18-640 Foundations of Computer Architecture

Recitation 1:

Simulation Basics and gem5 Tutorial

Guangshuo Liu and Naman Jain
September 3, 2014

References taken from http://www.m5sim.org/Tutorials

➢ Recommended Reference:
www.gem5.org

Electrical & Computer ENGINEERING

Announcements

- Lab group signup
  - Very few have signed up!
  - Follow the link on BlackBoard
- gem5 step-by-step instructions
  - Published on BlackBoard
  - Install gem5 and play with it before Lab 1 is out

Outline

- Simulation basics
  - Why simulation?
  - Why gem5?
- gem5 introduction
  - How to use gem5?
- Overview of all labs and homeworks
  - How are labs designed?
  - How are homeworks designed?
  - How will TAs grade your lab submissions?
- Other resources

Outline

- Simulation basics
  - Why simulation?
  - Why gem5?
- gem5 introduction
  - How to use gem5?
- Overview of all labs and homeworks
  - How are labs designed?
  - How are homeworks designed?
  - How will TAs grade your lab submissions?
- Other resources

What we are doing?

- Objective: to study a computer system
- Problem: the system might not be available
- Solutions
  - Build it – hardware prototyping
  - Model it – mathematics, statistics, machine learning
  - Simulate it – imitate what happens in hardware using software

Comparison

- Three factors
  - Complexity to build
  - Runtime
  - Accuracy

	Hardware prototyping	Mathematical modeling	Simulation
Advantages	• Runs fast • Accurate	• Runs fast • Flexible • Easy to build	• Flexible • Complexity is usually manageable • Can be accurate
Disadvantages	• Expensive in money and time • Not flexible	• Inaccurate?	• Runs slowly • Can be inaccurate

Simulator

- Input
  - Hardware system configuration
  - Software (instruction traces or executables)
- Output
  - Hardware performance counter measures
  - Timing
  - (Software output)
  - (Silicon area)
  - (Power / energy)

Simulator Classification – Complexity

- Full-system
  - Simulate the entire computer system
  - We can run virtually all the operating systems and software
  - Examples: gem5, simics
- User-level
  - Only simulate user-space code
  - OS is not simulated
  - Some system calls might be modeled (system call emulation)
  - Examples: gem5, SimpleScalar, Sniper

Simulator Classification – Level of Abstraction

- Instruction set
  - Simulates the functionality of each instruction
  - Does not simulate micro-architectural timing
  - Example: simics
- Micro-architecture
  - Does speculative, out-of-order multiprocessor timing simulation
  - May not implement functionality of full instruction set
  - Examples: gem5, SimpleScalar

Why gem5?

- Powerful
  - Full-system, cycle-accurate in micro-architectural simulation
- Flexible
  - Full system mode
  - System call emulation mode
- Easy to find help
  - Widely used in computer architecture research
  - Active community – you can register for their mailing list (gem5-users@gem5.org)
- Open-source and Free
  - Your contribution to gem5 is highly encouraged
  - But please do not submit lab solutions!

Outline

- Simulation basics
  - Why simulation?
  - Why gem5?
- gem5 introduction
  - How to use gem5?
- Overview of all labs and homeworks
  - How are labs designed?
  - How are homeworks designed?
  - How will TAs grade your lab submissions?
- Other resources

Build System

❖ Install Linux Ubuntu

Consider using VirtualBox or VMWare, if not using Linux system

❖ Download gem5 from our private release (AFS)

Copy `/afs/ece.cmu.edu/class/ece640/project/gem5.tar.bz2` in your system

Don't know how to access AFS:

- Use ECE machines to login, example, ece001.ece.cmu.edu
- Ask help@ece.cmu.edu to create an ECE account, if not already there.
- Login to Unix Andrew machines would not work!

❖ Untar file

```
tar -xvjf gem5.tar.bz2
```

Build System

❖ Install all dependencies

- `sudo apt-get update; sudo apt-get upgrade`
- `sudo apt-get install mercurial scons swig gcc m4 python python-dev libgoogle-perftools-dev g++ zlib1g-dev`

❖ Ready to build gem5!

- `scons build/X86/gem5.opt`
- Can make the build faster by using 'j' flag, but make sure you have allotted enough RAM and Processors for your virtual machine
  Example, for 4 concurrent processes use, `scons build/X86/gem5.opt -j 4`
- Notice the build target 'opt'

Build Targets

- **gem5.opt** has optimizations turned on and debugging functionality like asserts and DPRINTFs left in.
- This gives a good balance between the speed of the simulation and insight into what's happening in case something goes wrong.
- This version is best in most circumstances.

