gem5, Interoperability, and Improving Simulator Methodology

Jason Lowe-Power

jlowepower@ucdavis.edu

@jlowepower

Outline

What is gem5?

Recent gem5 features

gem5's future

My vision

Big news

Features coming soon

Created at Michigan by students of Steve Reinhardt, principally Nate Binkert.

"A tool for simulating systems"

UCDAVIS

Two Views of M5

1. A framework for event-driven simulation
   - Events, objects, statistics, configuration
2. A collection of predefined object models
   - CPUs, caches, busses, devices, etc.

- This tutorial focuses on #2
- You may find #1 useful even if #2 is not

UCDAVIS

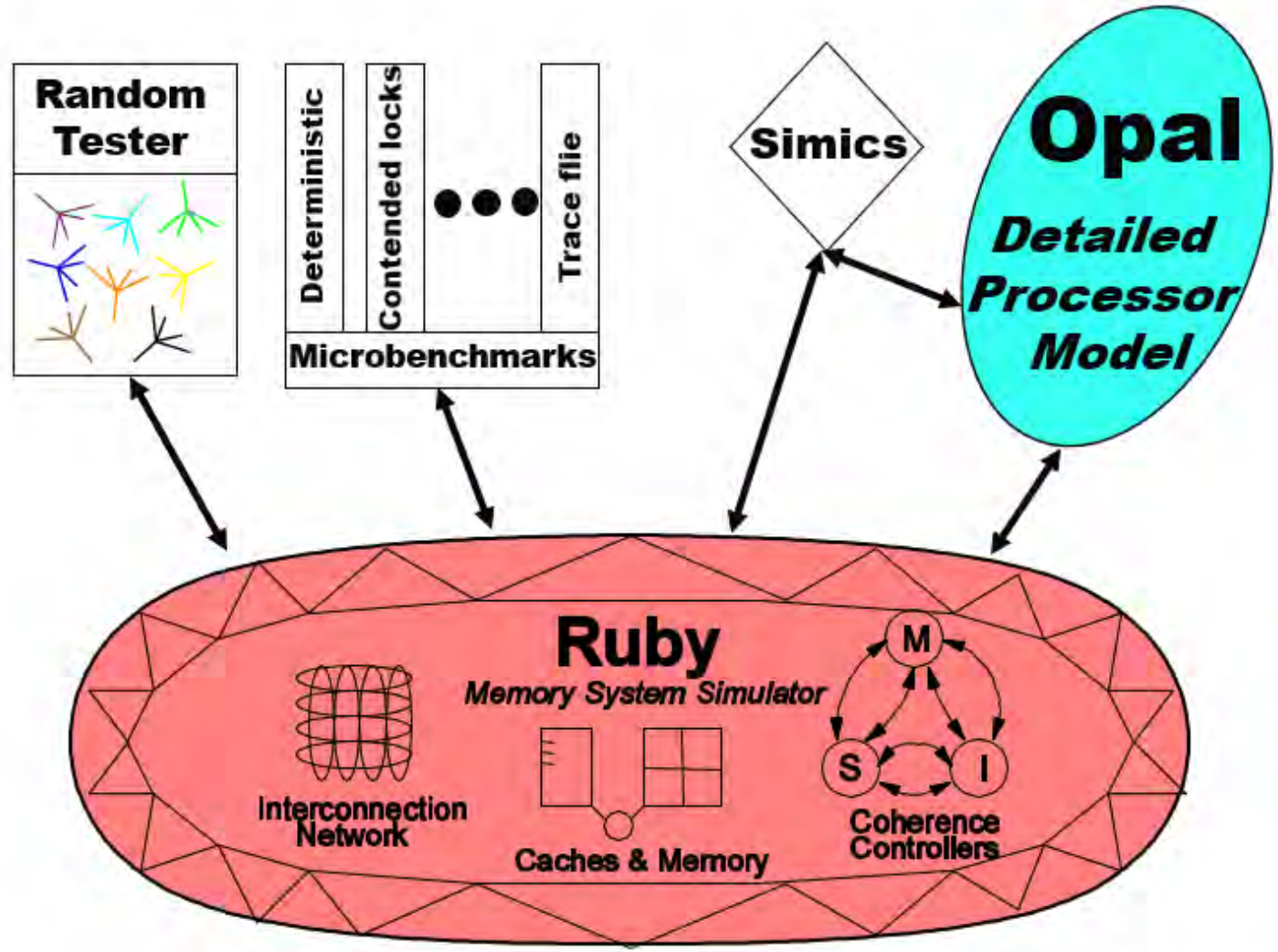
Created at Michigan by students of Steve Reinhardt, principally Nate Binkert.

"A tool for simulating systems"

Created at Wisconsin by students of Mark Hill and David Wood.

