



XPU-Point: Simulator-Agnostic Sample Selection Methodology for Heterogeneous CPU-GPU Applications

Alen Sabu, [Harish Patil](#), Wim Heirman, Changxi Liu, Trevor E. Carlson



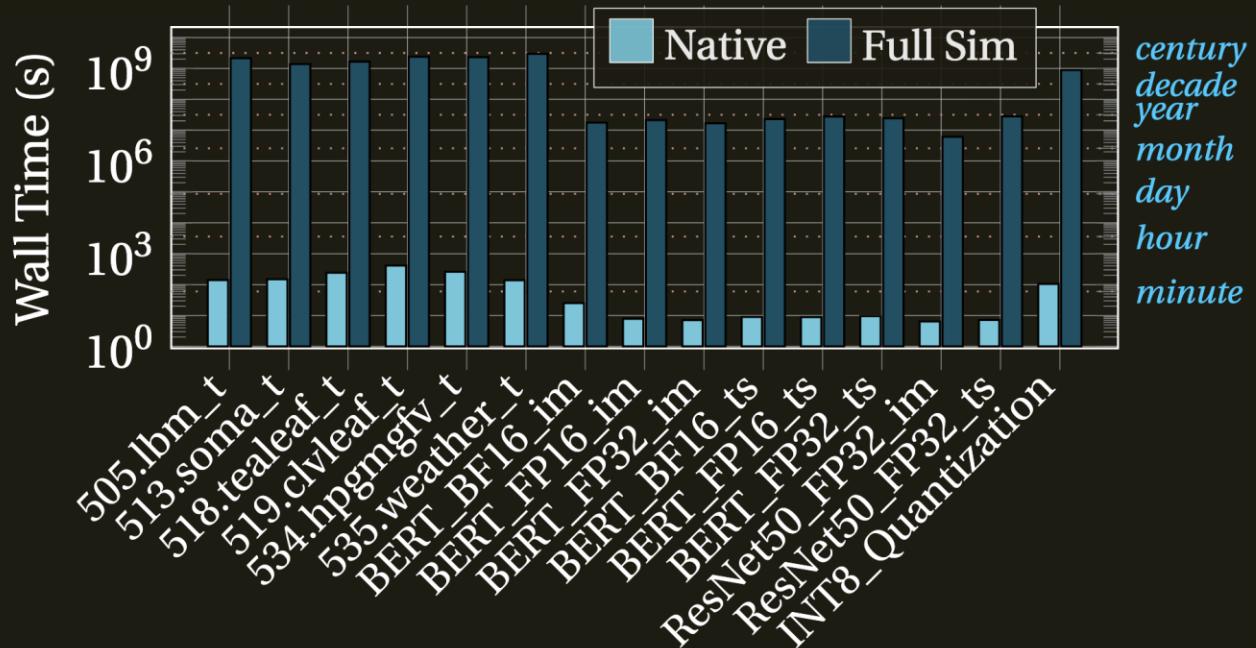
NUS
National University
of Singapore

intel

Parallel Architectures and Compilation Techniques (PACT)

November 06, 2025

Complex Architectures → Unrealistic Simulation Times



century
decade
year
month
day
hour
minute

Estimated Simulation Times:
gem5 (CPU portion) and
AccelSim (GPU portion)
heterogeneous CPU-GPU
benchmarks *SPEChpc 2021*
and *PyTorch/inference*

Modern architectures require smarter simulation techniques

Simulation: Key Questions

Where to Simulate?

Unit of Work/Simulation

- Repeatable across runs
- Microarchitecture-independent

How to Simulate?

- Trace-driven/Checkpoint-driven
- System-level/User-level

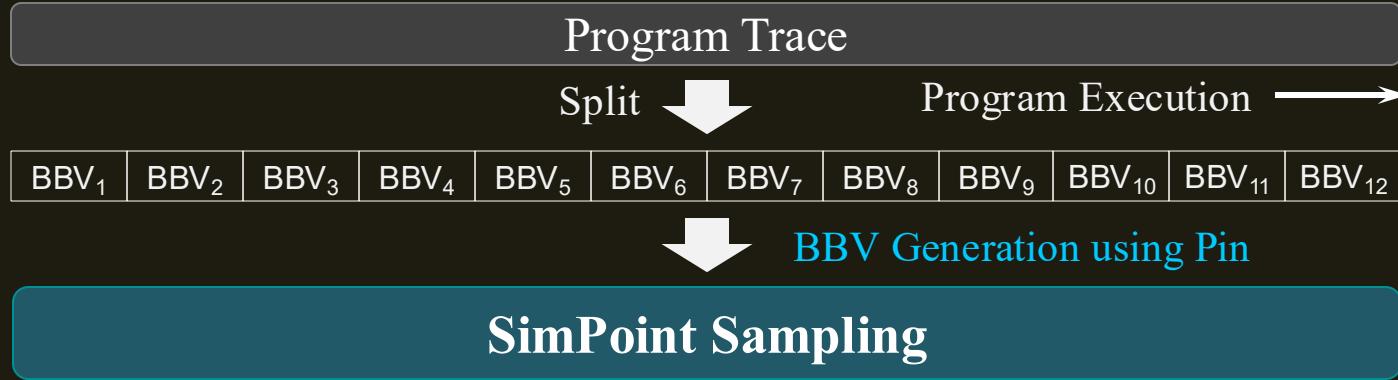
Are Simulation Regions Representative?

Compute Sampling Error

- Using simulation
- Using native execution (**simulator-agnostic**)

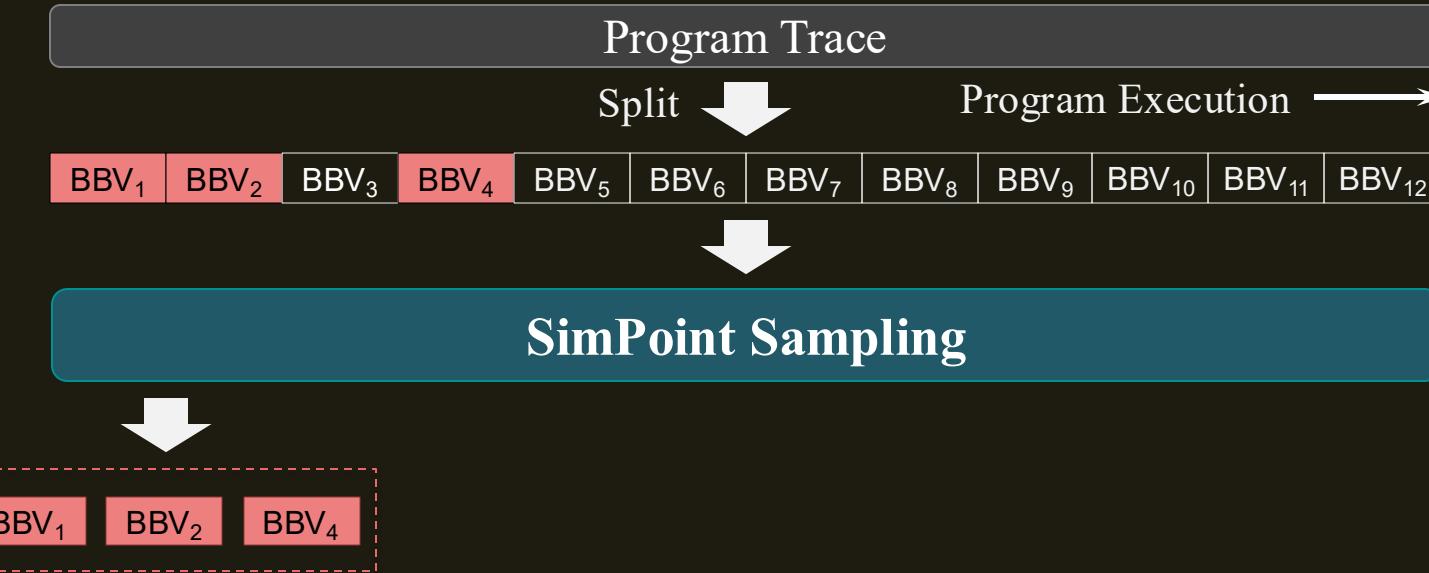
Selection of Regions of Interest

Where to Simulate?



