Computer architecture simulation

A presentation by

Prof. Jason Lowe-Power

Outline

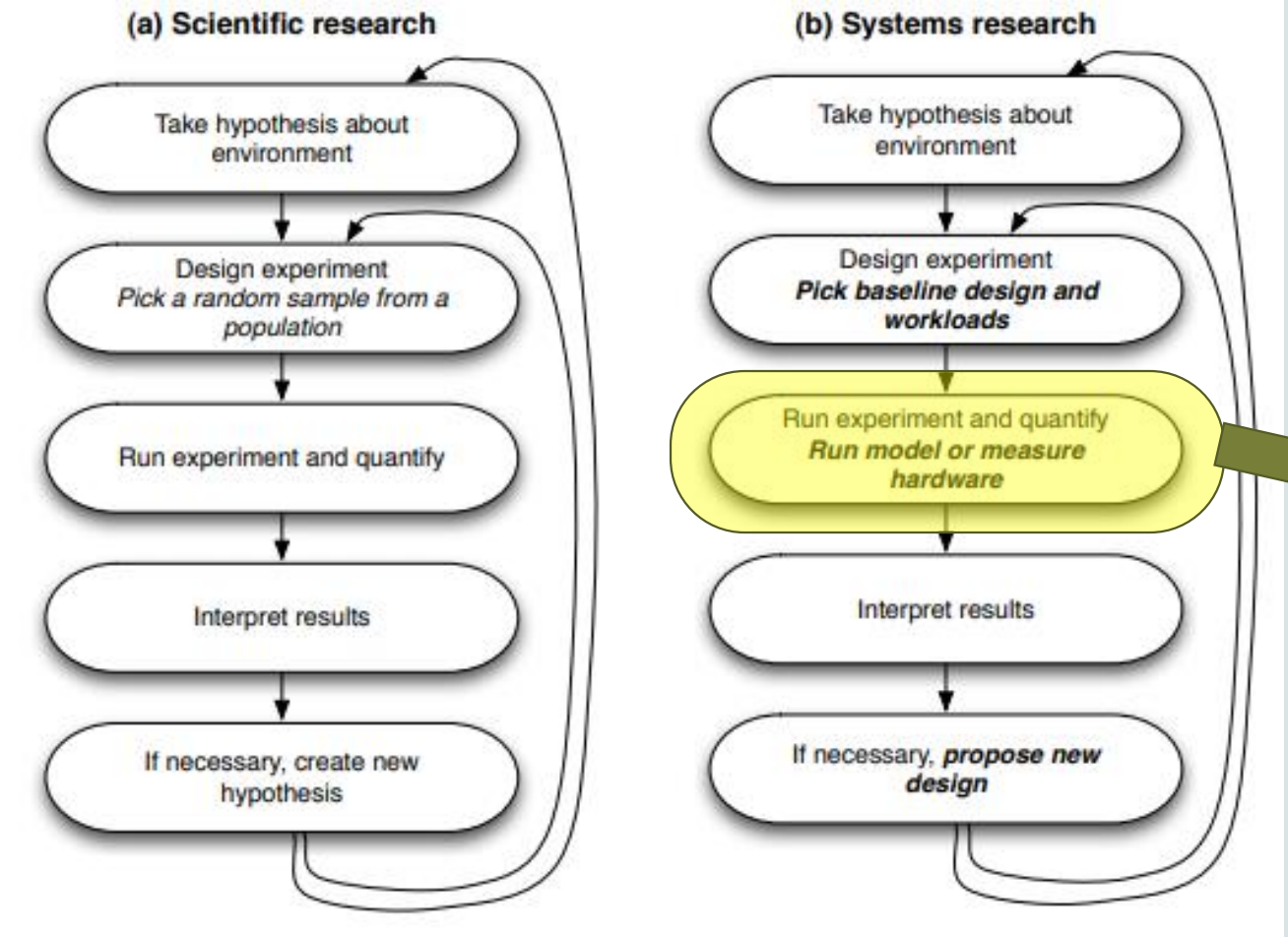
Computer systems research

Types of simulation

Nomenclature that we'll use

Different levels of simulation

Computer systems research/engineering



From Computer Architecture Performance Evaluation Methods by Lieven Eeckhout

Computer architecture simulation!

gem5

Why simulation?

