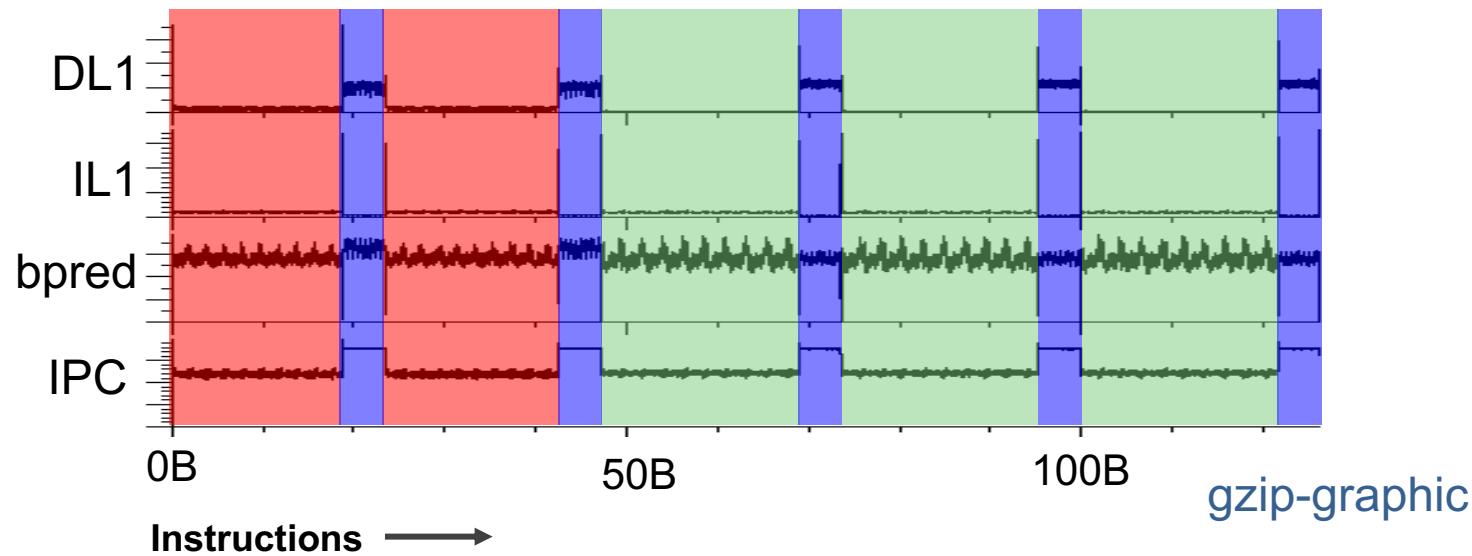
# Selection of Regions of Interest

Program executions are structured as phases



# Selection of Regions of Interest

Program executions are structured as phases



# Selection of Regions of Interest

Sampling using *SimPoint*

|                  |                  |                  |                  |                  |                  |                  |                  |                  |                   |                   |                   |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------------|-------------------|-------------------|
| BBV <sub>1</sub> | BBV <sub>2</sub> | BBV <sub>3</sub> | BBV <sub>4</sub> | BBV <sub>5</sub> | BBV <sub>6</sub> | BBV <sub>7</sub> | BBV <sub>8</sub> | BBV <sub>9</sub> | BBV <sub>10</sub> | BBV <sub>11</sub> | BBV <sub>12</sub> |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------------|-------------------|-------------------|

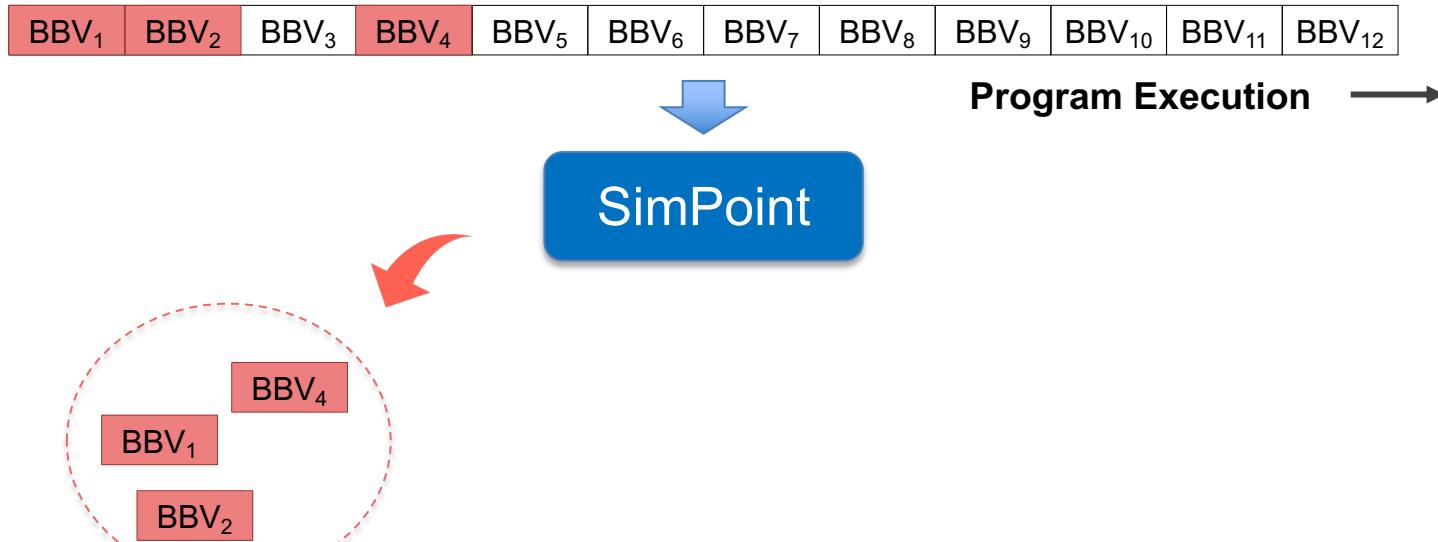


Program Execution →

SimPoint

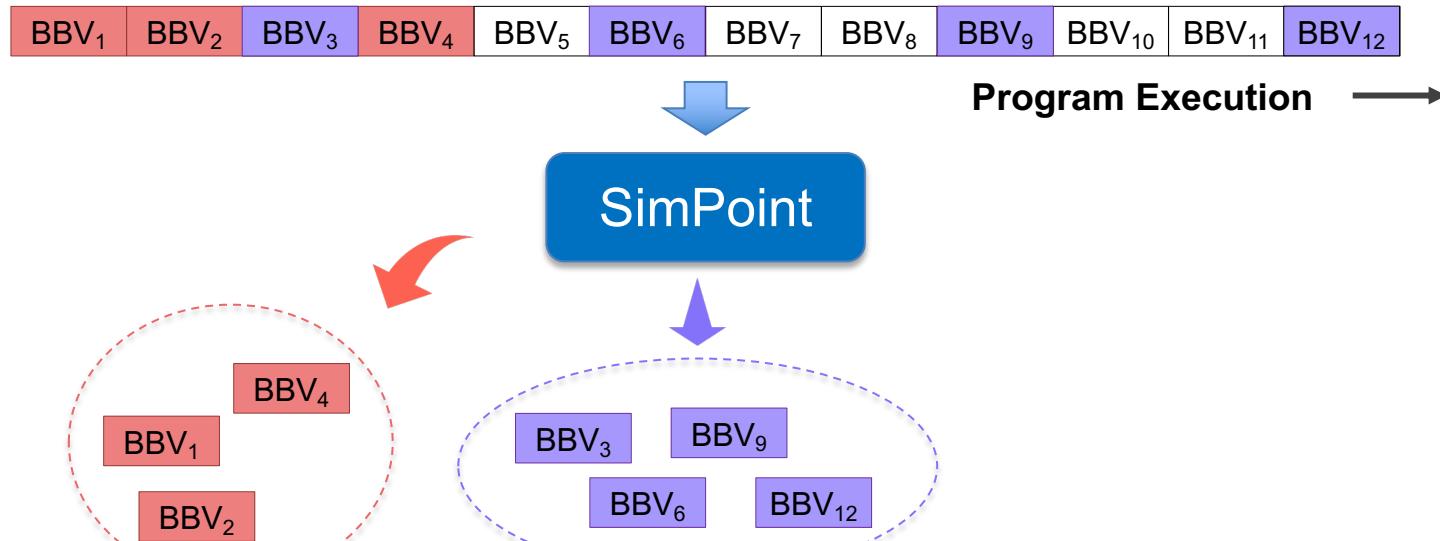
# Selection of Regions of Interest

Sampling using *SimPoint*



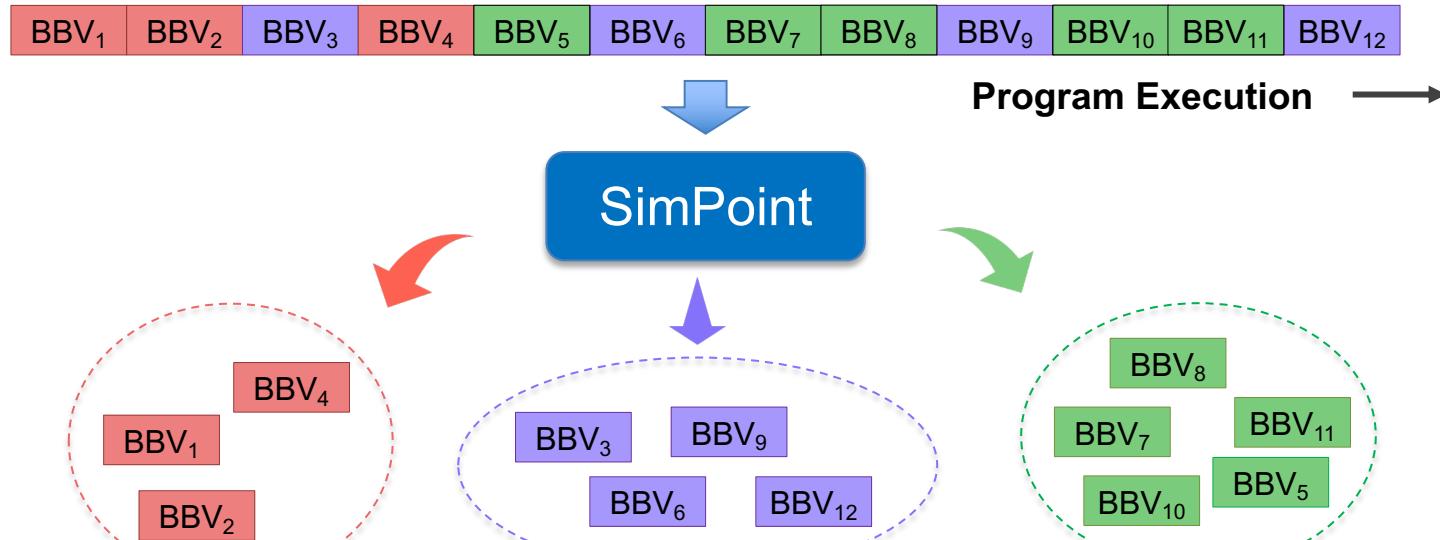
# Selection of Regions of Interest

## Sampling using *SimPoint*



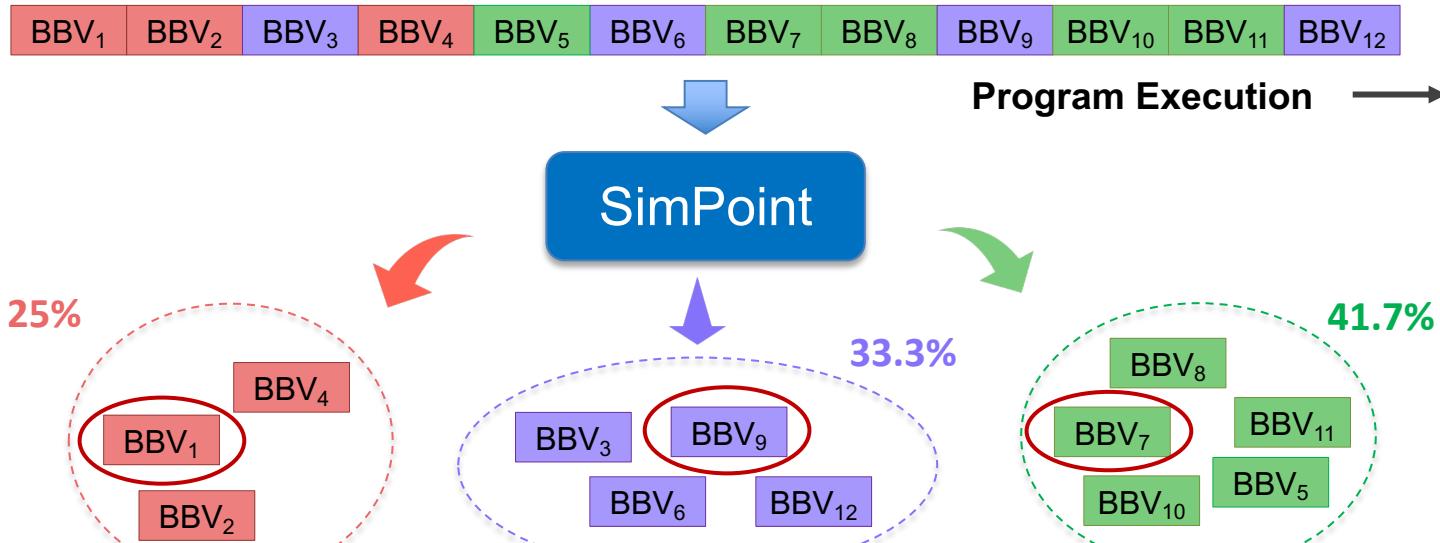
# Selection of Regions of Interest

## Sampling using *SimPoint*



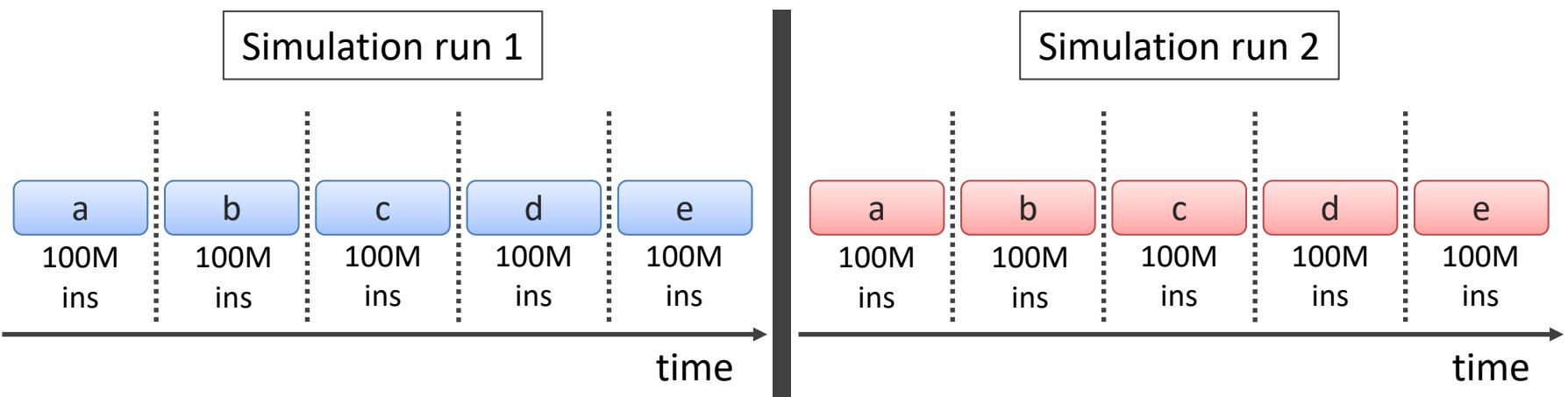
# Selection of Regions of Interest

## Sampling using *SimPoint*



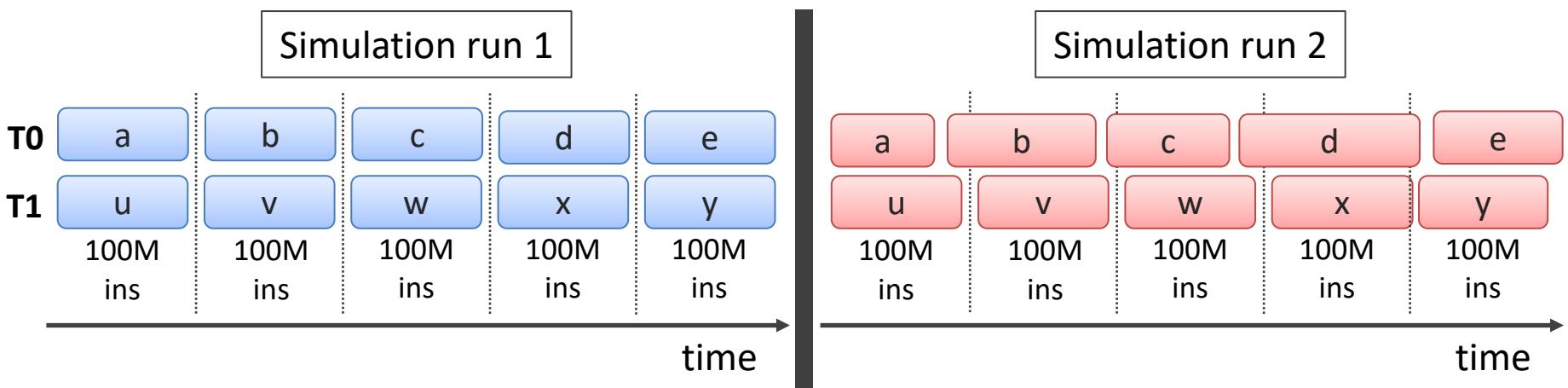
# Extending Single-threaded Techniques

- SimPoint or SMARTS ➤ Instruction count-based techniques
  - Works well for single-threaded applications



# Extending Single-threaded Techniques

- SimPoint or SMARTS ➤ Instruction count-based techniques
  - Inconsistent regions for multi-threaded applications