Binary name	Optimizations	Run time debugging support	Profiling support
gem5.debug		X	
gem5.opt	X	X	
gem5.fast	X		
gem5.prof	X		X
gem5.perf	X		X

Sample Run – SE Mode

```
build/X86/gem5.opt
                configs/example/se.py
                -c tests/test-progs/hello/bin/x86/linux/hello
```

```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Jul 17 2011 19:16:28
gem5 started Jul 17 2011 19:18:16
gem5 executing on zizzer
command line: ./build/ARM/m5.opt configs/example/se.py
Global frequency set at 1000000000000 ticks per second
0: system.remote_gdb.listener: listening for remote gdb #0 on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 3188500 because target called exit()
```

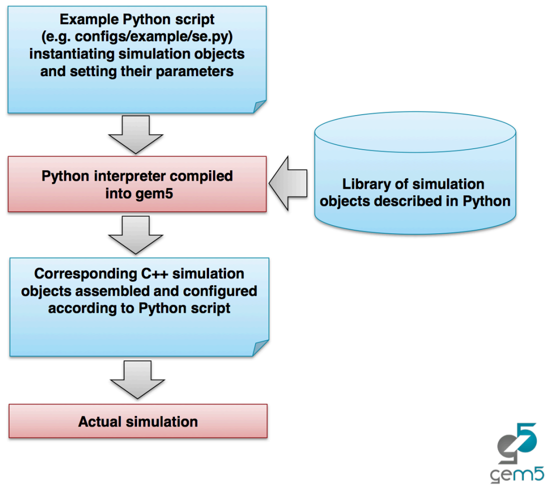

Sample Run – FS Mode

```
build/ALPHA/gem5.opt
                configs/example/fs.py
                --script=runscript.rcS
```

```
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Aug 11 2014 06:45:45
gem5 started Sep  2 2014 00:36:52
gem5 executing on ubuntu
command line: build/ALPHA/gem5.opt configs/example/fs.py --script=runscript.rcS
Global frequency set at 1000000000000 ticks per second
info: kernel located at: /home/namanjain/Documents/gem5/parsec/binaries/vmlinux
Listening for system connection on port 3456
```

Behind the Scenes



Understanding Commands

- Use `-h` flag to check various parameters that can be passed to the script
- Important Ones:
  - `-n Num` number of CPUs
  - `--cpu-type=CPU_TYPE` type of cpu to run with (detailed, inorder etc.)
  - `--caches` to enable caches
  - `--l2cache` to enable L2 Caches
  - `-c binary` executable in SE mode
  - `[-o options]` options to binary, if required
  - `--mem-size=MEM_SIZE` physical memory size

Stats Output

Output by default generated in `m5out/`

`.config` file is the configuration of the simulation performed

`.stat` file is the statistics generated at the end of simulation. You will have to extract data from this file to analyze information.

```
sim_seconds                          0.020540            # Number of seconds simulated
sim_ticks                         20540478500            # Number of ticks simulated
final_tick                        20540478500            # Number of ticks from beginning of simulation
sim_freq                        1000000000000            # Frequency of simulated ticks
host_inst_rate                         192635            # Simulator instruction rate (inst/s)
host_op_rate                           360417            # Simulator op (including micro ops) rate (op/s)
host_tick_rate                       92938401            # Simulator tick rate (ticks/s)
host_mem_usage                         664740            # Number of bytes of host memory used
host_seconds                           221.01            # Real time elapsed on the host
sim_insts                            42574565            # Number of instructions simulated
sim_ops                              79656464            # Number of ops (including micro ops) simulated
…
system.cpu.numCycles                 41080958            # number of cpu cycles simulated
…
system.cpu.committedInsts            42574565           # Number of Instructions Simulated
system.cpu.committedOps              79656464            # Number of Ops (including micro ops) Simulated
system.cpu.cpi                       0.964918            # CPI: Cycles Per Instruction
system.cpu.cpi_total                 0.964918            # CPI: Total CPI of All Threads
system.cpu.ipc                       1.036358            # IPC: Instructions Per Cycle
system.cpu.ipc_total                 1.036358            # IPC: Total IPC of All Threads
```


Outline

- Simulation basics
  - Why simulation?
  - Why gem5?
- gem5 introduction
  - How to use gem5?
- Overview of all labs and homeworks
  - How are labs designed?
  - How are homeworks designed?
  - How will TAs grade your lab submissions?
- Other resources

Overview of Labs

- Labs are biweekly, covering
  - Branch prediction
  - Out-of-order execution
  - Cache
  - Multi-threading
  - (An open-ended one)
- Think yourselves as computer architects
  - Construct new components
  - Profile how well the system works
- Feedbacks are always welcome

Overview of Homeworks

- They are also biweekly
- Each HW will be graded out of 100 points
- Should be done **Individually**
- 4 Assignments, which are designed to be hand in hand with the lectures

Homework No.	Lectures
HW #1	1 - 5
HW #2	6 - 10
HW #3	11 - 16
HW #4	17 - 21

Grading

- Labs make 40% of the total grade
- TA's use grading scripts
  - Make sure your code compile with the support code we provide
  - Make sure your output format follows specifications if there is any
- TA's also read your code
  - Code readability is important
- All submission is based on ECE AFS
  - Your individual and group submission directories will be created
  - Sign up your group early

Outline

- Simulation basics
  - Why simulation?
  - Why gem5?
- gem5 introduction
  - How to use gem5?
- Overview of all labs and homeworks
  - How are labs designed?
  - How are homeworks designed?
  - How will TAs grade your lab submissions?
- Other resources

Other resources

- Andrew File System (AFS)
  https://userguide.its.cit.cmu.edu/services/software/afs-intro/
- CMU Virtual Private Networking (VPN)
  http://www.cmu.edu/computing/network/vpn/
- ECE labs
  https://userguide.its.cit.cmu.edu/resources/labs/