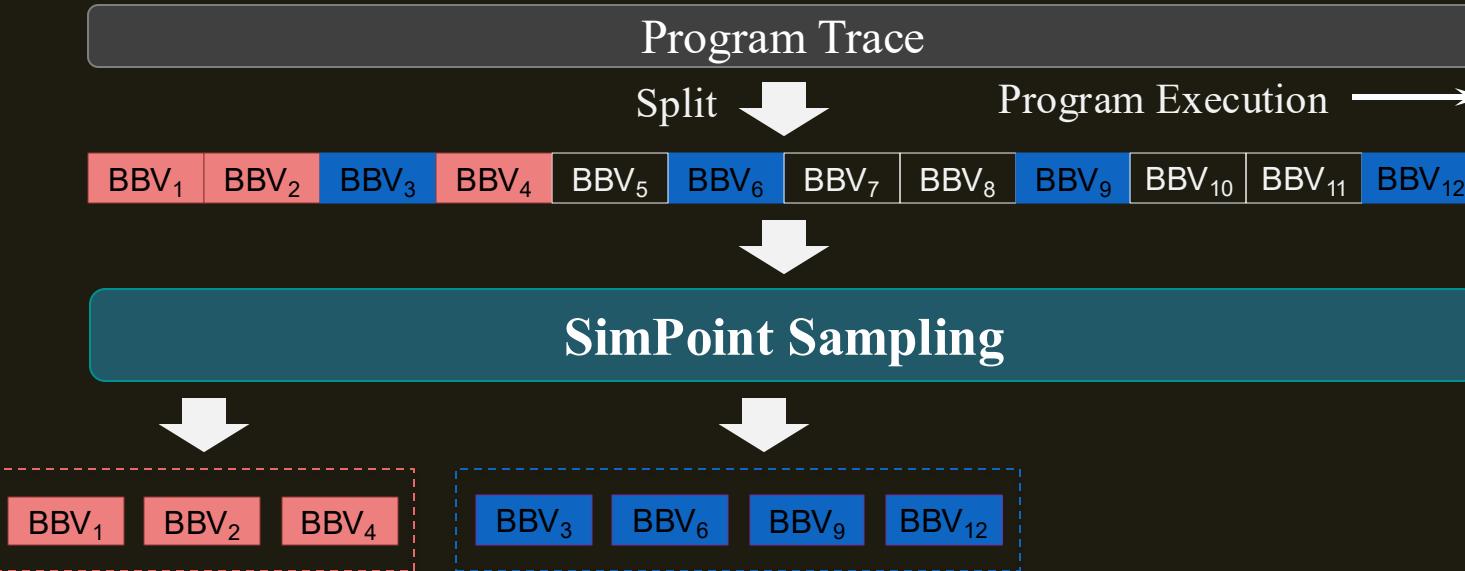
Selection of Regions of Interest

Where to Simulate?



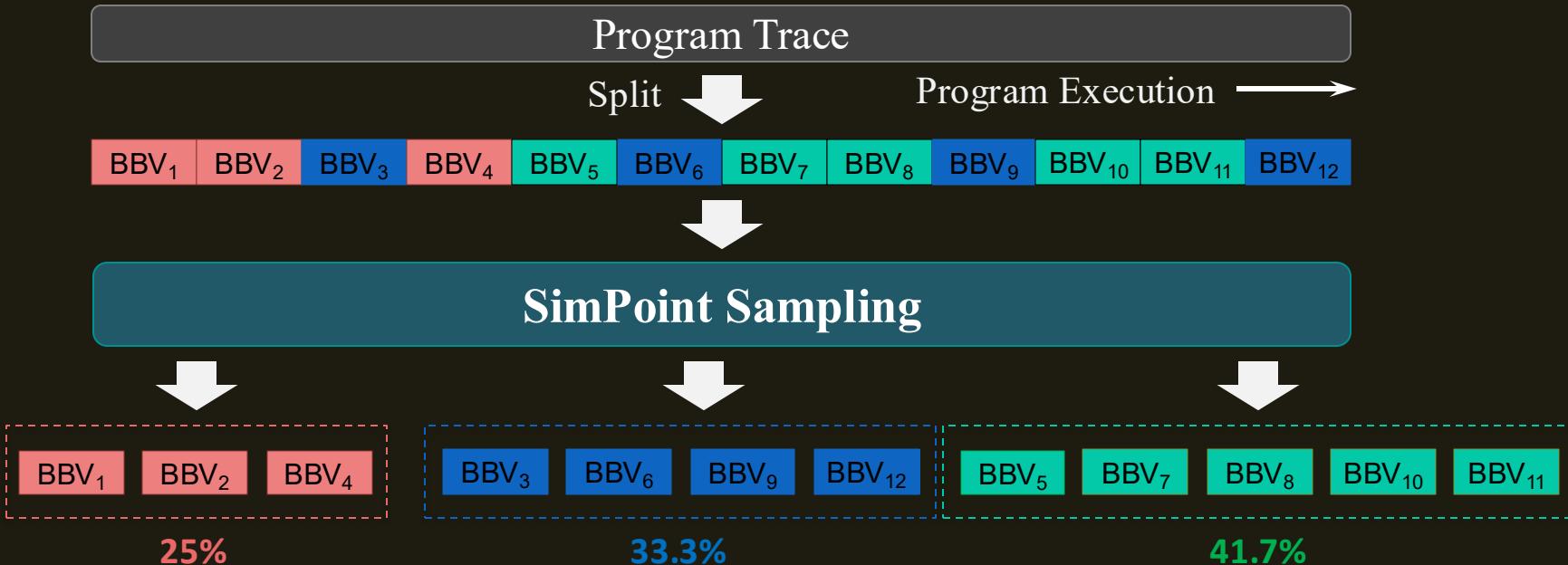
Selection of Regions of Interest

Where to Simulate?



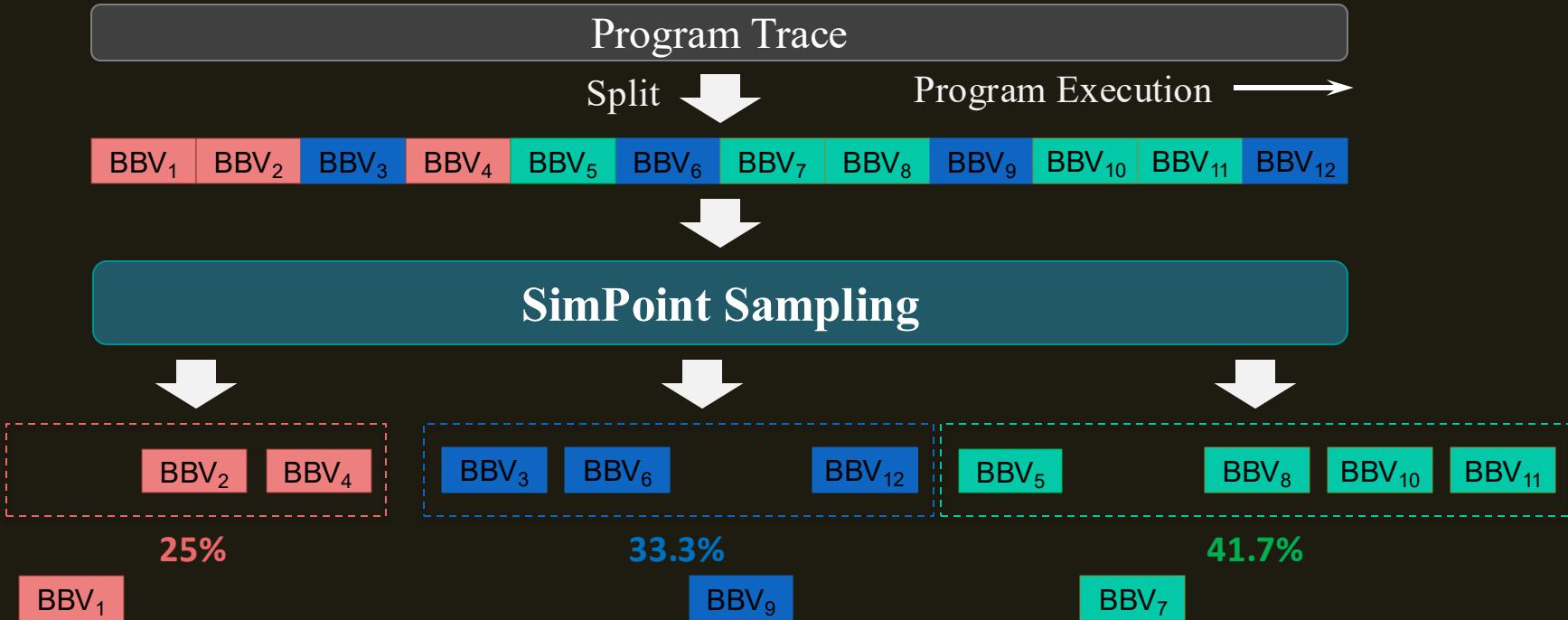
Selection of Regions of Interest

Where to Simulate?



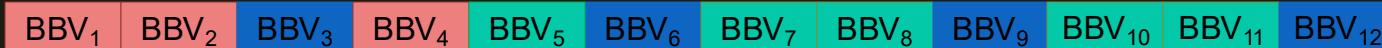
Selection of Regions of Interest

Where to Simulate?



