Learning gem5 – Part I

Getting started with gem5

Jason Lowe-Power
http://learning.gem5.org/
https://faculty.engineering.ucdavis.edu/lowepower/

UCDAVIS

What is gem5?

Michigan m5 + Wisconsin GEMS = gem5

"The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture."

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. **The gem5 simulator**. *SIGARCH Comput. Archit. News* 39, 2 (August 2011), 1-7. DOI=http://dx.doi.org/10.1145/2024716.2024718

Tutorial and book are open source!

http://learning.gem5.org/

https://github.com/powerjg/learning_gem5

See a problem?

Open a pull request or issue

Want to add new material? Let me know!

Want to do your own version of this? See http://learning.gem5.org/book/#notes-for-presentations

This tutorial

This is going to interactive

Work along with me for best results

Ask questions!!

Schedule

Learning Part I	8:30 – 10:00
Break	10:00 – 10:30
Learning Part II	10:30 – 12:00
Lunch	12:00 – 1:30
Learning Part IV & III	1:30 – 3:30
Break	3:30 – 4:00
gem5 Best Practices	4:00 – 5:00
Open forum	5:00 – 5:30

Learning Part III:

- Intro to Ruby
- Simple MSI protocol
- Configuring Ruby
- Debugging Ruby

- Memory system objects
- Simple cache model

- Building a simple CPU

UCDAVIS

Building gem5

http://learning.gem5.org/book/part1/building.html

UCDAVIS

Let's get started!

```
> git clone https://gem5.googlesource.com/public/gem5
> cd gem5
> git checkout -b asplos
> scons build/X86/gem5.opt -j5
```

and now we wait (about 8 minutes)

```
> scons build/X86/gem5.opt -j5
```

scons: the build system that gem5 uses (like make). See http://scons.org/

build/X86/gem5.opt: "parameter" passed to scons. gem5's *Sconscript* interprets this. Also, the patch to the gem5 executable.

X86: Specifies the default build options. See `build_opts/*`

opt: version of executable to compile (one of debug, opt, perf, fast)