# Multi-threaded Sampling is Complex

Instruction count-based  
techniques are unsuitable<sup>1</sup>

Threads progress differently  
due to load imbalance

Representing parallelism  
among threads

Differentiating thread  
waiting from real work

# Multi-threaded Sampling is Complex

Instruction count-based  
techniques are unsuitable<sup>1</sup>

Threads progress differently  
due to load imbalance

Identify a *unit of work* that is *invariant across executions*

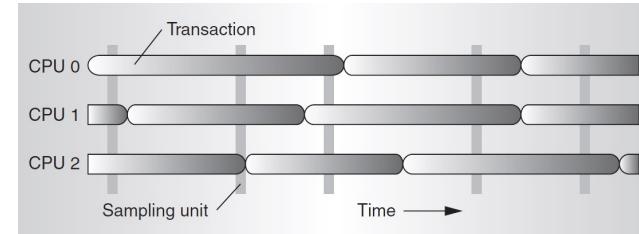
Representing parallelism  
among threads

Differentiating thread  
waiting from real work

# Multi-threaded Sampling: Prior works

## FlexPoints

- + Designed for non-synchronizing throughput workloads
- ✓ Instruction count-based sampling
- ✗ Assumes no thread interaction
- ✗ Requires simulation of the full application



# Multi-threaded Sampling: Prior works

## Time-based Sampling



Designed for synchronizing generic multi-threaded workloads



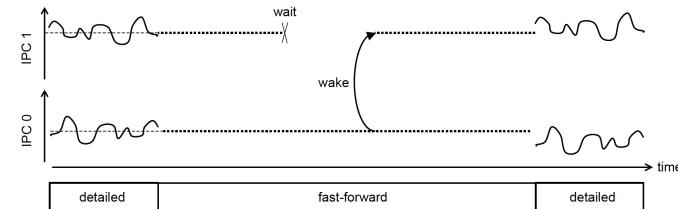
Applies to generic multi-threaded workloads



Extremely slow



Requires simulation of the full application



# Multi-threaded Sampling: Prior works

## BarrierPoint



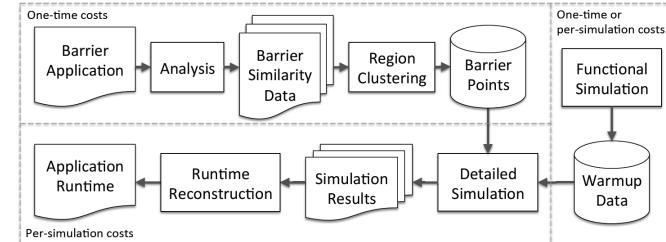
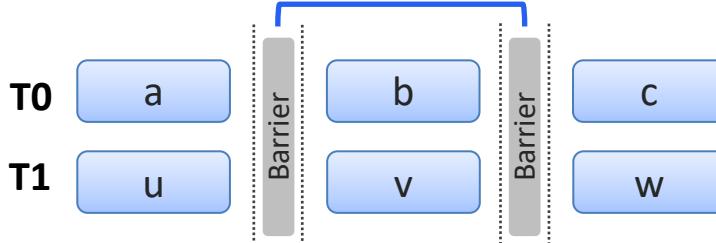
Designed for barrier-synchronized multi-threaded workloads



Scales well with number of barriers



Slow when *inter-barrier regions* are large



# Multi-threaded Sampling: Prior works

## TaskPoint



Designed for task-based workloads

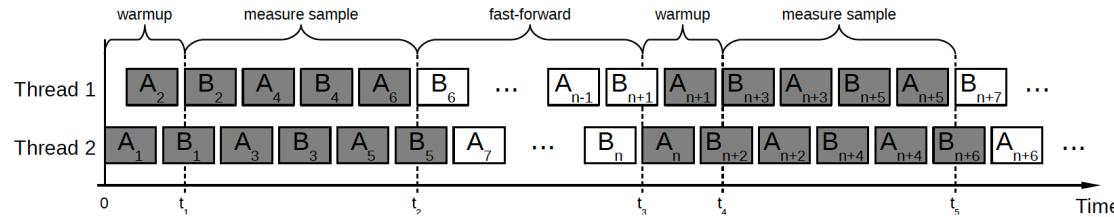


Uses analytical models to improve accuracy



Works only for the particular workload type

```
#pragma omp task  
label(task_type 1)  
do_something();
```



# The Unit of Work

## Single-threaded Sampling

SimPoint<sup>1</sup>      ➡      Instruction count  
SMARTS<sup>2</sup>

## Multiprocessor Sampling

Flex Points<sup>3</sup>      ➡      Instruction count

## Multi-threaded Sampling

Time-based sampling<sup>4</sup>      ➡      Time

## Application-specific Sampling

BarrierPoint<sup>5</sup>      ➡      Inter-barrier regions  
TaskPoint<sup>6</sup>      ➡      Task instances

<sup>1</sup>Sherwood et al., “Automatically Characterizing Large Scale Program Behavior”, ASPLOS’02

<sup>2</sup>Wunderlich et al., “SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling”, ISCA’03

<sup>3</sup>Wenisch et al., “SimFlex: statistical sampling of computer system simulation”, IEEE Micro’06

<sup>4</sup>Carlson et al., “Sampled Simulation of Multithreaded Applications”, ISPASS’13

<sup>5</sup>Carlson et al., “BarrierPoint: Sampled simulation of multi-threaded applications”, ISPASS’14

<sup>6</sup>Grass et al., “TaskPoint: Sampled simulation of task-based programs”, ISPASS’16

# The Unit of Work

## Single-threaded Sampling

SimPoint<sup>1</sup> → Instruction count  
SMARTS<sup>2</sup>

## Multiprocessor Sampling

Flex Points<sup>3</sup> → Instruction count

**We consider generic loop iterations as the unit of work**

Time-based sampling<sup>4</sup> → Time

BarrierPoint<sup>5</sup> → Inter-barrier regions  
TaskPoint<sup>6</sup> → Task instances

<sup>1</sup>Sherwood et al., "Automatically Characterizing Large Scale Program Behavior", ASPLOS'02

<sup>2</sup>Wunderlich et al., "SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling", ISCA'03

<sup>3</sup>Wenisch et al., "SimFlex: statistical sampling of computer system simulation", IEEE Micro'06

<sup>4</sup>Carlson et al., "Sampled Simulation of Multithreaded Applications", ISPASS'13

<sup>5</sup>Carlson et al., "BarrierPoint: Sampled simulation of multi-threaded applications", ISPASS'14

<sup>6</sup>Grass et al., "TaskPoint: Sampled simulation of task-based programs", ISPASS'16