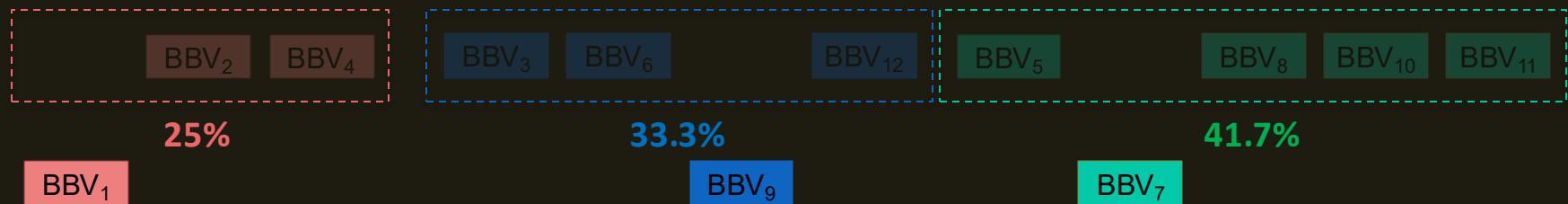
Projection Methodology

Instead of all regions...



...simulate only selected regions

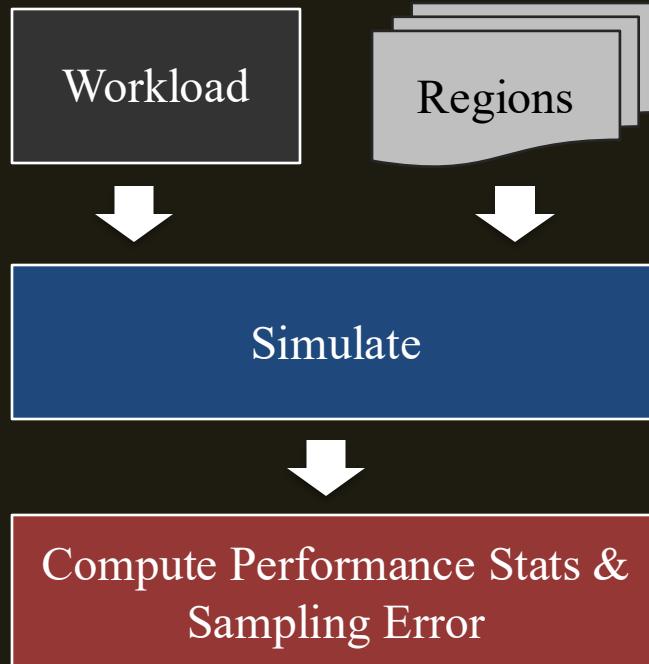
Project performance using weights



$$\text{Speedup} = 12/3 = 4$$

Simulation Region Validation With Simulation

Are Simulation Regions
Representative?



$$\text{Sampling Error} = \left| 1 - \frac{\text{Extrapolated Perf}}{\text{Actual Perf}} \right|$$

Challenge:

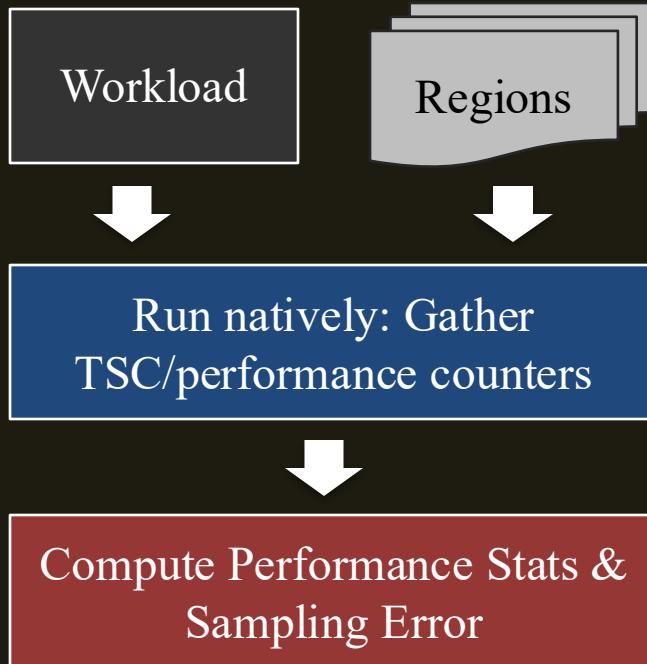
- Whole-program simulation is very slow

Workarounds:

- Use short workloads
- Use a fast, less accurate simulator

Simulation Region Validation With Native Execution

Are Simulation Regions
Representative?



$$\text{Sampling Error} = \left| 1 - \frac{\text{Extrapolated Perf}}{\text{Actual Perf}} \right|$$

Simulator-agnostic:

- Using native system as the simulator
- Much faster

Challenge:

- Precisely gathering region performance

Simulation Region Selection at Intel: Past 20 years

Methodology	Scope	Regions (Unit of Work/Simulation)	Sample Validation Technique	Comment
PinPoints (MICRO 2004)	Single-threaded/ Itanium	Fixed instructions	[simulator-agnostic] Pin (JIT) + perfmon	<i>Fixed-length intervals only</i>

Simulation Region Selection at Intel: Past 20 years

Methodology	Scope	Regions (Unit of Work/Simulation)	Sample Validation Technique	Comment
PinPoints (MICRO 2004)	Single-threaded/ Itanium	Fixed instructions	[simulator-agnostic] Pin (JIT) + perfmon	<i>Fixed-length intervals only</i>
Cross-binary Simulation Points (ISPASS 2007, 2015)	Single-threaded, multiple binaries/x86	Fixed instructions (binary 1)	CMP\$IM: Fast Pin-based cache simulator	<i>Less detailed simulator used</i>

Simulation Region Selection at Intel: Past 20 years

Methodology	Scope	Regions (Unit of Work/Simulation)	Sample Validation Technique	Comment
PinPoints (MICRO 2004)	Single-threaded/ Itanium	Fixed instructions	[simulator-agnostic] Pin (JIT) + perfmon	<i>Fixed-length intervals only</i>
Cross-binary Simulation Points (ISPASS 2007, 2015)	Single-threaded, multiple binaries/x86	Fixed instructions (binary 1)	CMP\$IM: Fast Pin-based cache simulator	<i>Less detailed simulator used</i>
GT-PinPoints (IISWC 2015)	OpenCL: GPU-only/Intel GPUs	GPU kernels	[simulator-agnostic] <i>CoFluent</i>	<i>GPU-only</i>

Simulation Region Selection at Intel: Past 20 years

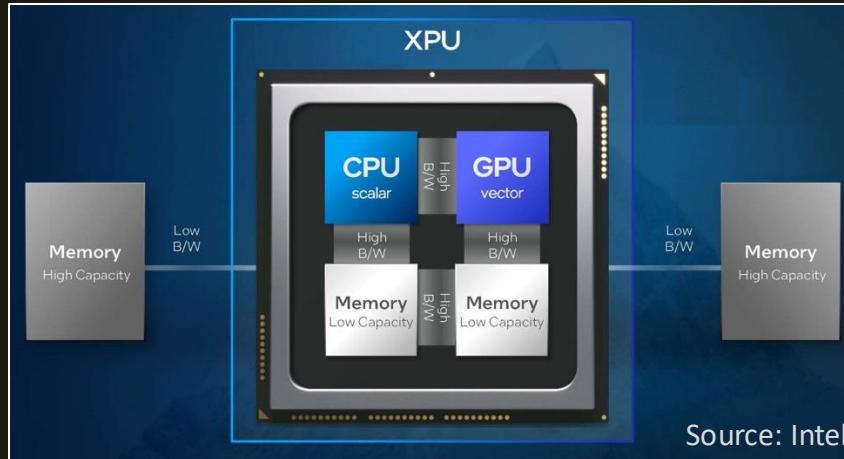
Methodology	Scope	Regions (Unit of Work/Simulation)	Sample Validation Technique	Comment
PinPoints (MICRO 2004)	Single-threaded/ Itanium	Fixed instructions	[simulator-agnostic] Pin (JIT) + perfmon	<i>Fixed-length intervals only</i>
Cross-binary Simulation Points (ISPASS 2007, 2015)	Single-threaded, multiple binaries/x86	Fixed instructions (binary 1)	CMP\$IM: Fast Pin-based cache simulator	<i>Less detailed simulator used</i>
GT-PinPoints (IISWC 2015)	OpenCL: GPU-only/Intel GPUs	GPU kernels	[simulator-agnostic] <i>CoFluent</i>	<i>GPU-only</i>
LoopPoint (HPCA 2022)	Multi-threaded/x86	Loop iterations	Sniper: Pin/SDE-based simulator	<i>SPEC ‘train’ runs used</i>

Simulation Region Selection at Intel: Past 20 years

Methodology	Scope	Regions (Unit of Work/Simulation)	Sample Validation Technique	Comment
PinPoints (MICRO 2004)	Single-threaded/ Itanium	Fixed instructions	[simulator-agnostic] Pin (JIT) + perfmon	<i>Fixed-length intervals only</i>
Cross-binary Simulation Points (ISPASS 2007, 2015)	Single-threaded, multiple binaries/x86	Fixed instructions (binary 1)	CMP\$IM: Fast Pin-based cache simulator	<i>Less detailed simulator used</i>
GT-PinPoints (IISWC 2015)	OpenCL: GPU-only/Intel GPUs	GPU kernels	[simulator-agnostic] <i>CoFluent</i>	<i>GPU-only</i>
LoopPoint (HPCA 2022)	Multi-threaded/x86	Loop iterations	Sniper: Pin/SDE-based simulator	<i>SPEC ‘train’ runs used</i>
XPU-Point (PACT 2025)	Heterogeneous CPU-GPU	GPU kernel: end to end	[simulator-agnostic] Pin (probe) + GT-Pin & NVBit	<i>Co-analysis of CPU and GPU</i>

Why Heterogeneous Architectures?

