Full-System Workloads and Asymmetric Multi-Core Simulation

Anthony Gutierrez

atgutier@umich.edu

Advanced Computer Architecture Laboratory

University of Michigan, Ann Arbor, MI

Outline

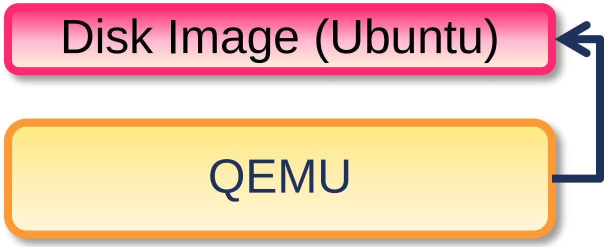
- Part I: Using Full-System Workloads
  - Available Full-System Workloads
  - Beyond SPEC CPU: Java Workloads
  - BBench: Example Interactive Workload
  - Interactive Workload Challenges and What We Need
- Part II: Asymmetric Multi-Core Simulation
  - Modeling an Asymmetric Multi-Core Simulation
  - Thread Migration in gem5
  - What is Still Missing

Full-System Workloads, What's out There? (ARM)

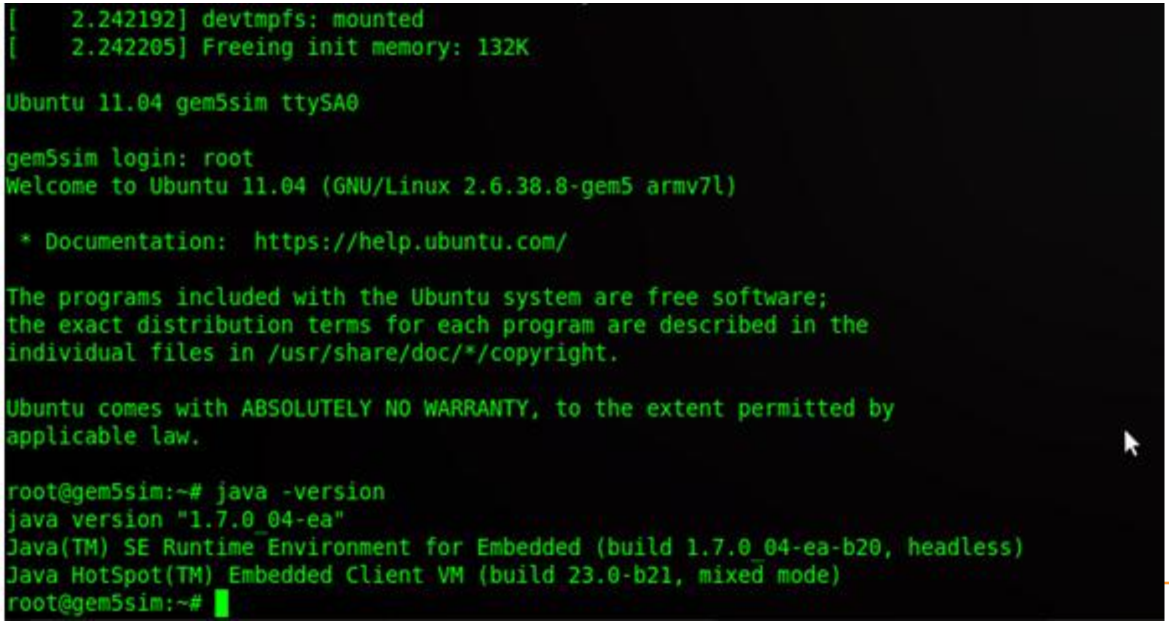
- gem5 can support workloads for Android/Linux out of the box
  - Models RealView/Versatile Express development boards
- Have successfully run Android and Ubuntu
  - With gui support over VNC
- Need to compile OS, kernel, and workloads for proper target
  - May also need to modify startup scripts and other file on image
- Pre-compiled disk images and kernels exist as well
  - Linaro (Ubuntu) and BBench (Android) images

Beyond SPEC CPU: Java Workloads

- DaCapo Benchmarks
  - Real-world, open-source Java benchmarks
- Need full-system simulation
  - Can’t really compile statically
  - Need Java VM and associated libraries
  - Appropriate OS: Ubuntu
- Can utilize QEMU to install required packages quickly



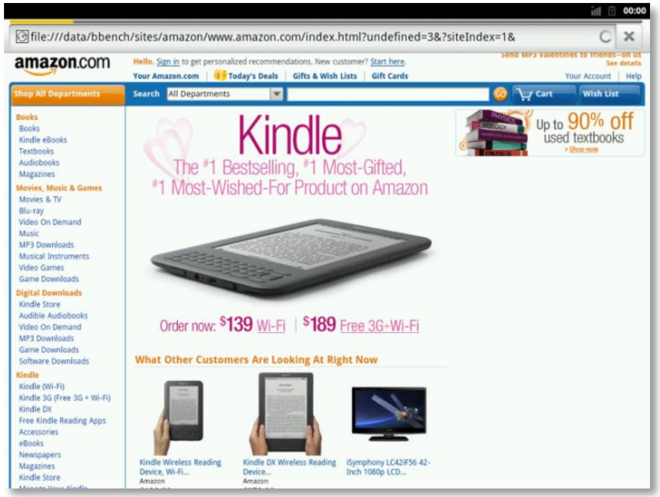
Mount image
Chroot image’s root dir
apt-get install libs
apt-get install JVM, etc.

