Benchmarking and Simulation

Computer Architecture: Design and Simulation
CMU 15-346, Spring 2022

Why design a computational device?

- Achieve a particular function or goal
  - Computer architecture is about designing something more efficient

- So how do you know your idea is good?

Benchmarks

- Run program(s) on lots of machine configurations
  - Know where your program runs best

- Why is this difficult?
  - Expensive hardware
  - Load may not be representative
  - Code may be proprietary
  - …

- Thus benchmarks are proxy programs for
  - Architects, marketing, users

Benchmarks Suites

- SPEC
- PARSEC / SPLASH
- EEMBC / dhrystone
- TPC-C/E/H/…

Different Metrics of Efficiency

Plane	DC to Paris	Top Speed	Passen-gers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

Performance of Benchmarks (comp arch)

- MIPS
  - ISAs are not equal
- GFLOPS
  - Code is not equal

Run time

- **CPU Time**
  - **Cycle count x cycle time (i.e., inverse of clock rate)**
  - **Cycle count = CPI x IC**

$$\text{Cycles per instruction} = \text{CPI} = \frac{\text{clock cycle count}}{\text{Instruction Count}}$$

$$\Rightarrow \text{clock cycle count} = \text{CPI} \times \text{IC}$$

$$\therefore \text{CPU time} = \text{IC} \times \text{CPI} \times \text{CT}$$

- **Improvements**
  - **IC (instruction count)**
    - **Compiler, algorithms**
  - **CPI**
    - **ILP**
  - **Cycle time**
    - **More pipelining, device improvements, simpler ISA**

RISC vs CISC

- **Reduced instruction set computers**
  - **Impact on CPU time (udd)**
- **Complex instruction set computers**
  - **Impact on CPU time (duu)**
- **Which is best?**
  - **It depends**

CPU Time lies

- Synchronization and parallel code

- Speedup

$$\text{Speedup} = \frac{\text{Time baseline}}{\text{Time with X}}$$

Amdahl's Law

- 1- s – a component of the program
- p – speedup of that component

$$\text{speedup} \leq \frac{1}{s + \frac{1-s}{p}}$$

- The more time something takes
  - The more speedup small improvements make

- Example:
  - $1 Billion for computer that does foo 100,000x faster
  - Foo is 90% of your workload

Gustafson's Law

- Maybe faster components can do more rather than less (time)

- How many more foo can we run in a day?

Example of means

	A	B
Prog 1	4	2
Prog 2	4	7
Hmean	4	3.11
Amean	4	4.5

Table holds rates in instructions per second

- Which is faster, A or B?
  - Consider running an average instruction from prog 1 followed by one from prog 2:
  - for A: (1/4 + 1/4) = 1/2
  - for B: (1/2 + 1/7) = 9/14

  Convert rates to times – then we can add the times together
  - A runs the two instructions faster (1/2 < 9/14), thus A is better
  - Now look at the harmonic mean (Hmean) vs. the arithmetic mean (Amean).
    - Hmean says A has a higher rate than B (4 vs. 3.11) so <u>A is better</u>
    - Amean says B has a higher rate than A (4.5 vs. 4) so B is better, but that's wrong!
- If you used the wrong method to combine the numbers, you would buy the slower machine!
- Note also that the definition of harmonic mean is just the average of the rates converted to times, then converted back to rates

Simulation

- How do we evaluate an idea?

Evaluating an architectural idea: simulation

- Architects evaluate architectural decisions quantitatively using chip simulators
  - Let's say an architect is considering adding a feature
  - Architect runs simulations with new feature, runs simulations without new feature, compare simulated performance
  - Simulate against a wide collection of benchmarks

- Design detailed simulator to test new architectural feature
  - Very expensive to simulate a parallel machine in full detail
  - Often cannot simulate full machine configurations or realistic problem sizes (must scale down workloads significantly!)
  - Architects need to be confident scaled down simulated results predict reality (otherwise, why do the evaluation at all?)