Why simulation

Need a tool to evaluate systems that don't exist (yet)

- Performance, power, energy, etc.

Very costly to actually make the hardware

Computer systems are complex with many interdependent parts

- Not easy to be accurate without the full system

Simulation can be parameterized

- Design-space exploration
- Sensitivity analysis

Alternatives to cycle-level simulation

Analytic models

Amdahl's Law:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Queueing models:

gem5

Kinds of simulation

Functional simulation

Instrumentation-based

Trace-based

Execution-driven

Full system

Kinds of simulation

Functional simulation

- Executes programs correctly. Usually no timing information
- Used to validate correctness of compilers, etc.
- RISC-V Spike, QEMU, gem5 “atomic” mode

Instrumentation

- Often binary translation. Runs on actual hardware with callbacks
- Like trace-based. Not flexible to new ISA. Some things opaque (caches)
- PIN, CMP$im, NVBit

Kinds of simulation

Trace-based simulation

- Generate addresses/events and re-execute
- Can be fast (no need to do functional simulation). Reuse traces
- *If execution depends on timing, this will not work!*
- "Specialized" simulators for single aspect (e.g., just cache hit/miss)

Execution-driven

- Functional and timing simulation is *combined*
- gem5 and many others
- gem5 is "execute in execute" or "timing directed"

Full system simulation

Components modeled with enough fidelity to run mostly unmodified apps

Often “Bare metal” simulation

All of the program is functionally emulated by the simulator

Often means running the OS in the simulator, not faking it

“Full system” simulators are often combine functional and execution-based

Nomenclature

Host: the actual hardware you're using

Running things directly on the hardware:

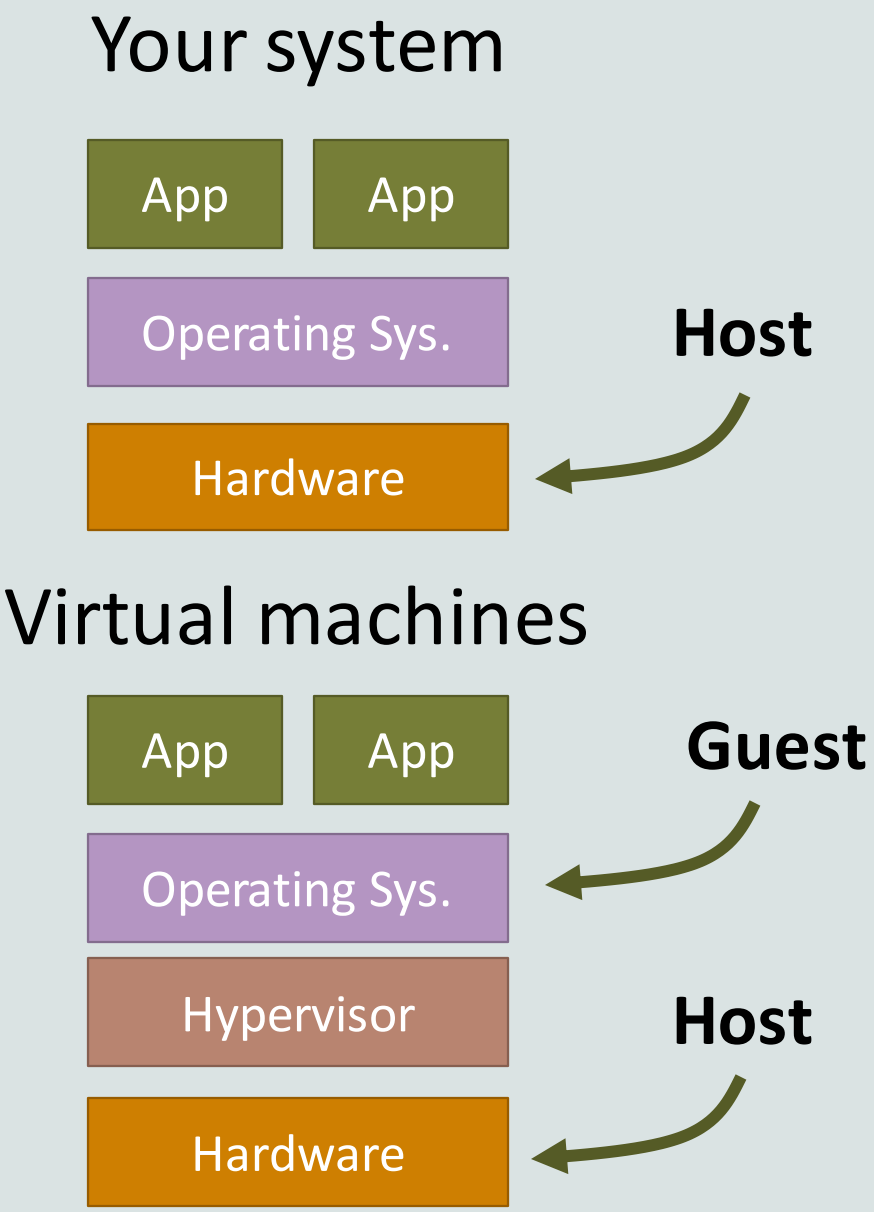
Native execution

Guest: Code running on top of "fake" hardware

OS in virtual machine is guest OS

Running "on top of" hypervisor

Hypervisor is emulating hardware



gem5

Nomenclature

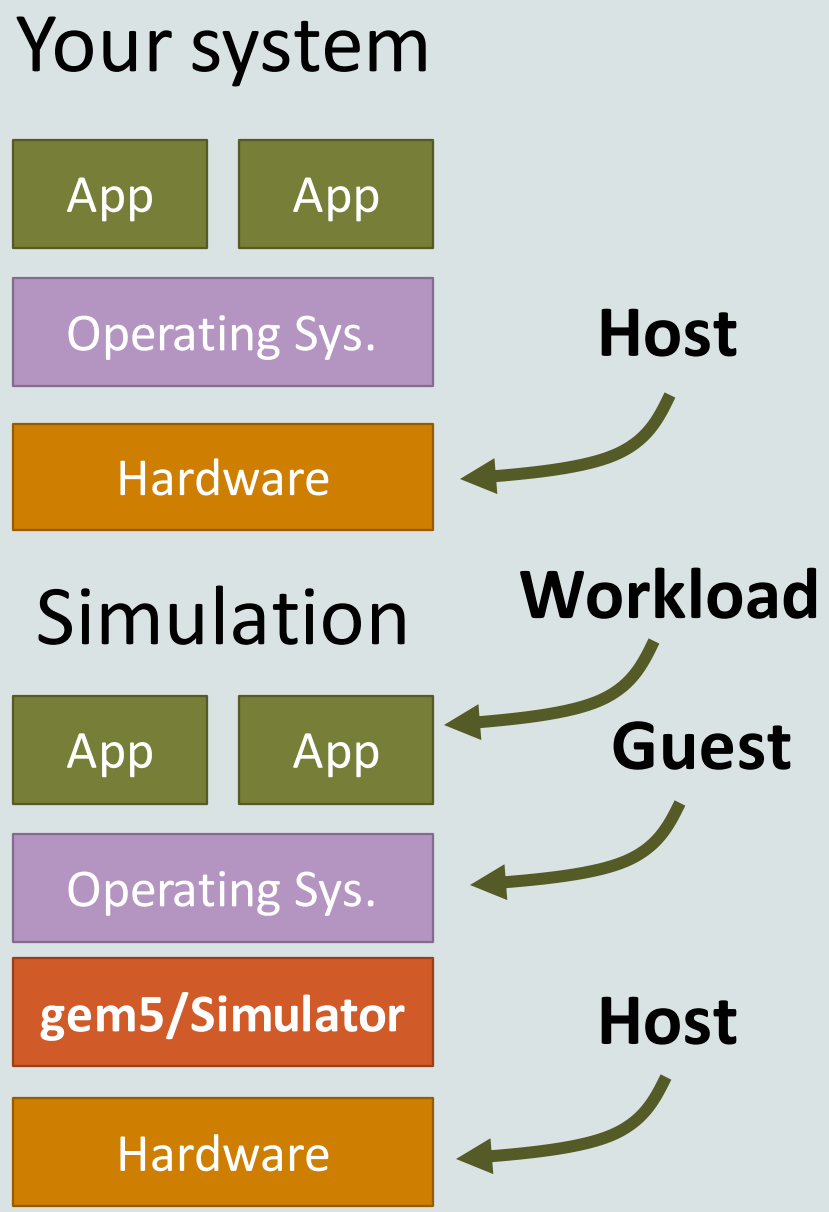
Host: the actual hardware you're using

Simulator: Runs on the host
Exposes hardware to the guest

Guest: Code running on *simulated* hardware
OS running on gem5 is guest OS
gem5 is simulating hardware

Simulator's code: Runs natively
executes/emulates the guest code

Guest's code: (or benchmark, workload, etc.)
Runs on gem5, not on the host.