- Multi-cores aren't scaling well¹ – power and thermal constraints
- XPU: Heterogeneous system w/ CPU, GPU, and memory co-packaged



Source: Intel

Simulation of Heterogeneous Architectures

Heterogeneous CPU-GPU simulation is extremely challenging

Simulation Slowdowns



CPU simulation $>10,000 \times$ slowdown¹

GPU simulation $>1,000,000,000 \times$ slowdown²

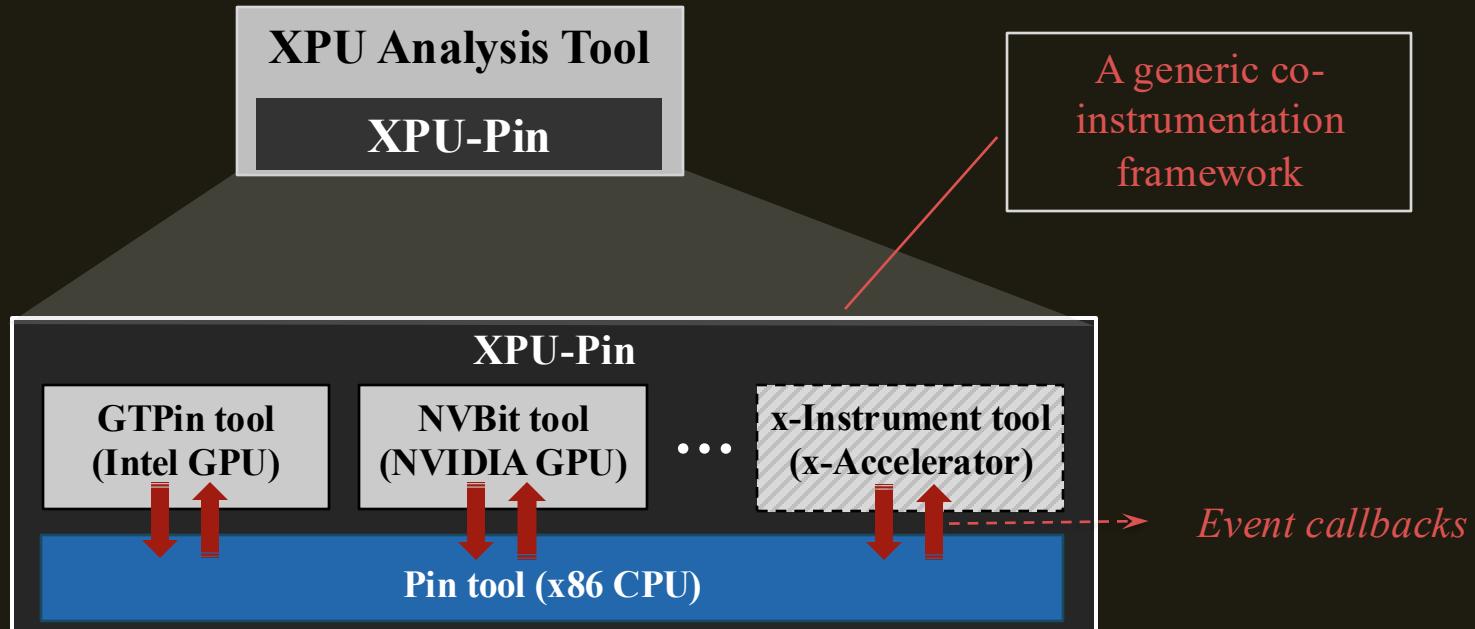
Phase-based CPU-GPU Region Selection

- Modern CPU-GPU workloads are **co-operative** (Ex. GROMACS)
- Need CPU and GPU co-analysis for **combined phase detection**



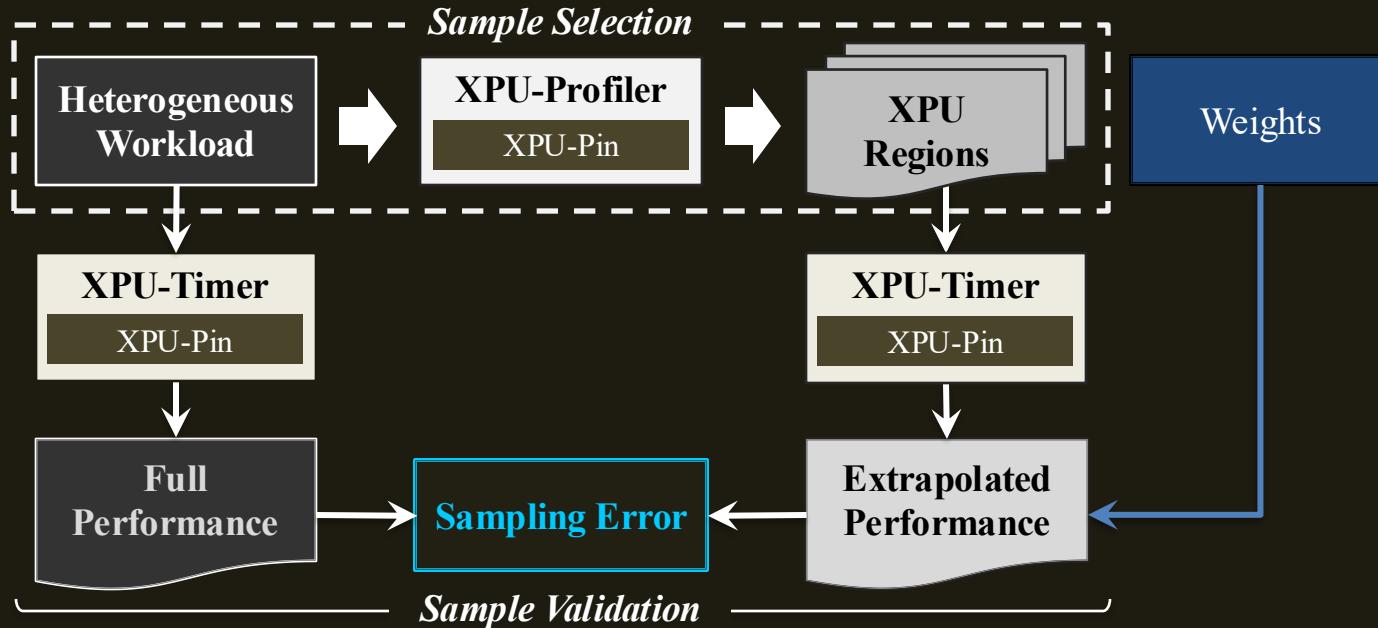
Challenge: No framework for simultaneous CPU and GPU analysis