# LoopPoint Methodology

Where to simulate

How to simulate

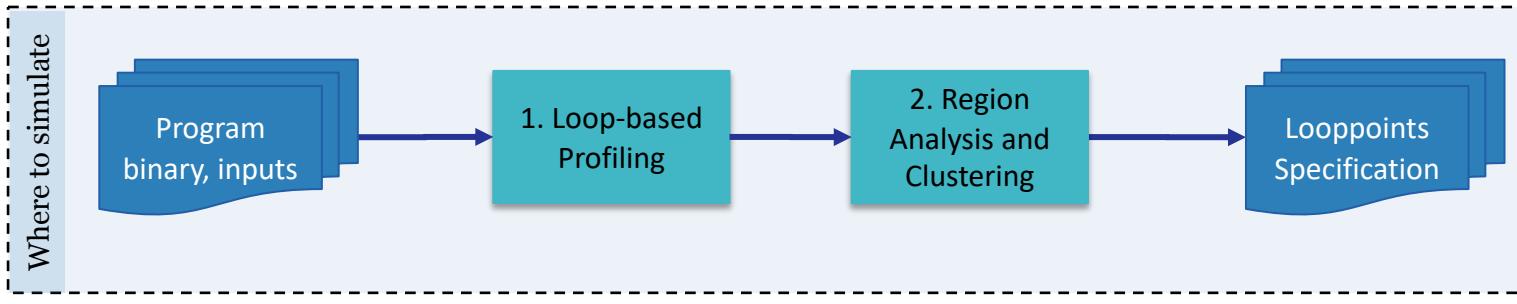
# LoopPoint Methodology

Where to simulate

Program  
binary, inputs

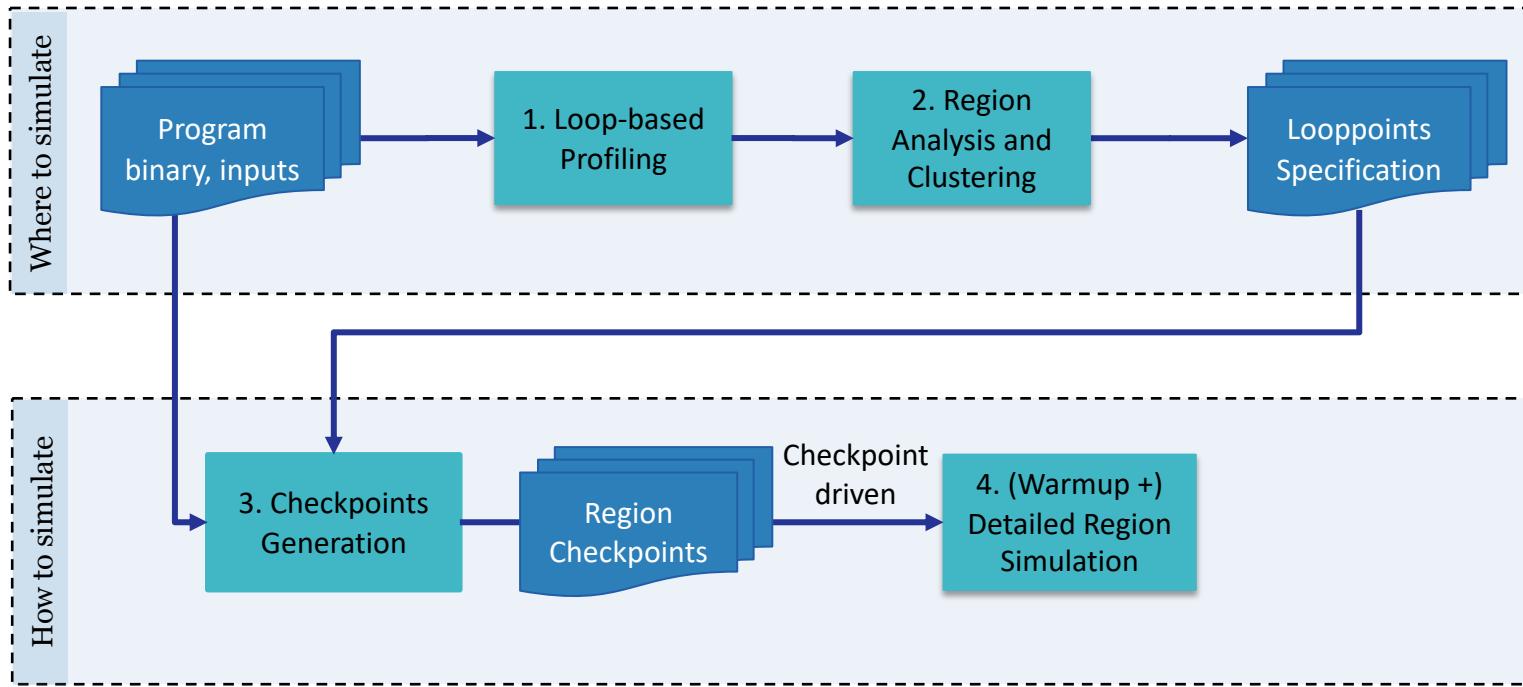
How to simulate

# LoopPoint Methodology

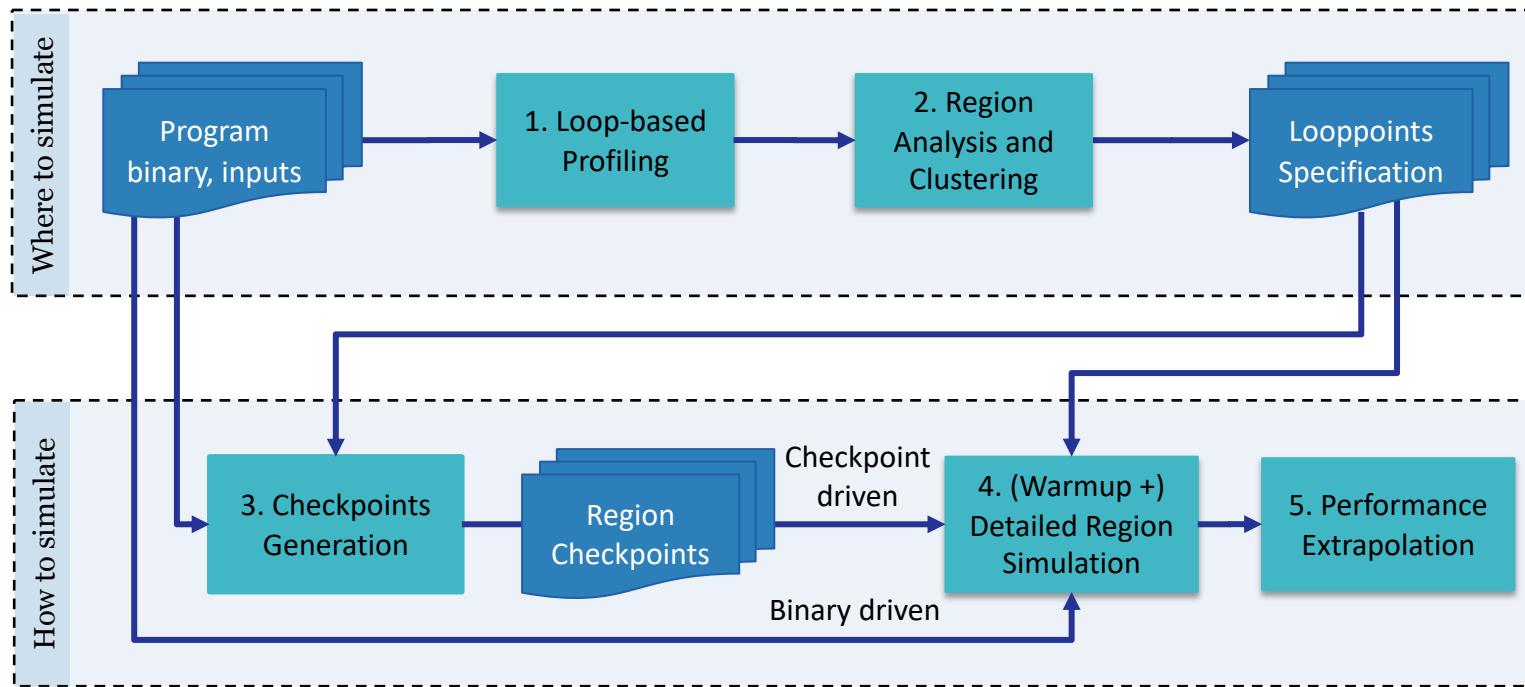


How to simulate

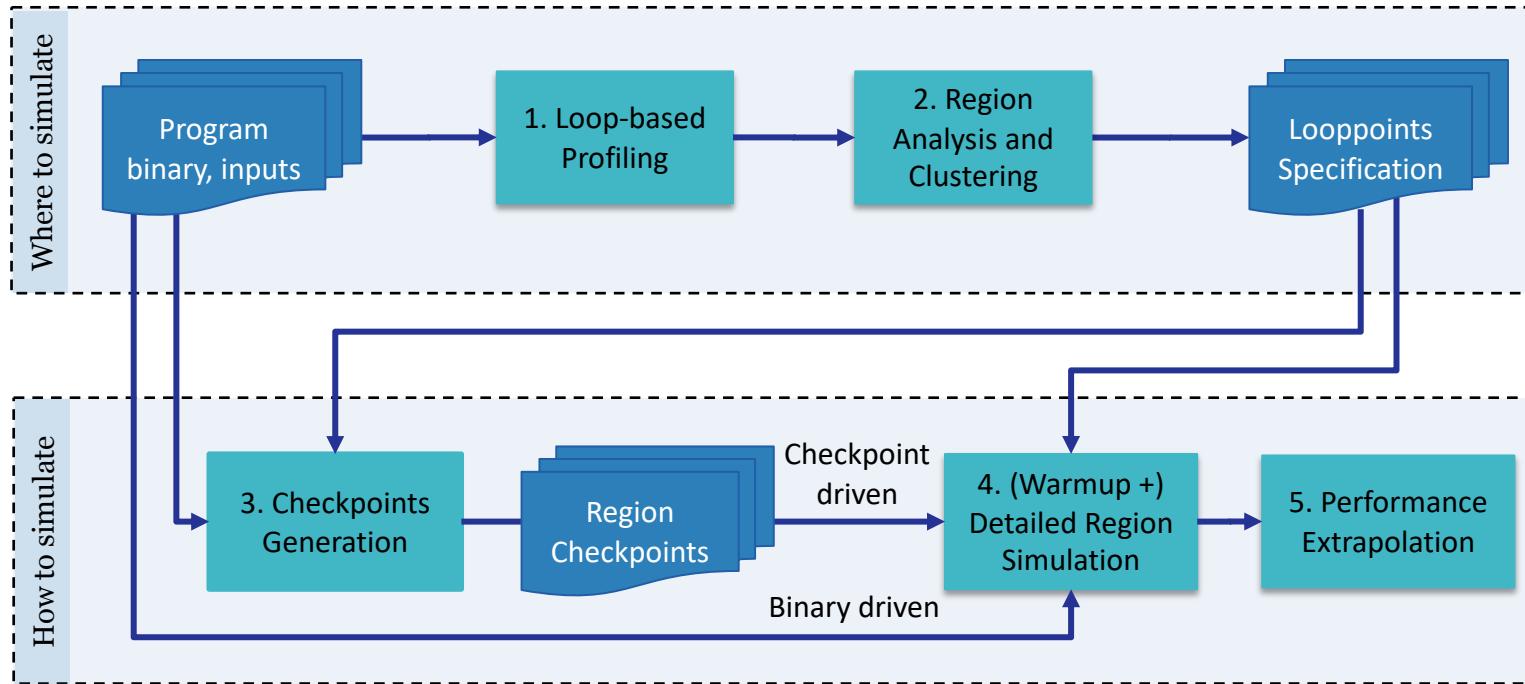
# LoopPoint Methodology



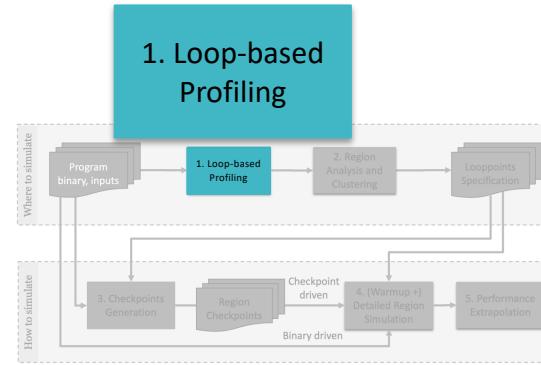
# LoopPoint Methodology



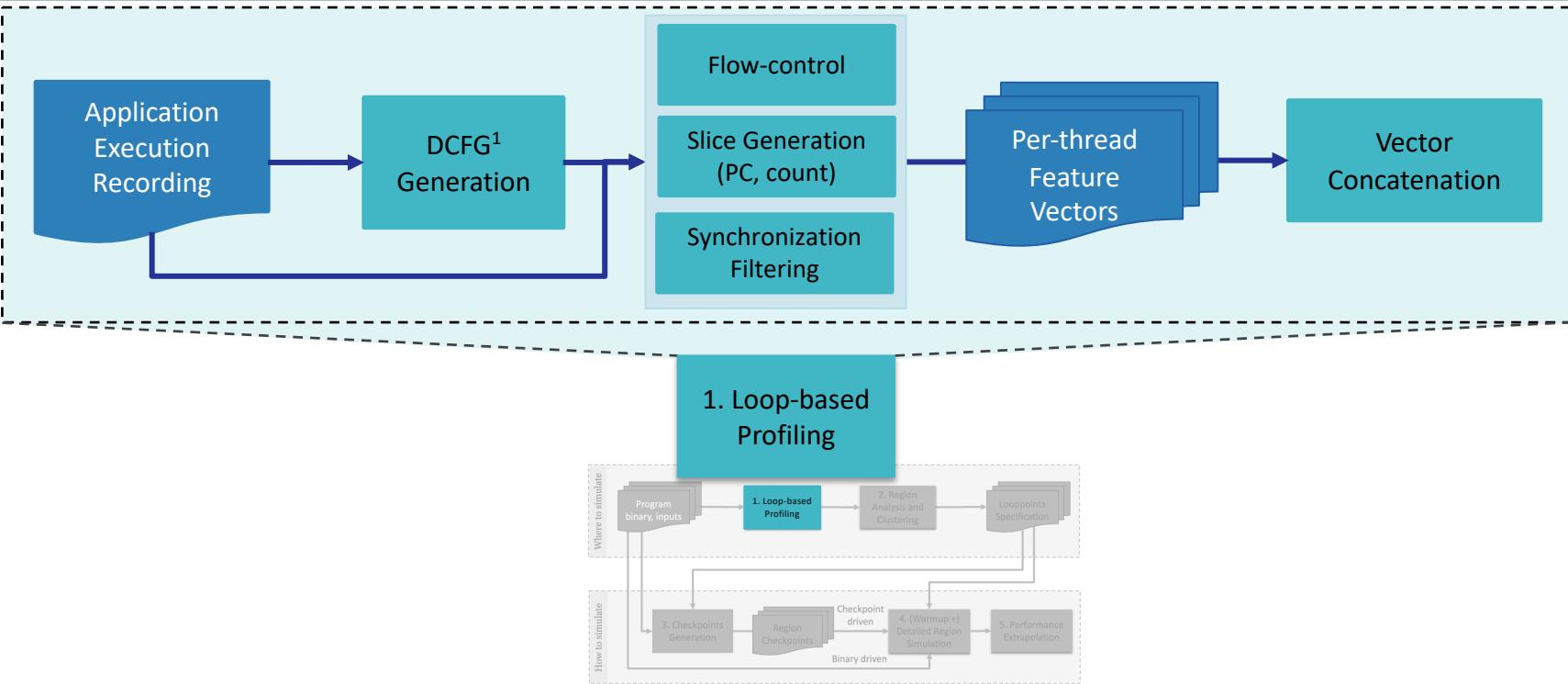
# Loop-based Profiling



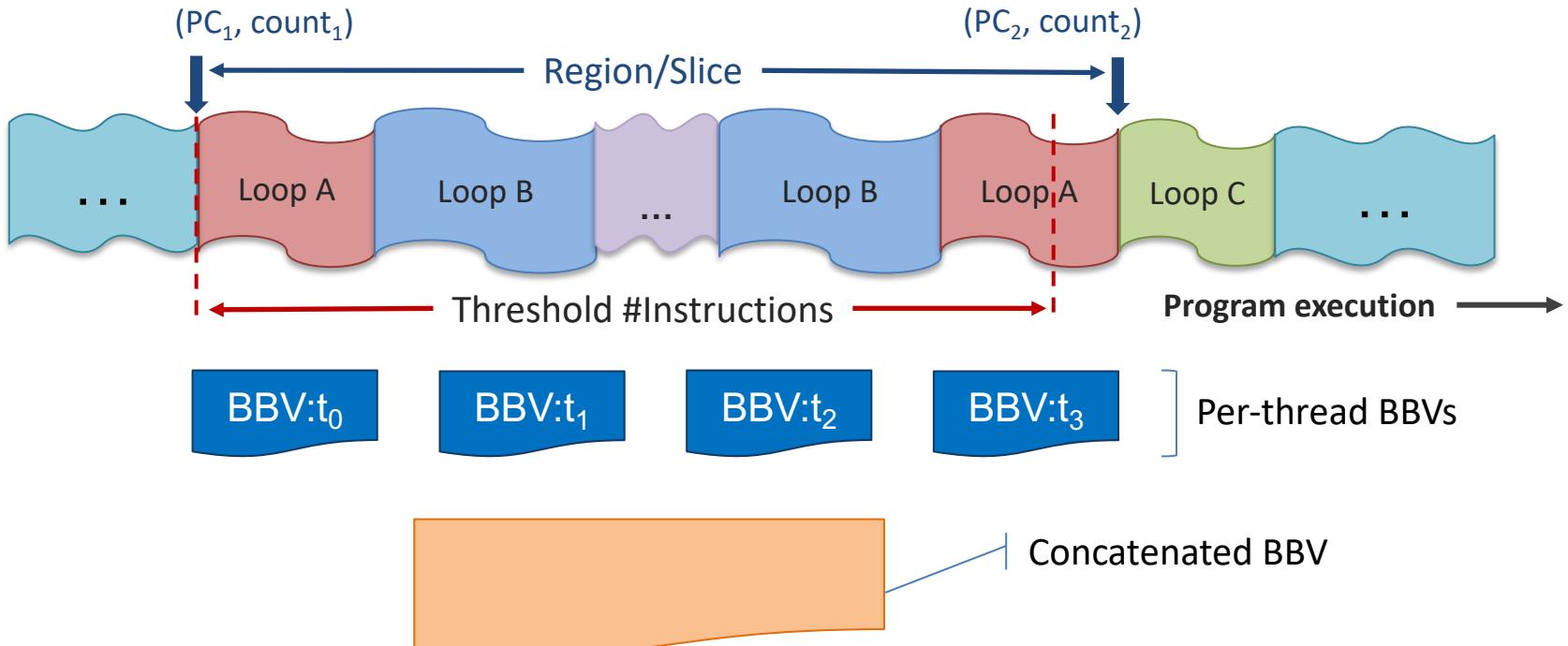
# Loop-based Profiling



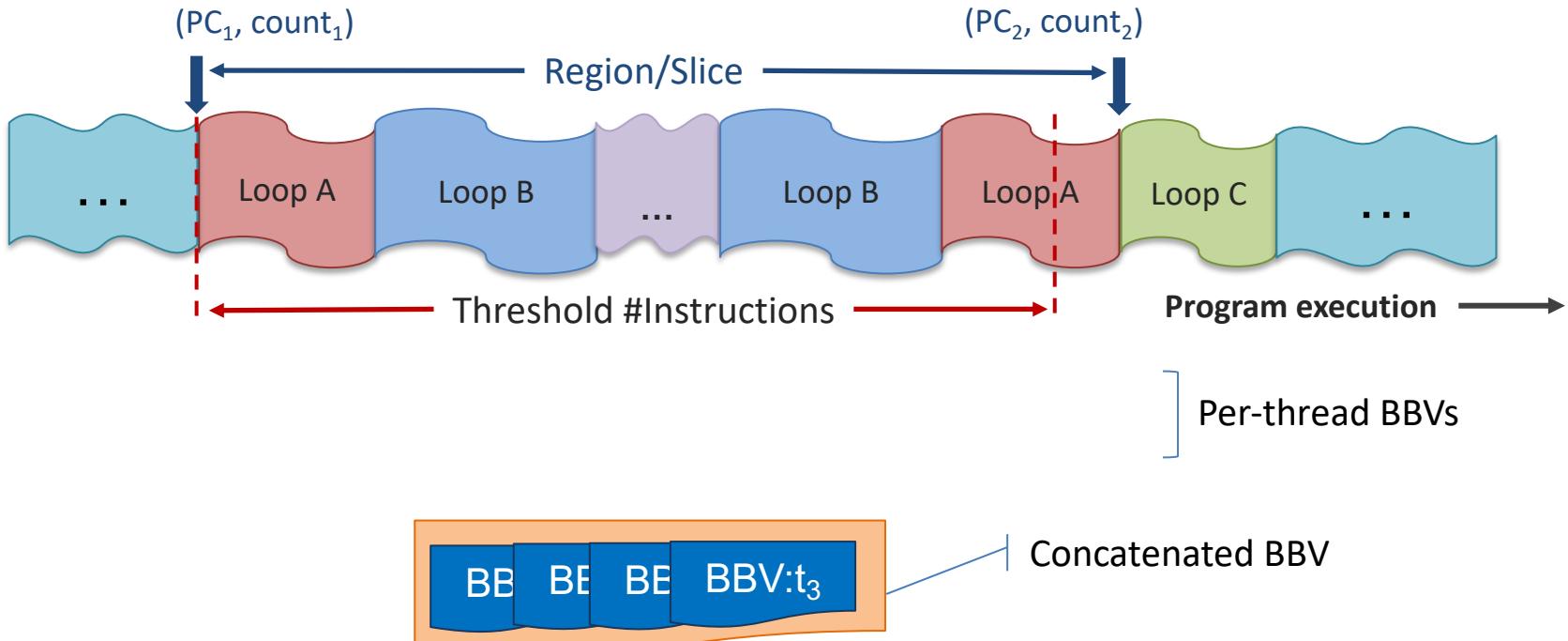
# Loop-based Profiling



# Region Representation

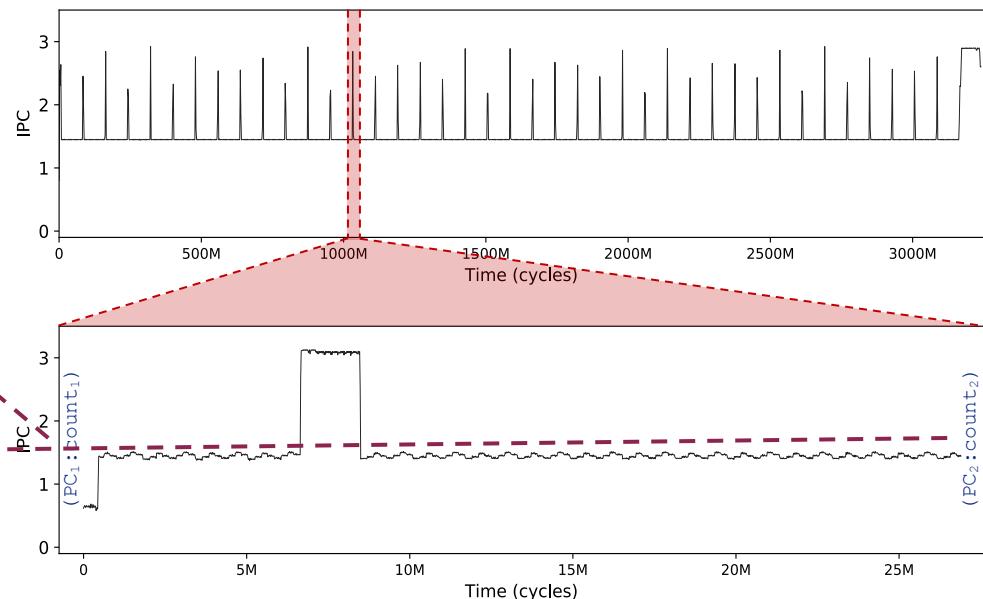


# Region Representation



# A LoopPoint Region

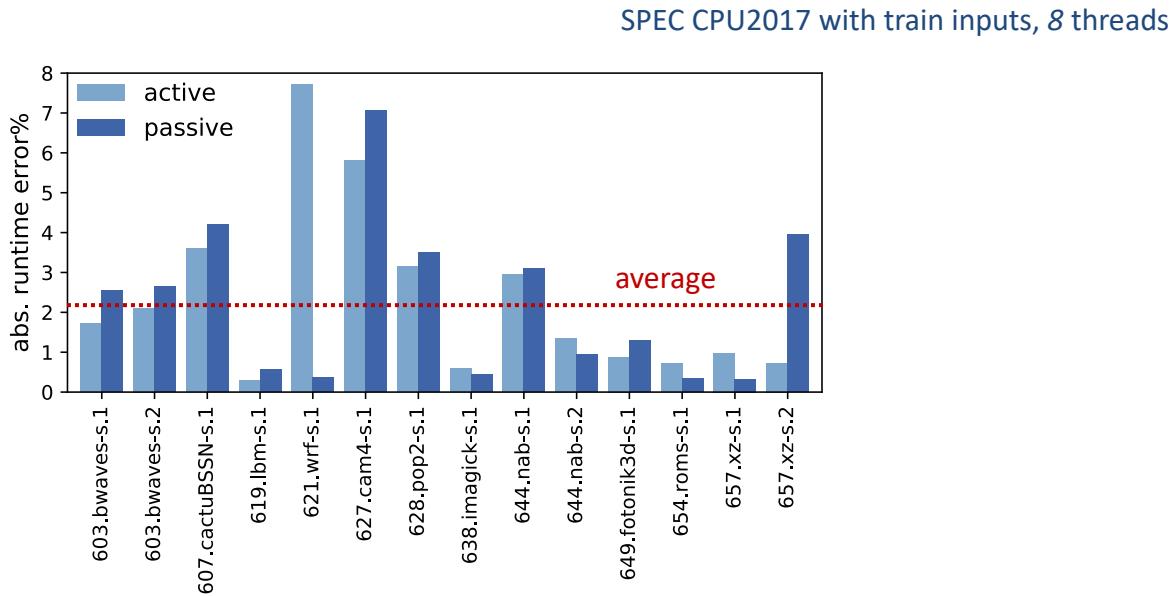
```
638.imagick_s/magick/morphology.c
2842 #if defined(MAGICKCORE_OPENMP_SUPPORT)
2843     #pragma omp parallel for schedule(static,4) shared(progress,status) \
2844         magick_threads(image,result_image,image->rows,1)
2845 #endif
2846     for (y=0; y < (ssize_t) image->rows; y++)
2847     {
2848         ...
2849         for (x=0; x < (ssize_t) image->columns; x++)
2850         {
2851             ...
2852             for (v=0; v < (ssize_t) kernel->height; v++) {
2853                 for (u=0; u < (ssize_t) kernel->width; u++, k--) {
2854                     ...
2855                     } /* u */
2856                     ...
2857                     } /* v */
2858                 } /* x */
2859             } /* y */
2860             ...
2861             }
```



638.imagick\_s, train input, 8 threads

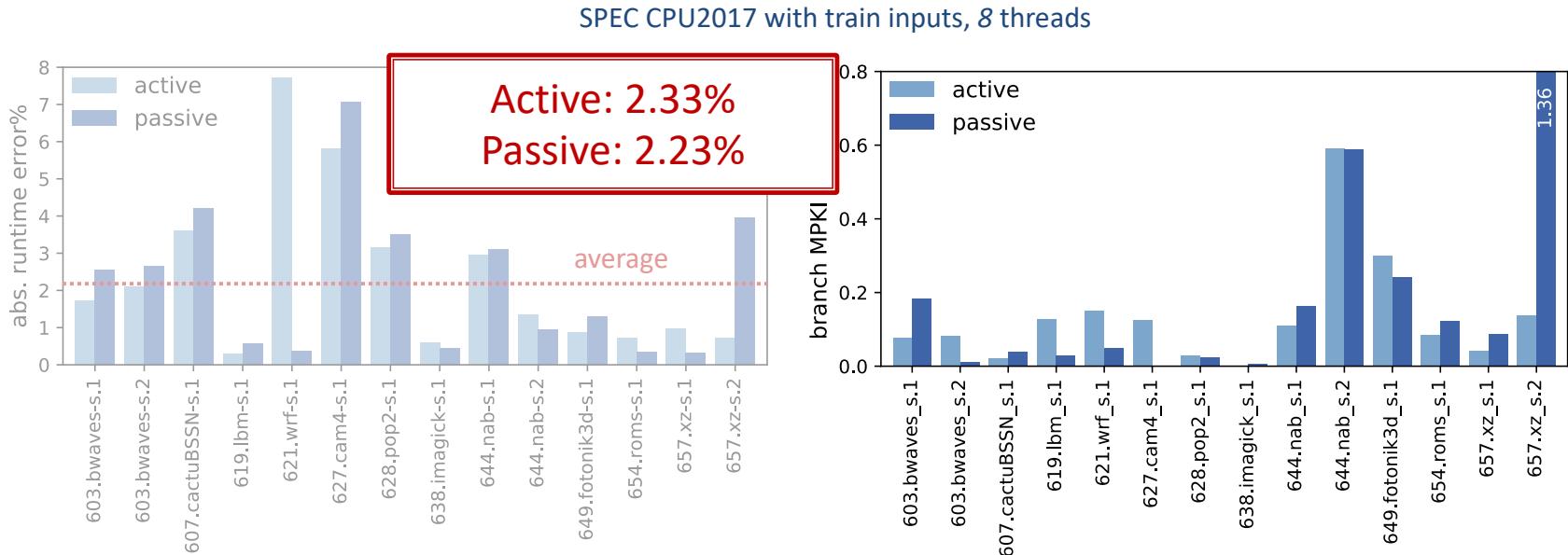
# Accuracy Results

Prediction error wrt. performance of whole application



# Accuracy Results

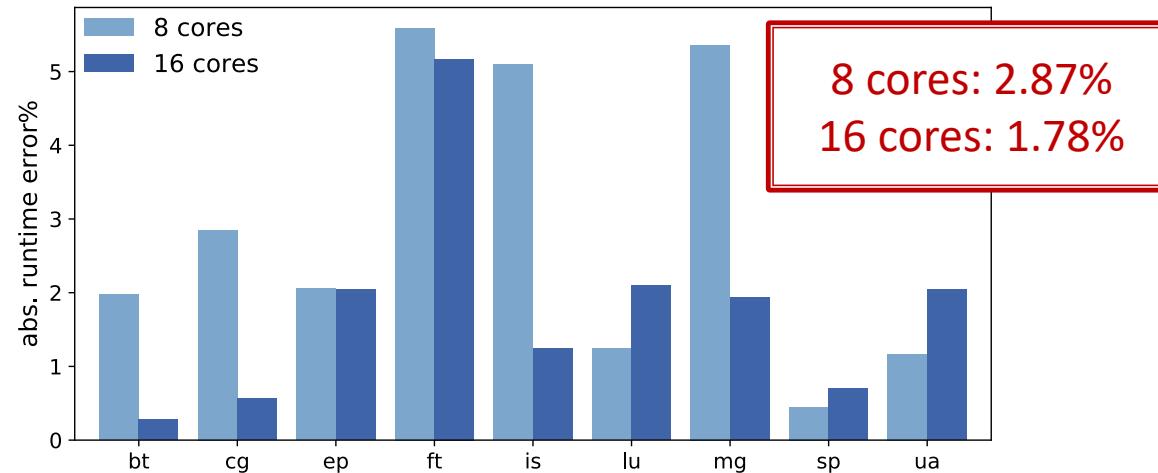
Prediction error wrt. performance of whole application