Simulation Accuracy

- Accuracy versus precision
  - Precise is being consistent
  - Accurate is being correct

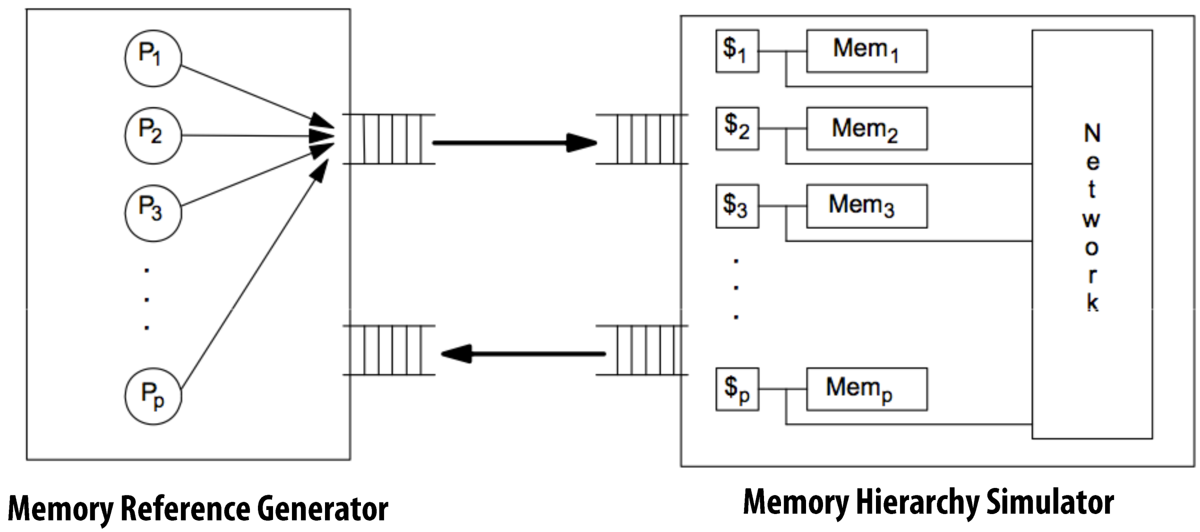
- Paper reference for simulator (in)accuracies

Level of detail

- Transistor
- Gate
- Circuit
- Functional

- Simulations always have to assume some things work

Execution-driven simulator



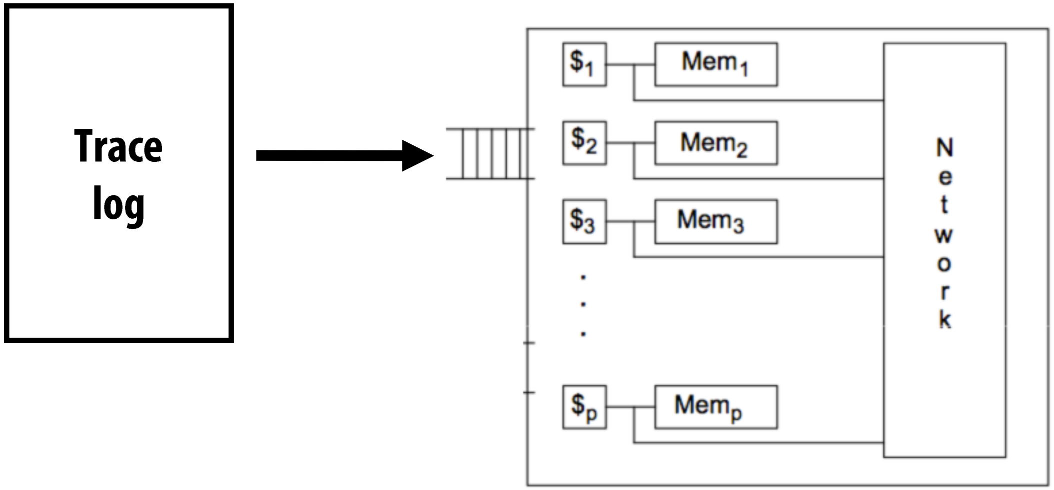
- Executes simulated program in software
  - Simulated processors generate memory references, which are processed by the simulated memory hierarchy

- Performance of simulator is typically inversely proportional to level of simulated detail
  - Simulate every instruction? Simulate every bit on every wire?