Nomenclature

Host: the actual hardware you're using

Simulator: Runs on the host

Exposes hardware to the guest

Simulator's performance:

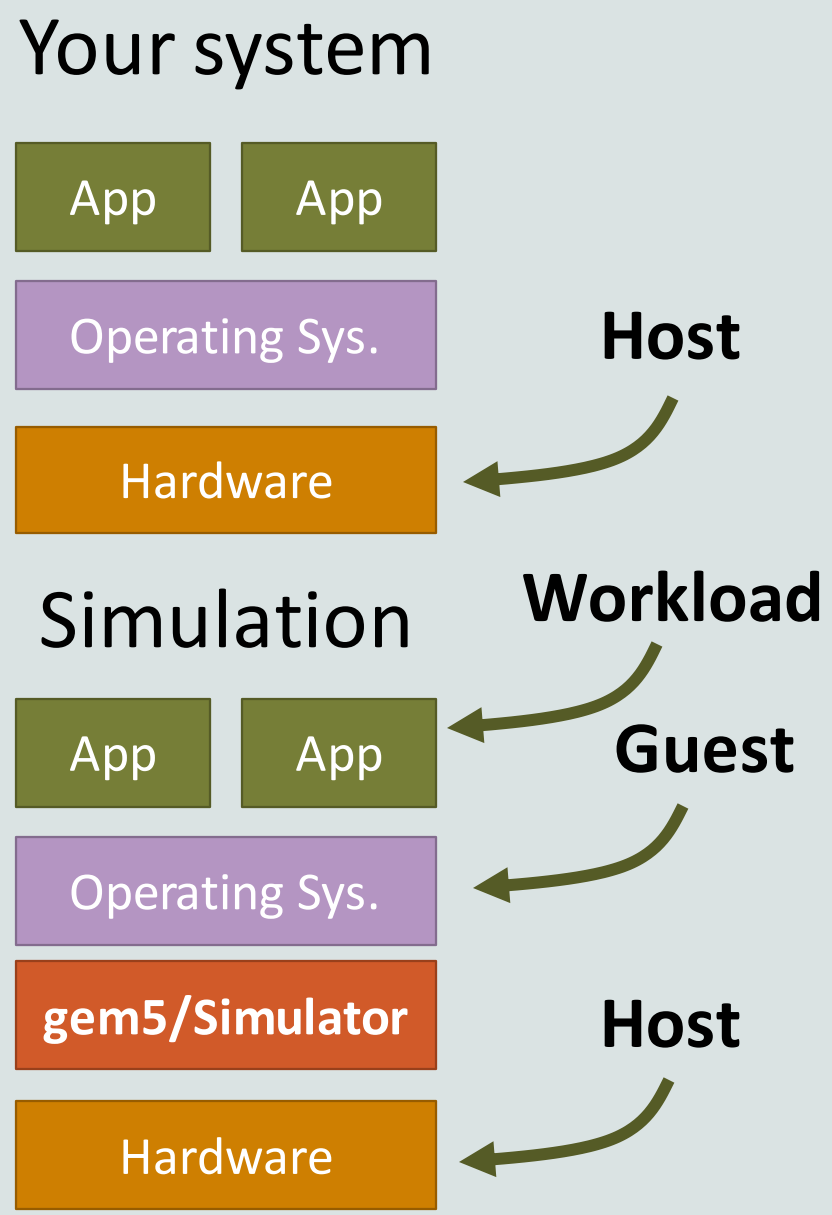
Time to run the simulation on host

Wallclock time as you perceive it

Simulat*ed* performance:

Time predicted by the simulator

Time for guest code to run on simulator



Tradeoffs

Development time: time to make the simulator/models

Evaluation time: wallclock time to run the simulator

Accuracy: How close is the simulator to *real* hardware

Coverage: How broadly can the simulator be used?

