gem5: Some cool things now and what the future holds

Jason Lowe-Power

jlowepower@ucdavis.edu

DArchR
DAVIS ARCHITECTURE RESEARCH

gem5

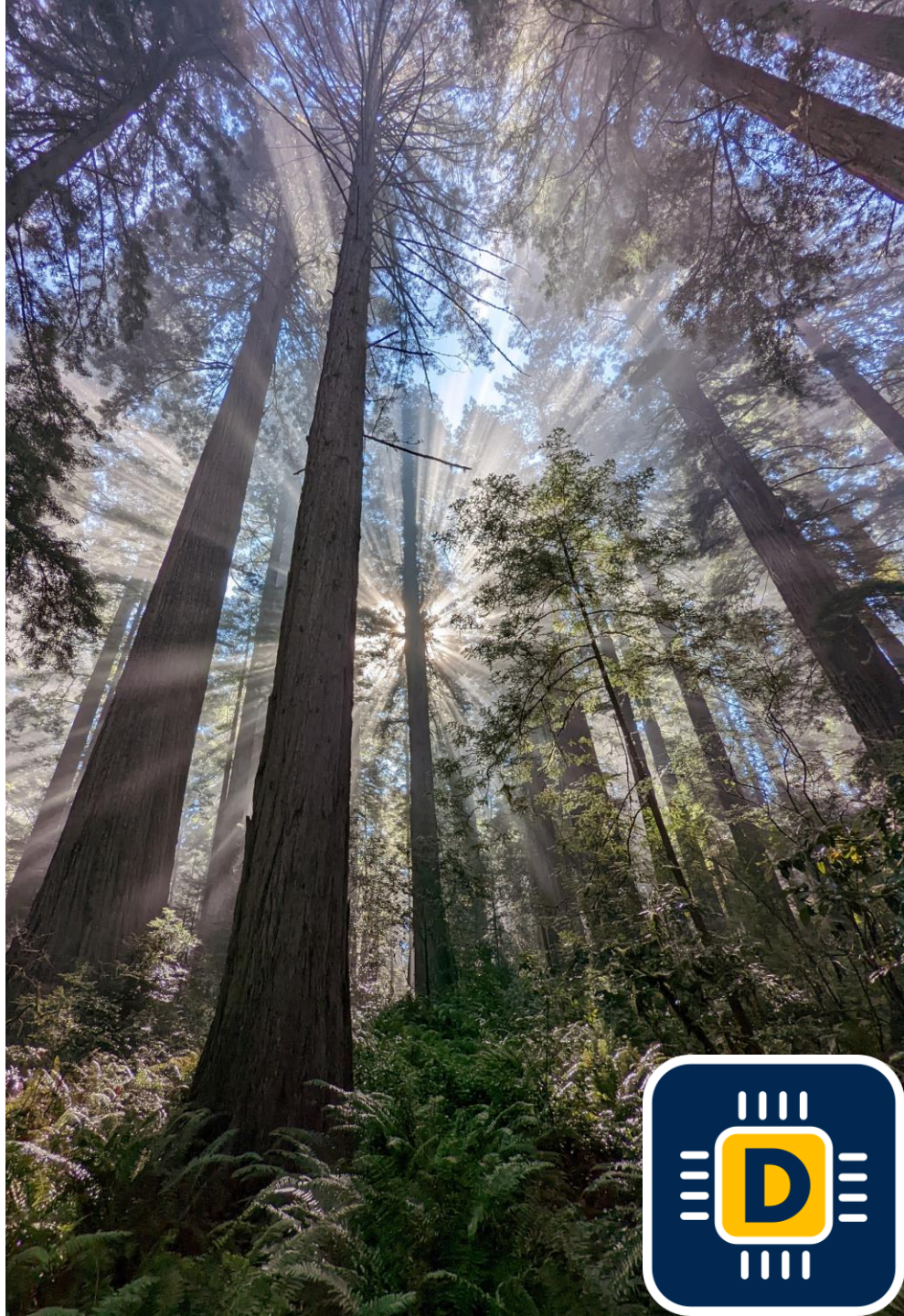
UCDAVIS
COMPUTER SCIENCE

Outline

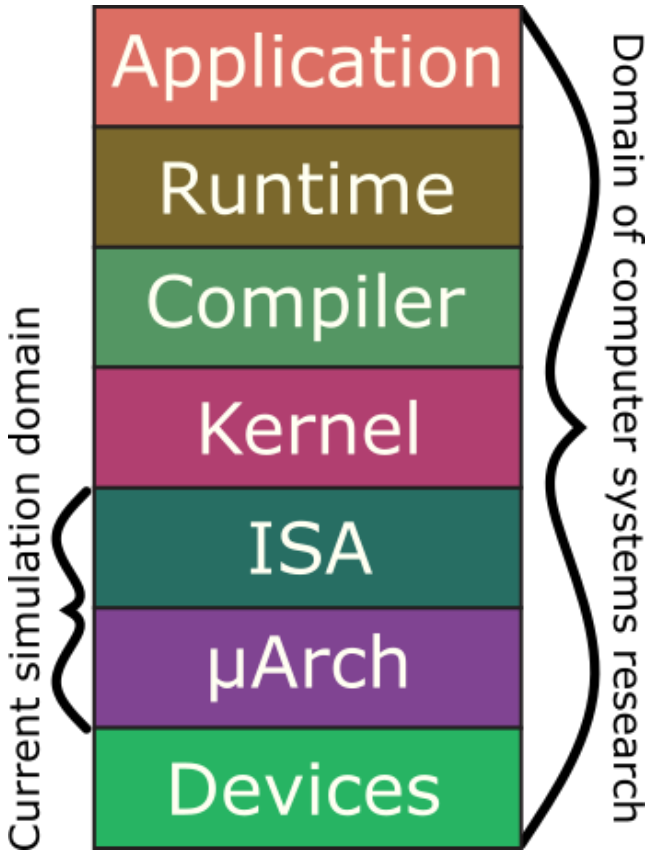