Detailed memory system

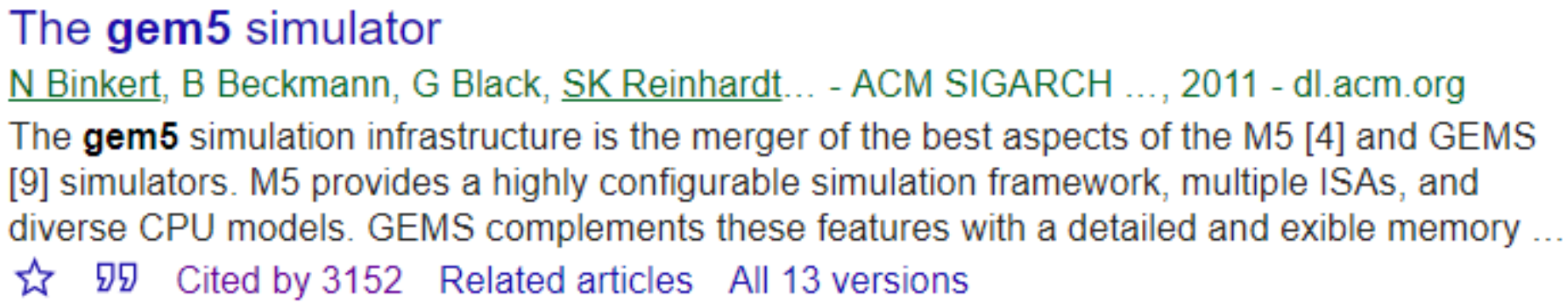
GEMS From 50,000 Feet



gem5 today

gem5 2011 paper cited over 3000 times

15-2



Thri

Average of 70-ish commits per month

About 100 unique contributors over last 2 years

today

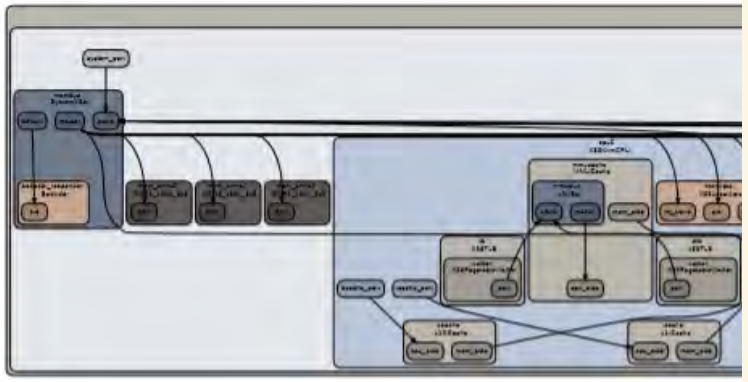
Over 400 “models”

- Over 4000 parameters!
- 3 timing-based CPU models (simple, in order, out of order)
- 8 ISAs (ARM, RISC-V, x86, Alpha, Power, SPARC, MIPS, GCN3)
- 12 memory models (DDR3, DDR4, HBM, HMC, etc.)
- 42 devices (PCI, Arm platform, x86 platform, storage)
- AMD GPGPU
- 12 cache coherence protocols
- Network on chip (Garnet)

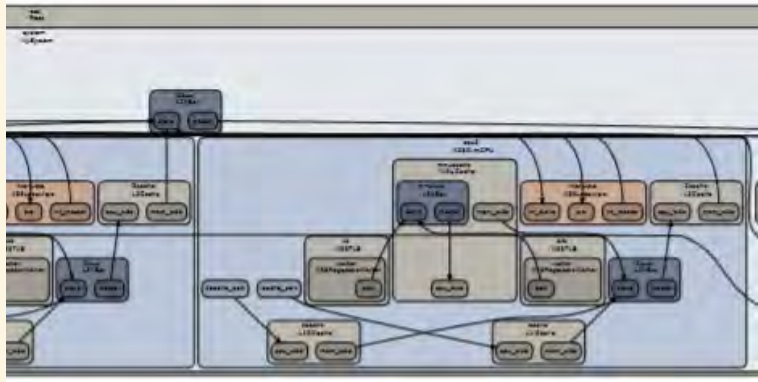
Programmatic configuration

```python
import m5
from m5.objects import *

system = System()

system.clk_domain = SrcClockDomain()
system.clk_domain.clock = '1GHz'
system.clk_domain.voltage_domain = VoltageDomain()

system.mem_mode = 'timing'
system.mem_ranges = [AddrRange('512MB')]

system.cpu = TimingSimpleCPU()

system.membus = SystemXBar()

system.cpu.icache_port = system.membus.slave
system.cpu.dcache_port = system.membus.slave

system.cpu.createInterruptController()
system.cpu.interrupts[0].pio = system.membus.master
system.cpu.interrupts[0].int_master = system.membus.slave
system.cpu.interrupts[0].int_slave = system.membus.master

system.system_port = system.membus.slave

system.mem_ctrl = DDR3_1600_8x8()
system.mem_ctrl.range = system.mem_ranges[0]
system.mem_ctrl.port = system.membus.master

process = Process()
process.cmd = ['tests/test-progs/hello/bin/x86/linux/hello']
system.cpu.workload = process
system.cpu.createThreads()

root = Root(full_system = False, system = system)
m5.instantiate()

print("Beginning simulation!")
exit_event = m5.simulate()
print('Exiting @ tick {} because {}'
      .format(m5.curTick(), exit_event.getCause()))
```

Programmatic configuration



```
# create the system we are going to simulate
system = MySystem(opts, no_kvm=False)

# set up the root SimObject and start the simulation
root = Root(full_system = True, system = system)

if system.getHostParallel():
    # Required for running kvm on multiple host cores.
    # Uses gem5's parallel event queue feature
    # Note: The simulator is quite picky about this number!
    root.sim_quantum = int(1e9) # 1 ms

# instantiate all of the objects we've created above
m5.instantiate()

print("Running the simulation")
exit_event = m5.simulate()

if exit_event.getCause() == "m5_exit instruction encountered":
    # switch to timing CPU
    system.switchCpus(system.cpu, system.timingCpu)

# Continue simulating
m5.simulate()
```