Trace-driven simulator

- Instrument real code running on real machine to record a trace of all memory accesses
  - Statically (or dynamically) modify program binary
  - Example: Intel's PIN (www.pintool.org)
    Contech (http://bprail.github.io/contech/)
  - May also need to record timing information to model contention in subsequent simulation

- Or generate trace using an execution-driven simulator
- Then play back trace on simulator



Event-driven Simulation

- Does the simulation happen on time or events?
  - Cache simulator
    - Does the simulator check each cycle if the current memory request is complete?
    - Does the simulator skip in time to when the request completes?
  - Can be complicated when different simulator components want different notions of "time"

Analytical Model

- **Hybrid approach**
  - Where possible use static analysis of input
  - Use performance counters, instrumentation or simulation for remaining inputs
  - Combine effects of different inputs independently

$$CPI = CPI_{steadystate} + CPI_{brmisp} + CPI_{icachemiss} + CPI_{dcachemiss}$$

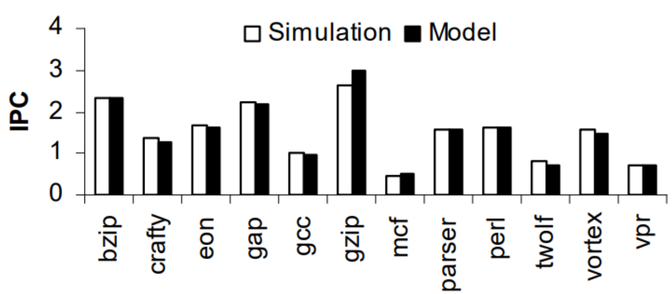


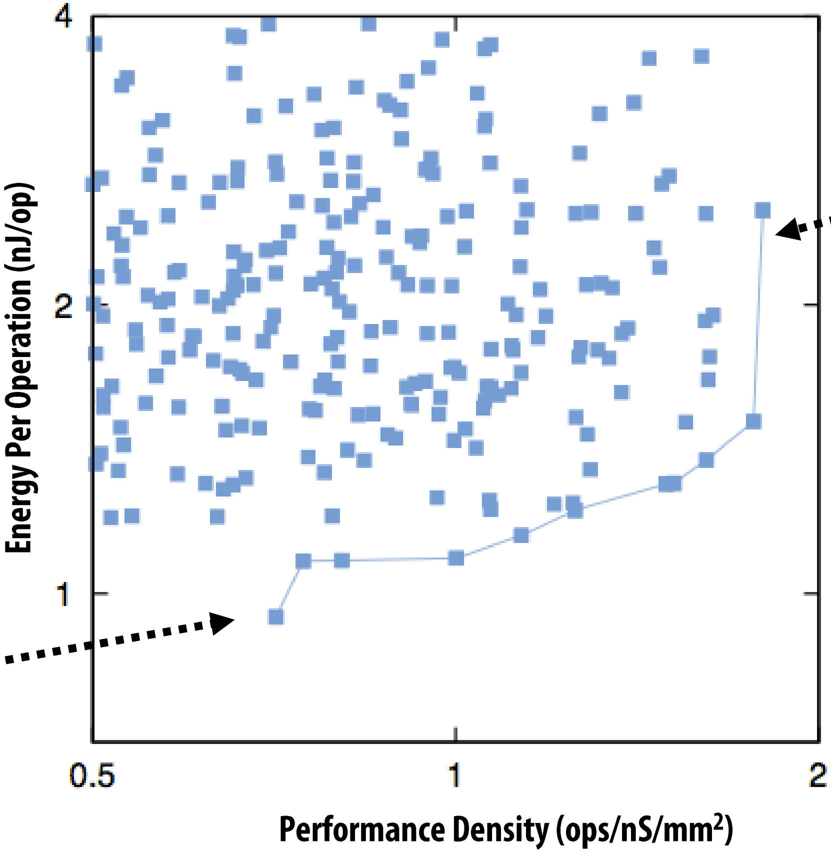
Figure 15: Comparison of performance predicted by our first-order model and from simulation.

A First-Order Superscalar Processor Model, Karkhanis and Smith, ISCA31, 2004.

Architectural simulation state space

- Another evaluation challenge: dealing with large parameter space of machines
  - Num processors, cache sizes, cache line sizes, memory bandwidths, etc.

Pareto Curve: (here: plots energy/perf trade-off)



Maximum performance per unit chip area (but high energy consumption)

High energy efficiency (but lower performance density)

Reducing Simulation Time

- Simulation fidelity
- Simulation inputs
  - Shorter traces
  - Smaller workloads
  - Targeted points (SimPoints, Barrier Interval)