XPU-Pin: Framework for Co-Analysis of Heterogeneous Execution



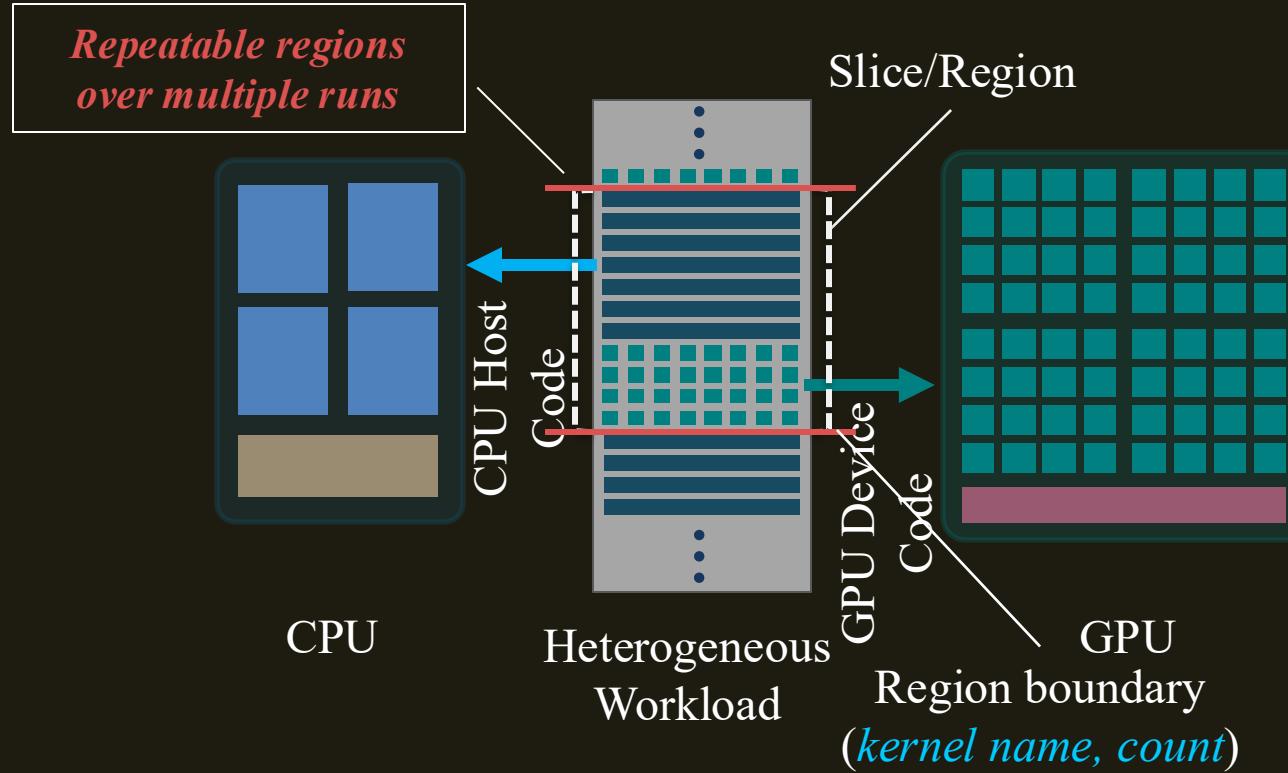
Support for generic accelerators: Need instrumentation tool as shared library

XPU-Point: End-to-End Workflow

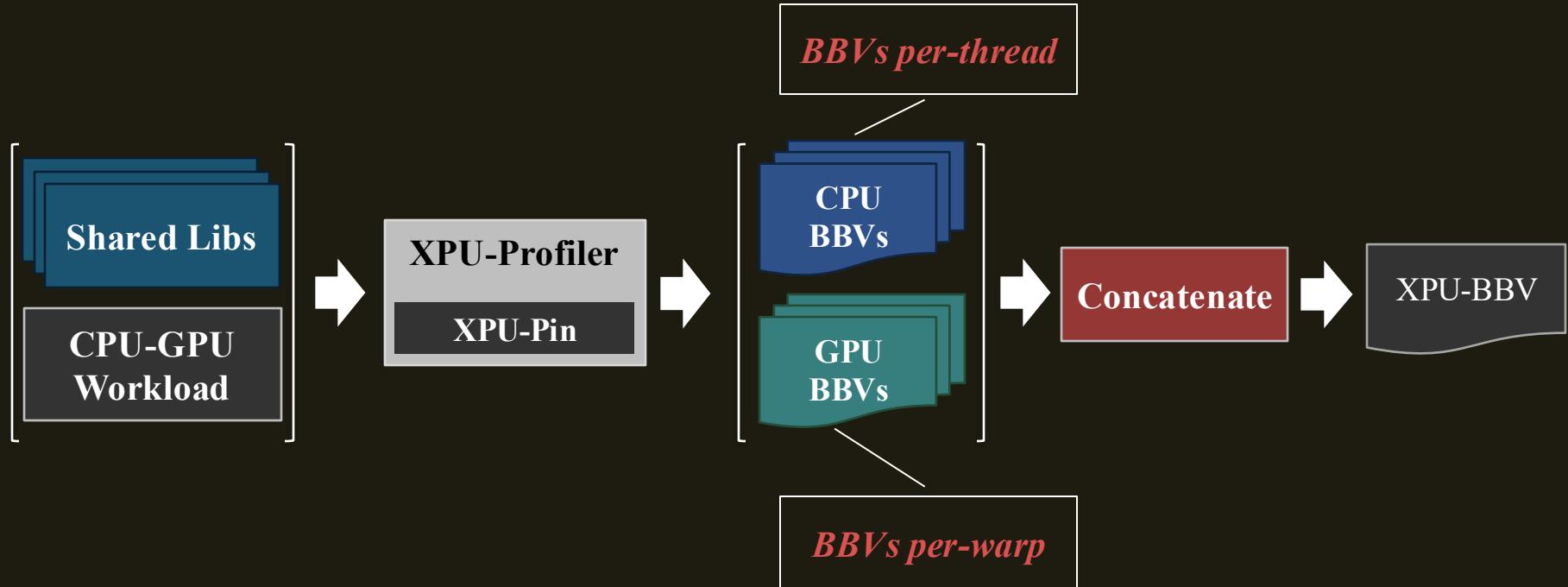


$$\text{Sampling Error} = \left| 1 - \frac{\text{Extrapolated Perf}}{\text{Actual Perf}} \right|$$