	Development time	Evaluation time	Accuracy	Coverage
functional simulation	excellent	good	poor	poor
instrumentation	excellent	very good	poor	poor
specialized cache and predictor simulation	good	good	good	limited
full trace-driven simulation	poor	poor	very good	excellent
full execution-driven simulation	very poor	very poor	excellent	excellent

https://www.morganclaypool.com/doi/abs/10.2200/S00273ED1V01Y201006CAC010

What “level” should we simulate?

Ask yourself: What fidelity is required for this question?

Example: New register file design

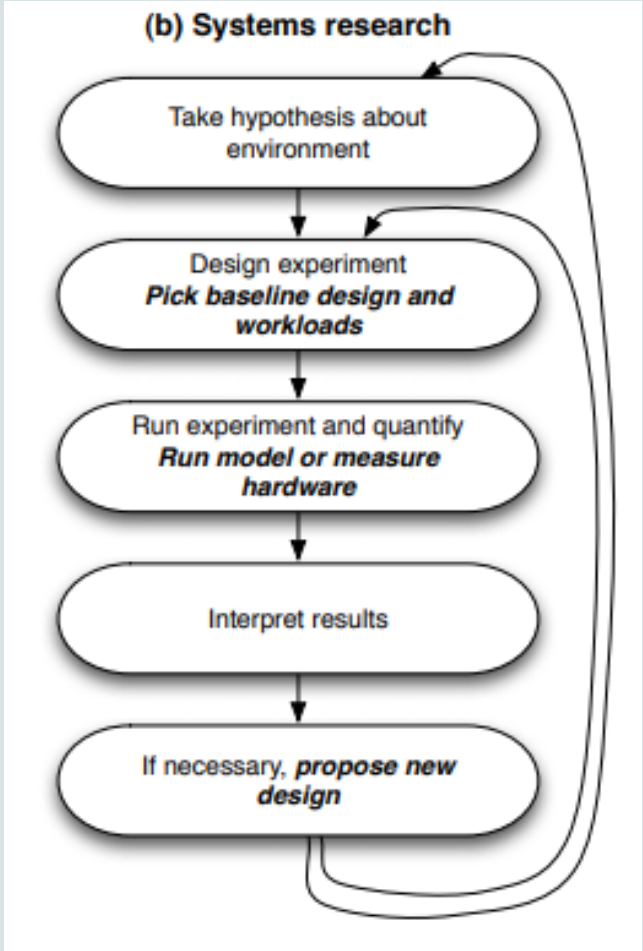
Often, the answer is a mix.

gem5 is well suited for this mix

Models with different fidelity

Drop-in replacements for each other

“Cycle level” vs “cycle accurate”



RTL simulation

RTL: Register transfer level/logic

- The “model” is the hardware design
- You specify every wire and every register
- Close to the actual ASIC

This is “***cycle accurate***” as it should be the same in the model and in an ASIC

Very high fidelity, but at the cost of configurability

- Need the entire design
- More difficult to combine functional and timing

Cycle-***level*** simulation

Models the system cycle-by-cycle

Often “event-driven” (we’ll see this soon)

Can be quite accurate

- Not the *exact same* cycle-by-cycle as the ASIC, but similar timing

Easily parameterizeable

- No need for a full hardware design

Faster than cycle-accurate

- Can “cheat” and functionally emulate some things

Accelerating simulation

Parallelism

- Difficult to make cycle-level simulators parallel
- Can run many different simulations in parallel (throughput, not latency)
- May be able to get parallelism with multiple nodes? (See SST ☺)

Use FPGAs

- Quite useful for RTL simulation (See FireSim)
- Can be cycle-accurate
- Requires close to hardware design (RTL, HLS, etc.)
- Less flexible than CPU-based simulation

Summary & important terms

Functional simulation

Just executing the code correctly. No timing.

Timing/execution-based simulation

The timing of each operation is simulated with the functionality

Full system

All guest code is run inside the simulator

Host vs guest

Host: the thing on your desk. The simulator runs on the host.

Guest: The application you're simulating.

The guest application/OS runs on the simulator.