Some examples of gem5's uses

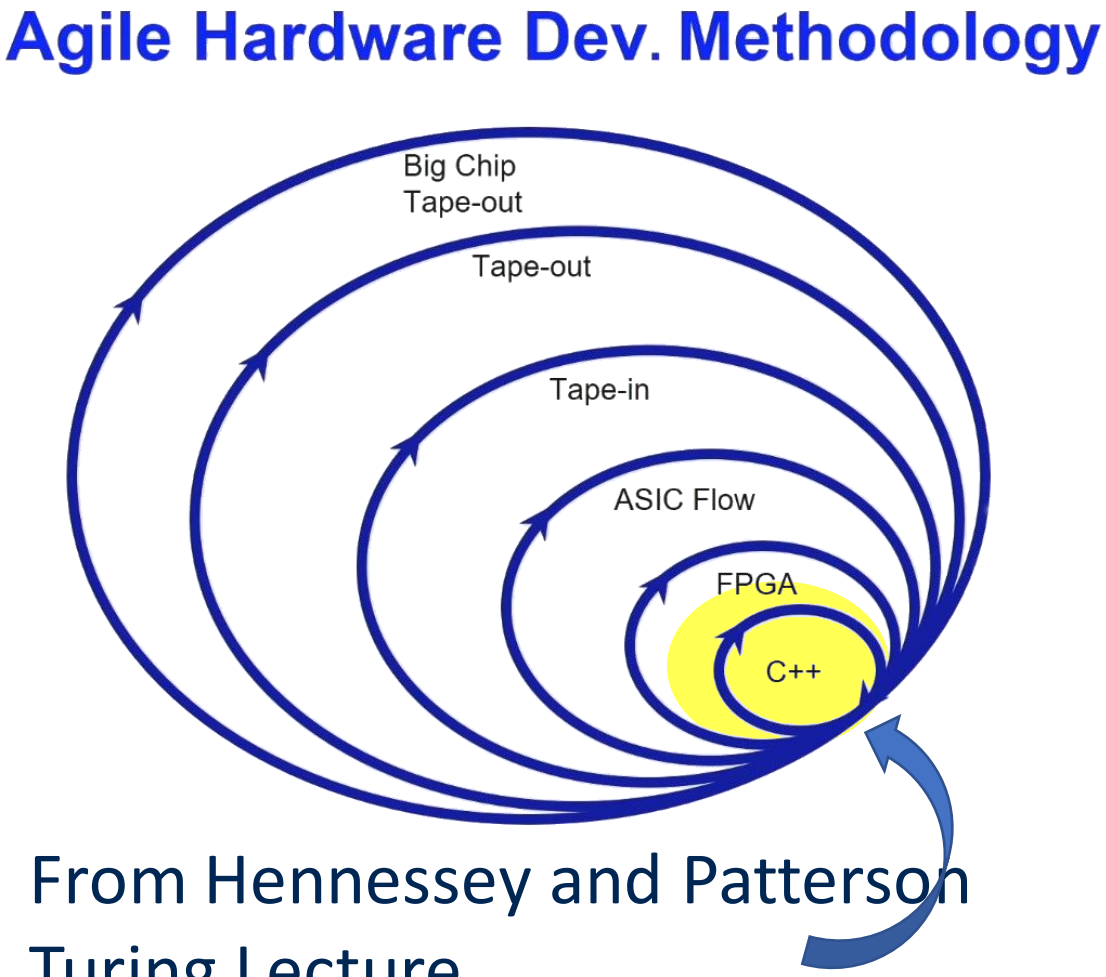
Future of gem5: supercomputers

What has made gem5 a success?



gem5's goals





From Hennessey and Patterson Turing Lecture

Cool use cases for gem5