Unit of Work for XPU-Point

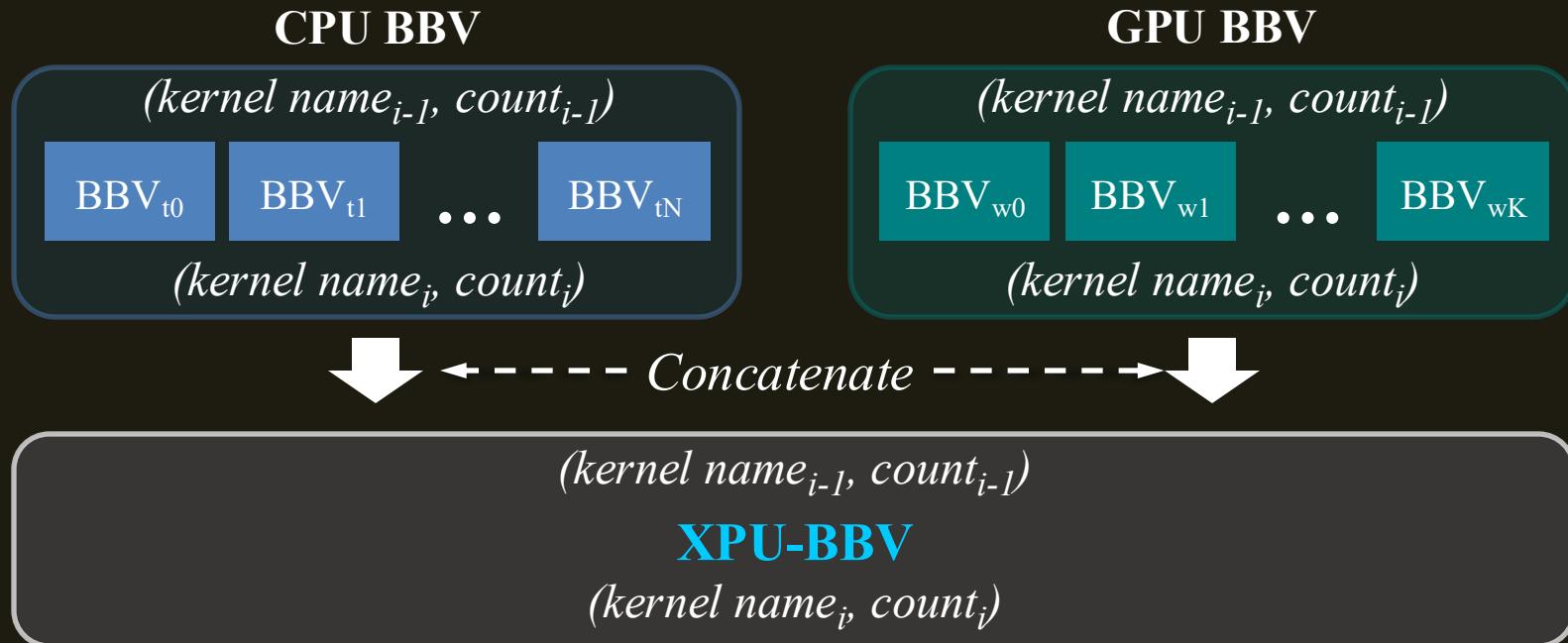


XPU-Profiler: CPU-GPU BBV Generation

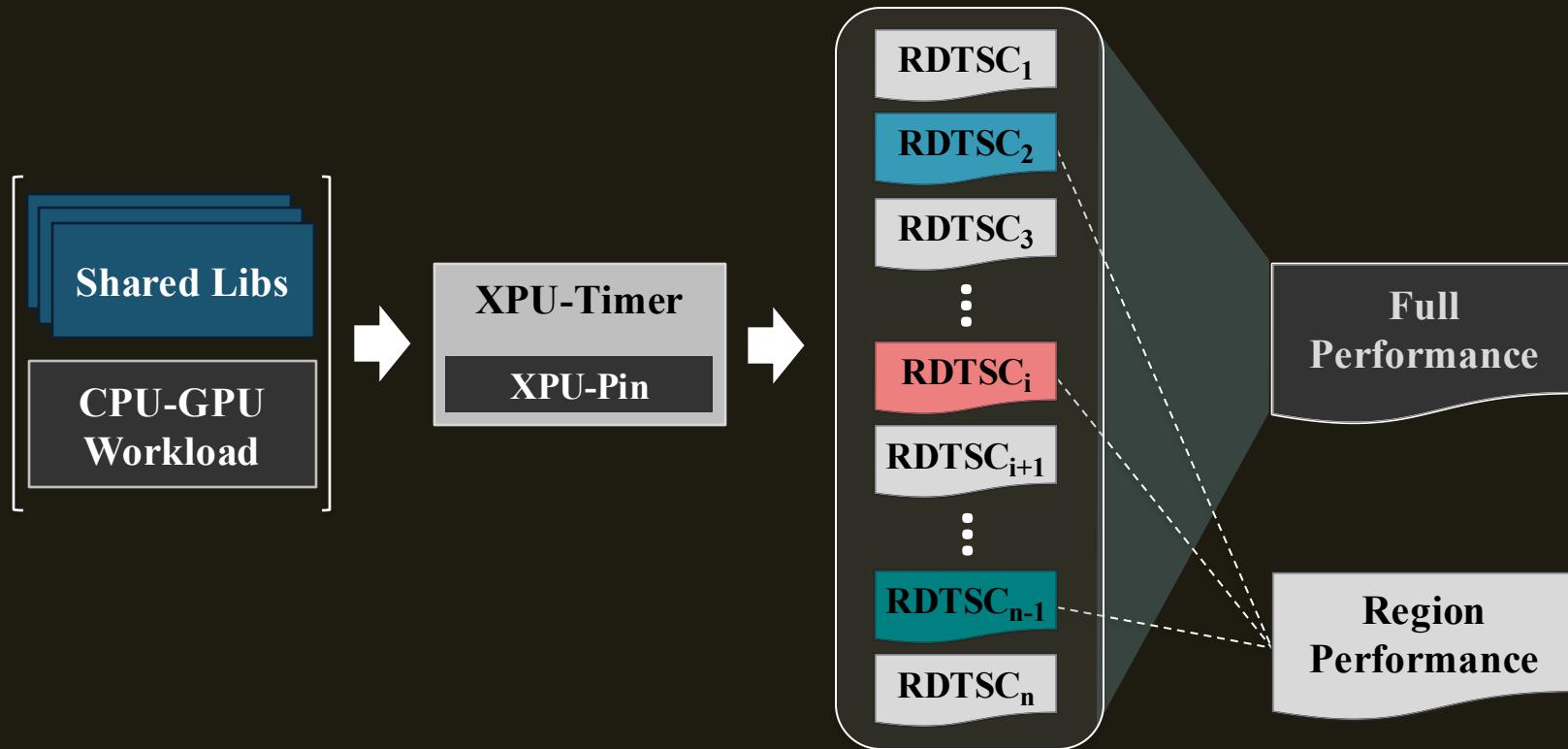


Challenge: Overhead of profiling → Be selective (shared libraries)

XPU-BBVs : CPU-GPU BBV Concatenation



XPU-Timer: Time Stamps for CPU-GPU Regions



Experimental Setup

- CPUs
 - Multiple Intel Client/Server CPUs
- GPUs
 - Intel: Iris Xe (Integrated), Discrete Graphics 2 (DG2), Ponte Vecchio (PVC)
 - NVIDIA: A100, GeForce GTX 1080, Titan XP
- Compilers
 - Intel OneAPI, GNU, NVCC

Results Reported

1. Sampling Error

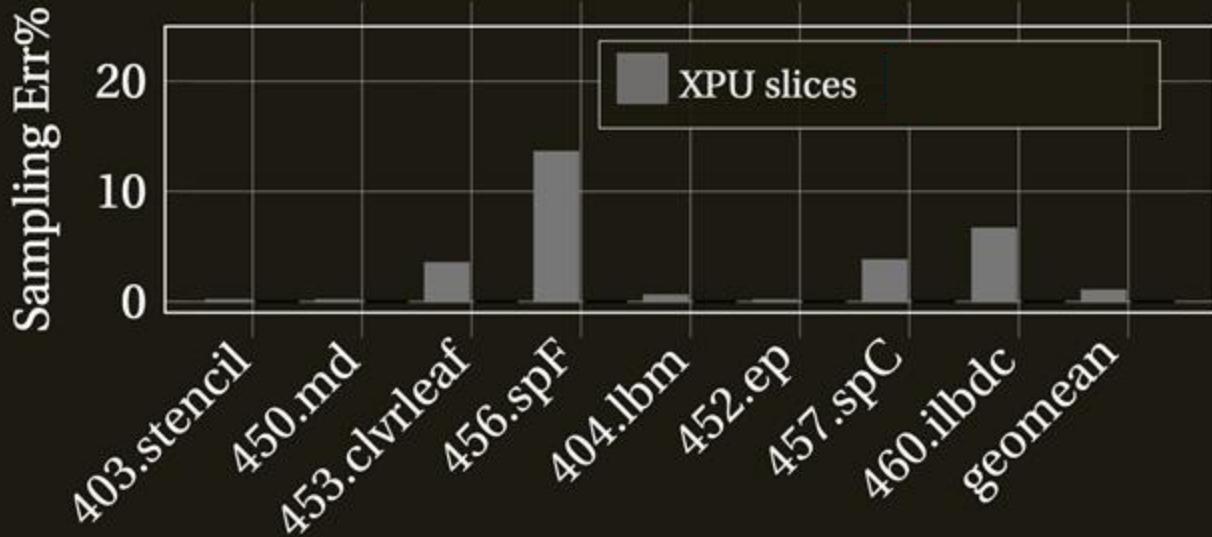
$$\text{Sampling Error} = \left| 1 - \frac{\text{Extrapolated Perf}}{\text{Actual Perf}} \right|$$

2. Speedup

$$\text{Speedup} = \frac{\text{Number of Total Regions}}{\text{Number of Simulation Regions}}$$

- Base analysis
 - BBV generation and error measurement on the same machine
- Cross analysis
 - Profiling ($\text{Machine}_1 / \text{GPU}_1$) → Measurement ($\text{Machine}_2 / \text{GPU}_2$)

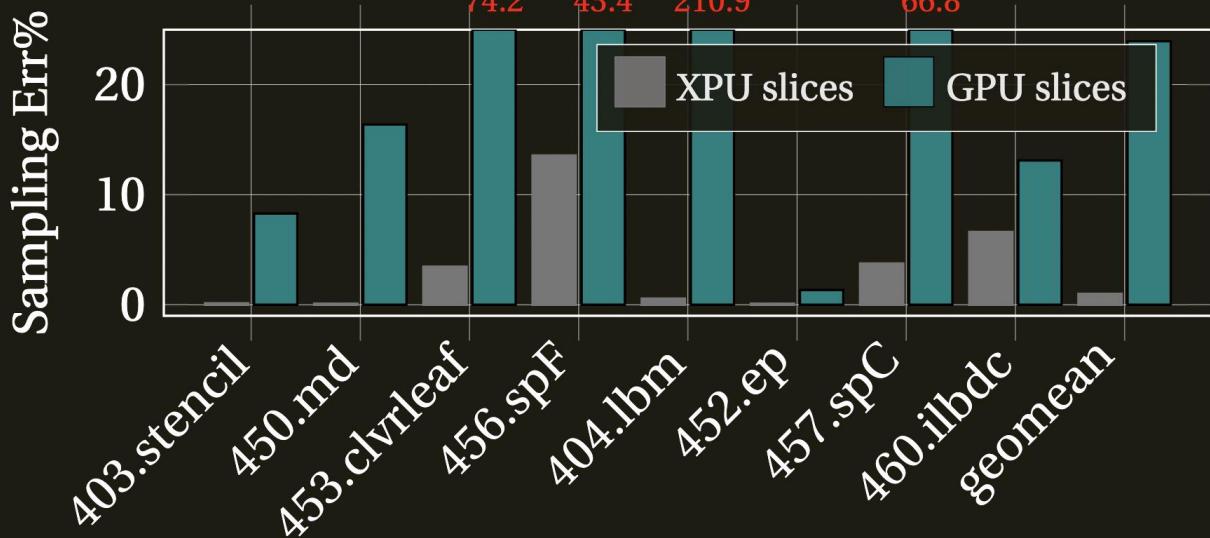
Results: SPECaccel2023



XPU slices:

- Combined CPU-GPU phase detection

Results: SPECaccel2023



XPU slices:

