Fast, Accurate, and Novel Performance Evaluations with PinCPU

Nicholas Mosier
Stanford University
Stanford, CA, USA
nmosier@stanford.edu

Hamed Nemati
KTH Royal Institute of Technology
Stockholm, Sweden
hnnemati@kth.se

John C. Mitchell
Stanford University
Stanford, CA, USA
jcm@stanford.edu

Caroline Trippel
Stanford University
Stanford, CA, USA
trippel@stanford.edu

We present PinCPU, the first fast-forwarding processor model for the gem5 computer architecture simulator [1] to allow users to dynamically instrument x86-64 userspace workloads. PinCPU uses Intel Pin [5] to enable users to fast-forward gem5 simulations at near-native speeds and exposes a plugin interface that allows users to register their own instrumentation callbacks at runtime. First, we overview PinCPU's design (§1). Then, we describe how researchers can use PinCPU plugins to (i) generate simpoints [4] quickly and accurately, (ii) translate simpoints between binaries compiled from the same source for novel hardware-software performance comparisons, and (iii) fast-forward binaries featuring custom ISA extensions (§2). We have open sourced PinCPU[1] and plan on upstreaming it into gem5.

1 PinCPU Design

We implement PinCPU as a new fast-forwarding[2] CPU model in gem5. Fig. 1 depicts PinCPU's design, which is inspired by gem5's existing fast-forwarding model (the KvmCPU). PinCPU fast-forwards a workload as follows. First, it spawns an Intel Pin subprocess [5], consisting of the userspace workload itself, the core PinCPU pintool, and a shim (Fig. 1). The core pintool implements low-level primitives necessary for setting up and managing the workload, and registers user instrumentation plugins (including those in §2.1–2.3). The shim is responsible for communicating with the PinCPU model in the main gem5 process and orchestrating the workload (via commands to the pintool) in the Pin subprocess. We emulate virtual memory by mapping the workload's virtual addresses into a shared memory file at an offset equal to the physical address (Fig. 1).

2 PinCPU Plugins

PinCPU's usefulness and novelty comes from its ability to load user-provided workload instrumentation plugins. We have designed three plugins to enable faster, more accurate, and novel performance evaluation methodologies.

2.1 Plugin 1: Fast and Accurate SimPoint Generation

SimPoints [4] is a widely used, phase-based evaluation technique that identifies a few representative dynamic instruction intervals (simpoints) from a basic block vector (BBV) profile collected during program execution. To estimate the total simulated runtime, one simulates each simpoint and computes the weighted sum of their individual runtimes.

However, gem5's official method to generate a BBV profile for SimPoints [6] requires full simulation of the entire workload, which takes months for the SPEC CPU2017 benchmarks [2] for *reference*-sized inputs (Fig. 2a). As an expedient alternative, some researchers use valgrind's BBV profile generator [9], which produces an inaccurate profile due to differences in program execution paths between valgrind and gem5, e.g., due to gem5's disabling of some hardware capabilities, like AVX.

To resolve this issue, we present a BBV profiler plugin for PinCPU that is *over 7 times faster* than valgrind's but as *accurate* as gem5's full-simulation approach. Our PinCPU BBV profiler's accuracy comes from the fact that the workload takes the same execution path under PinCPU as under gem5's simulated CPUs, since PinCPU intercepts and hands off all platform-specific operations (such as `CPUID` and system calls) to gem5 for emulation. Our talk will include a brief example of how to use PinCPU's BBV profiler to quickly generate simpoints, as well as empirical results demonstrating the PinCPU BBV profiler's accuracy.

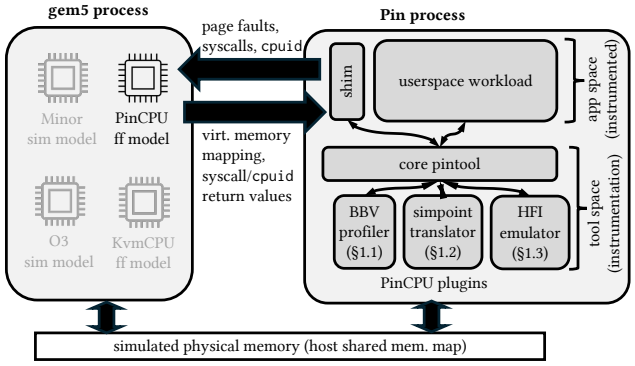
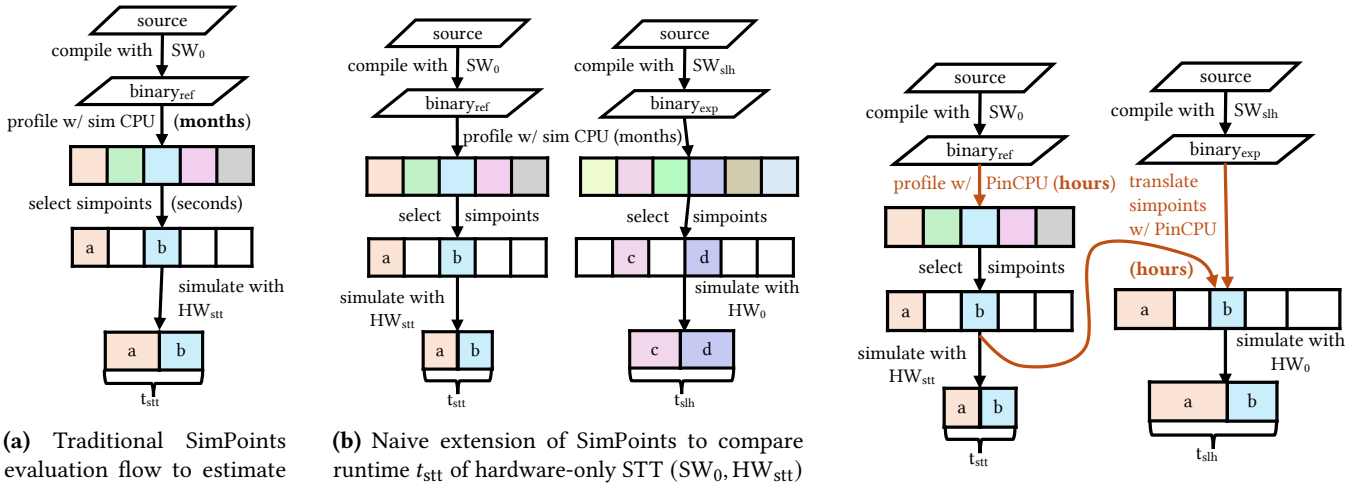