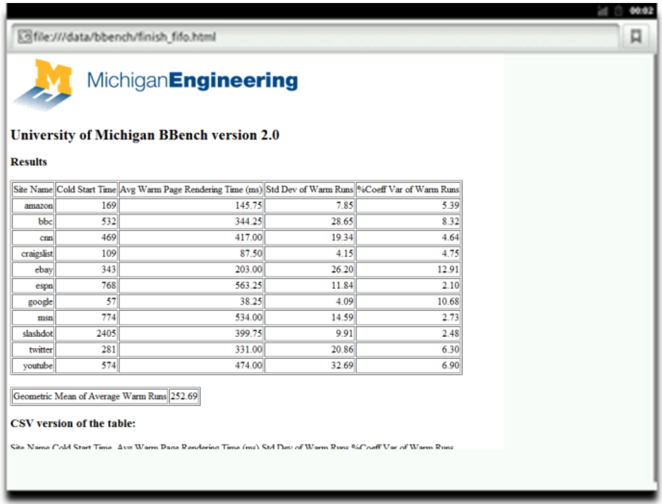


```
[    2.242192] devtmpfs: mounted
[    2.242205] Freeing init memory: 132K

Ubuntu 11.04 gem5sim ttySA0

gem5sim login: root
Welcome to Ubuntu 11.04 (GNU/Linux 2.6.38.8-gem5 armv7l)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@gem5sim:~# java -version
java version "1.7.0_04-ea"
Java(TM) SE Runtime Environment for Embedded (build 1.7.0_04-ea-b20, headless)
Java HotSpot(TM) Embedded Client VM (build 23.0-b21, mixed mode)
root@gem5sim:~# 
```

BBench: Web-Browser Benchmarking

- Web-browser benchmark
  - Collection of several relevant pages scraped from the web in 2011
  - JavaScript automates the rendering of each page
  - Ported to gem5 on both Gingerbread and ICS





University of Michigan BBench version 2.0

Results

Site Name	Cold Start Time	Avg Warm Page Rendering Time (ms)	Std Dev of Warm Runs	%Coeff Var of Warm Runs
amazon	169	145.75	7.85	5.39
bbc	532	344.25	28.65	8.32
cnn	469	417.00	19.34	4.64
craigslist	109	87.50	4.15	4.75
ebay	343	203.00	26.20	12.91
espn	768	563.25	11.84	2.10
google	57	38.25	4.09	10.68
msn	774	534.00	14.59	2.73
slashdot	2405	399.75	9.91	2.48
twitter	281	331.00	20.86	6.30
youtube	574	474.00	32.69	6.90

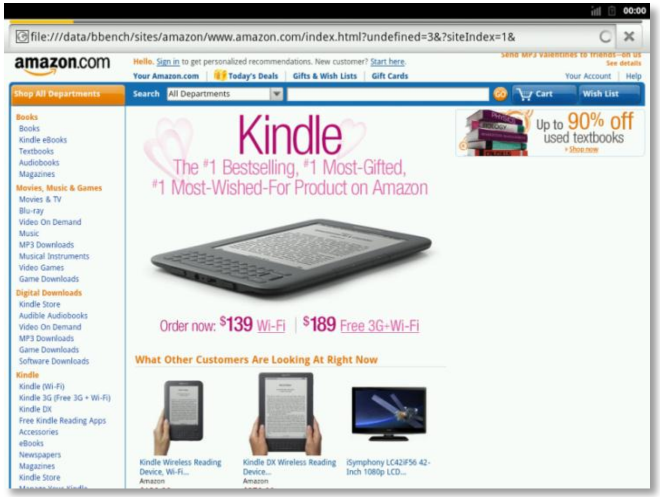
Geometric Mean of Average Warm Runs	252.69

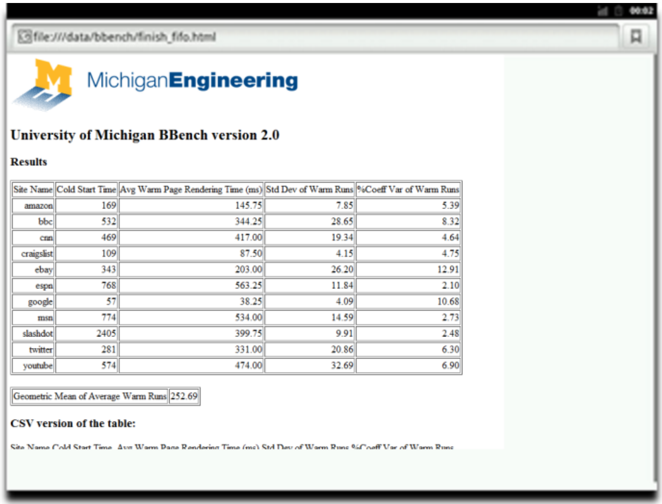
CSV version of the table:

Site Name,Cold Start Time, Avg Warm Page Rendering Time (ms),Std Dev of Warm Runs,%Coeff Var of Warm Runs

BBench: Web-Browser Benchmarking

- Web-browser benchmark
  - Collection of several relevant pages scraped from the web in 2011
  - JavaScript automates the rendering of each page
  - Ported to gem5 on both Gingerbread and ICS
- Challenges
  - Can’t interact very easily – Ensure BBench is fully automated





University of Michigan BBench version 2.0

Results

Site Name	Cold Start Time	Avg Warm Page Rendering Time (ms)	Std Dev of Warm Runs	%Coeff Var of Warm Runs
amazon	169	145.75	7.85	5.39
bbc	532	344.25	28.65	8.32
cnn	469	417.00	19.34	4.64
craigslist	109	87.50	4.15	4.75
ebay	343	203.00	26.20	12.91
espn	768	563.25	11.84	2.10
google	57	38.25	4.09	10.68
msn	774	534.00	14.59	2.73
slashdot	2405	399.75	9.91	2.48
twitter	281	331.00	20.86	6.30
youtube	574	474.00	32.69	6.90

