Simulators and such…

Mats Brorsson & Mladen Nikitovic
ICT
Dept of Electronic, Computer and Software Systems (ECS)

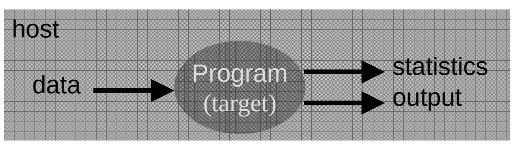
1

Outline

- What defines a simulator?
- Why are simulators needed?
- Classifications
- Case studies
- Benchmark suites
- New challenges
- References

2

What defines a simulator?



*From Wikipedia: "A **simulation** is an imitation of some real thing, state of affairs, or process. The act of simulating something generally entails representing certain key characteristics or behaviors of a selected physical or abstract system."*

- Simulation gives you the opportunity to model non-existing components, to collect statistics about its performance etc.

3

What about emulation?

Wikipedia:

A **software emulator** allows computer programs to run on a platform (computer architecture and/or operating system) other than the one for which they were originally written. Unlike simulation, which only attempts to reproduce a program's behavior, emulation attempts to model to various degrees the state of the device being emulated.

4

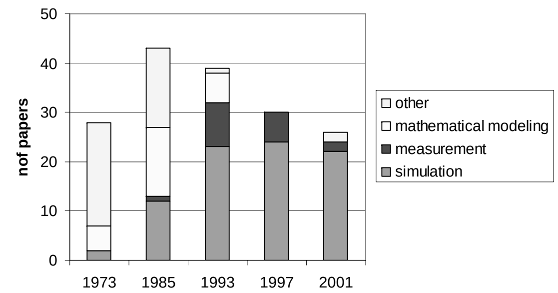
Why use simulation?

- Understanding real systems
- Higher degree of *observability*
- Less dangerous
- Fault injection
- Debugging
- Prototype hardware before expensive implementations
- Develop software in parallel with hardware

5

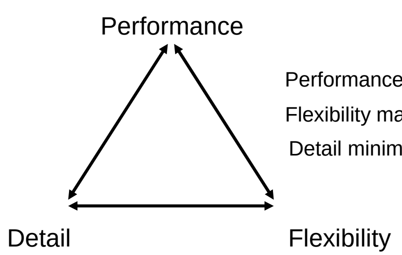
Research trends

Performance evaluation methodologies used in a sampling of papers frm ISCA [K. Skadron]



6

Simulator classifications



Performance speeds design cycle

Flexibility maximizes design scope

Detail minimizes risk

- Design goals drives optimization towards any corner in the triangle

7

Simulator categories

- Full-system vs microarchitecture simulators
  - Full system simulators are slow, but models OS-overheads and a more "complete picture" of the results
  - Architecture simulators are faster, usually more accurate than full system simulators.
- Functional vs performance simulators
  - functional sims are faster but less accurate
  - functional sims are also more flexible
  - performance allow accurate modeling of more complex architectures with out-of-order execution

8

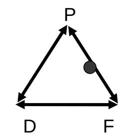
More simulator categories

- Interpreters vs. instrumented code
  - Interpreters can support multiple targets, thus are more flexible
  - Intstrumented code runs much faster on a host than using an interpreter, however one must watch out for probing effects
- Trace-driven vs execution-driven simulators
  - traces are pre-recorded streams of instructions, which allows for a deterministic simulation each time
  - execution-driven simulations allows exploration of speculative execution and also side-effects of the operating system

9

SimpleScalar

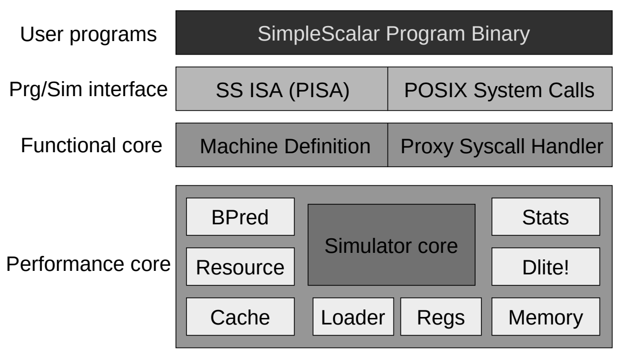
- A microarchitectural simulator suite (T. Austin 92')
  - www.simplescalar.com