1. Ease of use with the standard library and gem5 resources

2. Modeling new types of devices: CXL and disaggregated memory

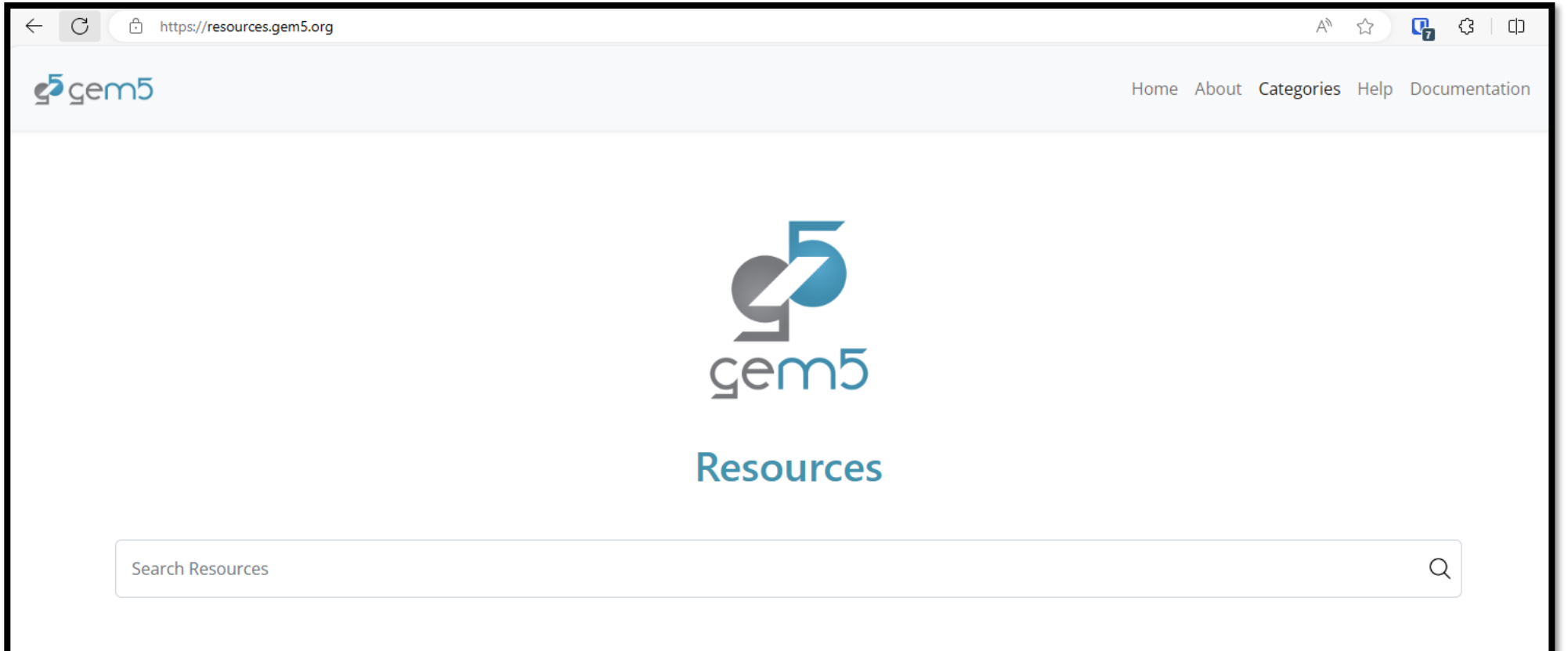
3. Full-system DRAM caches
4. Hardware-software co-design for security
5. Modeling accelerators