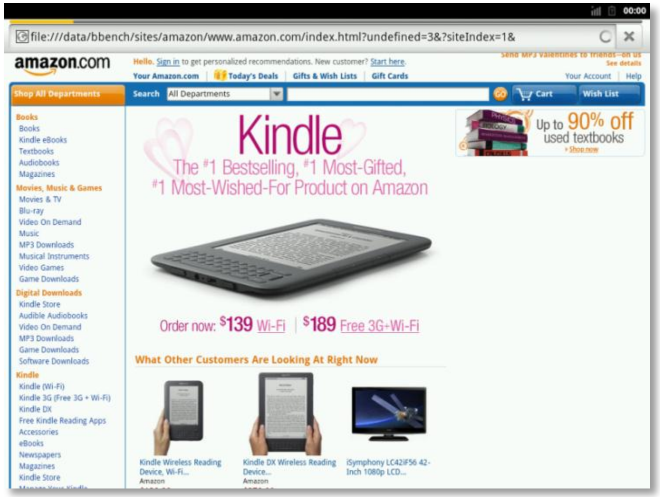
Geometric Mean of Average Warm Runs	252.69

CSV version of the table:

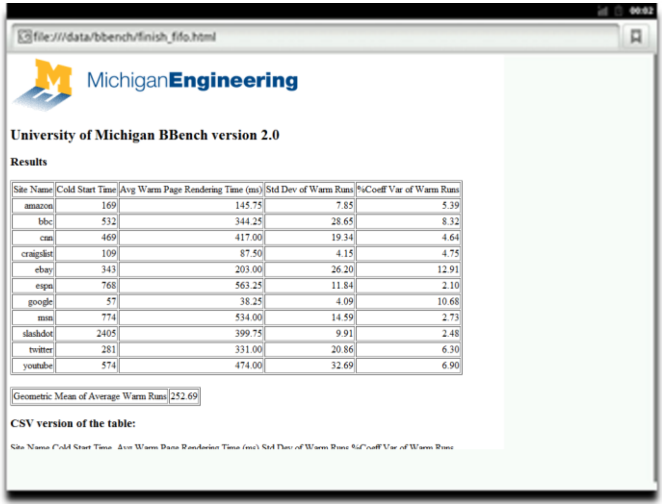
Site Name Cold Start Time Avg Warm Page Rendering Time (ms) Std Dev of Warm Runs %Coeff Var of Warm Runs

BBench: Web-Browser Benchmarking

- Web-browser benchmark
  - Collection of several relevant pages scraped from the web in 2011
  - JavaScript automates the rendering of each page
  - Ported to gem5 on both Gingerbread and ICS
- Challenges
  - Can’t interact very easily – Ensure BBench is fully automated
  - Terminate when benchmark finishes – Tricky scripting to terminate run



file:///data/bbench/sites/amazon/www.amazon.com/index.html?undefined=3&?siteIndex=1&



file:///data/bbench/finish_fifo.html

University of Michigan BBench version 2.0

Results

Site Name	Cold Start Time	Avg Warm Page Rendering Time (ms)	Std Dev of Warm Runs	%Coeff Var of Warm Runs
amazon	169	145.75	7.85	5.39
bbc	532	344.25	28.65	8.32
cnn	469	417.00	19.34	4.64
craigslist	109	87.50	4.15	4.75
ebay	343	203.00	26.20	12.91
espn	768	563.25	11.84	2.10
google	57	38.25	4.09	10.68
msn	774	534.00	14.59	2.73
slashdot	2405	399.75	9.91	2.48
twitter	281	331.00	20.86	6.30
youtube	574	474.00	32.69	6.90

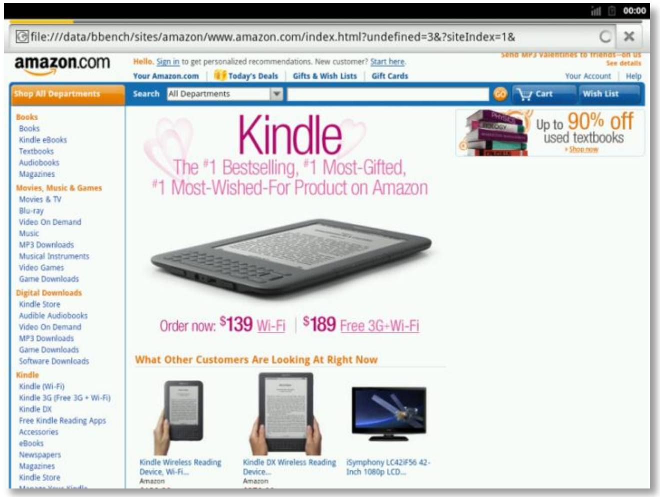
Geometric Mean of Average Warm Runs	252.69

CSV version of the table:

Site Name Cold Start Time Avg Warm Page Rendering Time (ms) Std Dev of Warm Runs %Coeff Var of Warm Runs

BBench: Web-Browser Benchmarking

- Web-browser benchmark
  - Collection of several relevant pages scraped from the web in 2011
  - JavaScript automates the rendering of each page
  - Ported to gem5 on both Gingerbread and ICS
- Challenges
  - Can’t interact very easily – Ensure BBench is fully automated
  - Terminate when benchmark finishes – Tricky scripting to terminate run
  - Prevent screen from locking – Modify Android FS source to prevent lock



file:///data/bbench/sites/amazon/www.amazon.com/index.html?undefined=3&?siteIndex=1&

Hello. Sign in to get personalized recommendations. New customer? Start here.