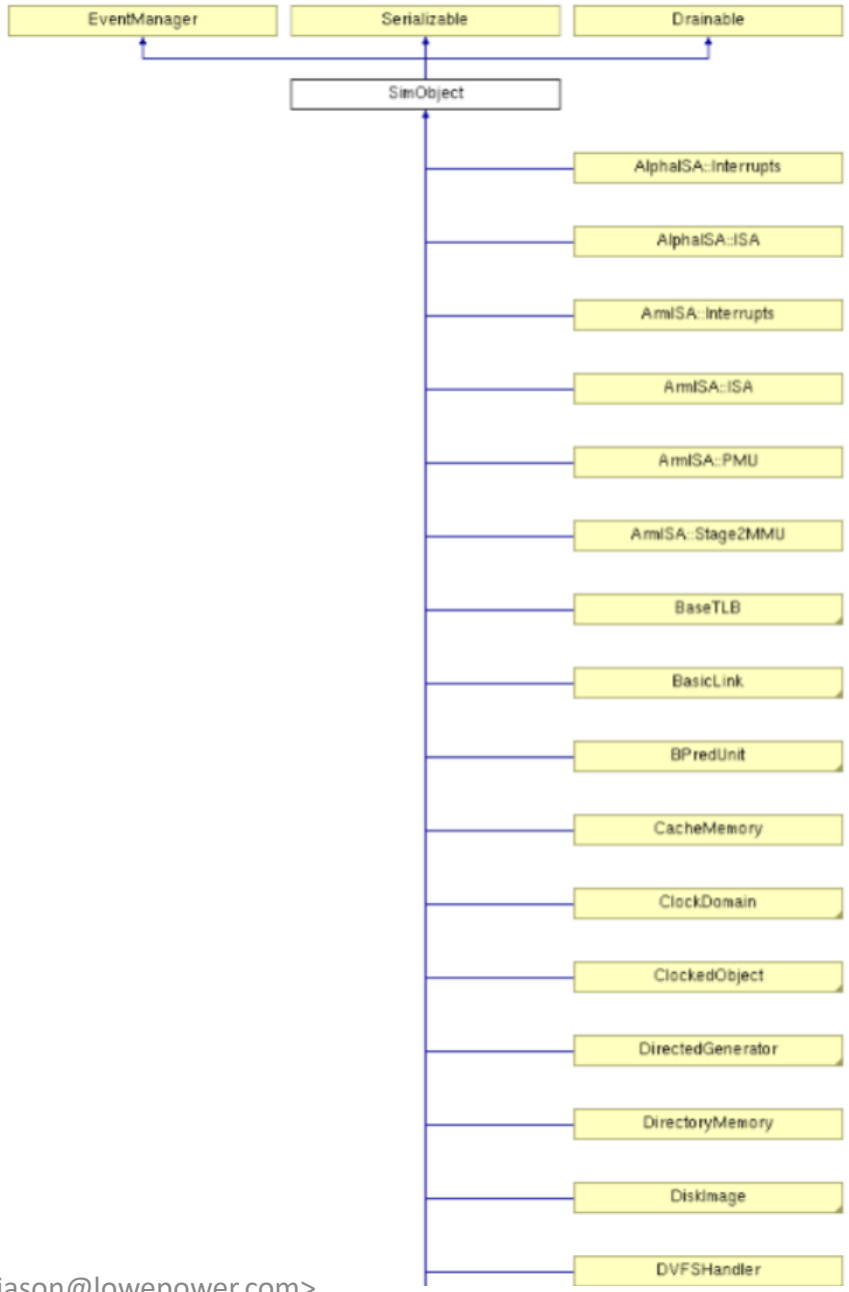
UCDAVIS

gem5 architecture

gem5 consists of "**SimObjects**"

Most C++ objects in gem5 inherit from **`class SimObject`**

Represent physical system components



gem5 architecture

gem5 is a **discrete event simulator**

Event Queue

1) Event at head dequeued
2) Event executed
3) More events queued

We'll cover more after the break

All SimObjects can enqueue events to the event queue

gem5 configuration scripts

http://learning.gem5.org/book/part1/simple_config.html

http://learning.gem5.org/book/part1/cache_config.html

gem5 user interface

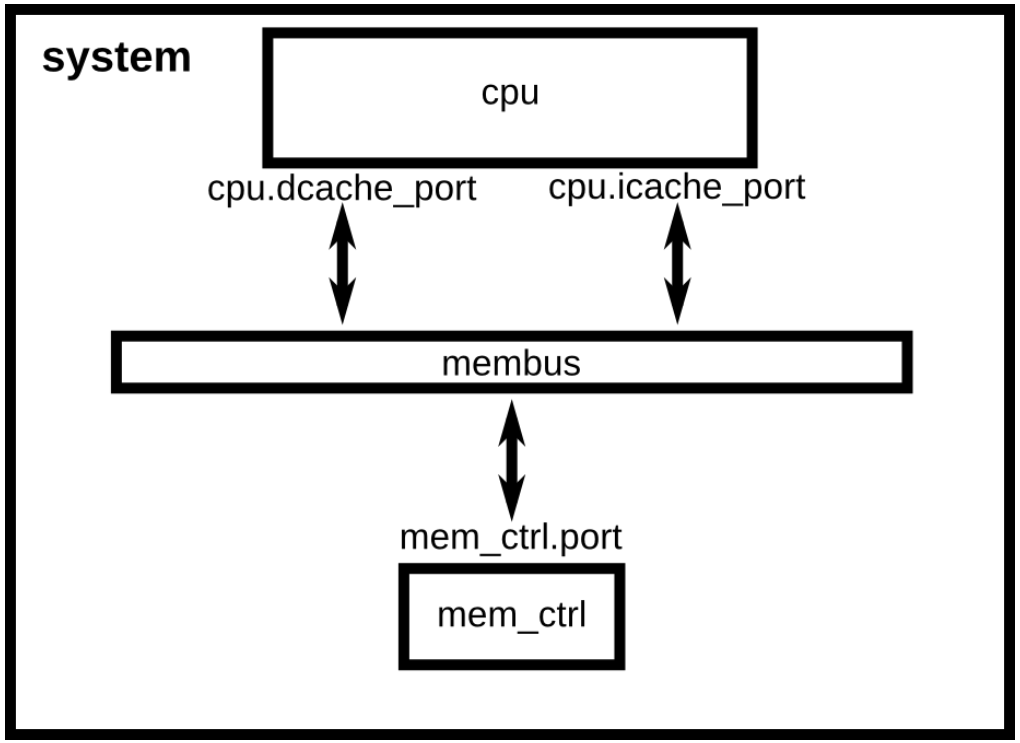
gem5 completely controlled by **Python scripts**