- Development tools
  - Compilers, assembler, linker & libraries
  - All source code included
- Simulators
  - Functional and performance simulators
  - Execution and trace-driven
  - Trace genarator

10

Simplescalar structure



11

Simplescalar simulator suite

Execution & trace-driven simulators

sim-fast	functional simulation
sim-safe	sim-fast with error detection
sim-cache	functional cache simulator
sim-cheetah	cache simulator (multiple configurations)
sim-outorder	performance out-of order execution

Detail

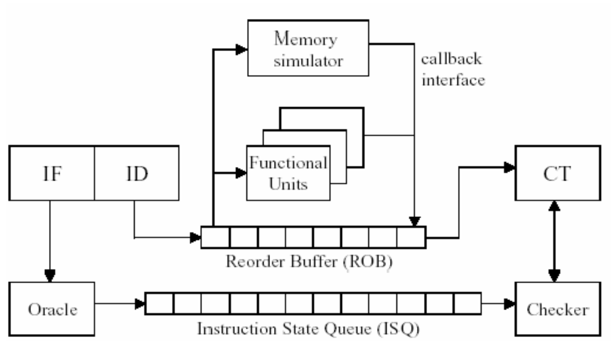
Performance

Trace generator

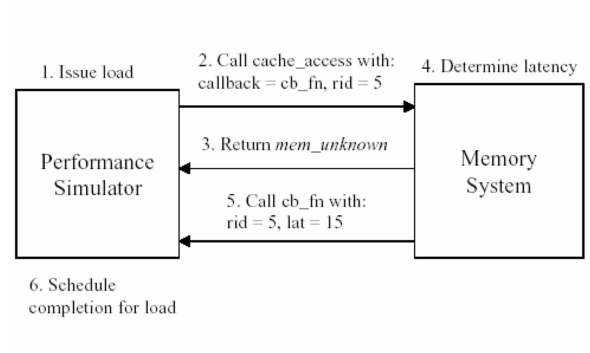
sim-eio	i/o-tracing & check-pointing

12

MASE structure

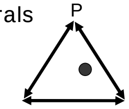


MASE callback interface

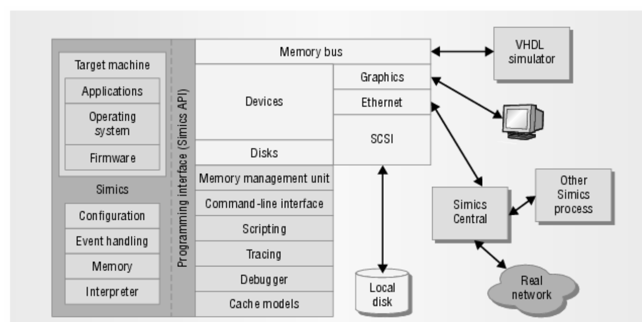


SimICS

- Commercial Full-system simulator (SICS 92´)
  - models entire operating system including uni/multi-processors, caches, networks, and peripherals
- User can attach own modules through interfaces
- Many targets and hosts supported
  - Powerpc, Sparc, x86, x86-64, Alpha, ARM, IA32/64, MIPS
  - Linux, Windows, Solaris
- Uses *images* to load system configurations
- Supports checkpointing and tracing
- http://www.simics.net, http://www.virtutech.com



SimICS Architecture



Simics' three modes

- Fast mode: No cache simulation. Just in time compilation.
- Normal mode: Simple cache simulation.
- Out-of-order mode: MAI (Micro Architecture Interface). Supports speculative execution, such as, branch and valute prediction. Cache simulation etc.