Send MP3 Valentines to friends—on us See details

Kindle

The #1 Bestselling, #1 Most-Gifted, #1 Most-Wished-For Product on Amazon

Order now: $139 Wi-Fi | $189 Free 3G+Wi-Fi

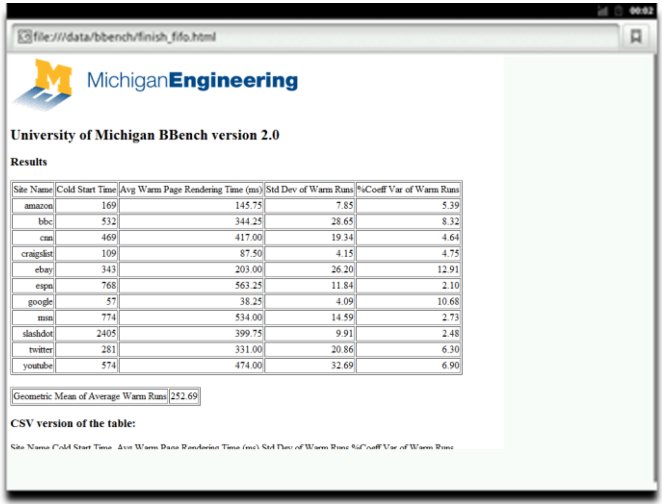
Up to 90% off used textbooks › Shop now

What Other Customers Are Looking At Right Now

Kindle Wireless Reading Device, Wi-Fi... Amazon

Kindle DX Wireless Reading Device... Amazon

iSymphony LC42iF56 42-Inch 1080p LCD...



file:///data/bbench/finish_fifo.html

MichiganEngineering

University of Michigan BBench version 2.0

Results

Site Name	Cold Start Time	Avg Warm Page Rendering Time (ms)	Std Dev of Warm Runs	%Coeff Var of Warm Runs
amazon	169	145.75	7.85	5.39
bbc	532	344.25	28.65	8.32
cnn	469	417.00	19.34	4.64
craigslist	109	87.50	4.15	4.75
ebay	343	203.00	26.20	12.91
espn	768	563.25	11.84	2.10
google	57	38.25	4.09	10.68
msn	774	534.00	14.59	2.73
slashdot	2405	399.75	9.91	2.48
twitter	281	331.00	20.86	6.30
youtube	574	474.00	32.69	6.90

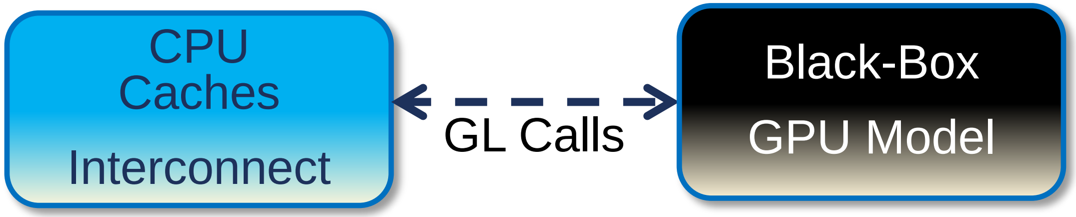
Geometric Mean of Average Warm Runs	252.69

CSV version of the table:

Site Name Cold Start Time Avg Warm Page Rendering Time (ms) Std Dev of Warm Runs %Coeff Var of Warm Runs

Challenges with Interactive Applications

- Running interactive applications
  - How do we automate these apps?
  - How do me model interactivity?
- What if the application relies on devices?
  - GPS, GPU, radio, etc.
  - E.g., BBench on gem5 spends majority of time in SW rendering – no GPU
- Things I'd like to seem in gem5:
  - Support for more realistic devices
    - Care about interaction with devices, so functional modeling could be enough
  - A centralized location for available workloads



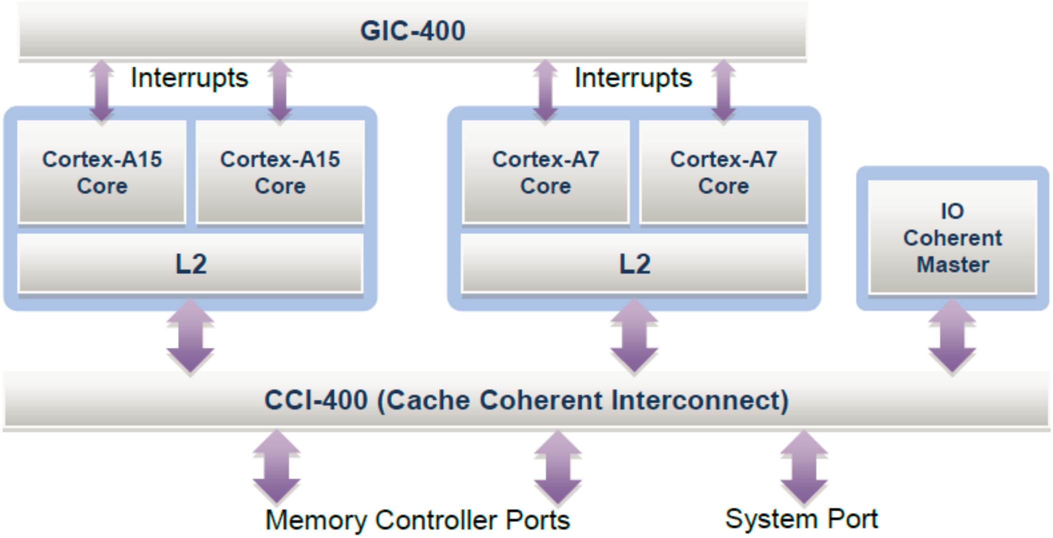
Outline

- Part I: Using Full-System Workloads
  - Available Full-System Workloads
  - Beyond SPEC CPU: Java Workloads
  - BBench: Example Interactive Workload
  - Interactive Workload Challenges and What We Need
- Part II: Asymmetric Multi-Core Simulation
  - Modeling an Asymmetric Multi-Core Simulation
  - Thread Migration in gem5
  - What is Still Missing

Modeling an Asymmetric Multi-Core System

- gem5 supports several CPU models
  - Out-of-order, in-order, single-cycle timing, atomic
- Generic interface between allows for multiple types at once
  - Out-of-order <-> in-order
  - Out-of-order <-> timing
  - In-order <-> timing
  - Atomic and timing models don’t mix well
- Setup everything in Python config scripts

ARM big.LITTLE Processing



Source: Greenhalgh, 2011. ARM white paper.