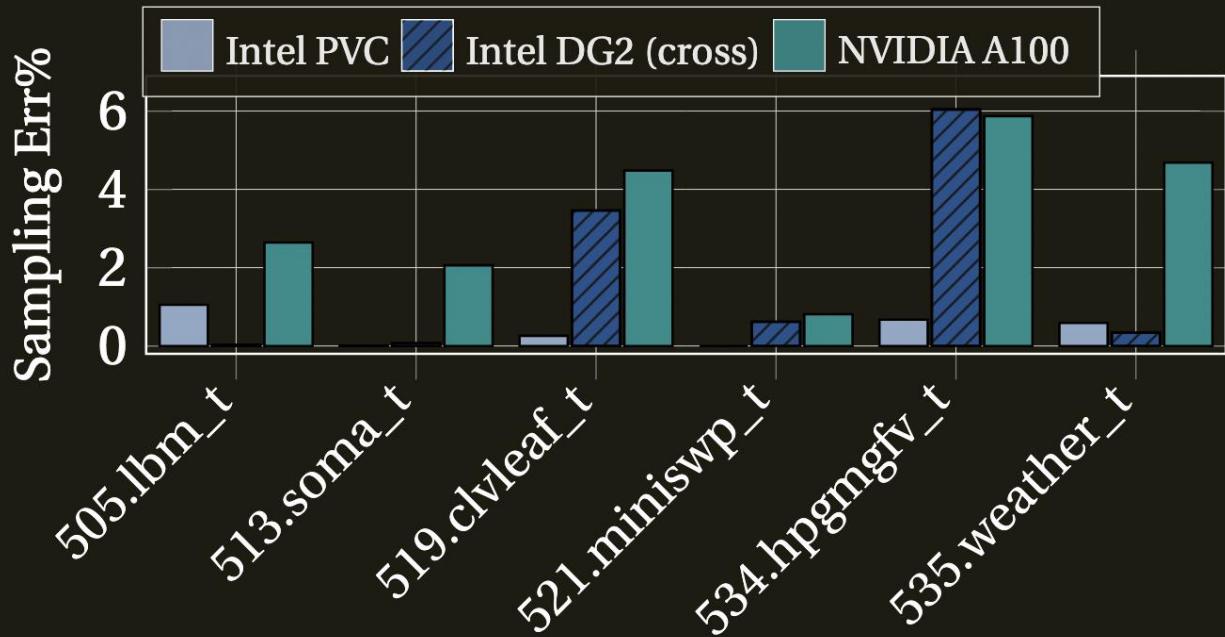
- Combined CPU-GPU phase detection

GPU slices:

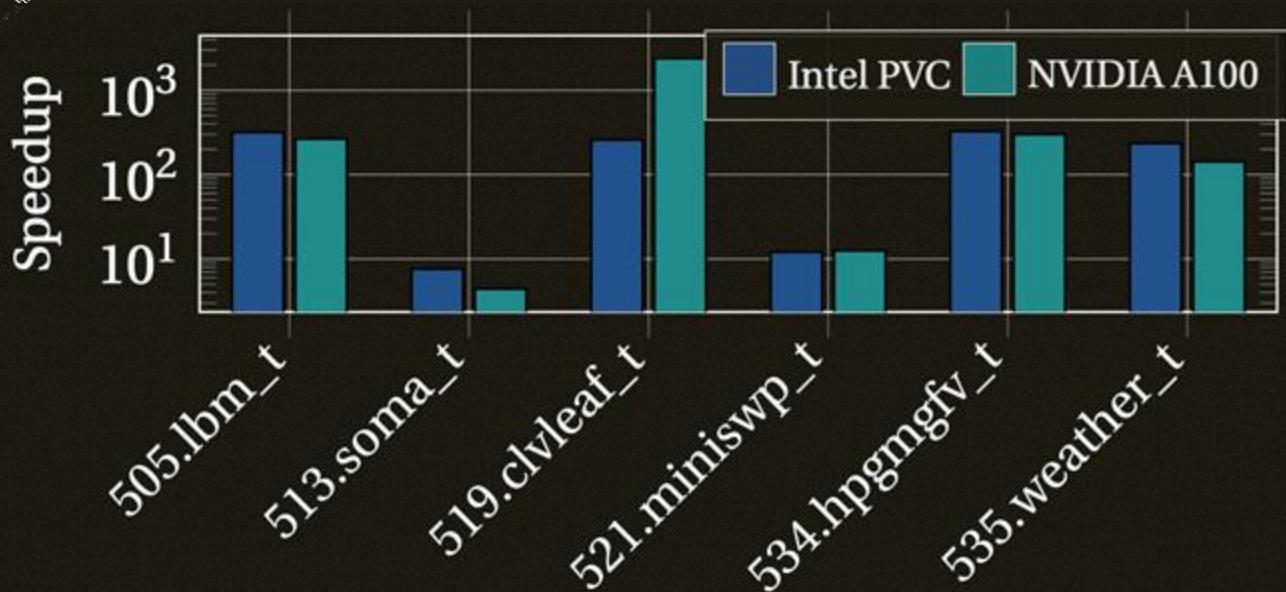
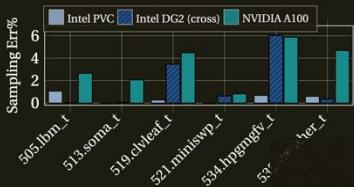
- GPU-only phase detection

Focusing on GPU-only evaluation could lead to inaccurate decisions

Results: SPEChpc2021



Results: SPEChpc2021

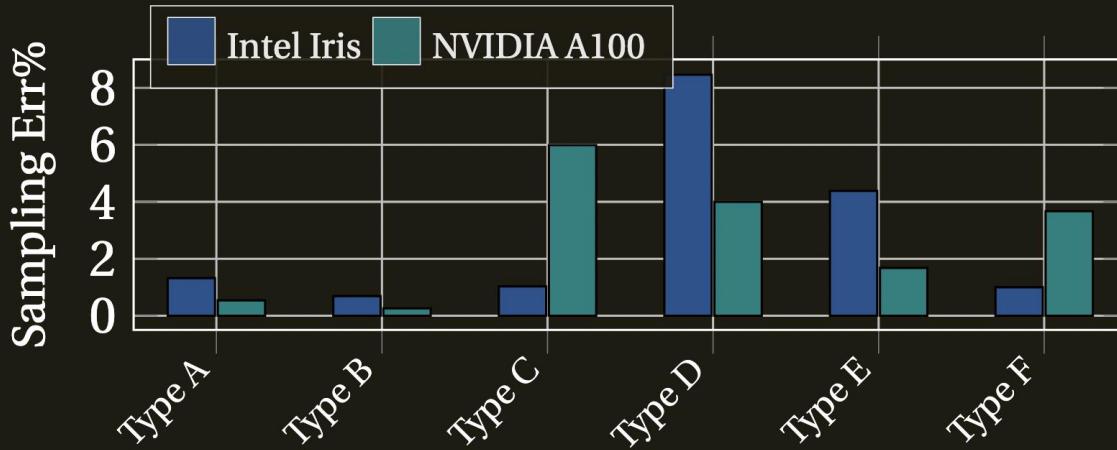


GROMACS: Various Configurations

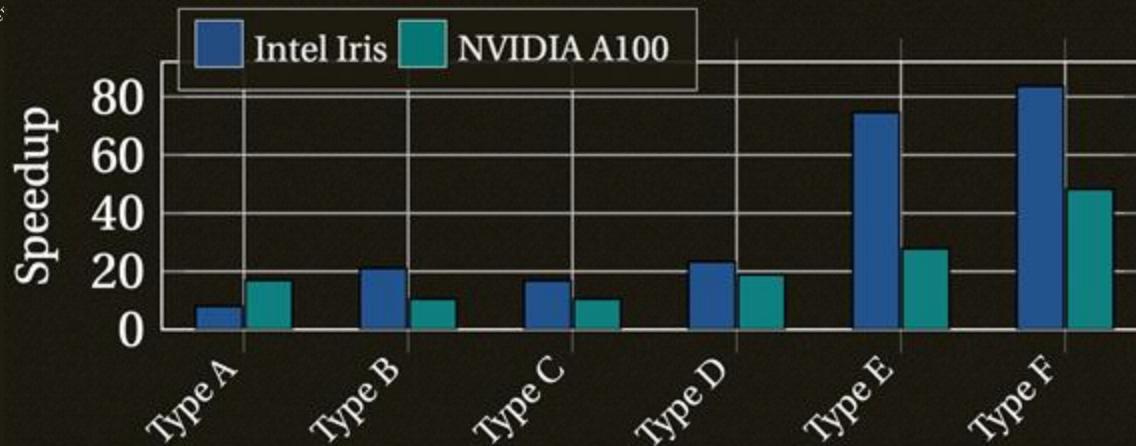
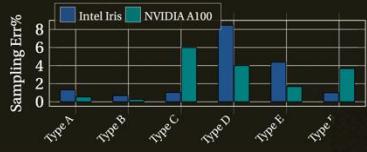
Type	nb	pme	pmefft	bonded	update	#slices
A	GPU	CPU	CPU	CPU	CPU	305
B	GPU	CPU	CPU	GPU	CPU	506
C	GPU	GPU	CPU	CPU	CPU	707
D	GPU	GPU	CPU	GPU	CPU	908
E	GPU	GPU	GPU	CPU	CPU	3730
F	GPU	GPU	GPU	GPU	CPU	3931

The classification of GROMACS based on the offloading device for the execution of each calculation. We also use -nsteps 200 with -notunepme for all types.