Scripts define system to model

All (C++) SimObjects exposed to Python

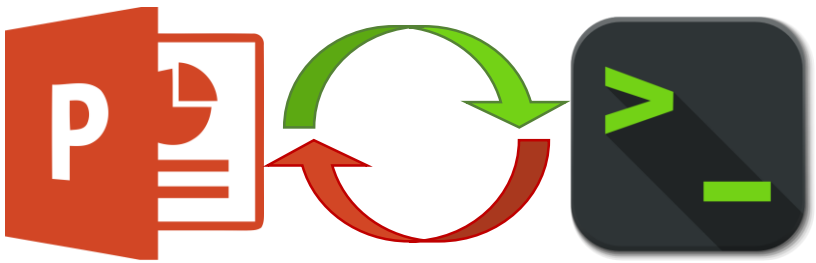
So… let's make one!

Simple config script



Single CPU connected to a memory bus

Switch!



Simple config script

configs/learning_gem5/part1/simple.py

Running gem5

```
> build/X86/gem5.opt
        configs/tutorial/simple.py
```

build/X86/gem5.opt: the gem5 binary to run

configs/…/simple.py: the configuration script (config script)

Port interface

```
system.cpu.icache_port = system.membus.slave
system.cpu.dcache_port = system.membus.slave
...
system.mem_ctrl.port = system.membus.master
```

Ports connect **MemObjects**

To register a connection between master and slave, use '=' in Python

Syscall Emulation (SE) mode

```
process = Process()
process.cmd = ['tests/.../hello']
system.cpu.workload = process
...
root = Root(full_system = False)
```

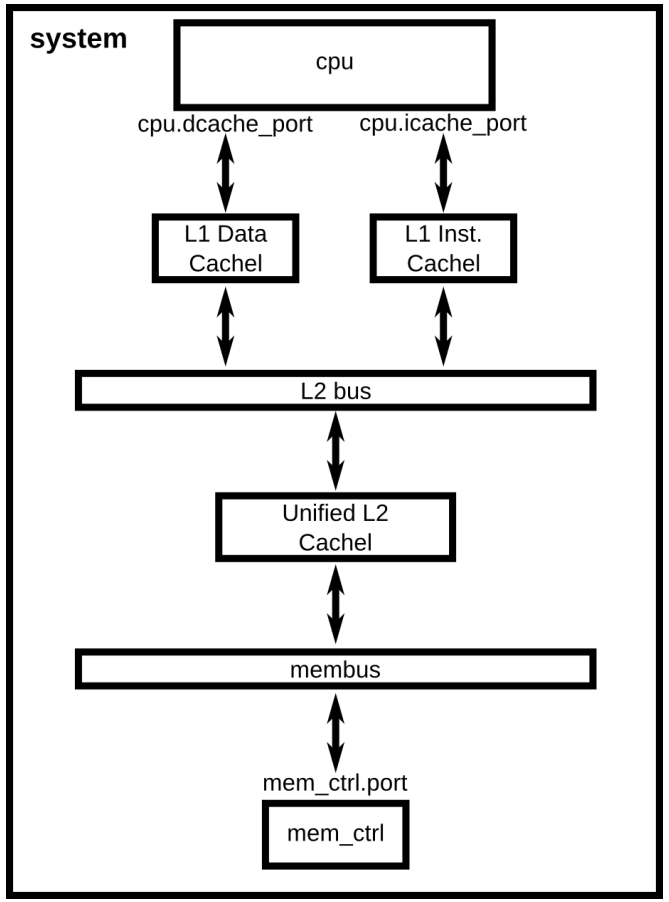
SE mode *emulates* the operating system (Linux) syscalls. No OS runs.

process: an *emulated* process with *emulated* page tables, file descriptors, etc.