# Changing Thread Count

Runtime prediction error wrt. whole application runtime

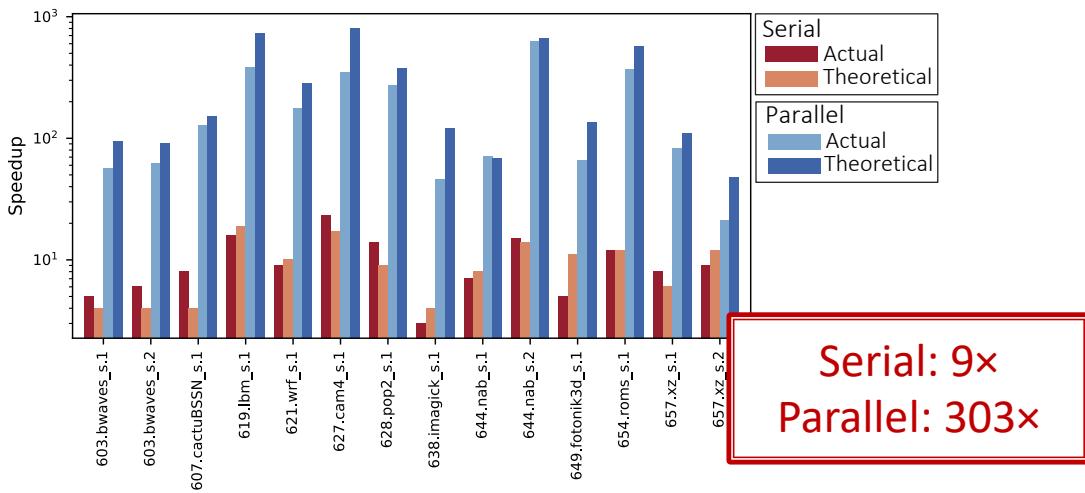
NPB 3.3 with *Class C* inputs, 8 and 16 threads, *passive* wait-policy



# Speedup

Parallel and serial speedup achieved for LoopPoint

SPEC CPU2017 with *train* inputs, 8 threads, *active* wait-policy

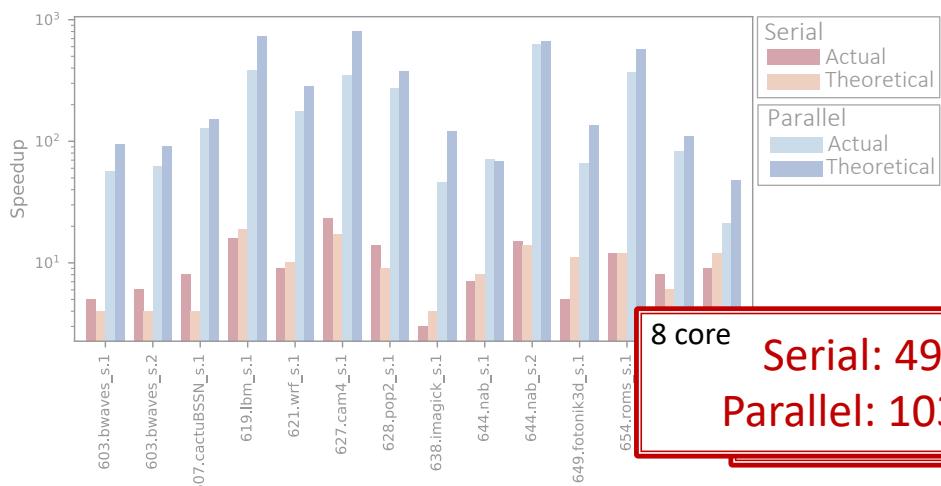


Serial: 9x  
Parallel: 303x

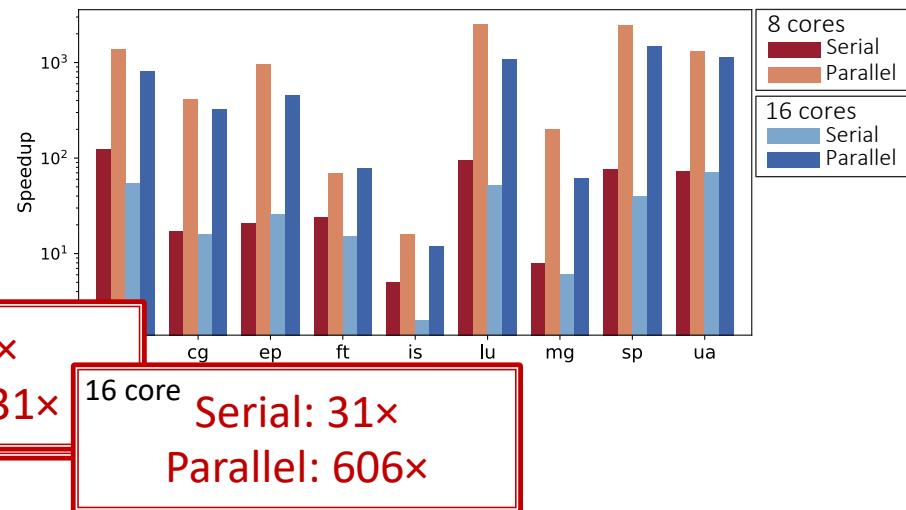
# Speedup

## Parallel and serial speedup achieved for LoopPoint

SPEC CPU2017 with *train* inputs, 8 threads, *active* wait-policy



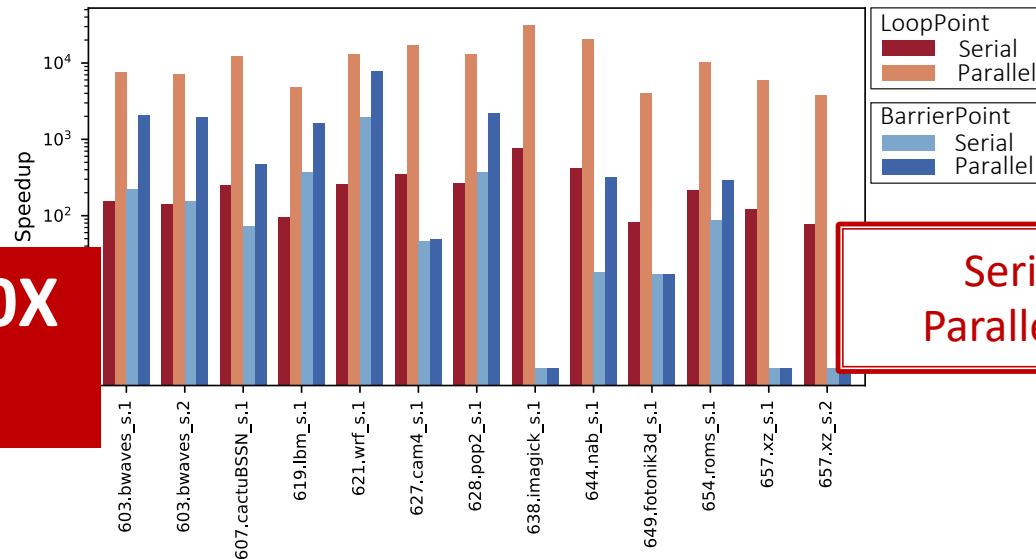
NPB with *Class C* inputs, 8 and 16 threads, *passive* wait-policy



# Speedup

## Theoretical Speedup comparison with BarrierPoint

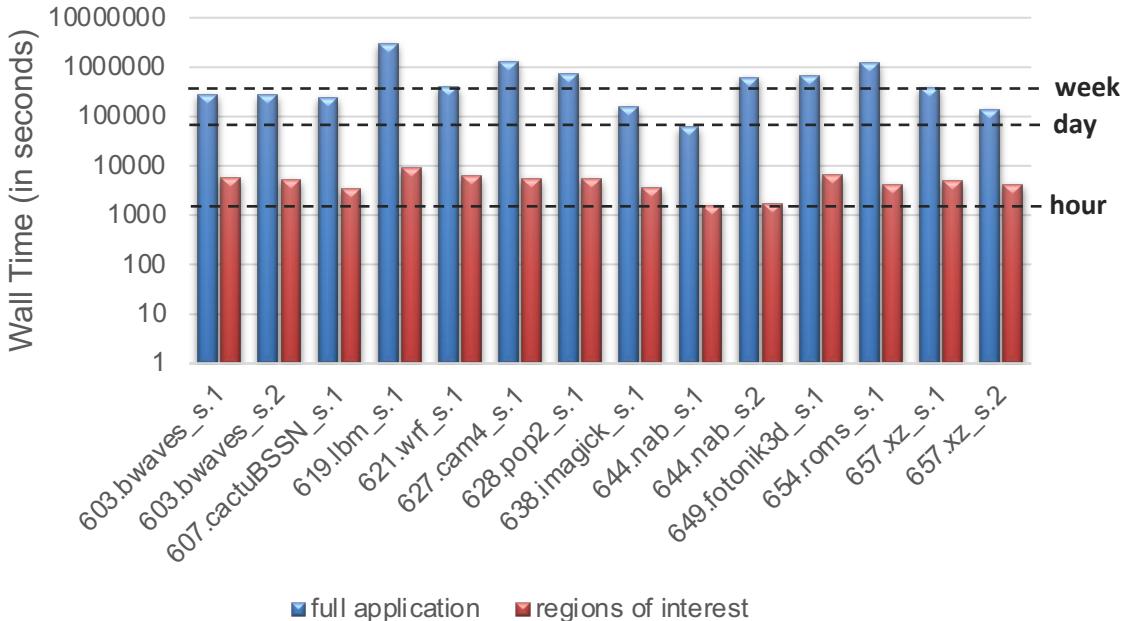
SPEC CPU2017 with *ref* inputs, 8 threads, *passive* wait-policy



Up to 31000X  
speedup!

Serial: 244x  
Parallel: 11587x

# Full Applications Vs. Regions of Interest



A comparison of simulation wall times of SPEC CPU2017 benchmarks using train inputs and 8 threads on SDE-based IWPS (CLX)

| Application       | Train | Ref  |
|-------------------|-------|------|
| 603.bwaves_s.1    | 33.33 | 1.01 |
| 603.bwaves_s.2    | 32.79 | 1.03 |
| 607.cactubSSN_s.1 | 26.81 | 0.45 |
| 619.lbm_s.1       | 4.86  | 0.65 |
| 621.wrf_s.1       | 9.28  | 0.47 |
| 627.cam4_s.1      | 4.78  | 0.23 |
| 628.pop2_s.1      | 6.27  | 0.46 |
| 638.imagick_s.1   | 25.93 | 0.13 |
| 644.nab_s.1       | 21.15 | 0.32 |
| 644.nab_s.2       | 9.74  | —    |
| 649.fotonik3d_s.1 | 12.93 | 1.55 |
| 654.roms_s.1      | 3.98  | 0.71 |
| 657.xz_s.1        | 21.43 | 0.74 |
| 657.xz_s.2        | 42.55 | 1.26 |

Fraction of regions to be simulated in detail for SPEC CPU2017 benchmarks using 8 threads

# More Information

- Links
  - SPEC2017 Looppoint-based checkpoints (ELFies<sup>1</sup>)
    - 603.bwaves\_s.1: <https://mynbox.nus.edu.sg/u/lAb1PcAG5X6GBuU-/9491eb22-b988-4bc5-9865-991a66d20944?l>
    - 603.bwaves\_s.2: <https://mynbox.nus.edu.sg/u/8L13qqS9d8DivXeB/6d14af21-6c4e-413a-b046-3c6115a211fc?l>
  - LoopPoint GitHub: <https://github.com/nus-comparch/looppoint>
  - ELFies GitHub: <https://github.com/intel/pinball2elf>
  - Web page: <https://looppoint.github.io>
  - Short talk: <https://youtu.be/Tr609MKT42g>
  - Questions: [alen@u.nus.edu.sg](mailto:alen@u.nus.edu.sg), [tcarlson@nus.edu.sg](mailto:tcarlson@nus.edu.sg)



# Agenda

| Time (Eastern) | Speaker            | Topic                                      |
|----------------|--------------------|--------------------------------------------|
| 13.20 to 13.30 | Trevor E. Carlson  | Overview of the tutorial                   |
| 13.30 to 14.20 | Akanksha Chaudhari | Performance analysis, simulation, sampling |
| 14.20 to 15.20 | Harish Patil       | Using tools: Pin, PinPlay, SDE, ELFies     |
| 15.20 to 15.40 |                    | Break                                      |
| 15.40 to 16.20 | Alen Sabu          | Multi-threaded sampling and LoopPoint      |
| 16.20 to 17.00 | Changxi Liu        | Sniper and LoopPoint demo                  |
| 17.00 to 17.40 | Zhantong Qiu       | Using LoopPoint with gem5                  |

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis

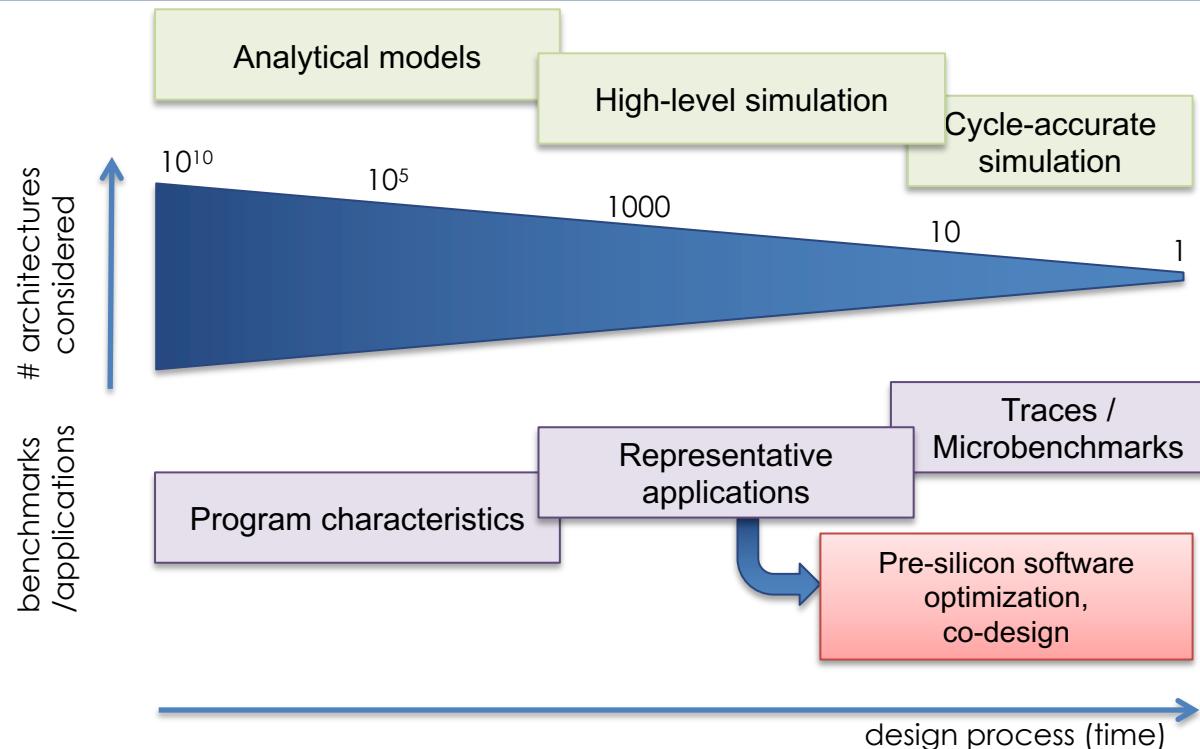


Session 4

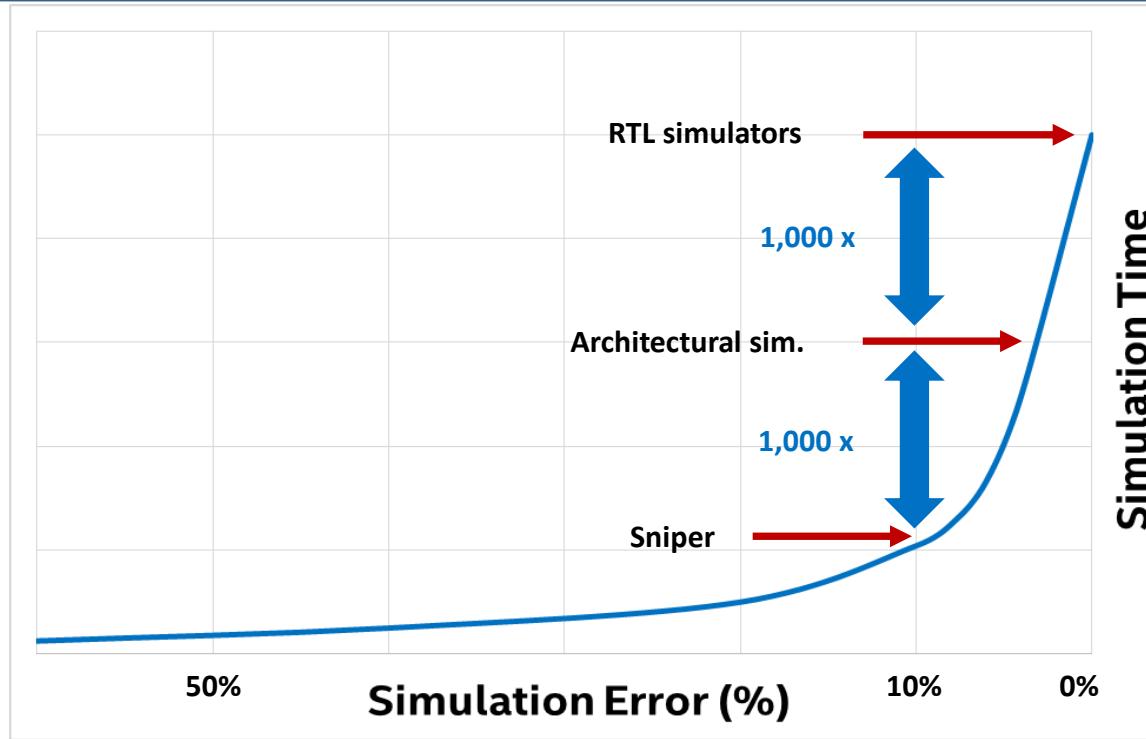
# Sniper and LoopPoint Demo

CHANGXI LIU, PHD CANDIDATE  
NATIONAL UNIVERSITY OF SINGAPORE

# The Architect's Tools – Design Waterfall



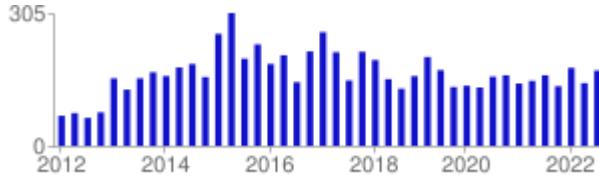
# Fast or accurate?