Two Ways to Model Asymmetric Cores

- 1) All cores are always active
  - Inside your config scripts define CPUs of multiple types:

    ```
    test_sys.cpus = [DerivO3CPU(cpu_id=0), InOrderCPU(cpu_id=1)]
    ```

  - Then, just run simulation as normal

- 2) Only cores of a certain type are active
  - Define multiple lists of CPUs and switch back-and-forth:

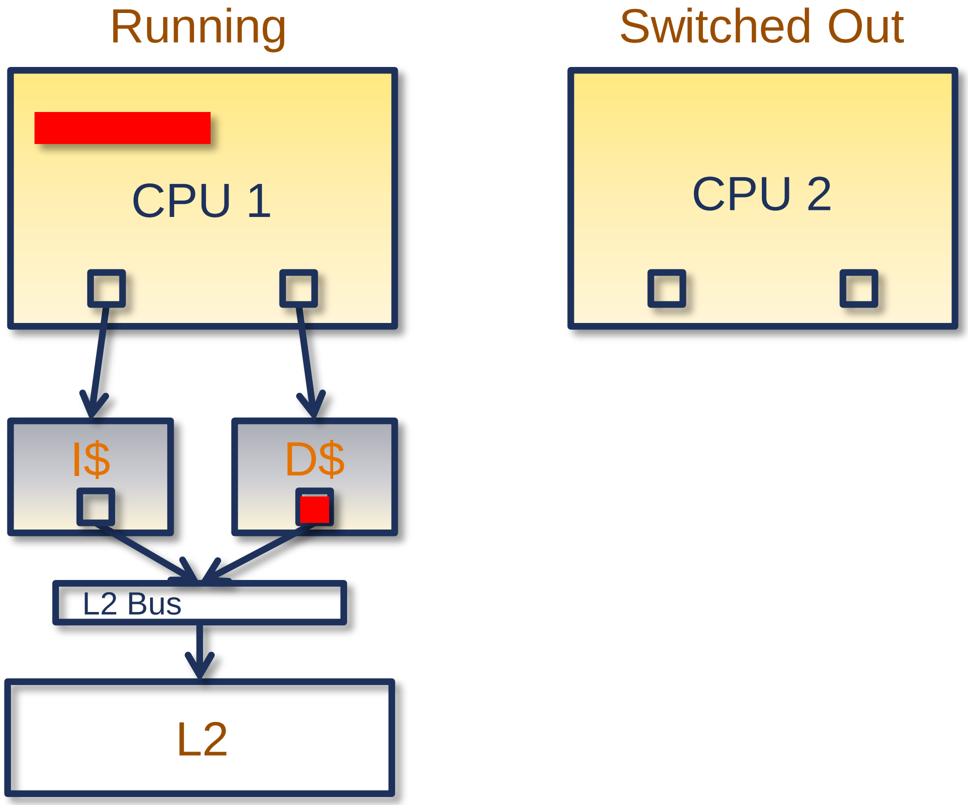
    ```
    test_sys.big_cpus = [DerivO3CPU(cpu_id=0), DerivO3CPU(cpu_id=1)]
    test_sys.big_cpus = [InOrderCPU(cpu_id=0), InOrderCPU(cpu_id=1)]
    switch_cpu_list = [(test_sys.big_cpu[i], test_sys.little_cpu[i]) for i in xrange(np)]
    ```

  - Then, on a switch event, use switching infrastructure:

    ```
    m5.drain(test_sys) # drains all objects
    m5.switchCpus(switch_cpu_list) # switches the CPUs & transfers state
    m5.resume(test_sys) # tell all objects to resume
    ```

Modeling Thread Migration

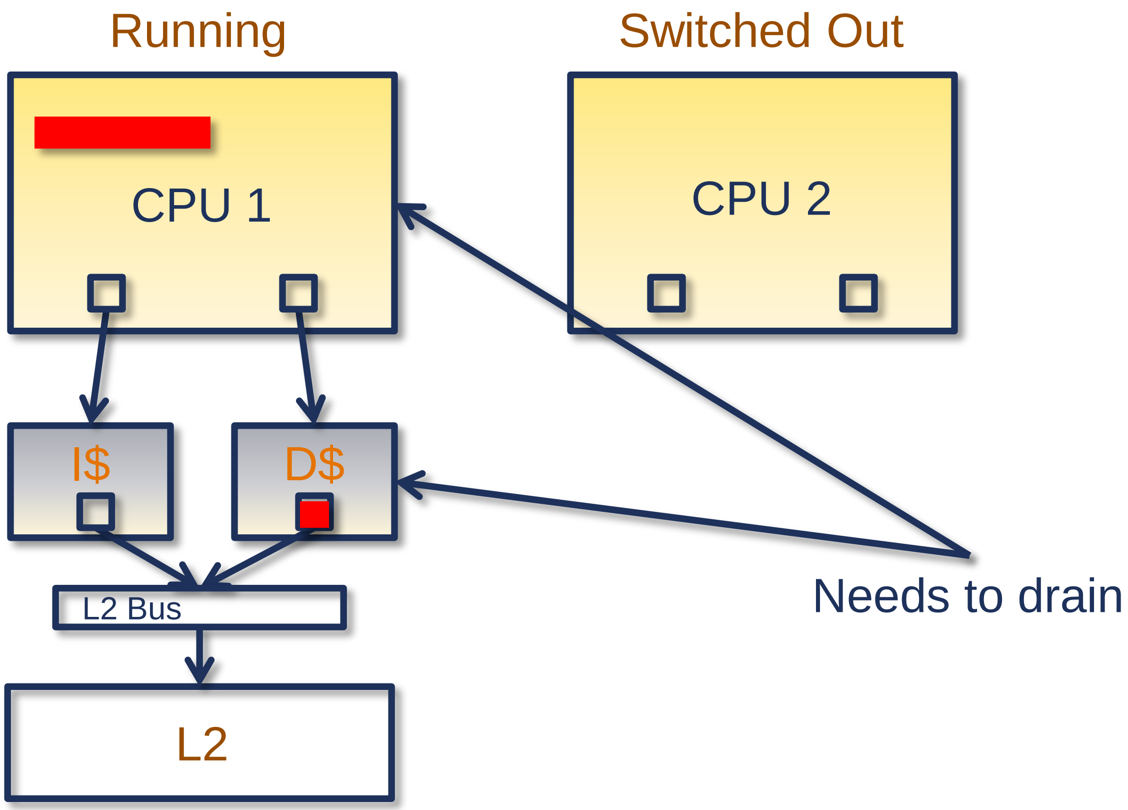
- gem5's built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()