Full system mode runs a full OS as if gem5 is a "bare metal" system. Like full virtualization.

Going further: **Adding caches**

http://learning.gem5.org/book/part1/cache_config.html

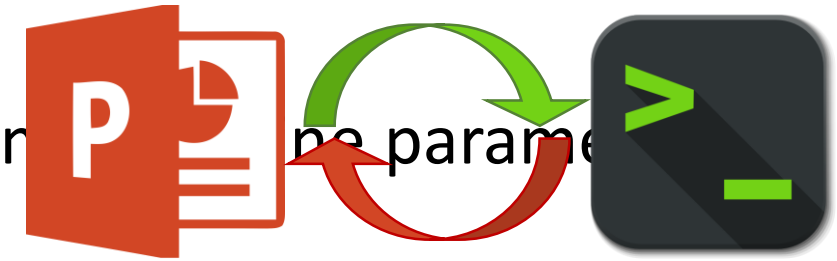


Extending SimObjects in Python config

Object-oriented configuration

Adding configuration parameters

Switch!



It's just Python!

```
class L1Cache(Cache):
    ...

class L1ICache(L1Cache):
    def connectCPU(self, cpu):
        self.cpu_side = cpu.icache_port
...
```

Use good object-oriented design!

See text for details

Debugging config files is easy. Just add some print statements!

Use Python builtins to provide support for command line parameters.

Understanding gem5 output

http://learning.gem5.org/book/part1/gem5_stats.html

UCDAVIS

Understanding gem5 output

```
> ls m5out

config.ini   config.json   stats.txt
```

config.ini: Dumps all of the parameters of all SimObjects. This shows exactly what you simulated.

config.json: Same as config.ini, but in json format.

stats.txt: Detailed statistic output. Each SimObject defines and updates statistics. They are printed here at the end of simulation.

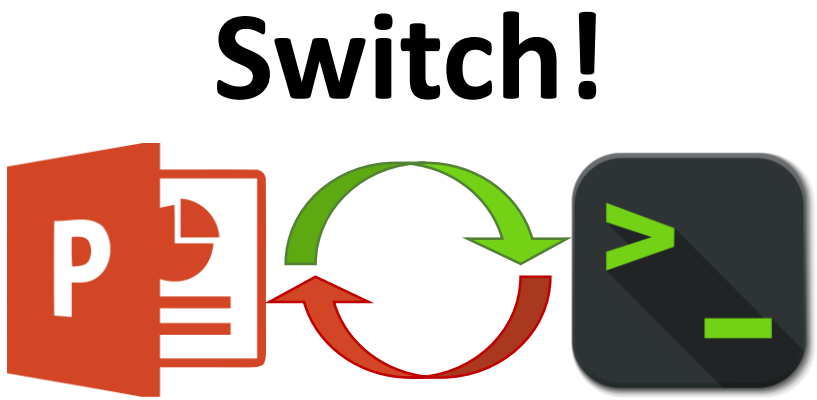
stats.txt

```
---------- Begin Simulation Statistics ----------
sim_seconds     0.000346      # Number of seconds simulated
sim_ticks       345518000     # Number of ticks simulated
final_tick      345518000     # Number of ticks from
sim_freq        1000000000000 # Frequency of simulate
...
sim_insts       5712          # Number of instructions simulated
sim_ops         10314         # Number of ops (including micro ...
...
system.mem_ctrl.bytes_read::cpu.inst  58264  # Nu
system.mem_ctrl.bytes_read::cpu.data  7167   # Nu
...
system.cpu.committedOps          10314  # Number
system.cpu.num_int_alu_accesses 10205  # Number of integer ...
```

sim_seconds: name of stat. This shows *simulated guest* time

Every SimObject can have its own stats. Names are what you used in the Python config file

Example scripts



Questions?

We covered

- gem5 history
- Downloading and building gem5
- gem5's user interface: python
- How to write a configuration script
- gem5's output
- Using the example scripts