Extending the Sparc-Sulima Computer Simulator

Peter Strazdins,
(with Bill Clarke and Adam Czejowski)

Department of Computer Science,
Australian National University,
and
The ANU-Fujitsu CAP Program

http://cs.anu.edu.au/Student/comp3800/sched3800.html

18 Oct 2002
1 Talk Outline

1. talk outline

2. performance analysis of computer systems and motivations for simulation (of computer systems via software)

3. introduction to Sparc-Sulima

4. structure of Sparc-Sulima

5. status of the Sparc-Sulima project

6. extending Sparc-Sulima
2 Motivations: Performance Analysis of Computer Systems

• One cannot understand the design tradeoffs or performance of multiprocessors without understanding the interaction of algorithms and architecture – John L. Hennessey, 1999
• systematic performance analysis techniques now drives design
  • driven by key applications (commercial and scientific)
  • includes the use of performance instrumentation libraries:
    • here, we insert calls to manipulate hardware event counter registers in the application’s source code
    • events affecting performance include branch mispredictions, TLB misses, cache misses etc (usually, memory-related events are the most important!)

and computer simulation:
  • use software to interpret programs and predict their performance
  • can have full visibility: actual H/W may not count all desired events
  • the simulated computer may not even exist! (essential for design!)
  • can vary parameters for architectural studies (eg. cache size)
  • for operating systems development / debugging
3 Introduction to Sparc-Sulima

- a complete machine simulator for an UltraSPARC SMP
  - UltraSPARC: an implementation of the 64-bit SPARC V9 architecture
  - highly complex instruction set – hardly RISC anymore!
- interprets instruction-by-instruction:
  \[
  \text{application (executable + data)} \rightarrow \text{simulator} \rightarrow \text{output, event counts}
  \]
  \[
  \uparrow \quad \text{simulated computer state} \quad \downarrow
  \]
- some implementation highlights:
  - used SLED toolkit to generate instruction decoder (6K lines of C; rather slow)
    - no bugs found in decoder once integrated! (found bug in Solaris assembler!)
  - uses a heavily Object-Oriented approach (20K lines C++ code)
  - extensively optimized: a slowdown of \(\approx 200\) (state-of-the art)
    - for running simulator on an UltraSPARC, use inline assembler to capture exact semantic effects of instruction (eg. floating point, graphics)
• better reliability, speed; loss of portability
  • ‘cache’ expensive calculations repeatedly carried out by simulator
• advanced debugging infrastructure: instruction-, exception- and call-level tracing
• has a Python scripting interface, allowing access to all main simulator components
  • using SWIG (Simplified Wrapper and Interface Generator)
  • eg. runsim-swig.sh MatFact.sim -w 64 -C 100

  runsim-swig.sh contains:

  ... import SparcSulima
  im = SparcSulima
  bus = sim.SPARCV9SystemBus("bus")
  cpu = sim.UltraSAN("cpu", 200L)
  cpu.bus = bus
  user = sim.UserSim("user")
  user.exe_filename = "$1"
  user.set_args(['printf '"%s",' "$@";printf "\n"'])
  prom = sim.ROM("prom")
  ...
  sim.reset()
  sim.run()
4 Structure of Sparc Sulima

Clock
- event queues, polling

BasicModule
- logging, scripting interface

Module
- install()
- reset()

CPU
- generic CPU class
  - deliver_interrrupt()
  - run()

SparcV9CPU
- US CPU state (e.g., registers)
  - ASI, exception defns

SparcMMU
- TLBs, ASI-accessible registers
  - fetch()
  - data_access()

Caches
- I-, D-, and E-cache implementation

SparcV9Bus
- generic bus

SparcV9SystemBus
- memory, bus registers
  - devices, console

UltraSAN
- interpreter ‘core’
  - simulator-specific state

UserSim
- system call emulation
5 The Sparc-Sulima Project

- **2002–2002**: the ANU-Fujitsu CAP Program:
  - 08/00: begun design
    - based on the Sulima simulator infrastructure (DiSY Group, UNSW)
  - 12/01: 1st code release (under GPL/BSD), + Developer’s Manual
  - 08/02: 2nd second release
  - has full functionality of an UltraSPARC I/II computer system (down to 2nd-level buses)

- **2003–2005**: ARC Grant on optimization of quantum chemistry applics. on non-uniform memory access SMPs (Rendell/Strazdins/ Gaussian/Sun)
  - extend for UltraSPARC III, for performance analysis a large-scale applic on a large-scale SMP
  - 1 Research Associate, 1 PhD + ...
6 Extending Sparc-Sulima

- ‘generic’ extensions (click here for more details)
  - extend for configurability: easy change of architectural parameters
  - may involve some re-design; re-design for maintenance may also be valuable
  - extend for portability: run on many platforms
  - advanced debugging support: eg. \texttt{gdb} interface
- add checkpointing support: important for large applications
  - challenging because of complex OO structure
- systematic (regression) testing (challenging: test space is enormous!)
- extend for a cluster configuration (some non-trivial research issues)
- such simulators are very complex software systems: good SE approaches and practices are important!