17

Target vs. host

- The *target* is the simulated system
- The *host* is the computer that runs Simics
- The different prompts:
  - target# — the targets prompt: root on target system
  - host$ — the host prompt: user on the host system (xterm etc)
  - simics> — the Simics prompt

18

Simulators summary

- SimpleScalar
  - Free uniprocessor simulator w tools, can simulate cache hieriarchy with a cycle-accurate processor model
- Simics
  - Commercial full-system uni/multi-processor simulator, flexible and portable, reasonably fast
- Simics extensions
  - Multifacet GEMS: http://www.cs.wisc.edu/gems/
  - Simflex: http://www.ece.cmu.edu/~simflex/

19

Benchmark suites : Spec

- Standard Performance Evaluation Corporation
- Consists of many categories e.g CPU
- Ver. 92, 95, 2000, current version 2006
  - CINT 2006 Integer benchmarks (12)
  - CFP 2000 Floating point benchmarks (17)
- Base vs Peak depending on optimization level
- Input versions
  - reference
  - reduced input (CPU 2000, obsolete)

20

Benchmark suites : MiBench

- Freely available benchmark suite, resembles EEMBC, a standardized (non-free) suite
  http://www.eecs.umich.edu/mibench/
- Range of 1 billion executed instructions
- Consists of 35 applications from 6 different areas in embedded computing:
  automotive/industrial (sorting)
  consumer (image and text compression)
  office (document-related operations)
  network (routing and encryption/decryption)
  security (encryption/decryption)
  telecom (encryption/decryption, speech encoding)

Benchmark suites : Mediabench

- Purpose is to represent applications common in embedded multimedia and communication environments
- Voice compression, image rendering & compression, encryption/decryption of text,
- Ranges from few millon to a billion executed instructions, quick to simulate
- Free source code at
  http://cares.icsl.ucla.edu/MediaBench/

Possible problems

- Can I trust the results that my simulator has produced?
- How do I verify my results?

New challenges: Accuracy

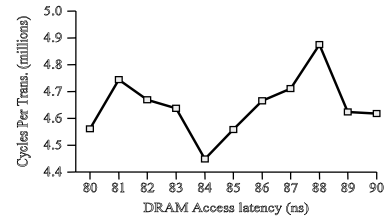
- *Absolute accuracy* shows how close one is to the real world whereas *relative accuracy* shows how correct a model is between different configuration settings
- Absolute accuracy is increasingly complex to achieve due to modeling limitations such as timing variations due to physical phenomenons
- Therefore, relative accuracy is more feasable to achieve today, but harder to verify

New challenges: Increased complexity

- Increasingly complex architectures are modeled e.g multiprocessor systems with complex networks, operating system behavior, running multiple processes/threads and so on..
- With current simulation speed one would wait years for a simulation session to finish
- Several ideas to reduce simulation time
  - Reduce binary size
  - Vary simulator accuracy during a session
  - Fast forward between sections of code

New challenges: Workload variability

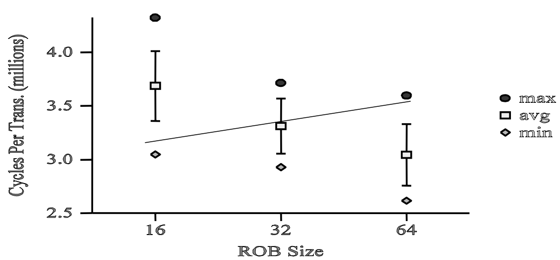
- A workload consists of multiple processes or threads executed at various instants of time
- Running one workload scenario will not give you an accurate result due to variations in I/O response, OS services and schedulers



[E. Larson]

New challenges: Workload variability

- One need to run a workload scenario multiple times to increase confidence in results



References

- Addressing workload variability in architectural simulations,
  E. Larson et al., IEEE Computer, 2001.
- Challenges in Computer Architecture Evaluation,
  K. Skadron et al., IEEE Computer, 2003.
- Simulating a 2M commercial server on a 2K PC,
  A. R. Alameldeen et al., IEEE Computer, 2003.
- SimpleScalar: An infrastructure for computer System modeling,
  T. Austin et al, IEEE Computer, 2002.
- MASE: A novel infrastructure for detailed microarchitectural modeling
  E. Larson et al., Int. Symp. on Performance Analysis of Systems and Software, 2001.

References

- Asim: A performance model framework
  J. Emer et al., IEEE Computer 2002.
- Simics: A full system simulation platform
  P. S. Magnusson, IEEE Computer 2002.
- RSIM: Simulating shared-memory multiprocessors with ILP processors
  C. J. Hughes, IEEE Computer, 2002.
- Picking statistically valid and early simulation points
  E. Perelman, Intl. Conf. On Parallel Architectures and Compilation Techniques (PACT), 2003.