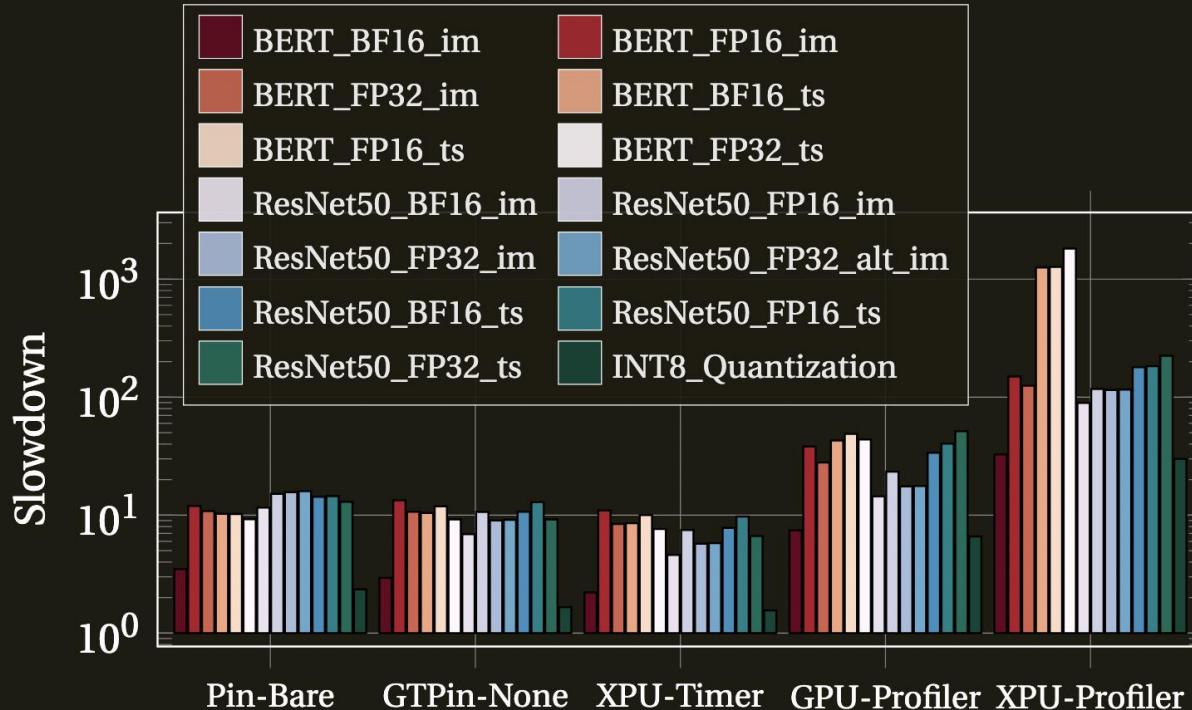
Results: GROMACS



Results: GROMACS



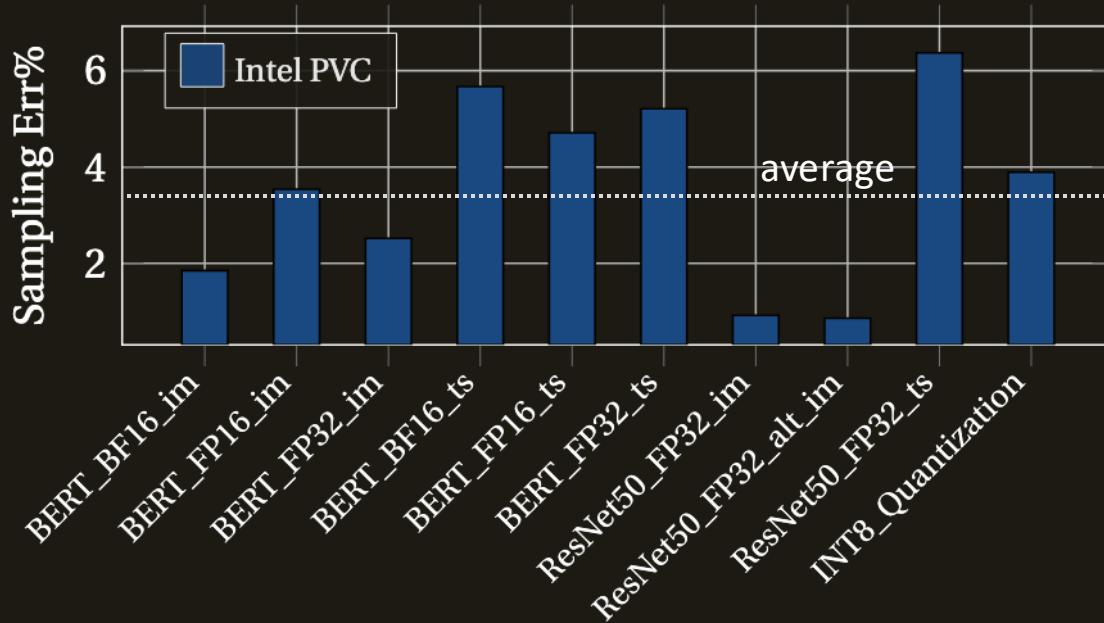
PyTorch Inference Workloads: Overheads



PyTorch Inference runs evaluated on platform with Intel Sapphire Rapids CPU and Intel Ponte Vecchio GPU

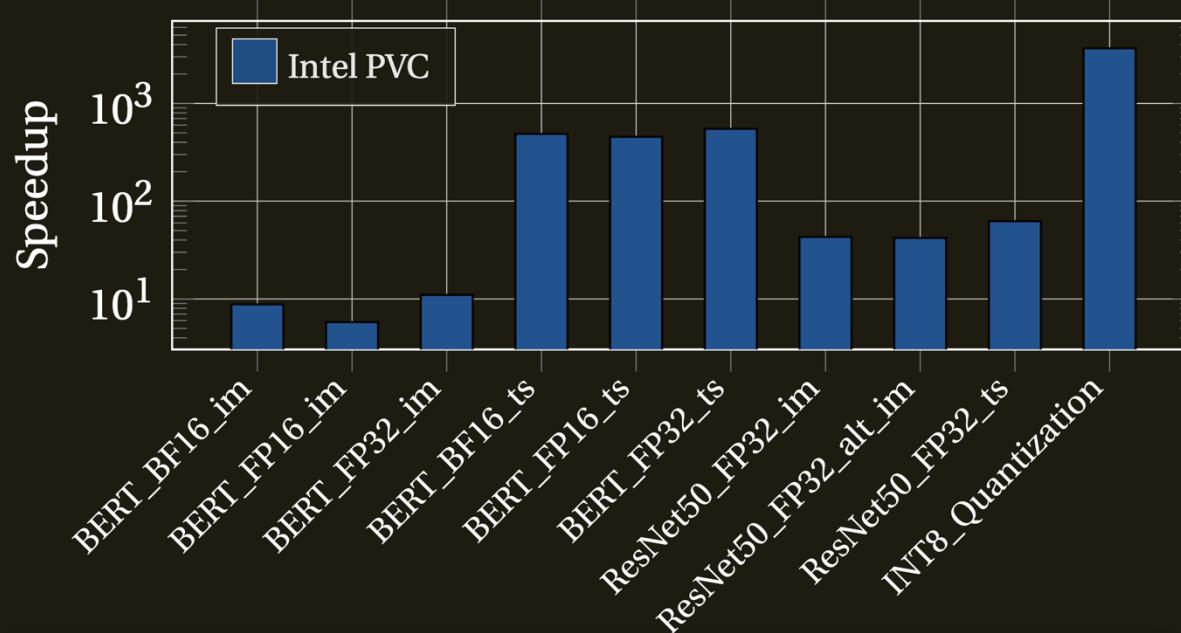
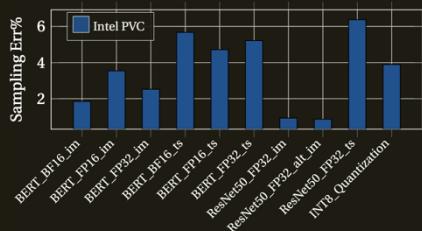
Challenge: Overhead of profiling → Be selective (shared libraries)

Results: PyTorch Inference



PyTorch Inference (selective profiling) runs evaluated on Intel Ponte Vecchio GPU

Results: PyTorch Inference



PyTorch Inference (selective profiling) runs evaluated on Intel Ponte Vecchio GPU

Summary

- XPU-Point is the first to enable accelerated heterogeneous simulation through **CPU-GPU co-sampling**
- Works for both **Intel-** and **NVIDIA-based** CPU-GPU platforms
- XPU-Point tools are open-sourced on GitHub
 - <https://github.com/nus-comparch/xpupoint>
- Acknowledgments
 - Roland Schulz, Edward Mascarenhas, Aleksandr Bobyr, Intel **GTPin** Team