tmux

[0] 0:..ode/gem5/gem5*

CPU 07/08 14:35:27

jlp@amarillo ~/Code/gem5/gem5 ‹feature/simplefs*›

$ build/X86/gem5.opt configs/myconfigs/runkvm.py --c

Code/gem5/gem5 ‹feature/simplefs*›

erm 3457

Pause 00:00:00 Select Area Audio Record Pointer

2:35 PM 8/7/2019

gem5 today

Support for complex ML stacks

See https://github.com/KyleRoarty/gem5_docker/

Support for ARM SVE instructions

Integration with other simulators

DRAMSim2	SST
DSENT	SystemC
McPAT	GPGPU-Sim

UCDAVIS

Simulator integration

Choose a ***driver***

Choose an interface

gem5 + SST

Multiple implementations

SST used as driver for gem5

But ***two*** event queues

Interface is memory requests

Separate setup for each simulator

SST + gem5 = A Scalable Simulation Infrastruct
High Performance Computing

Mingyu Hsieh
Sandia National Labs P.O.Box 5800
Albuquerque, NM

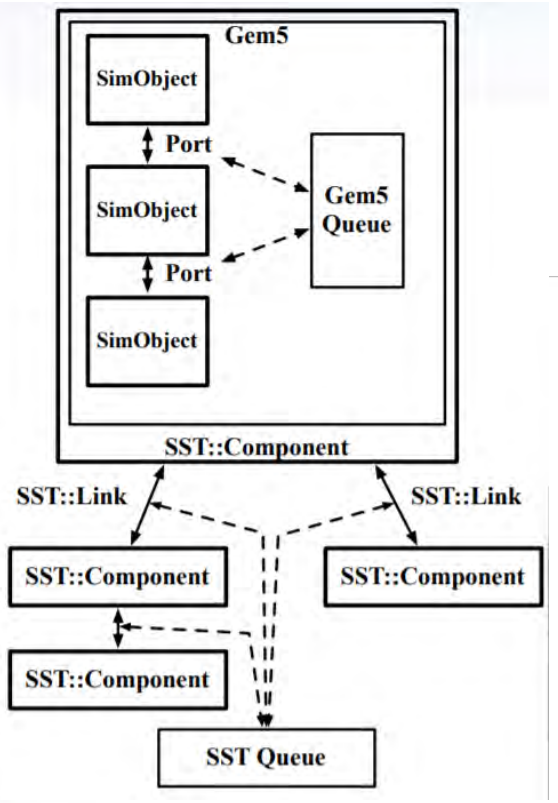
Jie Meng
Boston University
ECE Department
Boston, MA
jiemeng@bu.edu

Michael Levenh
Sandia National
P.O.Box 580
Albuquerque, N
mjleven@sandi

Kevin Pedretti
Sandia National Labs
P.O.Box 5800
Albuquerque, NM
ktpedre@sandia.gov

Ayse Coskun
Boston University
ECE Department
Boston, MA
acoskun@bu.edu

Arun Rodrigu
Sandia National
P.O.Box 580
Albuquerque, N
afrodri@sandia



gem5 + SystemC

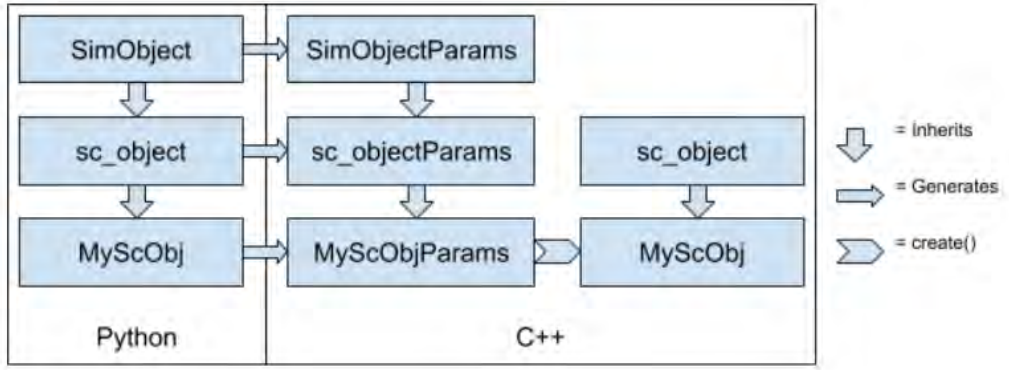
Multiple implementations…

SystemC is an IEEE standard: defines interfaces, etc.