Modeling Thread Migration

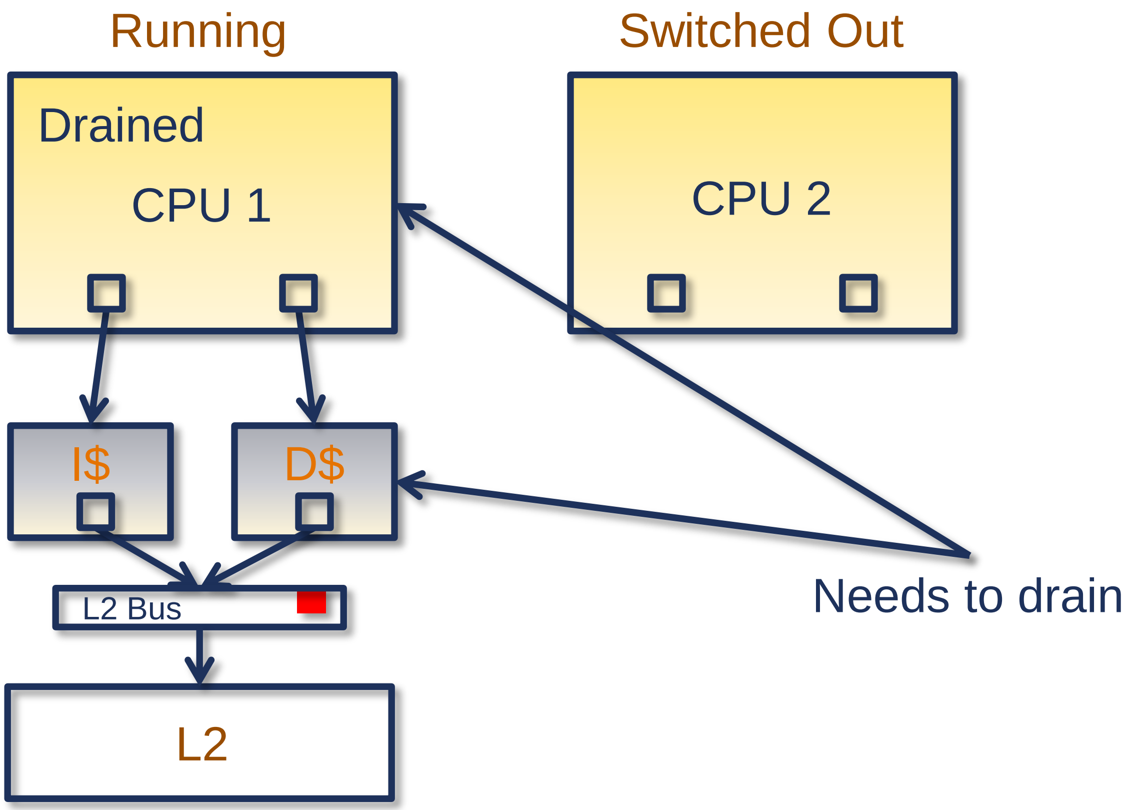
- gem5's built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()

Modeling Thread Migration

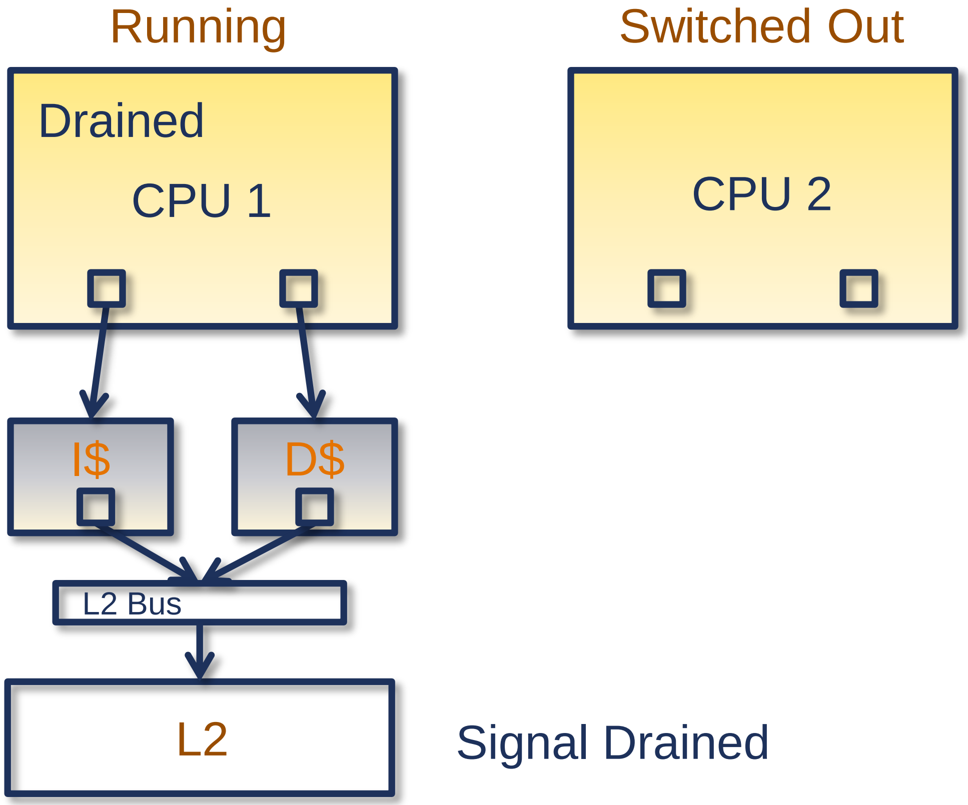
- gem5's built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()