# Sniper History

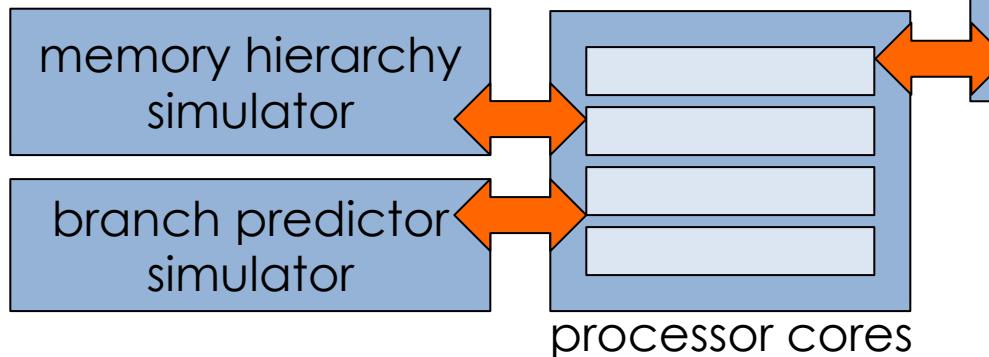
- August 2010: Sniper forked from MIT Graphite
- November 2011: SC'11 paper, first public release
- Today:
  - Interval and Instruction-window-centric core models
  - 7000+ downloads from 100+ countries
  - Active mailing list
  - 1200+ citations (SC'11 & TACO'12 papers)

snipersim.org downloads by quarter



# Simulation in Sniper

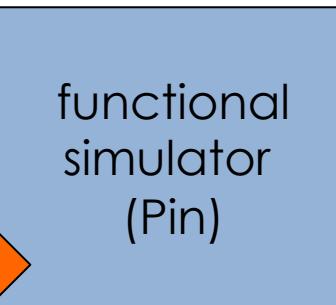
Execution-driven simulation



A single-process,  
multithreaded  
workload (v1.0)

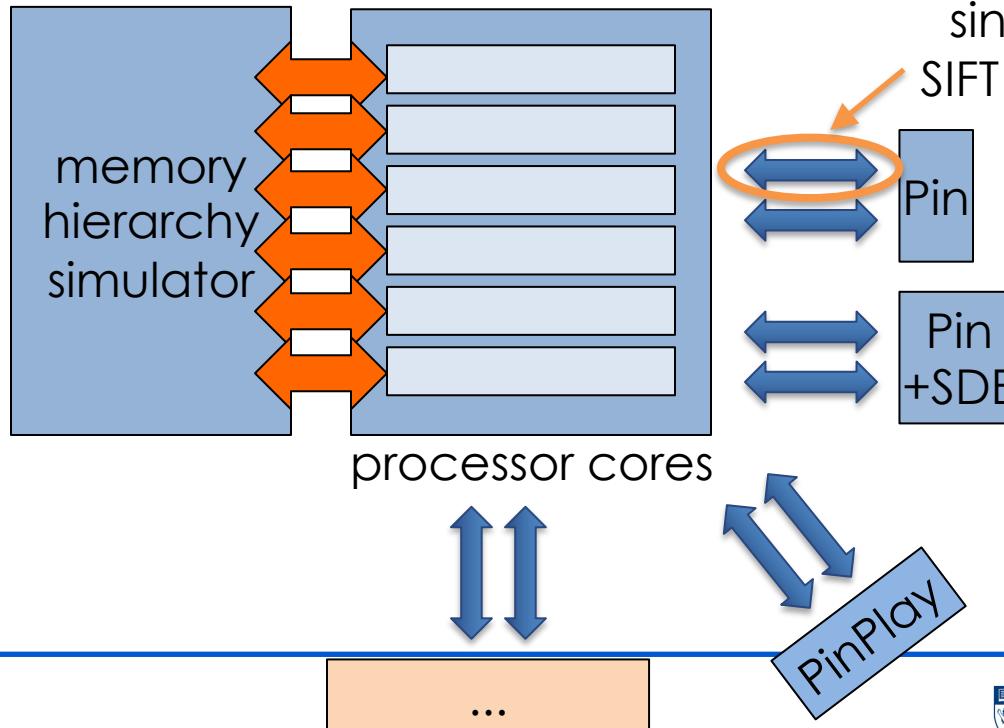
Trace-driven simulation

Multiple,  
single-threaded  
workloads (v2.0)



# Simulation in Sniper with SIFT

Functional-first simulation + timing-feedback



# Running Sniper

Configuration Region of interest markers in code **Workload command line**

```
$ run-sniper -c gainestown --roi -- ./test/fft/fft -p2
[SNIPER] Start
[SNIPER] -----
[SNIPER] Sniper using Pin frontend
[SNIPER] Running pre-ROI region in CACHE_ONLY mode
[SNIPER] Running application ROI in DETAILED mode
[SNIPER] Running post-ROI region in FAST_FORWARD mode
[SNIPER] -----
```

FFT with Blocking Transpose  
1024 Complex Doubles  
2 Processors

```
[SNIPER] Enabling performance models
[SNIPER] Setting instrumentation mode to DETAILED
[SNIPER] Disabling performance models
[SNIPER] Leaving ROI after 2.08 seconds
[SNIPER] Simulated 1.1M instructions, 0.9M cycles, 1.22 IPC
[SNIPER] Simulation speed 545.5 KIPS (272.8 KIPS / target core - 3666.2ns/instr)
[SNIPER] Setting instrumentation mode to FAST_FORWARD
```

PROCESS STATISTICS

```
...
[SNIPER] End
[SNIPER] Elapsed time: 5.97 seconds
```

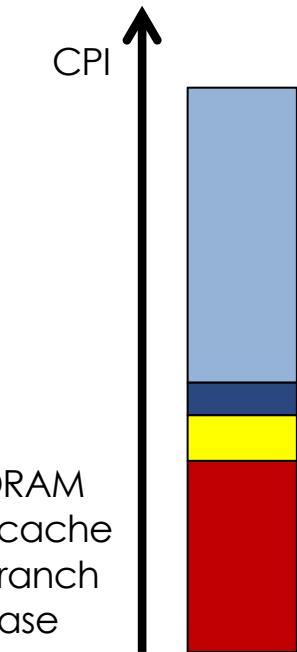
# Simulation results

sim.out: Quick overview of basic performance results

|                        | Core 0 | Core 1 |
|------------------------|--------|--------|
| Instructions           | 506505 | 505562 |
| Cycles                 | 469101 | 468620 |
| Time (ns)              | 176354 | 176173 |
| Branch predictor stats |        |        |
| num incorrect          | 1280   | 1218   |
| misprediction rate     | 7.70%  | 7.42%  |
| mpki                   | 2.53   | 2.41   |
| Cache Summary          |        |        |
| Cache L1-I             |        |        |
| num cache accesses     | 46642  | 46555  |
| num cache misses       | 217    | 178    |
| miss rate              | 0.47%  | 0.38%  |
| mpki                   | 0.43   | 0.35   |
| Cache L1-D             |        |        |
| num cache accesses     | 332771 | 332412 |
| num cache misses       | 517    | 720    |
| miss rate              | 0.16%  | 0.22%  |
| mpki                   | 1.02   | 1.42   |
| Cache L2               |        |        |
| num cache accesses     | 984    | 1090   |
| num cache misses       | 459    | 853    |

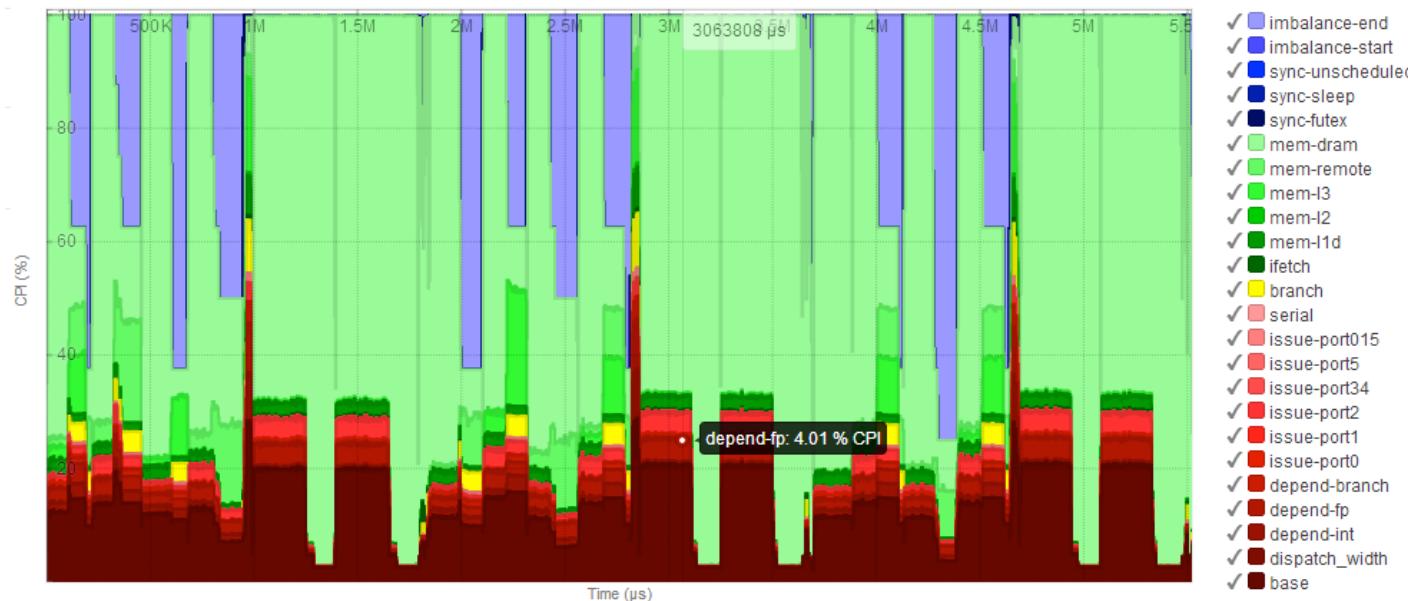
# Cycle stacks

- Where did my cycles go?
  - Cycles/time per instruction
  - Broken up in components
    - Base: ideal execution, no bottlenecks
    - Add “lost” cycles due to each HW structure
  - Normalize by either
    - Number of instructions (CPI stack)
    - Execution time (time stack)
- *Different from miss rates:*  
cycle stacks directly quantify the effect on performance
- (Also: top-down analysis in VTune)



# Advanced visualization

- Cycle stacks through time



# Downloading Sniper 8.0

- Clone from <https://github.com/snipersim/snipersim>
- `export CC=gcc-9; export CXX=g++-9`
- `make`
- Set `SNIPER_ROOT` to point to the Sniper base directory
- All set to use Sniper 8.0!
- Testing:
  - `make -C test/fft`

# Downloading LoopPoint

- Prerequisites
  - x86-based Linux machine
  - Require GCC 9
  - Python
  - Docker

# Downloading LoopPoint

- Opensource code
  - <https://github.com/nus-comparch/looppoint.git>
  - Clone the repo

```
[REDACTED] $ git clone https://github.com/nus-comparch/looppoint.git
Cloning into 'looppoint'...
remote: Enumerating objects: 320, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (141/141), done.
remote: Total 320 (delta 27), reused 148 (delta 21), pack-reused 152
Receiving objects: 100% (320/320), 15.74 MiB | 13.79 MiB/s, done.
Resolving deltas: 100% (56/56), done.
Checking connectivity... done.
[REDACTED] $ ls
looppoint
```

# Building LoopPoint

- make build
  - Build docker image

```
Created wheel for tabulate: filename=tabulate-0.8.9-py2-none-any.whl size=33171 sha256=c170d0c5148145e2deb57b20db0b76d241909980d4dcea24  
278faa8f3e0a3136  
Stored in directory: /tmp/pip-ephem-wheel-cache-5zZe7v/wheels/0a/4b/e1/d0e504a346ed0882b93f971fe1122b9de64fabebd9b1d81b9f  
Successfully built tabulate  
Installing collected packages: tabulate  
Successfully installed tabulate-0.8.9  
Removing intermediate container f962cd7c7f48  
--> fdccc13883e7  
Step 11/11 : RUN pip3 install --no-cache-dir --upgrade pip &&     pip3 install --no-cache-dir numpy  
--> Running in 89fa1a2a269a  
Collecting pip  
  Downloading https://files.pythonhosted.org/packages/a4/6d/6463d49a933f547439d6b5b98b46af8742cc03ae83543e4d7688c2420f8b/pip-21.3.1-py3-n  
one-any.whl (1.7MB)  
Installing collected packages: pip
```

Successfully built b006ee297a64  
Successfully tagged ubuntu:18.04-looppoint

```
Installing collected packages: numpy  
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is  
recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv  
Successfully installed numpy-1.19.5  
Removing intermediate container 89fa1a2a269a  
--> b006ee297a64  
[Warning] One or more build-args [TZ_ARG] were not consumed  
Successfully built b006ee297a64  
Successfully tagged ubuntu:18.04-looppoint
```

# Building LoopPoint

- make build
- make
  - Run the docker image

```
[REDACTED] looppoint (main)$ make
docker run --rm -it -v "[REDACTED]" looppoint:[REDACTED] looppoint"
--user 2014:100 -w "[REDACTED]" looppoint" ubuntu:18.04-looppoint
I have no name!@9b31dd16ef4e: [REDACTED] looppoint$ ls
Dockerfile-ubuntu-18.04 README.md lplib.py run-looppoint.py tools
Makefile           apps      preprocess suites.py
I have no name!@9b31dd16ef4e: [REDACTED] looppoint$
```

# Building LoopPoint

- make build
- make
- make apps
  - Build the demo applications
  - Source code of the apps
    - apps/demo/matrix-omp
    - apps/demo/dotproduct-omp

```
I have no name!@9b31dd16ef4e: [REDACTED] looppoint$ make apps
make -C apps/demo/matrix-omp
make[1]: Entering directory '[REDACTED] looppoint/apps/demo/matrix-omp'
g++ -g -O3 -fopenmp -o matrix-omp matrix-omp-init.cpp matrix-omp.cpp -static
/usr/lib/gcc/x86_64-linux-gnu/9/libgomp.a(target.o): In function 'gomp_target_init':
(.text+0x358): warning: Using 'dlopen' in statically linked applications requires at
runtime the shared libraries from the glibc version used for linking
ln -s matrix-omp base.exe
make[1]: Leaving directory '[REDACTED] looppoint/apps/demo/matrix-omp'
make -C apps/demo/dotproduct-omp
make[1]: Entering directory '[REDACTED] looppoint/apps/demo/dotproduct-omp'
g++ -g -O3 -fopenmp -o dotproduct-omp dot_product_openmp.cpp
ln -s dotproduct-omp base.exe
make[1]: Leaving directory '[REDACTED] looppoint/apps/demo/dotproduct-omp'
I have no name!@9b31dd16ef4e: [REDACTED] looppoint$ █
```

# Building LoopPoint

- make build
- make
- make apps
- make tools
  - Build Sniper and LoopPoint tools

Downloading  
Sniper

```
I have no name!@f3f87f6c10eb: [REDACTED]looppoint$ make tools
Downloading SDE kit
--2022-06-19 09:04:36-- https://downloadmirror.intel.com/684899/sde-external-9.0.0-2021-11-07-lin.tar.xz
Resolving downloadmirror.intel.com (downloadmirror.intel.com)... 13.33.88.124, 13.33.88.27, 13.33.88.68, ...
Connecting to downloadmirror.intel.com (downloadmirror.intel.com)|13.33.88.124|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26240092 (25M) [binary/octet-stream]
Saving to: 'STDOUT'

-          100%[=====] 25.02M 10.8MB/s   in 2.3s

2022-06-19 09:04:38 (10.8 MB/s) - written to stdout [26240092/26240092]
```

Downloading  
Intel SDE

```
Downloading Sniper from https://github.com/snipersim/snipersim
make -C tools/sniper
make[1]: Entering directory '[REDACTED]looppoint/tools/sniper'
Using SDE kit
Building for x86 (intel64)
[DOWNLO] SDE 9.0.0
[DOWNLO] pinplay-scripts
[DOWNLO] Pin 3.18-98332
[DOWNLO] mbuilder
[DOWNLO] xed
[INSTAL] xed
[PYTHON VERSION] 2.7.17
[GIT VERSION] v10.0-298-g2be2d28
[GCC VERSION] 9
.
.
.
make[4]: Entering directory '[REDACTED]looppoint/tools/sniper/sift/recorder'
make[4]: Leaving directory '[REDACTED]looppoint/tools/sniper/sift/recorder'
make[3]: Leaving directory '[REDACTED]looppoint/tools/sniper/sift/recorder'
make[2]: Leaving directory '[REDACTED]looppoint/tools/sniper/sift'
make[2]: Entering directory '[REDACTED]looppoint/tools/sniper/standalone'
[DEP ] standalone/standalone.d
[DEP ] standalone/exceptions.d
[CXX ] standalone/exceptions.o
[CXX ] standalone/standalone.o
[LD  ] lib/sniper
```