gem5 resources

Setting up simulations is complicated

Kernel, operating system, libraries, workloads, annotations, inputs, etc.

gem5 resources contains everything you need

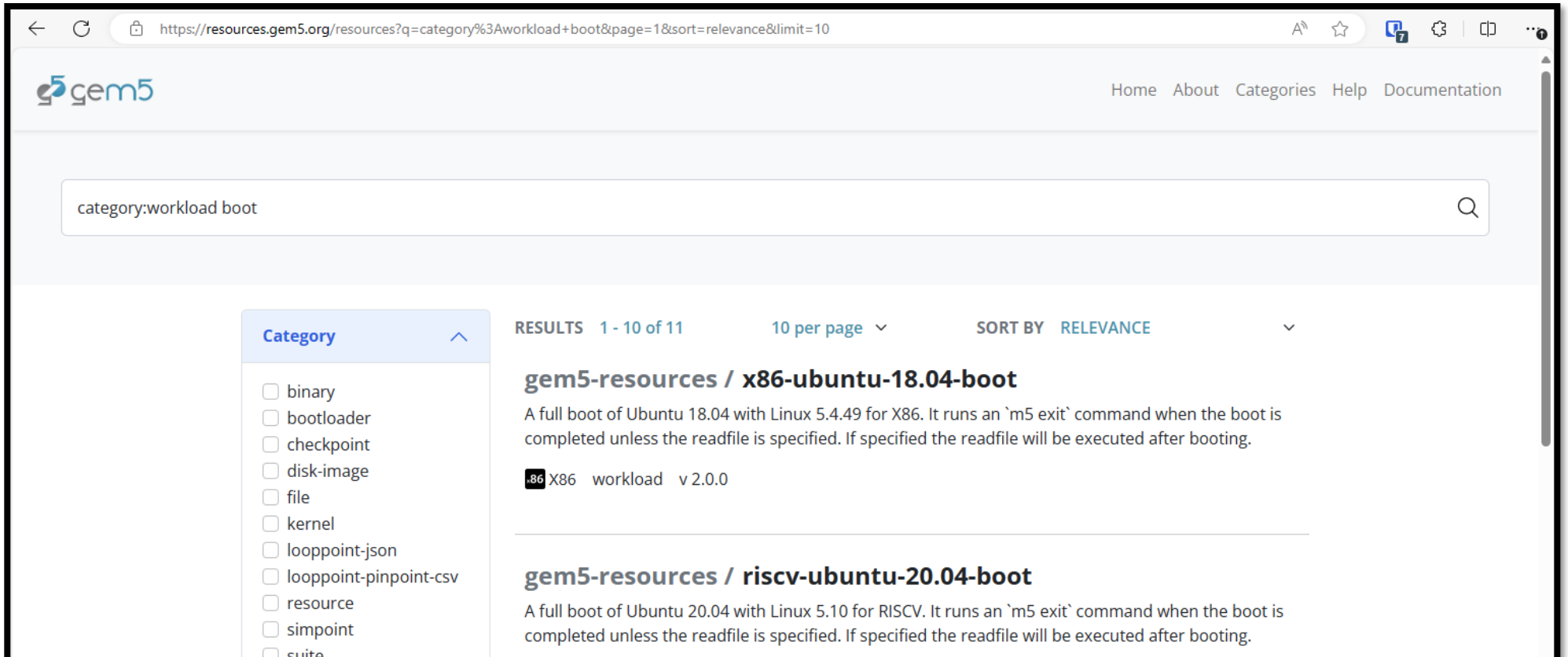


gem5 resources

Setting up simulations is complicated

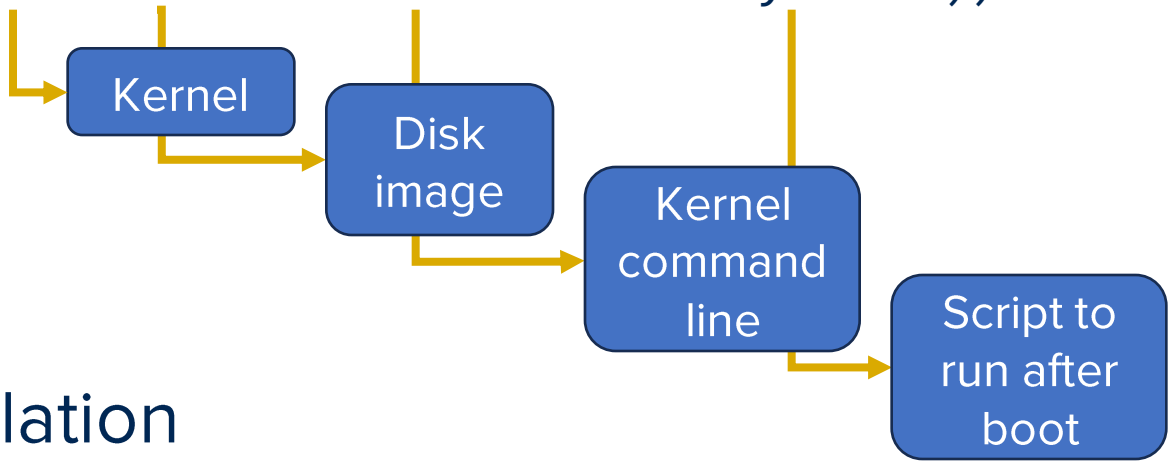
Kernel, operating system, libraries, workloads, annotations, inputs, etc.

gem5 resources contains everything you need



gem5 resources

```
board = X86Board()
board.set_workload(obtain_resource("x86-ubuntu-22.04-boot-with-systemd"))
simulator = Simulator(board=board)
simulator.run()
```