gem5 implements SystemC

Deeply integrated with gem5's Python API

Interface can be at any arbitrary boundary



Simulator integration

Choose a ***driver***

Choose an interface

Requires well-defined API

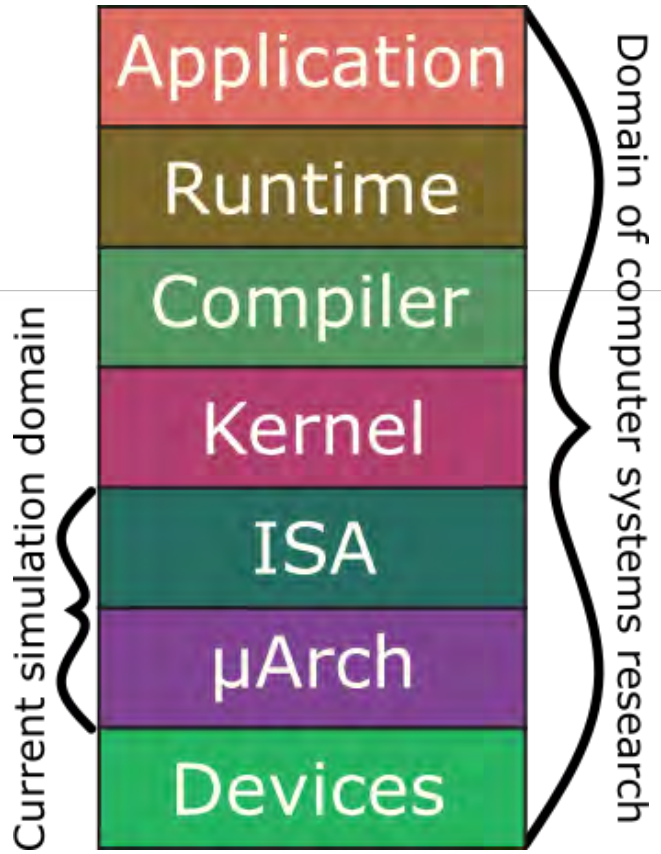
gem5 is currently a moving target

Upstreaming bridge with tests required

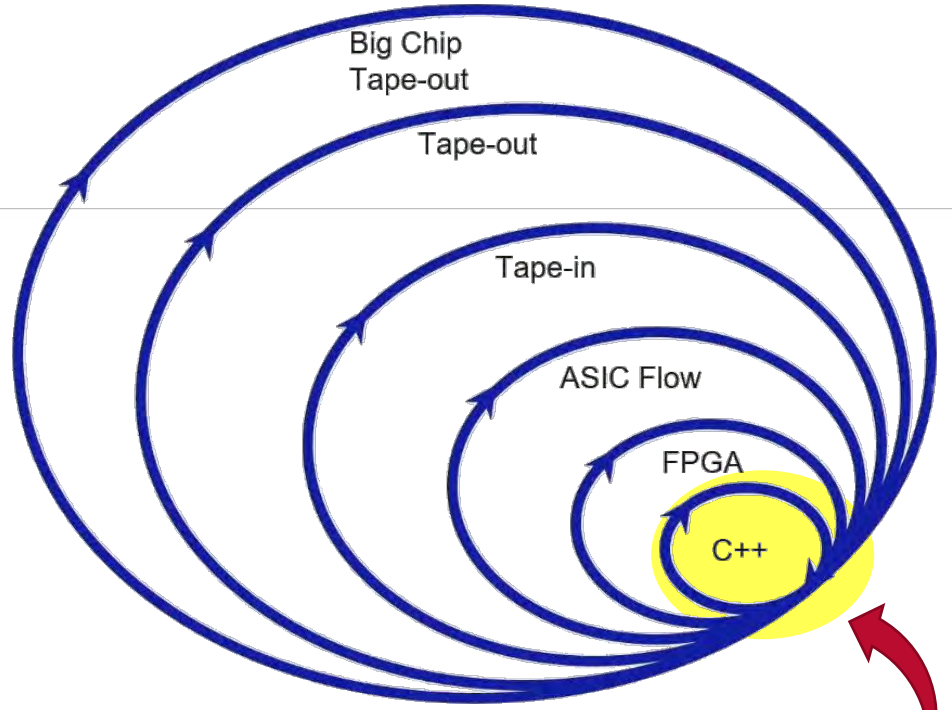
Future vision

UCDAVIS

Future vision



Agile Hardware Dev. Methodology



From Hennessey and Patterson

Turing L

Future vision

Anyone (including non-architect) can download and use gem5

Used for cross-stack research:

Change kernel, change runtime, change hardware, all in concert

Run full ML stacks, full AR/VR stacks… other emerging apps

We’re close… just a lot of rough edges!

RE-gem5

Building a sustainable future

UCDAVIS

RE-gem5 Building a sustainable future

Big news: NSF funding for 3 years to “enhance community infrastructure”

Thanks to

Co-PI Matt Sinclair (U-Wisc)

AMD/Arm collaborators

Hiring a software engineer!

RE-invigorate

RE-juvinate

RE-vitalize

RE-vamp

RE-furbish

RE-novate

RE-model

RE-pair

UCDAVIS

Jason's *unofficial* roadmap

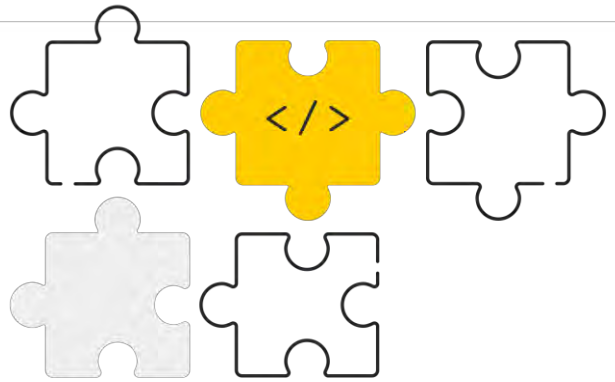
Providing stability

We will have set releases: Likely 3-4 per year

- Releases will be much more thoroughly tested **and documented**
- Can cite “used gem5 release X”