Modeling Thread Migration

- gem5’s built-in drain()/takeOverFrom()/switchOut()/resume() functionality

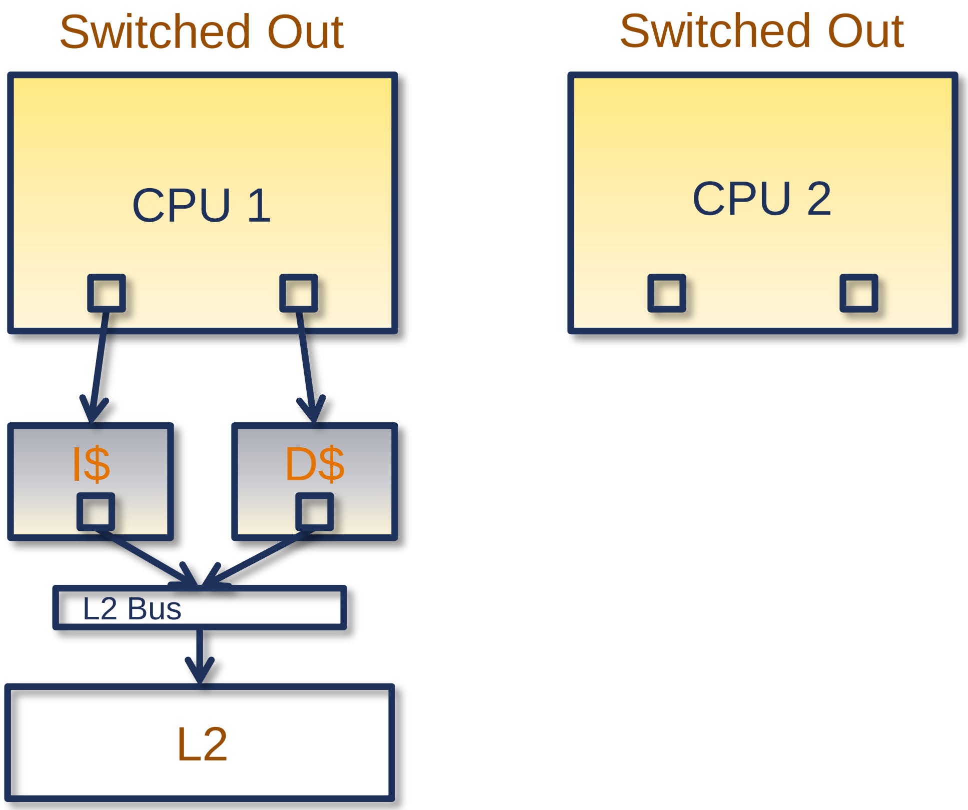


1. drain()

Signal Drained

Modeling Thread Migration

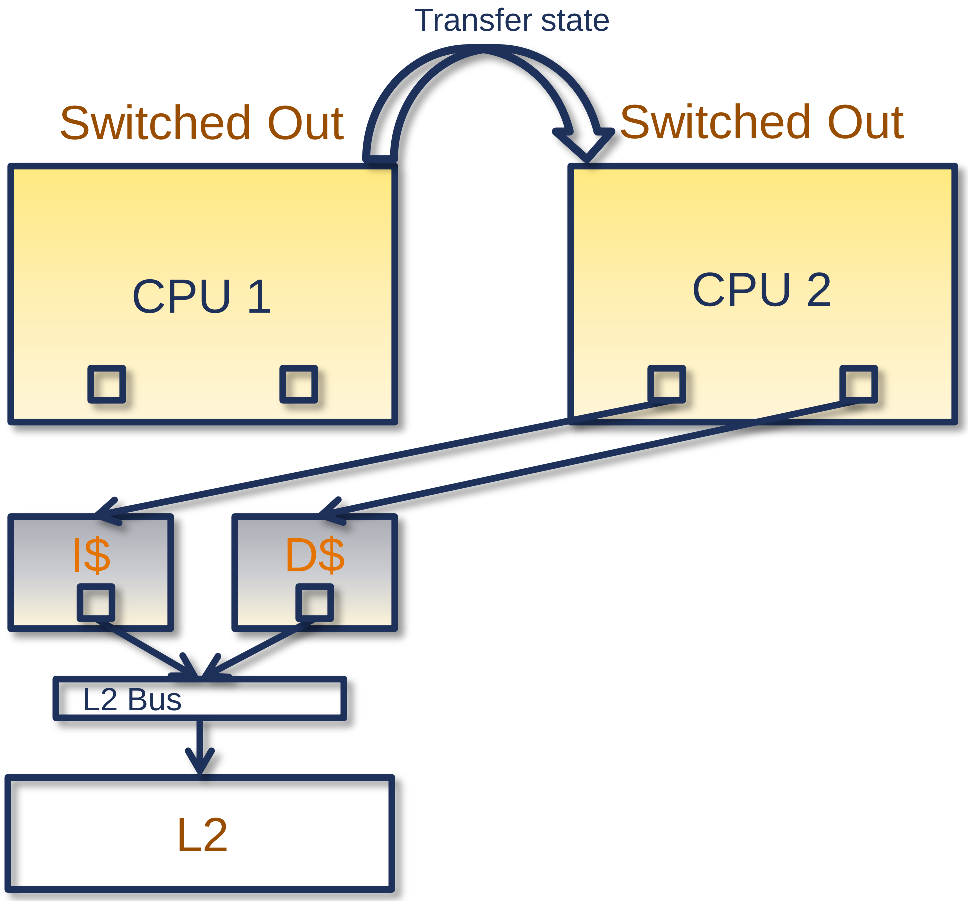
- gem5's built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()
2. switchOut()

