Simics: A Full System Simulation Platform

Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav H?llberg, Johan H?gberg, Fredrik Larsson, Andreas Moestedt, Bengt Werner

Presented for ECE 259 by Mohammad Mottaghi

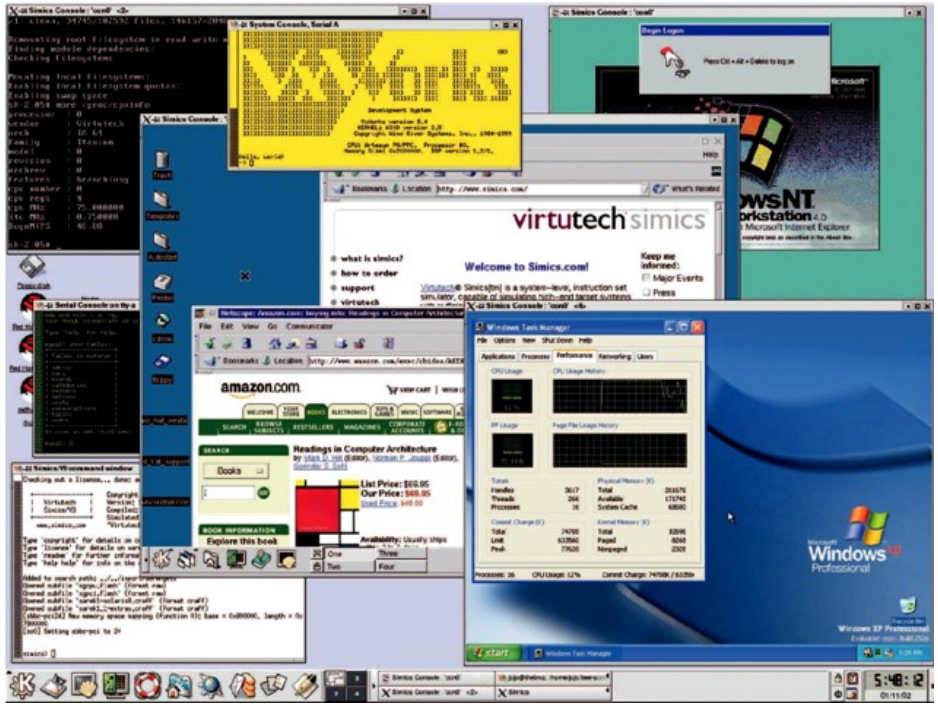
March 29th 2010

Simulation tradeoffs

- A perfect model: simulation with total accuracy
  - problems: cost, time to completion, specification inaccuracies
  - Serious problem: workload
    - Highly accurate models → very small toy benchmarks
    - Result: Accurate answers to irrelevant questions
- Simics: Balance between accuracy and performance → Sufficiently:
    - Abstract
    - Functional accurate → commercial workloads
    - Time accurate → interface with hardware

As a flexible full system simulator

- Simics supports a broad variety of tasks:
  - microprocessor design, operating system development, fault injection …
- Simulates different processors / operating systems / devices
  - Multiple sessions, each separate simulation
  - Can be networked together
    - Simics central
  - Sample:
    - Fast enough → interactive
    - 30 secs → load forum webpage
- Record & timestamp events(User inputs), Access mem traffic anywhere, breakpoints …



Simics applications

- Reduces time-to-market
  - relaxes many dependencies by providing a single platform across the development cycle
  - each task can begin within an abstract context
- Applications
  - Microprocessor design
    - Simulates processors @ the instruction-set level
    - Resolves traditional trace-based simulation limitations
      - Cache and I/O timing + interleaving mem ops + OS mem mgmnt
    - Data dependence tracking and roll back support
    - Provides functional model (no timing model)
  - Operating system emulation
  - OS development
  - High-availability testing

Simics applications, cntd.

- Memory studies
  - represented by one or more memory spaces (address spaces)
    - Examples: physical cacheable memory, PCI bus spaces
  - extension: connecting timing model
    - Mem op latency, cache input, …
- Device development
  - Can communicate with external programs
    - A single device is simulated by an ext program
- Debugging
  - Reproduce errors → repeat exact event flow
    - play back KB, mouse, NWK traffic
  - Implement advanced breakpoints
    - correlated locks, timing breakpoints
  - Use well-known debugger interface
    - gdb through gdb-remote

Simics features 1

- **Simics Central**, a tool that:
  - synchronizes the virtual time between Simics simulators
  - distributes simulated traffic between nodes
    - Acts as a router
  - halts the simulation if one process consumes cycles slower than the rest
    - NWK sim speed → slowest Simics process
  - Entire distributed sim → fully deterministic
- **Devices**
  - For each target: a set of devices are supported → OS boot
    - e.g. x86(PC): 8254, 8237, 8259 (int contoller) …
  - Supports multiprocessor for all targets
- **Interfacing to other simulators**
  - Interface to a clock-cycle-accurate model written in Verilog

Simics features 2

- **Simics API**
  - > 200 functions + > 50 interfaces + data types
  - Extensible → write plug-in device models
- **Memory**
  - Mem ops: biggest perf. challenge for simulator
  - Simulator translation cache (STC)
    - Pointer to simulated mem – indexd by virtual adr
- **Event handling**
  - 2 event queues per processor: Step Q, Time Q
    - allows Simics to mix event-driven and time-driven components
  - Events in Step Q → fired after n PC steps
  - Events in Time Q → fired after n CPU clock cycles

Simics features 3

- **Configuration**
  - describe target sys → object-oriented conf. lang.
  - To add a device
    - write a class using the Simics API → loadable module
    - define an object of that class in the configuration file
- **CLI and scripting**
  - Simics controlled through command line interface
    - Also a runtime python env.
  - Scripts can be tied to certain events
    - e.g. TLB misses, I/O operations
  - [figure 5: example]

Questions and thoughts

- Performance Analysis toolkit: missing?
  - Simics: CPU + Capabilities of compiler (entire prog)
  - How much does Simics exploit this?
    - For speeding up the simlulation
    - Could give hints for Ideal ILP vs. real ILP
    - Ideal vs. real cache hits rates
- Hardware emulation?
  - Can we use FPGA or ASIC emulators to speedup simulation? (of a non-existing CPU)
- Time-to-market reduction: not as good as advertised, dangerous?
  - Might hide a critical failure of the target system
    - Data dependant heat distribution of a particular layout
  - Presumed to be successful
  - Upon this (false) assumption, investment on the next ongoing phases (compiler, os, …)