Sniper build completed

# Building LoopPoint

- Opensource code
  - <https://github.com/nus-comparch/looppoint.git>
  - Clone the repo
- LoopPoint script
  - `make build`
    - Build docker image
  - `make`
    - Run docker image
  - `make apps`
    - Build the demo applications
  - `make tools`
    - Build Sniper and LoopPoint tools

# Running LoopPoint

- Use LoopPoint driver script
  - `./run-looppoint.py -h`
  - Provides the information on how to run the tool

```
I have no name!@1fbfad8b73ce:[REDACTED] looppoint$ ./run-looppoint.py -h
Benchmarks:
demo:
dotproduct matrix

The tool runs end-to-end LoopPoint sampling methodology targeting multi-threaded applications.
Usage:
  run-looppoint.py
  [-h | --help]: Help
  [-n | --ncores=<num of threads> (8)]
  [-i | --input-class=<input class> (test)]
  [-w | --wait-policy=<comp wait policy> (passive)]
  [-p | --program=<suite-application-input> (demo-dotproduct-1]): Ex. demo-matrix-1,cpu2017-bwaves-1
  [-c | --custom-cfg=<cfg-file>]: Run a workload of interest using cfg-file in the current directory (See README.md for details)
  [--force]: Start a new set of end-to-end run
  [--reuse-profile]: Reuse the profiling data (used along with --force)
  [--reuse-fullsim]: Reuse the full program simulation (used along with --force)
  [--no-flowcontrol]: Disable thread flowcontrol during profiling
  [--use-pinplay]: Use PinPlay instead of SDE for profiling
  [--native]: Run the application natively (no sampling)
```

# Running LoopPoint

- Example run command

- `./run-looppoint.py -p demo-dotproduct-1 -n 8 --force`

```
I have no name!@1fbfad8b73ce: [REDACTED] looppoint$ ./run-looppoint.py -p demo-dotproduct-1 -n 8 --force
[LOOPPOINT] Generating fat pinball.
[PREPROCESS] dotproduct-omp
[PREPROCESS] apps/demo/dotproduct-omp/dotproduct-omp
[PREPROCESS] [REDACTED] looppoint/apps/demo/dotproduct-omp/dotproduct-omp
[PREPROCESS] symlinkng dotproduct-omp /tmp/tmpcdMI_d/base.exe
[PREPROCESS] apps/demo/dotproduct-omp/test
[PREPROCESS] [REDACTED] looppoint/apps/demo/dotproduct-omp/test
[PREPROCESS] symlinkng [REDACTED] looppoint/apps/demo/dotproduct-omp/test/dotproduct-omp.1.cfg /tmp/tmp
cdMI_d/dotproduct-omp.1.cfg
[PREPROCESS] Done
*** TRACING: START *** June 19, 2022 10:03:27
Script version $Revision:1.128$
Script: sde_pinpoints.py
Script args: --delete --mode mt --sdehome=[REDACTED] looppoint/tools/sde-external-9.0.0-20
21-11-07-lin --cfg [REDACTED] looppoint/apps/demo/dotproduct-omp/test/dotproduct-omp.1.cfg --log_options
-start_address main -log:fat -log:mp_atomic 0 -log:mp_mode 0 -log:strace -log:basename [REDACTED] loop
point/results/demo-dotproduct-1-test-passive-8-20220619100327/whole_program.1/dotproduct.1 --replay_options=repl
ay:strace -l
```

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
    - `make_mt_pinball` : Generate whole-program pinball
    - `gen_dcfg` : Generate DCFG file to identify loop information
    - `gen_bbv` : Generate feature vector of each region
    - `gen_cluster` : Cluster regions

# Fat Pinball

- Makes Pin-based analyses repeatable.
- Command:
  - `$SDE_KIT/pinplay-scripts/sde_pinpoints.py --mode mt --cfg=$CFGFILE --log_options="-start_address main -log:fat -log:basename $WPP_BASE" --replay_options="-replay:strace" -l`
- Generates a whole-program pinball for further profiling steps

# DCFG Generation

- A dynamic control-flow graph (DCFG) is a specialized control-flow graph that adds data from a specific execution of a program
- C++ DCFG APIs available for accessing the data
  - `DCFG_LOOP_CONTAINER::get_loop_ids`
    - Get the set of loop IDs
  - `DCFG_LOOP`
    - `get_routine_id` : get the function that the loop belongs to
    - `get_parent_loop_id` : get the parent loop

# DCFG Generation

- A dynamic control-flow graph (DCFG) is a specialized control-flow graph that adds data from a specific execution of a program
- C++ DCFG APIs available for accessing the data.
- More APIs can be found in
  - `tools/sde-external-9.0.0-2021-11-07-lin/pinkit/sde-example/include`
    - `dcfg_api.H`
    - `dcfg_pin_api.H`
    - `dcfg_trace_api.H`

# DCFG Generation

- Collect Loop Information
- Command:
  - `$SDE_BUILD_KIT/pinplay-scripts/replay.py --pintool=sde-global-looppoint.so --pintool_options "-dcfg -replay:deadlock_timeout 0 -replay:strace -dcfg:out_base_name $DCFG_BASE $WPP_BASE"`
  - `-dcfg` : enable DCFG generation
  - `DCFG_BASE` : the basename of DCFG that is generated

# BBV Generation

- Profiling the feature vector of each region
- Command:
  - `$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pintool="sde-global-looppoint.so" --global_regions --pccount_regions --cfg $CFG --whole_pgm_dir $WPP_DIR --mode mt -S $SLICESIZE -b --replay_options "-replay:deadlock_timeout 0 -global_profile -emit_vectors 0 -filter_exclude_lib libgomp.so.1 -filter_exclude_lib libiomp5.so -looppoint:global_profile -looppoint:dcfg-file $DCFG -looppoint:main_image_only 1 -looppoint:loop_info $PROGRAM.$INPUT.loop_info.txt -flowcontrol:verbose 1 -flowcontrol:quantum 1000000 -flowcontrol:maxthreads $NC0RES"`
  - `--pccount_regions` : (PC, count)-based region information
  - `-S $SLICESIZE`: The *global* instruction count for each region
  - `-filter_exclude_lib`: Exclude libraries from profiling information

# BBV Generation

- Profiling the feature vector of each region
- Command:
  - `$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pintool="sde-global-looppoint.so" --global_regions --pccount_regions --cfg $CFG --whole_pgm_dir $WPP_DIR --mode mt -S $SLICESIZE -b --replay_options "-replay:deadlock_timeout 0 -global_profile -emit_vectors 0 -filter_exclude_lib libgomp.so.1 -filter_exclude_lib libiomp5.so -looppoint:global_profile -looppoint:dcfg-file $DCFG -looppoint:main_image_only 1 -looppoint:loop_info $PROGRAM.$INPUT.loop_info.txt -flowcontrol:verbose 1 -flowcontrol:quantum 1000000 -flowcontrol:maxthreads $NCORES"`
  - `-looppoint:main_image_only`: Select only main image for choosing markers
  - `-looppoint:loop_info` : Utilize loop information as the marker of each region
  - `-flowcontrol:quantum` : synchronize each thread every `1000000` instructions

# Clustering

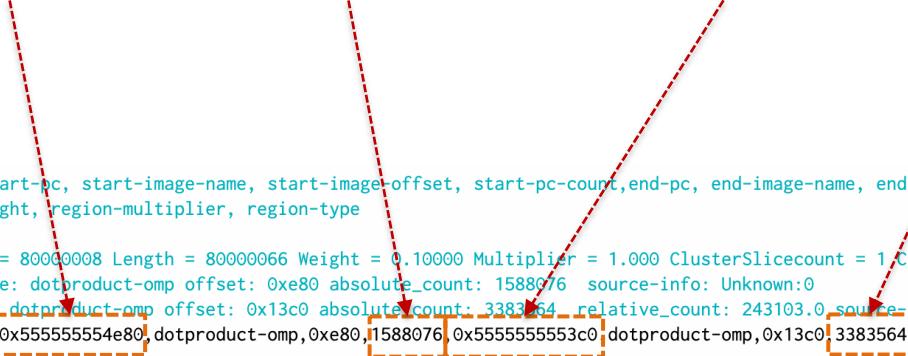
- Cluster all regions into several groups.
  - SimPoint [1]
  - Utilize feature vectors of all threads
  - kmeans algorithm

# Clustering

- Cluster all regions into several groups.
- Command
  - `$SDE_BUILD_KIT/pinplay-scripts/sde_pinpoints.py --pintool="sde-global-looppoint.so" --cfg $CFG --whole_pgm_dir $WPP_DIR -S $SLICESIZE --warmup_factor=2 --maxk=$MAXK --append_status -s --simpoint_options="-dim $DIM -coveragePct 1.0 -maxK $MAXK"`
  - **DIM** : The reduced dimension of the vector that BBVs are projected to
  - **MAXK** : Maximum number of clusters for kmeans

# Running LoopPoint

- The LoopPoint driver script
  - Profiling Results:
    - dotproduct.1\_52.global.pinpoints.csv
    - (start-pc, start-pc-count ), (end-pc, end-pc-count )



```
# comment,thread-id,region-id,start-pc, start-image-name, start-image-offset, start-pc-count,end-pc, end-image-name, end-image-offset, end-pc-count,end-pc-relative-count, region-length, region-weight, region-multiplier, region-type

# RegionId = 1 Slice = 1 Icount = 80000008 Length = 80000066 Weight = 0.10000 Multiplier = 1.000 ClusterSliceCount = 1 ClusterIcount = 80000066
#Start: pc : 0x555555554e80 image: dotproduct-omp offset: 0xe80 absolute_count: 1588076 source_info: Unknown:0
#End: pc : 0x555555553c0 image: dotproduct-omp offset: 0x13c0 absolute_count: 3383564 relative_count: 243103.0 source_info: Unknown:0
cluster 0 from slice 1,global,1 [0x555555554e80] dotproduct-omp,0xe80,1588076,[0x555555553c0] dotproduct-omp,0x13c0,3383564,243103,80000066,0.10000,1.000,simulation
```

# Running LoopPoint

- The LoopPoint driver script
  - Profiling Results:
    - dotproduct.1\_52.global.pinpoints.csv
    - (start-pc, start-pc-count ), (end-pc, end-pc-count)
    - Cluster group id

```
# comment,thread-id,region-id,start-pc, start-image-name, start-image-offset, start-pc-count,end-pc, end-image-name, end-image-offset, end-pc-count,end-pc-relative-count, region-length, region-weight, region-multiplier, region-type

# RegionId = 1 Slice = 1 Icount = 80000008 Length = 80000066 Weight = 0.10000 Multiplier = 1.000 ClusterSliceCount = 1 ClusterIcount = 80000066
#Start: pc : 0x555555554e80 image: dotproduct-omp offset: 0xe80 absolute_count: 1588076 source-info: Unknown:0
#End: pc : 0x5555555553c0 image: dotproduct-omp offset: 0x13c0 absolute_count: 3383564 relative_count: 243103.0 source-info: Unknown:0
cluster 0 from slice 1,global,1,0x555555554e80,dotproduct-omp,0xe80,1588076,0x5555555553c0,dotproduct-omp,0x13c0,3383564,243103,80000066,0.10000,1.000,simulation
```

# Running LoopPoint

- The LoopPoint driver script
  - Profiling Results:
    - dotproduct.1\_52.global.pinpoints.csv
    - (start-pc, start-pc-count ), (end-pc, end-pc-count)
    - Cluster group id
    - Cluster multiplier

```
# comment,thread-id,region-id,start-pc, start-image-name, start-image-offset, start-pc-count,end-pc, end-image-name, end-image-offset, end-pc-count,end-pc-relative-count, region-length, region-weight, region-multiplier, region-type

# RegionId = 1 Slice = 1 Icount = 80000008 Length = 80000066 Weight = 0.10000 Multiplier = 1.000 ClusterSliceCount = 1 ClusterIcount = 80000066
#Start: pc : 0x555555554e80 image: dotproduct-omp offset: 0xe80 absolute_count: 1588076 source-info: Unknown:0
#End: pc : 0x5555555553c0 image: dotproduct-omp offset: 0x13c0 absolute_count: 3383564 relative_count: 243103.0 source-info: Unknown:0
cluster 0 from slice 1,global,1,0x555555554e80,dotproduct-omp,0xe80,1588076,0x5555555553c0,dotproduct-omp,0x13c0,3383564,243103,80000066,0.10000,1.000,simulation
```

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
    - dotproduct.1\_52.global.pinpoints.csv
    - Sampled Simulation : (start-pc, start-pc-count ), (end-pc, end-pc-count), cluster group id
    - Extrapolation : cluster group id, cluster-multiplier

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
  - Sampled simulation of selected regions

# Simulation using Sniper

- LoopPoint support in Sniper 8.0
- Handle the beginning and ending of representative regions
  - Using PC-based markers
  - Sniper shifts simulation modes based on signals from Pin/SDE

# Simulation using Sniper

- LoopPoint support in Sniper 8.0
  - Handle the beginning and ending of representative regions
  - `./run-sniper -n 8 -gscheduler/type=static -cgainestown -ssimuserroi --roi-script --trace-args=-pinplay:control start:address:<PC>:count<Count>:global --trace-args=-pinplay:control stop:address:<PC>:count<Count>:global -- <app cmd>`
  - Region start: `-control start:address:<PC>:count<Count>`
  - Region end: `-control end:address:<PC>:count<Count>`
  - *PC, Count* : LoopPoint region boundaries
  - **Note:** Use `-control` if SDE is used instead of Pin/Pinplay

# Simulation using Sniper

Start PC and count

```
./run-sniper -n 8 -v -sprogresstrace:10000000 -gtraceinput/timeout=2000 -gscheduler/type=static -  
cgainestown --trace-args=-sniper:flow 1000 -ssimuserroi --roi-script --trace-args=-control start:address:  
0x5555555553c0:count8095299:global --trace-args=-control stop:address:0x5555555553c0:count16984191:global -  
gperf_model/fast_forward/oneipc/interval=100 -ggeneral/inst_mode_init=detailed -gperf_model/fast_forward/oneipc/  
include_memory_latency=true -- ./base.exe
```

Application

End PC and count

# Simulation using Sniper

Warmup  
ends

```
[PROGRESS] 700M instructions, 3198 KIPS, 2.37 IPC
[PROGRESS] 710M instructions, 6004 KIPS, 8.00 IPC
[PROGRESS] 720M instructions, 5526 KIPS, 8.00 IPC
[CONTROLLER] tid: 5 ip: 0x0000555555553e2 658579928 Start
[SNIPER] Enabling performance models
[PROGRESS] 730M instructions, 608 KIPS, 1.97 IPC
[PROGRESS] 740M instructions, 469 KIPS, 1.61 IPC
[PROGRESS] 750M instructions, 455 KIPS, 1.61 IPC
[PROGRESS] 760M instructions, 447 KIPS, 1.61 IPC
[PROGRESS] 770M instructions, 447 KIPS, 1.61 IPC
[PROGRESS] 780M instructions, 446 KIPS, 1.61 IPC
[PROGRESS] 790M instructions, 446 KIPS, 1.61 IPC
[PROGRESS] 800M instructions, 448 KIPS, 1.61 IPC
[CONTROLLER] tid: 4 ip: 0x0000555555553e2 669005339 Stop
[SNIPER] Disabling performance models
[SNIPER] Leaving ROI after 176.54 seconds
[SNIPER] Simulated 80.0M instructions, 708.4M cycles, 0.11 IPC
[SNIPER] Simulation speed 453.2 KIPS (56.6 KIPS / target core - 17654.0ns/instr)
[SNIPER] Sampling: executed 7.03% of simulated time in detailed mode
[SNIPER] Setting instrumentation mode to FAST_FORWARD
[PROGRESS] 810M instructions, 1918 KIPS, 4.23 IPC
```

Detailed simulation

Fast-forwarding  
the rest

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
  - Sampled simulation of selected regions
  - Extrapolation of performance results

# Extrapolation of Performance Result

- Runtime of corresponding representative region : `region_runtime`
- Scaling factor : `multiplier`

```
for regionid, multiplier in region_mult.iteritems():
    region_runtime = 0
    try:
        region_runtime = read_simstats(region_stats[regionid], region_config[regionid], 'runtime')
    except:
        print('[LOOPPOINT] Warning: Skipping r%s as the simulation results are not available' % regionid)
        continue
    cov_mult += multiplier
    extrapolated_runtime += region_runtime * multiplier
    if region_runtime > max_rep_runtime:
        max_rep_runtime = region_runtime
    sum_rep_runtime += region_runtime
```

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
  - Sampled simulation of selected regions
  - Extrapolation of performance results
    - Predicted runtime using sampled simulation

| application      | runtime<br>actual (ns) | runtime<br>predicted (ns) | error<br>(%) | speedup<br>(parallel) | speedup<br>(serial) | coverage<br>(%) |
|------------------|------------------------|---------------------------|--------------|-----------------------|---------------------|-----------------|
| dotproduct-omp.1 | 592169300.0            | 414953200.0               | 29.93        | 5.86                  | 1.68                | 100.0           |

# Running LoopPoint

- The LoopPoint driver script
  - Profiling the application
  - Sampled simulation of selected regions
  - Extrapolation of performance results
    - Predicted runtime using sampled simulation
    - The error rate of obtained using sampled simulation

| application      | runtime<br>actual (ns) | runtime<br>predicted (ns) | error<br>(%) | speedup<br>(parallel) | speedup<br>(serial) | coverage<br>(%) |
|------------------|------------------------|---------------------------|--------------|-----------------------|---------------------|-----------------|
| dotproduct-omp.1 | 592169300.0            | 414953200.0               | 29.93        | 5.86                  | 1.68                | 100.0           |

# Running Custom Workloads

- Create a config file in the application directory (format as below)

[Parameters]

program\_name: bwaves-s

input\_name: 1

command: ./speed\_bwaves.icc18.0.g02avx bwaves\_1 < bwaves\_1.in

**603.bwaves\_s.1.cfg**

Config file

- Run command:

```
$L0OPPOINT_R0OT/run-looppoint.py -c 603.bwaves_s.1.cfg -w active -n 8 --force
```

[LOOPPOINT] Collecting results for bwaves-s.1 using test input class and active OMP wait policy

| application | runtime<br>actual (ns) | runtime<br>predicted (ns) | error<br>(%) | speedup<br>(parallel) | speedup<br>(serial) | coverage<br>(%) |
|-------------|------------------------|---------------------------|--------------|-----------------------|---------------------|-----------------|
| bwaves-s.1  | 2108671000.0           | 2055671101.13             | 2.51         | 11.65                 | 1.03                | 100.0           |

# Agenda

| Time (Eastern) | Speaker            | Topic                                      |
|----------------|--------------------|--------------------------------------------|
| 13.20 to 13.30 | Trevor E. Carlson  | Overview of the tutorial                   |
| 13.30 to 14.20 | Akanksha Chaudhari | Performance analysis, simulation, sampling |
| 14.20 to 15.20 | Harish Patil       | Using tools: Pin, PinPlay, SDE, ELFies     |
| 15.20 to 15.40 |                    | Break                                      |
| 15.40 to 16.20 | Alen Sabu          | Multi-threaded sampling and LoopPoint      |
| 16.20 to 17.00 | Changxi Liu        | Sniper and LoopPoint demo                  |
| 17.00 to 17.40 | Zhantong Qiu       | Using LoopPoint with gem5                  |

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis



Session 5

# Using LoopPoint with gem5

ZHANTONG QIU, UNDERGRADUATE STUDENT  
UNIVERSITY OF CALIFORNIA, DAVIS

# Quick background on gem5

Main difference between gem5 and Sniper?

gem5 is an execute-in-execute simulator

Two “modes:”

Full-system: boots a Linux kernel, requires disk image, etc.