Modeling Thread Migration

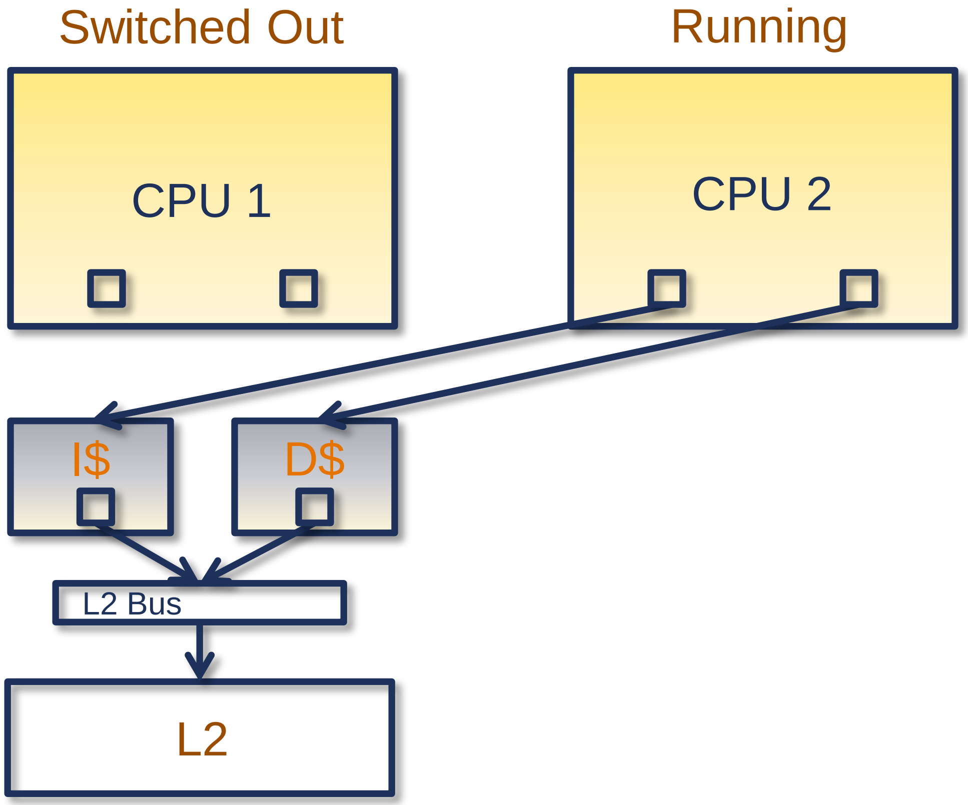
- gem5's built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()
2. switchOut()
3. takeOverFrom()

Modeling Thread Migration

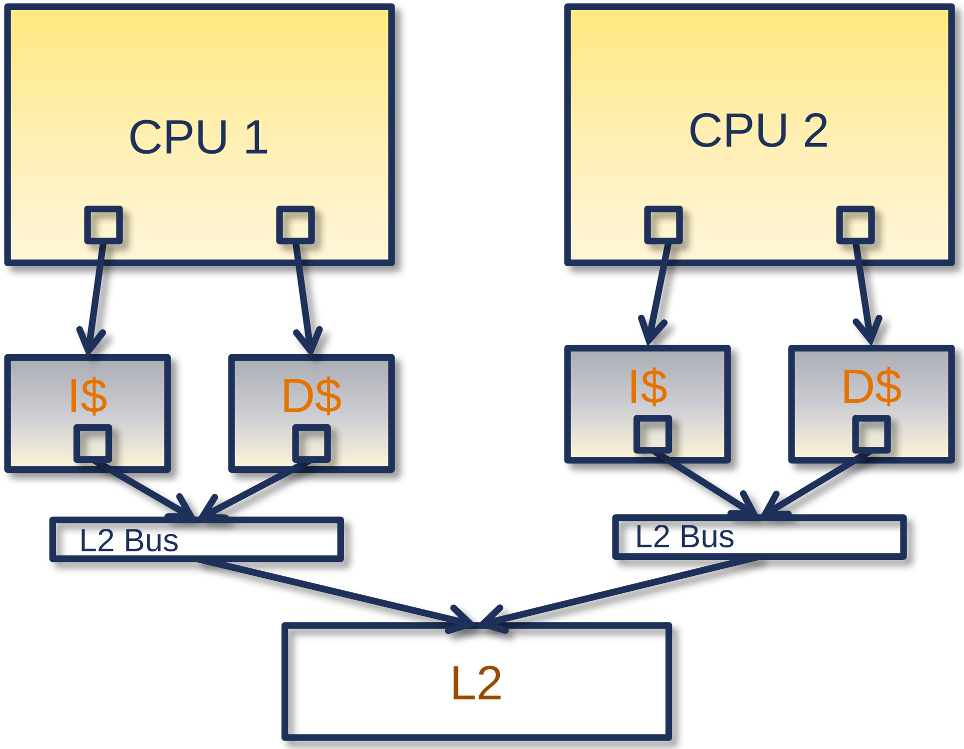
- gem5’s built-in drain()/takeOverFrom()/switchOut()/resume() functionality



1. drain()
2. switchOut()
3. takeOverFrom()
4. Resume()

No Cache Swapping

- More realistic migration modeling
  - Give each core their own L1 caches
  - In takeOverFrom(), don’t swap caches



Clean/Invalidate Cache during drain()

```
dCache.memWriteback();
iCache.memWriteback();
dCache.memInvalidate();
iCache.memInvalidate();
```

What’s Missing?

- Realistic timing of thread migration
  - Registers, caches, and all other thread context transferred (atomically)
- InOrderCPU support for ARM, x86
  - Currently using scaled-down O3, or TimingSimpleCPU to model InOrder
- Account for cache state transfer/cleaning overhead
  - Currently, caches are swapped between cores, or cleaned atomically

Questions?