Checkpoints

Full system and syscall emulation

Suites: Multiple workloads

With multiprocessing support*

*Coming soon

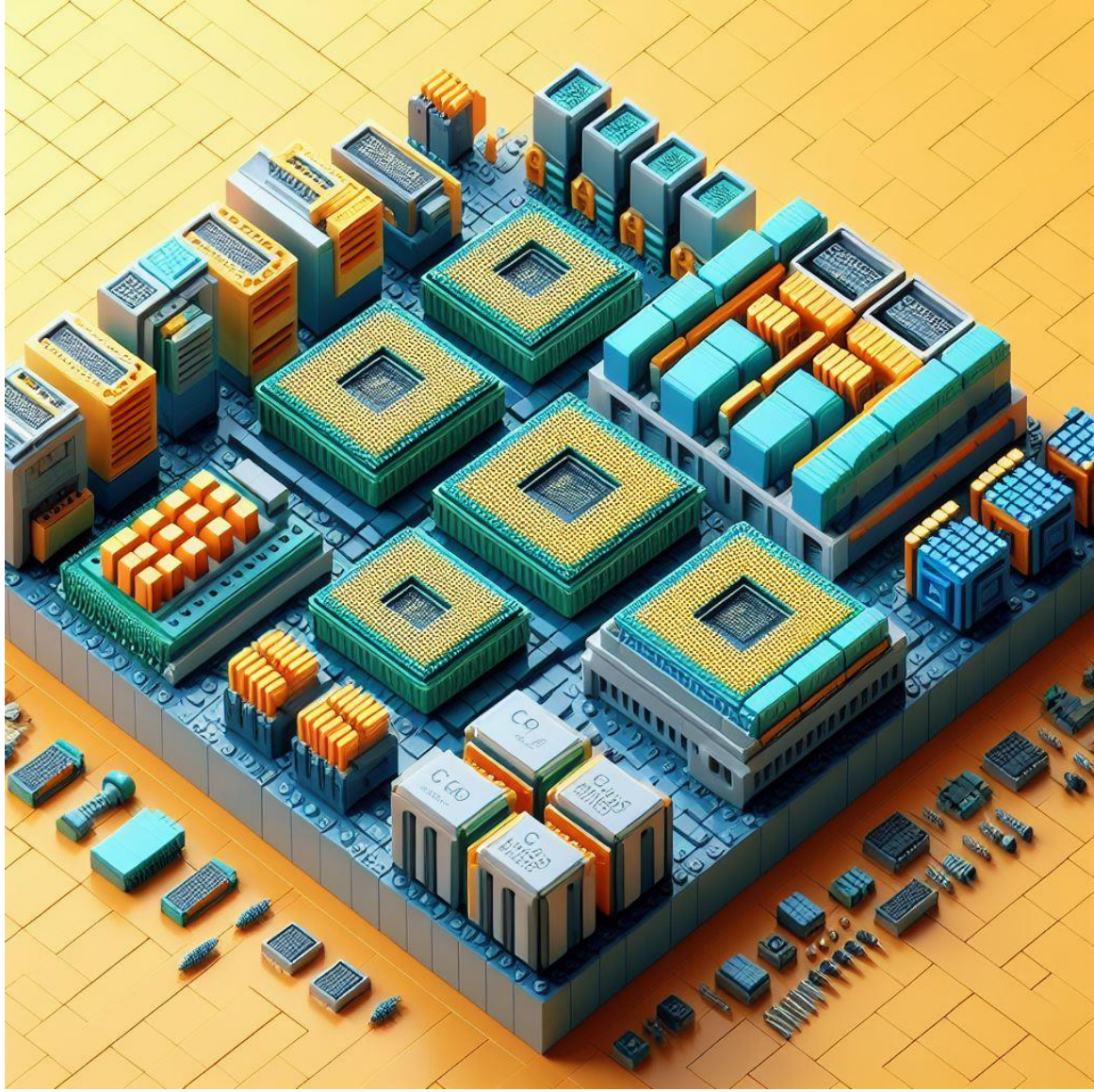
```
sims = []
for benchmark in obtain_resource('x86-speccpu2006'):
    board = X86Board()
    board.set_workload(benchmark)
    sims.append(Simulator(board=board))
multisim.add_simulators(sims)
multisim.run_all(processes=8)
```