Syscall emulation: “Fakes” the Linux system calls in gem5

We will be using **syscall emulation (SE) mode**

gem5 is a *python interpreter* which configures and controls simulation

We will show the python code needed to set up LoopPoints/ELFies



# How to perform LoopPoint sampling in gem5?

- **Our implementation focuses on using the checkpoint methodology**
  - We take a checkpoint at the beginning of the selected region with a fast and simple architecture setup, and restore the checkpoints with the desired architecture.
- **The LoopPoint module in gem5 is designed to use with the gem5 standard library**
  - The gem5 standard library provides flexible and convenience modules for simulation setups.
- **In gem5, we use checkpoint to save the state of the simulation. It allows us to restore and simulate a particular region of the whole simulation with different architectures.**



# A small example of what can and can not change when restoring a gem5 checkpoint

When taking checkpoints:

```
86 # When taking a checkpoint, the cache state is not saved, so the cache
87 # hierarchy can be changed completely when restoring from a checkpoint.
88 # By using NoCache() to take checkpoints, it can slightly improve the
89 # performance when running in atomic mode, and it will not put any restrictions
90 # on what people can do with the checkpoints.
91 cache_hierarchy = NoCache()
92
93 # Using simple memory to take checkpoints might slightly imporve the
94 # performance in atomic mode. The memory structure can be changed when
95 # restoring from a checkpoint, but the size of the memory must be maintained.
96 memory = SingleChannelDDR3_1600(size="2GB")
```

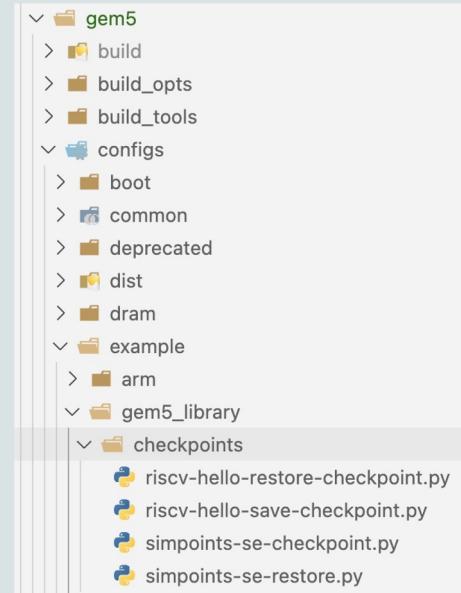
When restoring a checkpoint:

```
75 # The cache hierarchy can be different from the cache hierarchy used in taking
76 # the checkpoints
77 cache_hierarchy = PrivateL1PrivateL2CacheHierarchy(
78     l1d_size="32kB",
79     l1i_size="32kB",
80     l2_size="256kB",
81 )
82
83 # The memory structure can be different from the memory structure used in
84 # taking the checkpoints, but the size of the memory must be maintained
85 memory = DualChannelDDR4_2400(size="2GB")
```

A tutorial on checkpointing in gem5 was given as a part of the gem5 2022 Bootcamp. A recording of this event can be found within this link:

<https://gem5bootcamp.github.io/gem5-bootcamp-env/modules/extra%20topics/checkpointing-commmonitor>

You can find example scripts of taking checkpoints in the gem5 directory:



# How to take checkpoints for LoopPoint sampling?

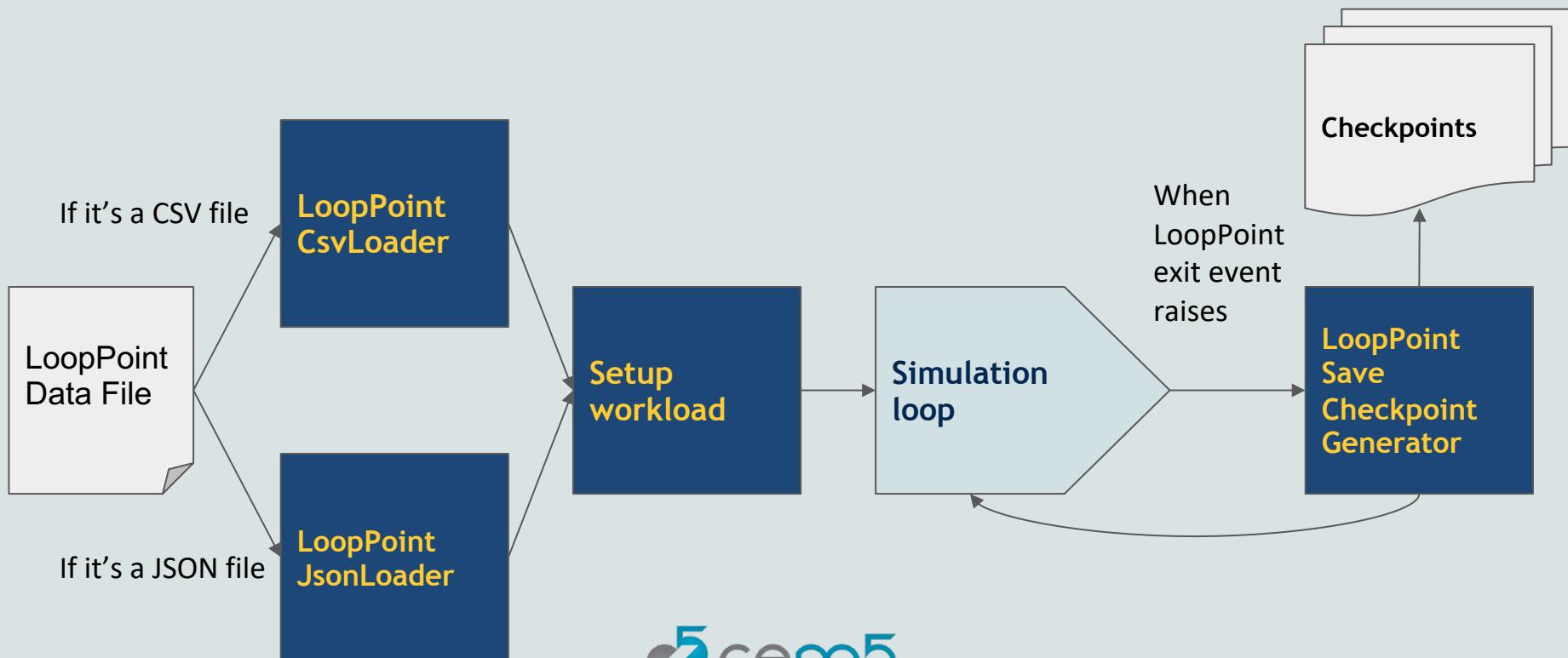


# The LoopPoint JSON file

```
region id
  __ simulation
    |__ start
      |__ pc
      |__ global
      |__ relative
    |__ end
      |__ pc
      |__ global
      |__ relative
  __ warmup # optional to region
    |__ start
      |__ pc
      |__ global
      |__ relative
    |__ end
      |__ pc
      |__ global
      |__ relative
  __ multiplier
```

```
"1": {
  "simulation": {
    "start": {
      "pc": 4221392,
      "global": 211076617,
      "relative": 15326617
    },
    "end": {
      "pc": 4221392,
      "global": 219060252,
      "relative": 23310252
    }
  },
  "multiplier": 4.0,
  "warmup": {
    "start": {
      "pc": 4221056,
      "count": 23520614
    },
    "end": {
      "pc": 4221392,
      "count": 211076617
    }
  }
},
"2": {
  "simulation": {
    "start": {
      "pc": 4206672,
      "global": 1
    },
    "end": {
      "pc": 4221392,
      "global": 6861604,
      "relative": 6861604
    }
  },
  "multiplier": 1.0
}
```

# How to take checkpoints for LoopPoint sampling?



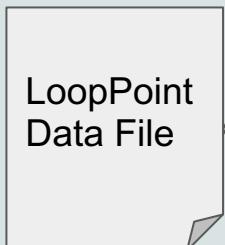
# A config script with a simple architecture

```
1  from gem5.simulate.exit_event import ExitEvent
2  from gem5.simulate.simulator import Simulator
3  from gem5.utils.requires import requires
4  from gem5.components.cachehierarchies.classic.no_cache import NoCache
5  from gem5.components.boards.simple_board import SimpleBoard
6  from gem5.components.memory.single_channel import SingleChannelDDR3_1600
7  from gem5.components.processors.simple_processor import SimpleProcessor
8  from gem5.components.processors.cpu_types import CPUTypes
9  from gem5.isas import ISA
10 from gem5.resources.resource import obtain_resource
11 from pathlib import Path
12
13 requires(isa_required=ISA.X86)
14 cache_hierarchy = NoCache()
15 memory = SingleChannelDDR3_1600(size="2GB")
16 processor = SimpleProcessor(
17     cpu_type=CPUTypes.ATOMIC,
18     isa=ISA.X86,
19     num_cores=9,
20 )
```

```
21 board = SimpleBoard(
22     clk_freq="3GHz",
23     processor=processor,
24     memory=memory,
25     cache_hierarchy=cache_hierarchy,
26 )
27 board.set_se_binary_workload(
28     binary=obtain_resource("x86-matrix-multiply-omp"),
29     # In here, we use the gem5 resource to obtain the binary
30     # we can also input the local path to the binary, i.e.
31     # binary=Path("path/to/binary")
32     arguments=[100, 8]
33 )
34
35 simulator = Simulator(
36     board=board
37 )
38 simulator.run()
```



If it's a CSV file



If it's a JSON file

```
41 from gem5.resources.looppoint import LooppointCreatorCSV  
  
53 looppoint = LooppointCreatorCSV(  
54     # Pass in the LoopPoint data file  
55     pinpoints_file=Path(  
56         obtain_resource(  
57             "x86-matrix-multiply-omp-100-8-global-pinpoints"  
58         ).get_local_path()  
59     )  
60 )
```

OR

```
54 from gem5.resources.looppoint import LooppointJsonLoader  
  
120 looppoint = LooppointJsonLoader(  
121     looppoint_file=Path(  
122         obtain_resource(  
123             "x86-matrix-multiply-omp-100-8-looppoint"  
124         ).get_local_path()  
125     )  
126 )  
127
```



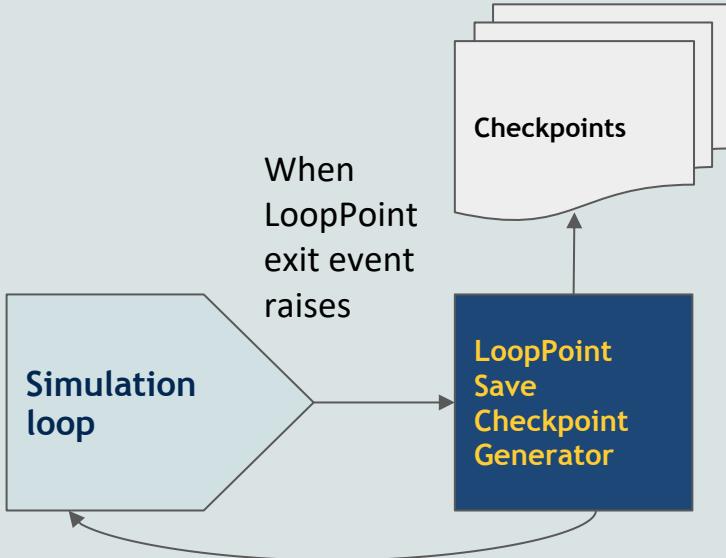
LoopPoint  
CsvLoader

LoopPoint  
JsonLoader

Setup  
workload

```
69 board.set_se_looppoint_workload(  
70     binary=obtain_resource("x86-matrix-multiply-omp"),  
71     arguments=[100, 8],  
72     # Pass LoopPoint module into the board  
73     looppoint=looppoint,  
74 )
```

When  
LoopPoint  
exit event  
raises



```
38     from gem5.simulate.exit_event_generators import (
39         looppoint_save_checkpoint_generator,
40     )
```

```
76     # name the path of where the checkpoints should be saved
77     dir = Path("looppoint_checkpoints_folder")
78     dir.mkdir(exist_ok=True)
79
80     simulator = Simulator(
81         board=board,
82         on_exit_event={
83             ExitEvent.SIMPOINT_BEGIN: looppoint_save_checkpoint_generator(
84                 checkpoint_dir=dir,
85                 looppoint=looppoint,
86                 # True if the relative PC count pairs should be updated during the
87                 # simulation. Default as True.
88                 update_relatives=True,
89                 # True if the simulation loop should exit after all the PC count
90                 # pairs in the LoopPoint data file have been encountered. Default
91                 # as True.
92                 exit_when_empty=True,
93             ),
94         },
95     )
```

```
97     simulator.run()
98
99     # Output the JSON file
100    looppoint.output_json_file()
```



```
27 from gem5.simulate.exit_event import ExitEvent
28 from gem5.simulate.simulator import Simulator
29 from gem5.utils.requires import requires
30 from gem5.components.cachehierarchies.classic.no_cache import NoCache
31 from gem5.components.boards.simple_board import SimpleBoard
32 from gem5.components.memory.single_channel import SingleChannelDDR3_1600
33 from gem5.components.processors.simple_processor import SimpleProcessor
34 from gem5.components.processors.cpu_types import CPUTypes
35 from gem5.isas import ISA
36 from gem5.resources.resource import obtain_resource
37 from pathlib import Path
38 from gem5.simulate.exit_event_generators import (
39     looppoint_save_checkpoint_generator,
40 )
41 from gem5.resources.looppoint import LooppointCreatorCSV
42
43 requires(isa_required=ISA.X86)
44
45 cache_hierarchy = NoCache()
46 memory = SingleChannelDDR3_1600(size="2GB")
47 processor = SimpleProcessor(
48     cpu_type=CPUTypes.ATOMIC,
49     isa=ISA.X86,
50     num_cores=9,
51 )
52
53 looppoint = LooppointCreatorCSV(
54     # Pass in the LoopPoint data file
55     pinpoints_file=Path(
56         obtain_resource(
57             "x86-matrix-multiply-omp-100-8-global-pinpoints"
58         ).get_local_path()
59     )
60 )
61
62 board = SimpleBoard(
63     clk_freq="3GHz",
64     processor=processor,
65     memory=memory,
66     cache_hierarchy=cache_hierarchy,
67 )
68
69 board.set_se_looppoint_workload(
70     binary=obtain_resource("x86-matrix-multiply-omp"),
71     arguments=[100, 8],
72     # Pass LoopPoint module into the board
73     looppoint=looppoint,
74 )
75
76 # name the path of where the checkpoints should be saved
77 dir = Path("looppoint_checkpoints_folder")
78 dir.mkdir(exist_ok=True)
79
80 simulator = Simulator(
81     board=board,
82     on_exit_event={
83         ExitEvent.SIMPOINT_BEGIN: looppoint_save_checkpoint_generator(
84             checkpoint_dir=dir,
85             looppoint=looppoint,
86             # True if the relative PC count pairs should be updated during the
87             # simulation. Default as True.
88             update_relatives=True,
89             # True if the simulation loop should exit after all the PC count
90             # pairs in the LoopPoint data file have been encountered. Default
91             # as True.
92             exit_when_empty=True,
93         )
94     },
95 )
96
97 simulator.run()
98
99 # Output the JSON file
100 looppoint.output_json_file()
```



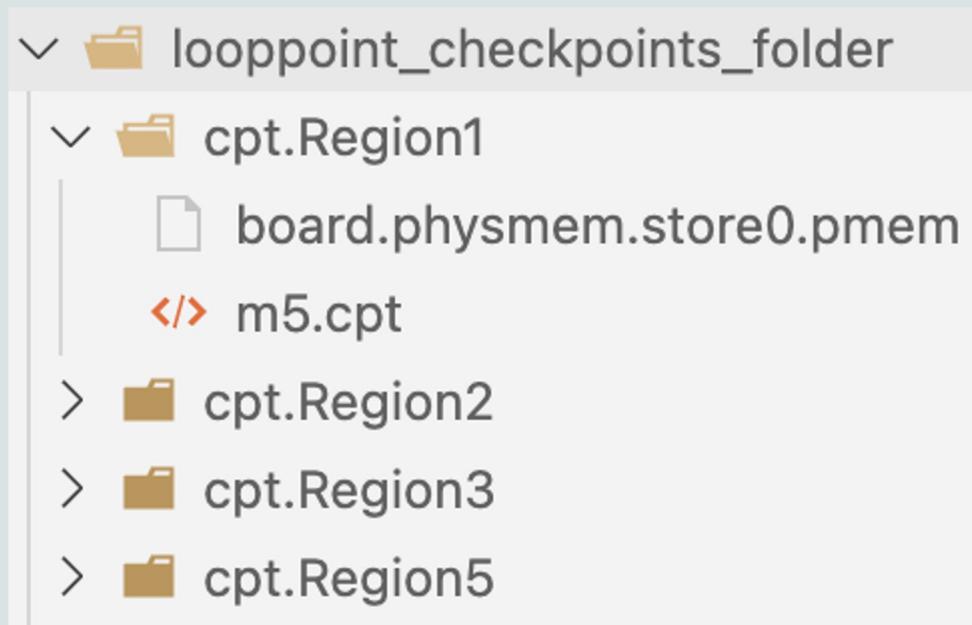
Example command to run the checkpoint script:

```
build/X86/gem5.opt create-checkpoints.py
```