Figure 1. PinCPU design.

[1] Available at https://github.com/StanfordPLArchSec/pincpu-gem5.
[2] *Fast-forwarding* CPU models execute the workload on the host, rather than simulating it.



(a) Traditional SimPoints evaluation flow to estimate runtime t_{stt} of hardware-only proposal STT [7] (SW_0, HW_{stt}): **slow** (takes months).

(b) Naive extension of SimPoints to compare runtime t_{stt} of hardware-only STT (SW_0, HW_{stt}) against runtime t_{slh} of software-only SLH [3] (SW_{slh}, HW_0): **slow** (takes months), **inaccurate** (simulates STT/SLH on different code regions).

(c) Our PinCPU-enabled evaluation flow for comparing runtime of STT and SLH: **fast** (takes hours), **accurate** (simulates STT/SLH on same code regions.)

Figure 2. PinCPU and its plugins enable fast, accurate hardware-software performance comparisons.

2.2 Plugin 2: SimPoint Translation for Hardware-Software Performance Comparisons

Suppose a researcher wants to evaluate the runtime performance of a hardware proposal (e.g., STT, a hardware Spectre defense [7]) against that of a software proposal (e.g., SLH, a software Spectre defense [3]) for a particular benchmark using SimPoints. For fairness, the hardware proposal should be evaluated on the unmodified binary, not the software proposal's modified binary. However, one cannot simply apply SimPoints to each binary individually as in Fig. 2b, since the simpoints selected for the unmodified vs. modified binaries may represent different parts of the execution.

To overcome this, we propose a new technique called *simpoint translation* leveraging PinCPU (Fig. 2c). First, we select simpoints for the unmodified binary, via the method described in §2.1. Then, we use a PinCPU plugin to dynamically translate the unmodified binary's simpoints to equivalent instruction intervals in the modified binary. As a case study, we will use simpoint translation to conduct and present the first performance comparison ever of STT versus SLH.

2.3 Plugin 3: Fast-Forwarding of Custom ISA Extensions

Suppose a researcher wants to evaluate the runtime performance of a proposed ISA extension (e.g., HFI, a hardware-assisted sandboxing proposal [8]) on a long-running workload using SimPoints, which requires fast forwarding. However, gem5's only pre-existing fast-forwarding CPU model, KvmCPU, does not support dynamic binary instrumentation and thus cannot model ISA extensions. Thus, the researcher must either resort to using an emulator to crudely approximate their proposal's performance (e.g., HFI's emulator [8]) or evaluate their proposal on a shorter-running, non-representative workload that can be simulated in full.

PinCPU resolves this limitation of gem5 by allowing users to write custom plugins that emulate custom ISA extensions by hooking and modifying the behavior of existing or new instructions. As a case study, we will present a PinCPU plugin that emulates HFI's ISA extension and evaluate HFI using our PinCPU-aided simpoint translation methodology.

3 Conclusion

Our open-source PinCPU extension allows computer architecture researchers to perform fast, accurate, and novel performance evaluations in gem5. We hope PinCPU and the new methodologies we have developed using it will improve the accuracy of performance results and enable new kinds of performance evaluations in future research.

References

[1] Nathan Binkert et al. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News.*
[2] James Bucek, Klaus-Dieter Lange, and Jóakim v. Kistowski. 2018. Spec cpu2017: next-generation compute benchmark. In *ICPE'18.*
[3] Chandler Carruth. 2018. *Speculative Load Hardening.* LLVM Project.
[4] Erez Perelman et al. 2003. Using simpoint for accurate and efficient simulation. In *PER'03.*
[5] Chi-Keung Luk et al. 2005. Pin: building customized program analysis tools with dynamic instrumentation. *Acm sigplan notices.*
[6] gem5 Developers. 2018. *Simpoints.* gem5 Project.
[7] Jiyong Yu et al. 2019. Speculative taint tracking (stt) a comprehensive protection for speculatively accessed data. In *ISCA'19.*
[8] Shravan Narayan et al. 2023. Going beyond the limits of sfi: flexible and secure hardware-assisted in-process isolation with hfi. In *ASPLOS'23.*
[9] Valgrind Developers. [n. d.] *BBV: an experimental basic block vector generation tool.* Valgrind Project.