gem5 standard library

Building blocks for systems

Plug *processor*, *cache hierarchy*, *memory* into a **board**

Like microcenter for simulations



Lead: Mahyar Samani

gem5 standard library

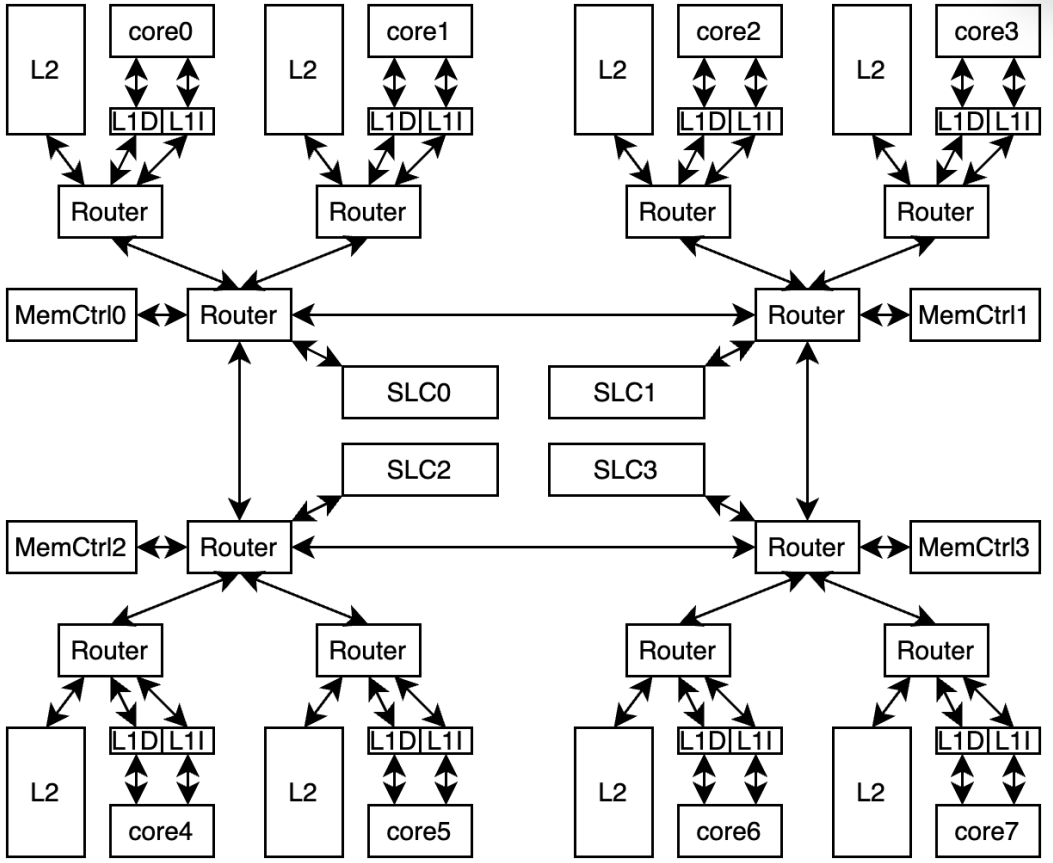
Goal: Accurate model of modern systems

Case study: Arm Neoverse N1 cores and CMN-700

Allow users to focus on their innovation

Provide simple interface

Enable complexity through *extension*, not *parameterization*



Extension in the standard library

```
board = NovoverseSystemFS("4GHz", num_cores, vlen)
board.set_workload(obtain_resource("test-latency"))
simulator = Simulator(board=board)
simulator.run()
```

```
class NovoverseSystemFS(ArmBoard):
    def __init__(self, clk_freq, num_cores, vlen):
        release = ArmDefaultRelease()
        platform = VExpress_GEM5_Foundation()
        sve_parameters = ARM_SVE_Parameters(vlen=vlen, is_fullsystem=True)
        processor = NovoVectorProcessor(num_cores, sve_parameters)
        memory = QuadChannelDDR4_2400_16x4()
        cache_hierarchy = CoherentMeshNetwork()
```