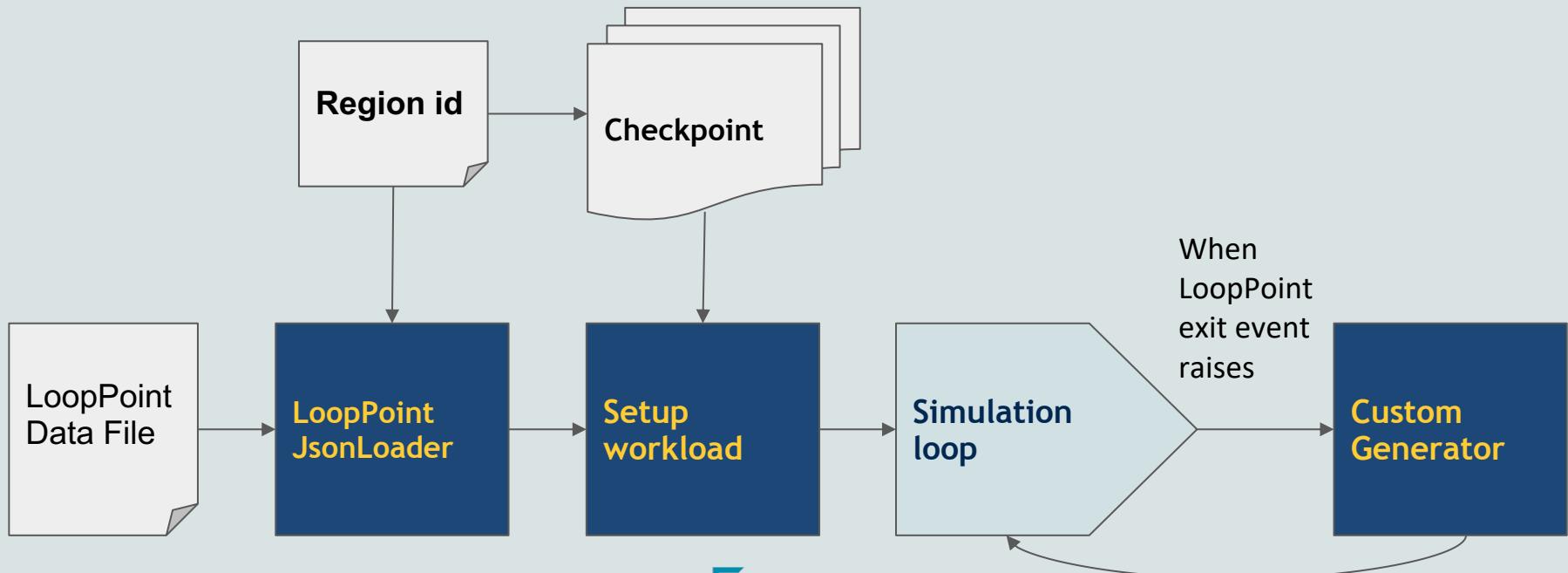
Example output

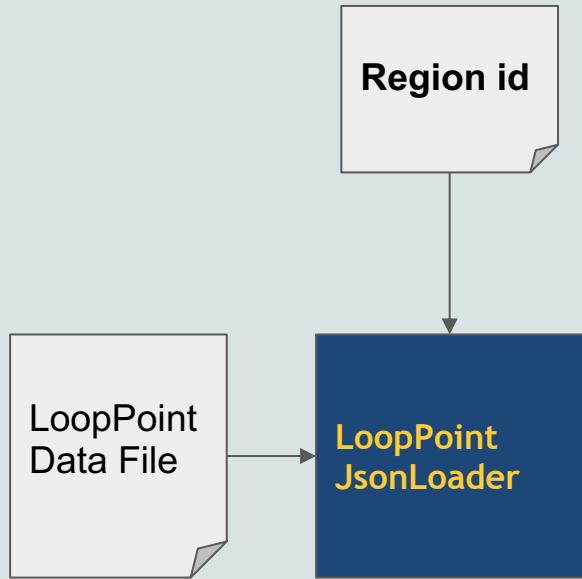
```
58 Writing checkpoint
59 build/X86/sim/simulate.cc:195: info: Entering event queue @ 138524528475. Starting simulation...
60 build/X86/sim/simulate.cc:195: info: Entering event queue @ 154979540325. Starting simulation...
61 build/X86/sim/simulate.cc:195: info: Entering event queue @ 163287976905. Starting simulation...
62 build/X86/sim/simulate.cc:195: info: Entering event queue @ 171528530436. Starting simulation...
63 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
64 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
65 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
66 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
67 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
68 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
69 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
70 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
71 Writing checkpoint
72 build/X86/sim/simulate.cc:195: info: Entering event queue @ 171528531102. Starting simulation...
73 build/X86/sim/simulate.cc:195: info: Entering event queue @ 187980745752. Starting simulation...
74 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
75 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
76 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
77 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
78 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
79 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
80 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
81 build/X86/sim/process.cc:382: warn: Checkpoints for pipes, device drivers and sockets do not work.
```

## Example checkpoints:

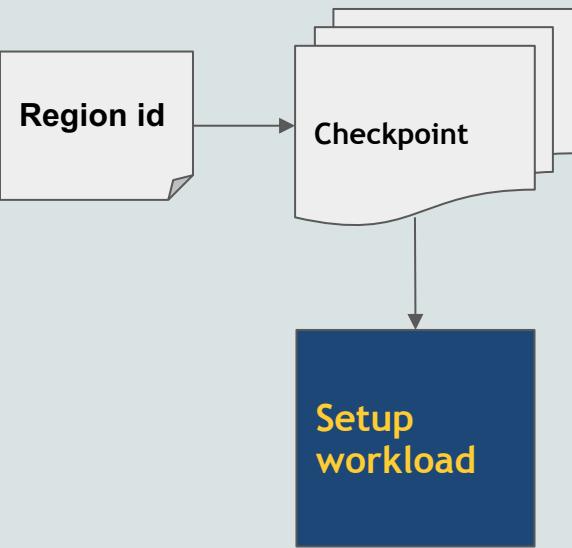


The process is similar for restoring a checkpoint

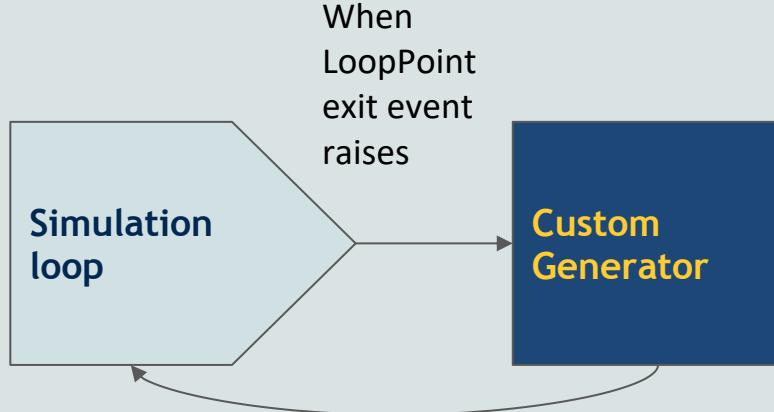




```
54     from gem5.resources.looppoint import LooppointJsonLoader  
  
107     looppoint = LooppointJsonLoader(  
108         looppoint_file=Path(  
109             obtain_resource(  
110                 "x86-matrix-multiply-omp-100-8-looppoint"  
111             ).get_local_path()  
112         ),  
113         # pass in the id of the region you want to restore  
114         # i.e. region_id=1  
115         region_id=args.checkpoint_region,  
116     )
```



```
118 board.set_se_looppoint_workload(  
119     binary=obtain_resource("x86-matrix-multiply-omp"),  
120     looppoint=looppoint,  
121     # Pass in the path to the checkpoint you want to restore  
122     # Each simulation can only restore one checkpoint  
123     # i.e. checkpoint_path=Path("looppoint_checkpoints_folder/cpt.Region1")  
124     checkpoint=obtain_resource(  
125         f"x86-matrix-multiply-omp-100-8-looppoint-checkpoint-region-{args.checkpoint_region}"  
126     ).get_local_path()  
127 )  
---
```



```

129 # This generator will dump the stats and exit the simulation loop when the
130 # simulation region reaches its end. In the case there is a warmup interval,
131 # the simulation stats are reset after the warmup is complete.
132 def reset_and_dump():
133     if len(looppoint.get_targets()) > 1:
134         print("Warmup region ended. Resetting stats.")
135         reset()
136         yield False
137     print("Region ended. Dumping stats.")
138     dump()
139     yield True
140

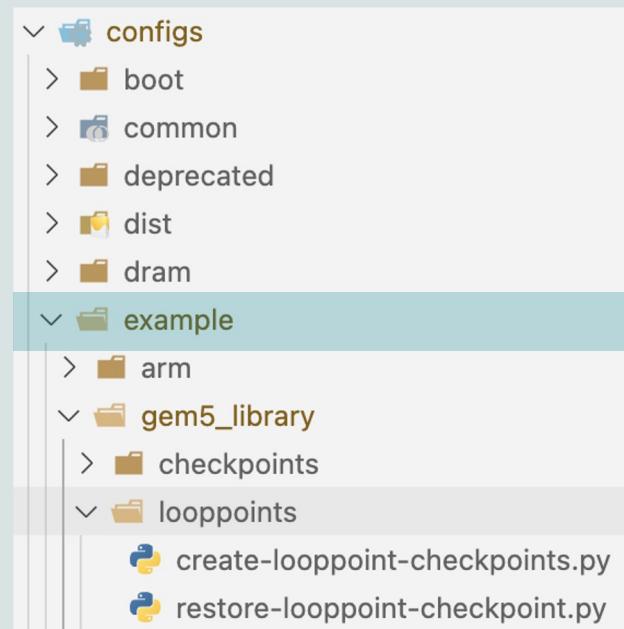
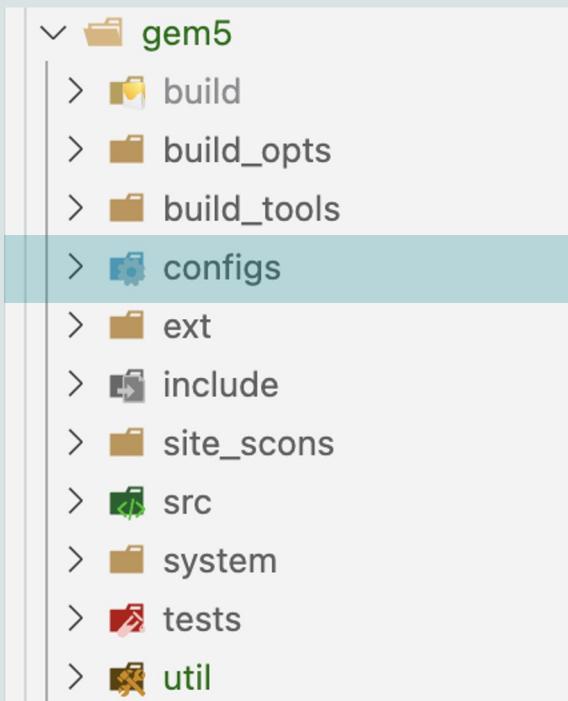
```

```

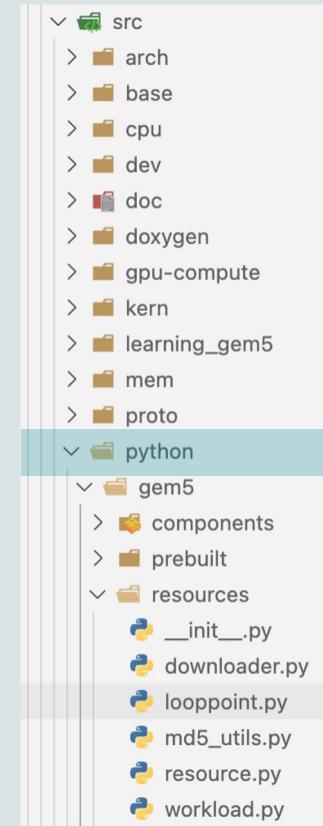
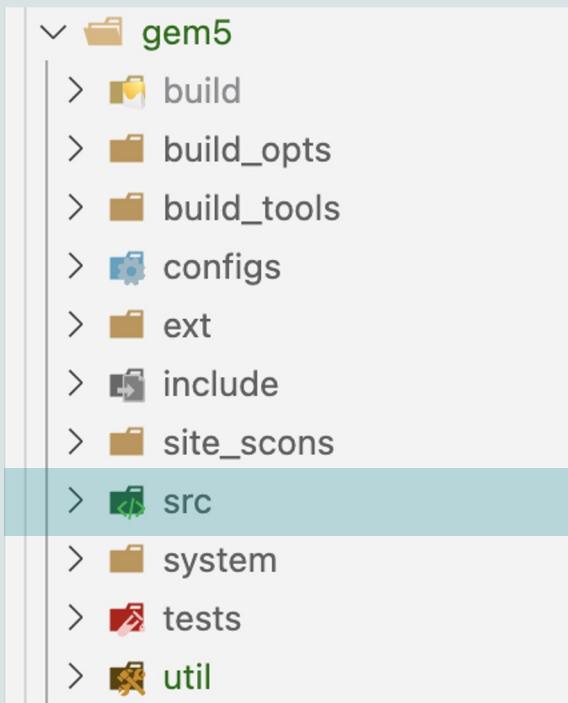
141 simulator = Simulator(
142     board=board,
143     on_exit_event={ExitEvent.SIMPOINT_BEGIN: reset_and_dump()},
144 )
145

```

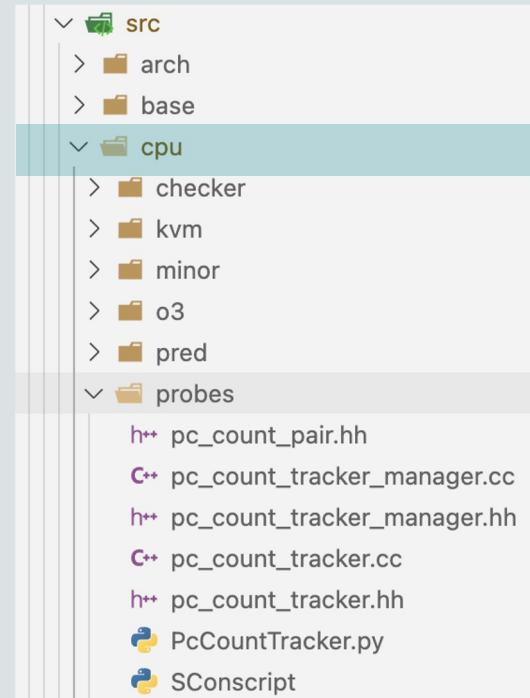
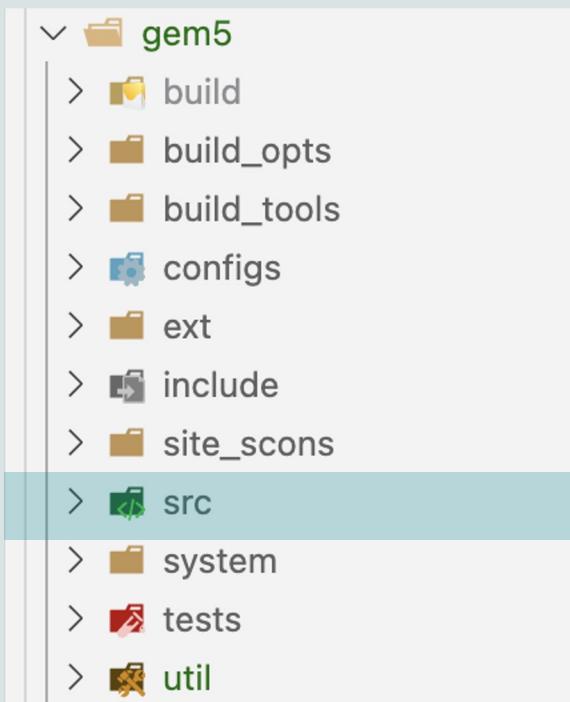
# Where to find the LoopPoint related files?



# Where to find the LoopPoint related files?



# Where to find the LoopPoint related files?



# Debug Flag

The debug flag is activated by passing the option in the command line. For example:

```
build/X86/gem5.opt --debug-flags=PcCountTracker example-script.py
```

It shows all the PCs and PC Count pairs that the simulation is tracking.

```
4      0: board.processor.cores0.core.probeListener.ptmanager: total 3 PCs in counter
5      0: board.processor.cores0.core.probeListener.ptmanager: all targets:
6      (4221392,603516434)
7      (4221392,156028351)
8      (4221392,179989993)
9      (4221392,250145350)
```

It also shows the PC Count pair that's struggling the exit event and the remaining pairs that haven't been encountered.

```
309  682559042715: board.processor.cores0.core.probeListener.ptmanager: pc:(4221392,642586697) encountered
310  682559042715: board.processor.cores0.core.probeListener.ptmanager: There are 0 targets remained
311  682559042715: board.processor.cores0.core.probeListener.ptmanager: all targets are encountered.
```

# Useful gem5 tutorials

Link to gem5 standard library tutorial:

<https://www.gem5.org/documentation/gem5-stdlib/overview>

Link to gem5 2022 bootcamp website:

<https://gem5bootcamp.github.io/gem5-bootcamp-env/>



Thank you!

# LoopPoint Tools: Sampled Simulation of Complex Multi-threaded Workloads using Sniper and gem5

Alen Sabu<sup>1</sup>, Changxi Liu<sup>1</sup>, Akanksha Chaudhari<sup>1</sup>, Harish Patil<sup>2</sup>, Wim Heirman<sup>2</sup>,  
Zhantong Qiu<sup>3</sup>, Jason Lowe-Power<sup>3</sup>, Trevor E. Carlson<sup>1</sup>

<sup>1</sup>National University of Singapore

<sup>2</sup>Intel Corporation

<sup>3</sup>University of California, Davis