APIs will be stable at the time of releases

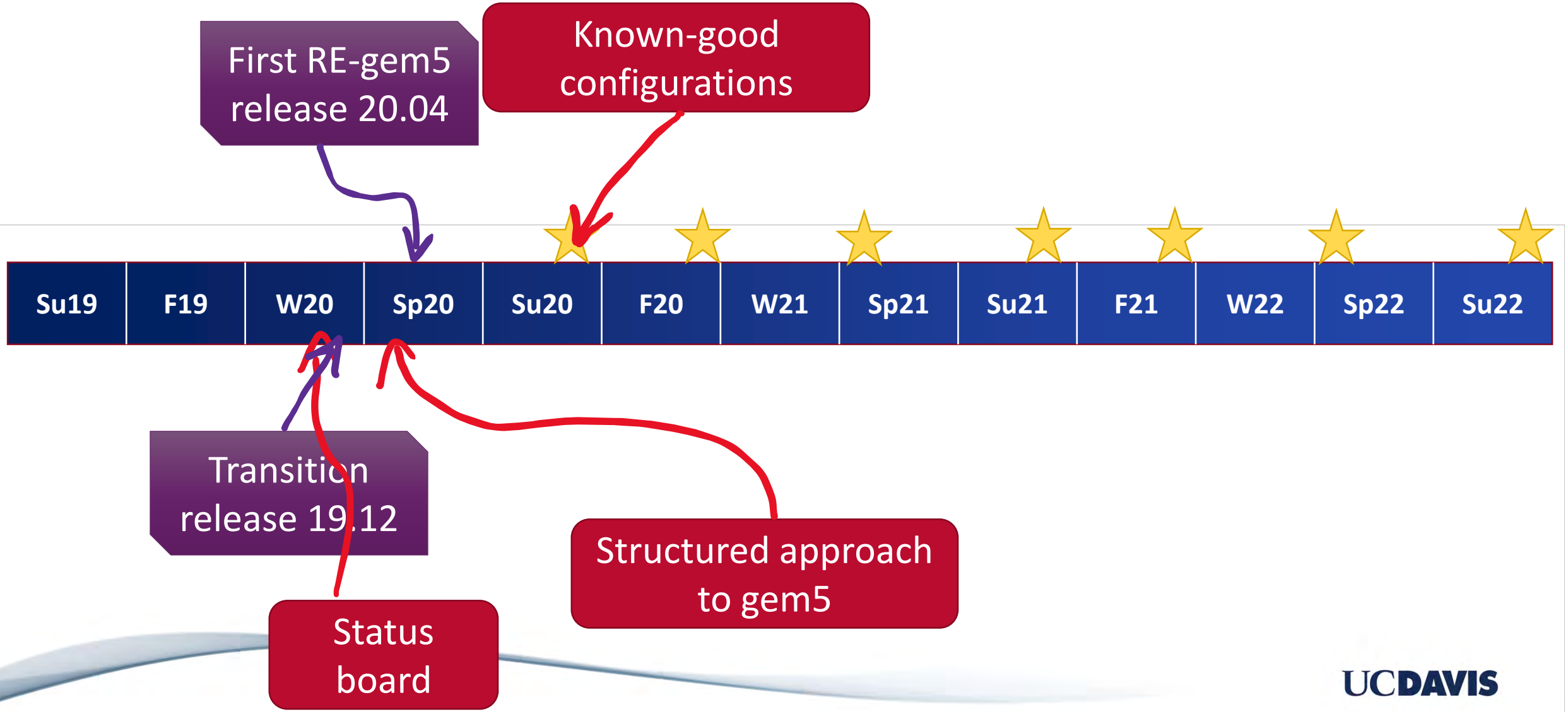
- No longer chasing a moving target
- Define stable APIs and give lead for changes



Improved testing

- Continuous integration tests

Jason's *unofficial* roadmap



Known-good configurations

What do you want to do with a simulator?

Take a baseline, add a new feature, and test

We will provide these baselines as "***known-good configurations***"

Tested at each gem5 release

Community can add workloads and new configurations

Known-good configurations

Researchers can concentrate on one part of the system

Improved methodology sections in papers

- Clearly state baselines
- Readers & reviewers can better understand

Give non-architects a starting point

```
from gem5.model_library import DesktopSystem
```

UCDAVIS

gem5 status & performance board

Shows what works and what doesn't

Show comparison between configurations

Show comparison to hardware

Updated on each gem5 release for all known-good configurations

Status board

Kernel: 5.2.3

Atomic CPU

5.2.3	atomic	8	init	Failure	69
5.2.3	atomic	8	systemd	kernel panic	1118
5.2.3	atomic	2	init	Success!	639
5.2.3	atomic	2	systemd	kernel panic	799
5.2.3	atomic	4	init	Success!	686
5.2.3	atomic	4	systemd	kernel panic	906
5.2.3	atomic	1	init	Running!	9
5.2.3	atomic	1	systemd	kernel panic	853

Out of order CPU

5.2.3	o3	8	init	timeout	10801
5.2.3	o3	8	systemd	timeout	10802
5.2.3	o3	2	init	timeout	10802
5.2.3	o3	2	systemd	timeout	10802
5.2.3	o3	4	init	timeout	10802
5.2.3	o3	4	systemd	timeout	10801
5.2.3	o3	1	init	Success!	9936
5.2.3	o3	1	systemd	kernel panic	10644

KVM CPU

5.2.3	kvm	8	init	Success!	4
5.2.3	kvm	8	systemd	Success!	12
5.2.3	kvm	2	init	Success!	3
5.2.3	kvm	2	systemd	Success!	10
5.2.3	kvm	4	init	Success!	5
5.2.3	kvm	4	systemd	Success!	13
5.2.3	kvm	1	init	Success!	3
5.2.3	kvm	1	systemd	Success!	8

Kernel: 4.14.134

Atomic CPU

4.14.134	atomic	8	init	Success!	1128
4.14.134	atomic	8	systemd	timeout	10801
4.14.134	atomic	2	init	Success!	938
4.14.134	atomic	2	systemd	timeout	10802
4.14.134	atomic	4	init	Success!	986
4.14.134	atomic	4	systemd	timeout	10802
4.14.134	atomic	1	init	Success!	841
4.14.134	atomic	1	systemd	timeout	10801

Out of order CPU

4.14.134	o3	8	init	timeout	10802
4.14.134	o3	8	systemd	timeout	10801
4.14.134	o3	2	init	timeout	10802
4.14.134	o3	2	systemd	kernel panic	5169
4.14.134	o3	4	init	kernel panic	4263
4.14.134	o3	4	systemd	kernel panic	5111
4.14.134	o3	1	init	timeout	10802
4.14.134	o3	1	systemd	timeout	10801

KVM CPU

4.14.134	kvm	8	init	Success!	8
4.14.134	kvm	8	systemd	Success!	15
4.14.134	kvm	2	init	Success!	3
4.14.134	kvm	2	systemd	Success!	9
4.14.134	kvm	4	init	Success!	5
4.14.134	kvm	4	systemd	Success!	11
4.14.134	kvm	1	init	Success!	3
4.14.134	kvm	1	systemd	timeout	10802

Kernel: 4.9.186

Atomic CPU

4.9.186	atomic	8	init	Success!	1114
4.9.186	atomic	8	systemd	timeout	10801
4.9.186	atomic	2	init	Success!	853
4.9.186	atomic	2	systemd	timeout	10801
4.9.186	atomic	4	init	Success!	935
4.9.186	atomic	4	systemd	timeout	10801
4.9.186	atomic	1	init	Success!	876
4.9.186	atomic	1	systemd	timeout	10802

Out of order CPU

4.9.186	o3	8	init	kernel panic	3250
4.9.186	o3	8	systemd	kernel panic	2682
4.9.186	o3	2	init	kernel panic	3201
4.9.186	o3	2	systemd	Running!	25
4.9.186	o3	4	init	kernel panic	3294
4.9.186	o3	4	systemd	Running!	11
4.9.186	o3	1	init	kernel panic	3261
4.9.186	o3	1	systemd	kernel panic	4706

KVM CPU

4.9.186	kvm	8	init	Success!	7
4.9.186	kvm	8	systemd	Success!	16
4.9.186	kvm	2	init	Success!	4
4.9.186	kvm	2	systemd	Success!	9
4.9.186	kvm	4	init	Success!	5
4.9.186	kvm	4	systemd	Success!	11
4.9.186	kvm	1	init	Success!	4
4.9.186	kvm	1	systemd	Success!	7

Kernel: 4.9.186

Atomic CPU

4.4.186	atomic	8	init	timeout	10801
4.4.186	atomic	8	systemd	timeout	10802
4.4.186	atomic	2	init	timeout	10802
4.4.186	atomic	2	systemd	timeout	10802
4.4.186	atomic	4	init	timeout	10801
4.4.186	atomic	4	systemd	timeout	10801
4.4.186	atomic	1	init	timeout	10802
4.4.186	atomic	1	systemd	timeout	10802

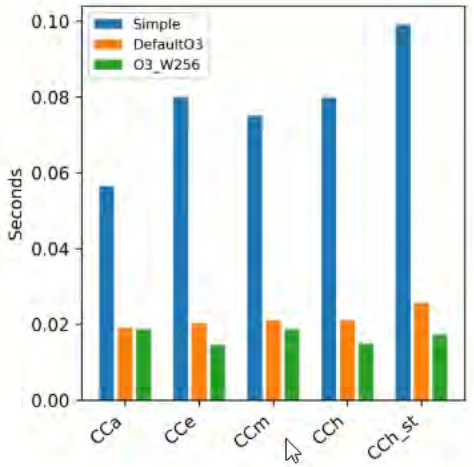
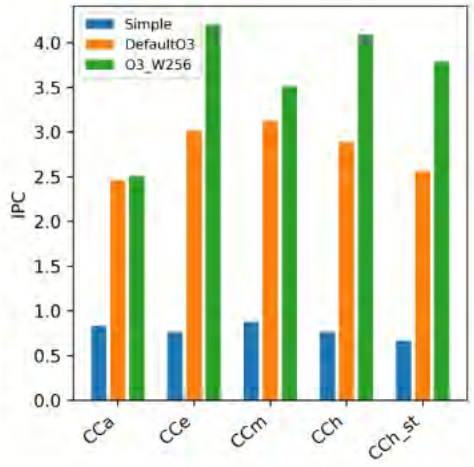
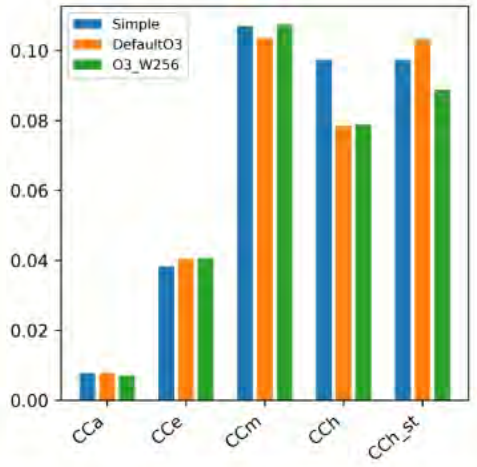
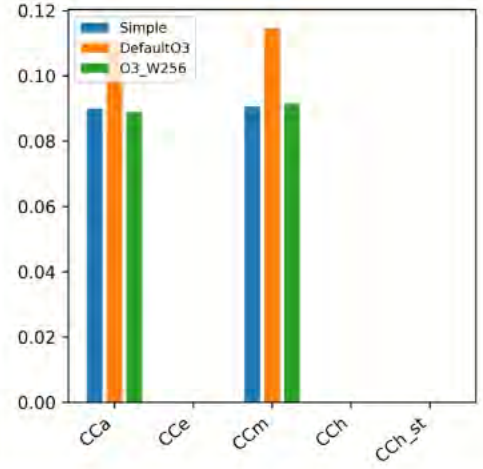
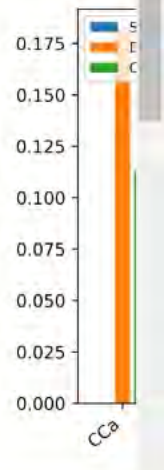
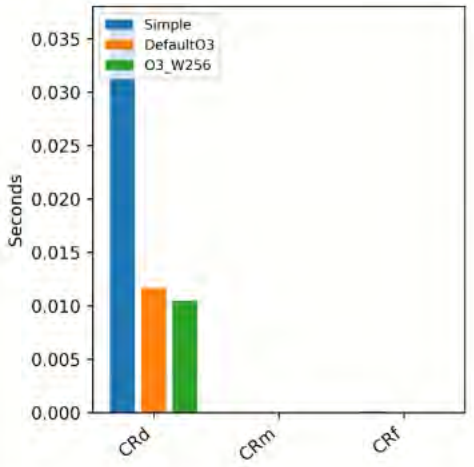
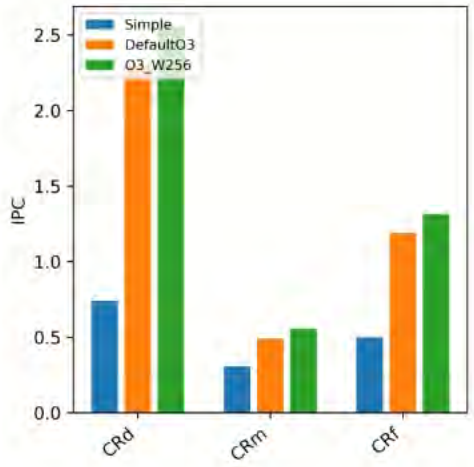
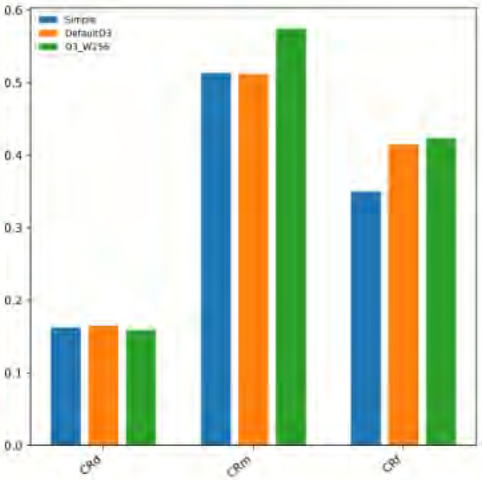
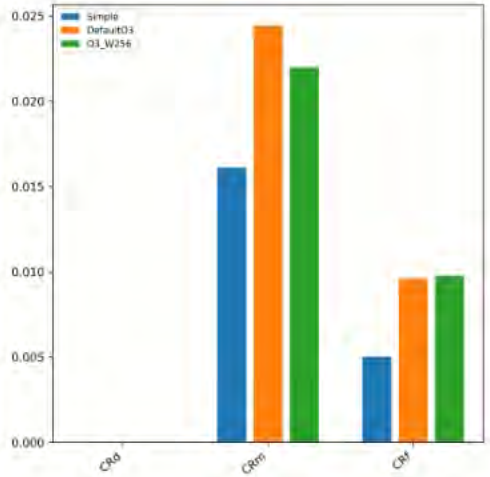
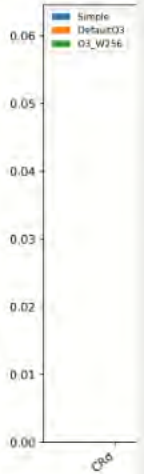
Out of order CPU

4.4.186	o3	8	init	timeout	10802
4.4.186	o3	8	systemd	timeout	10801
4.4.186	o3	2	init	sigterm	1216
4.4.186	o3	2	systemd	timeout	10801
4.4.186	o3	4	init	timeout	10801
4.4.186	o3	4	systemd	timeout	10801
4.4.186	o3	1	init	timeout	10801
4.4.186	o3	1	systemd	timeout	10801

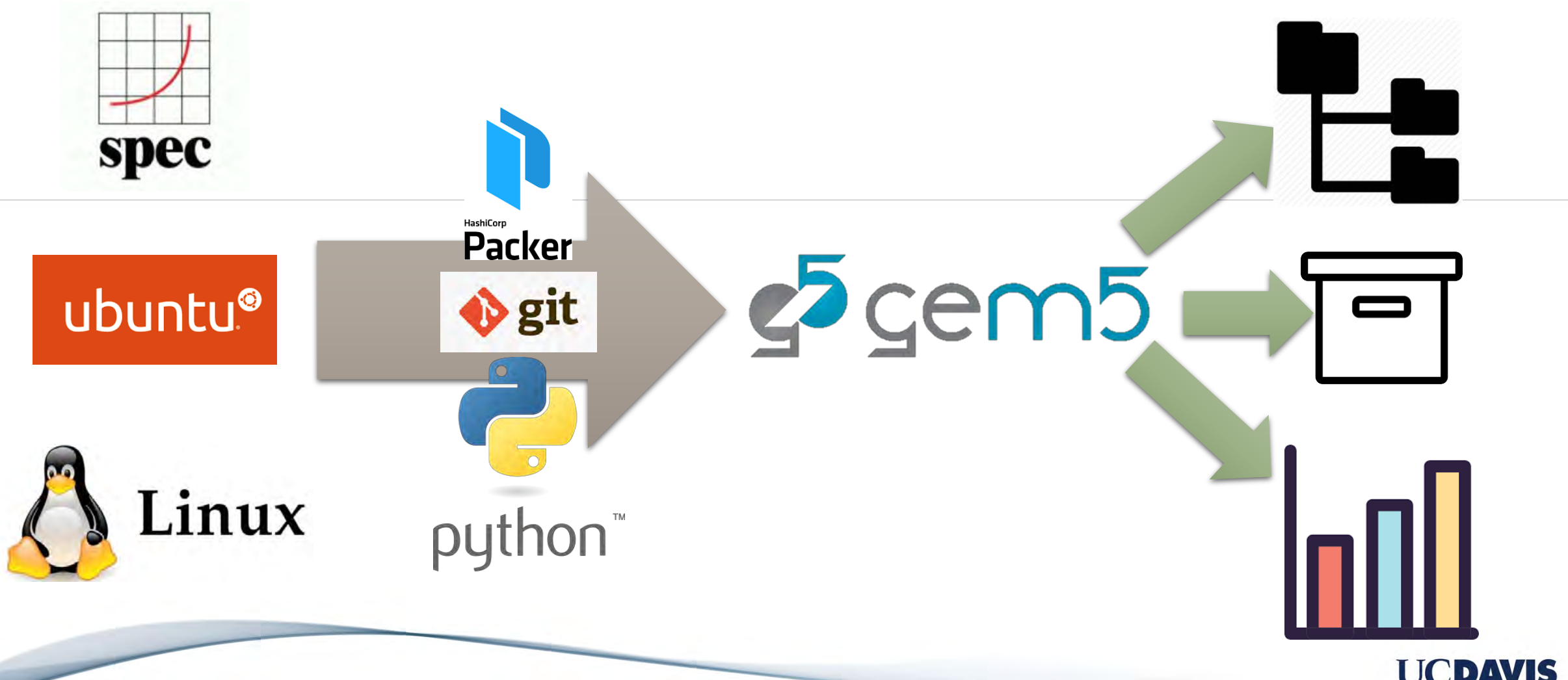
KVM CPU

4.4.186	kvm	8	init	Terminated	4
4.4.186	kvm	8	systemd	Terminated	4
4.4.186	kvm	2	init	Terminated	2
4.4.186	kvm	2	systemd	Terminated	2
4.4.186	kvm	4	init	Terminated	3
4.4.186	kvm	4	systemd	Terminated	3
4.4.186	kvm	1	init	Terminated	2
4.4.186	kvm	1	systemd	Terminated	2

Benchmark Centric Results for X86

Category	Sub-Category	Time	IPC	Branch Miss Prediction Rate	L1 Data Cache Miss Rate	L1 Instruction
Control Benchmarks	Conditional					
	Recursive					

gem5: A structured approach



gem5: A structured approach

Set of protocols for gem5 users

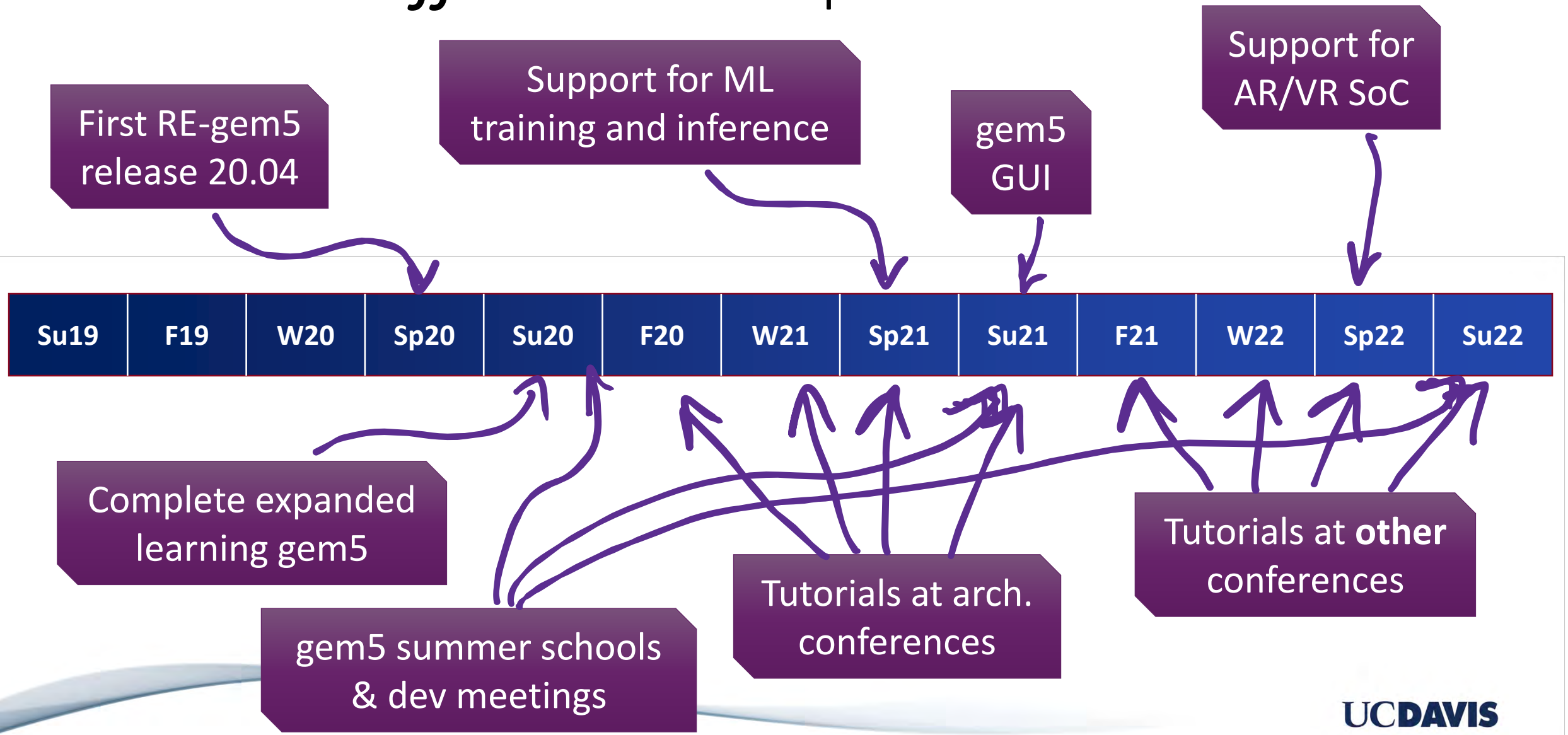
Automatically records and tracks dependencies

- Benchmarks
- Automatic disk image creation
- Linux kernel

Scripts to run gem5

- Set structure for output to make it easy to graph things
- Run jobs on external server

Jason's *unofficial* roadmap



Other plans

Improved testing and stability (tests for the 400 models!)

New website

Community code shepherding

Many more…

Conclusions

: Solid foundation with 15+ years of use

Thriving community

Large number of models

Flexible simulation

Building sustainable infrastructure for the next 15 years

From you

Feedback on roadmap and new features

What is missing for ***you*** to use gem5?

Ways to help:

- Join the advisory board
- Join the dev community
- Fund us!
- Learn more about gem5: http://gem5.org

Email jlowepower@ucdavis.edu

UCDAVIS