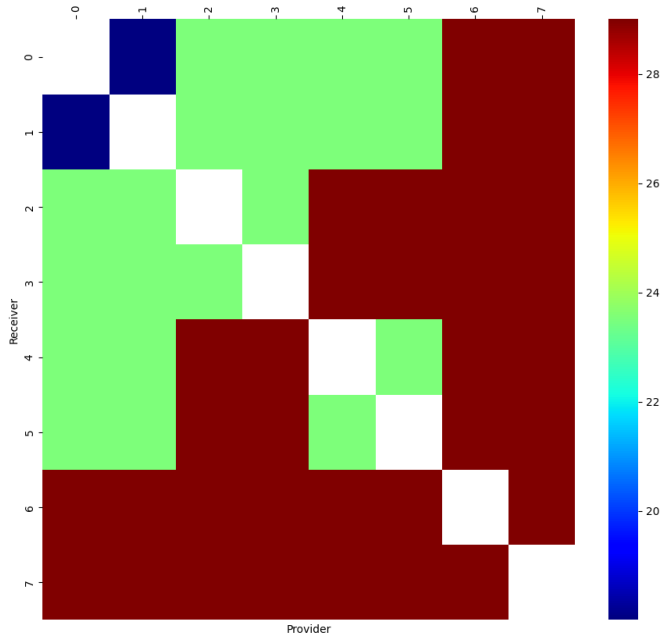
https://github.com/darchr/novoverse

Example validation

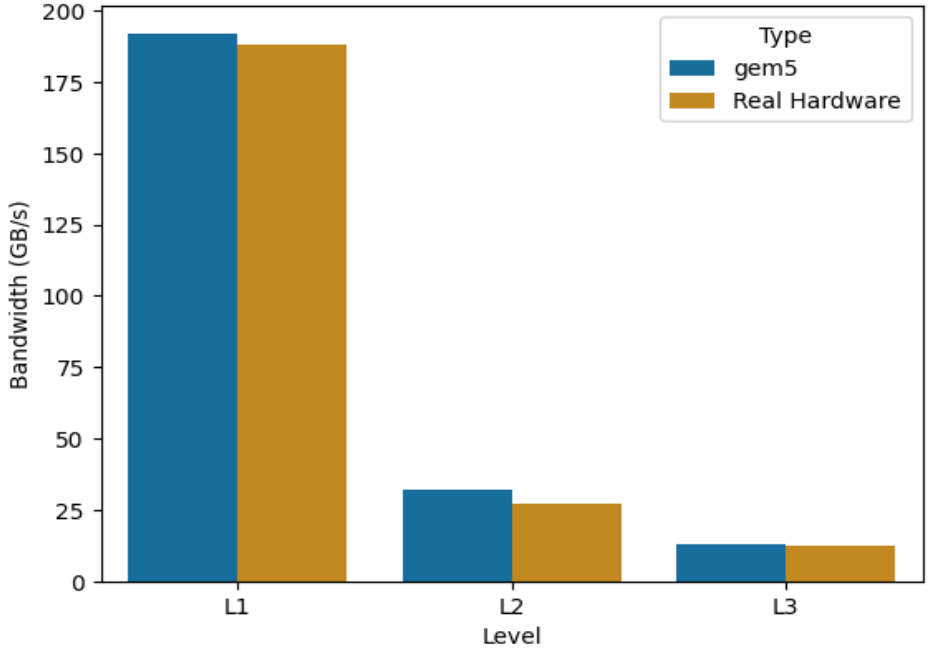
Validating component-by-component

- Memory, cache-to-cache latency, cache-to-core bandwidth
- Synthetic workloads

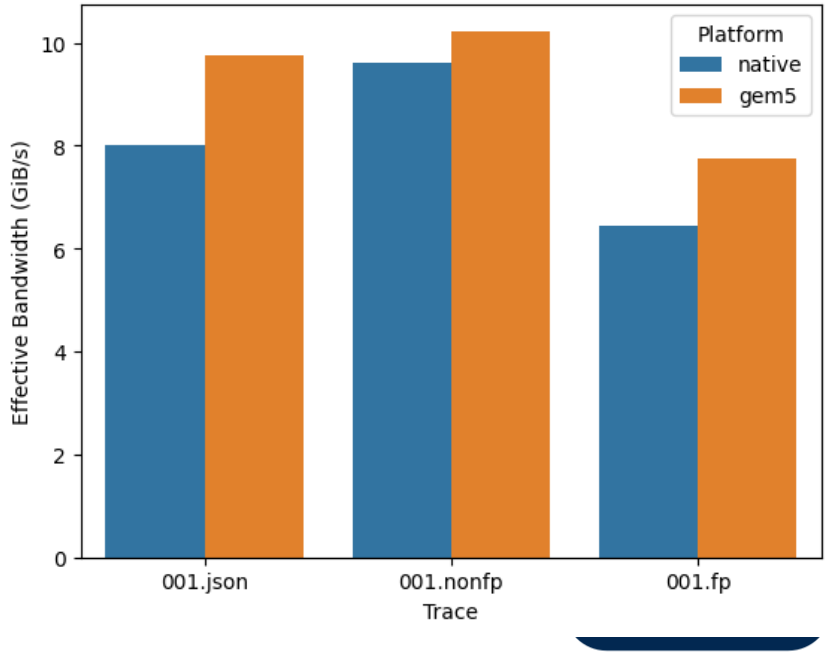
Core to core latency



Bandwidth

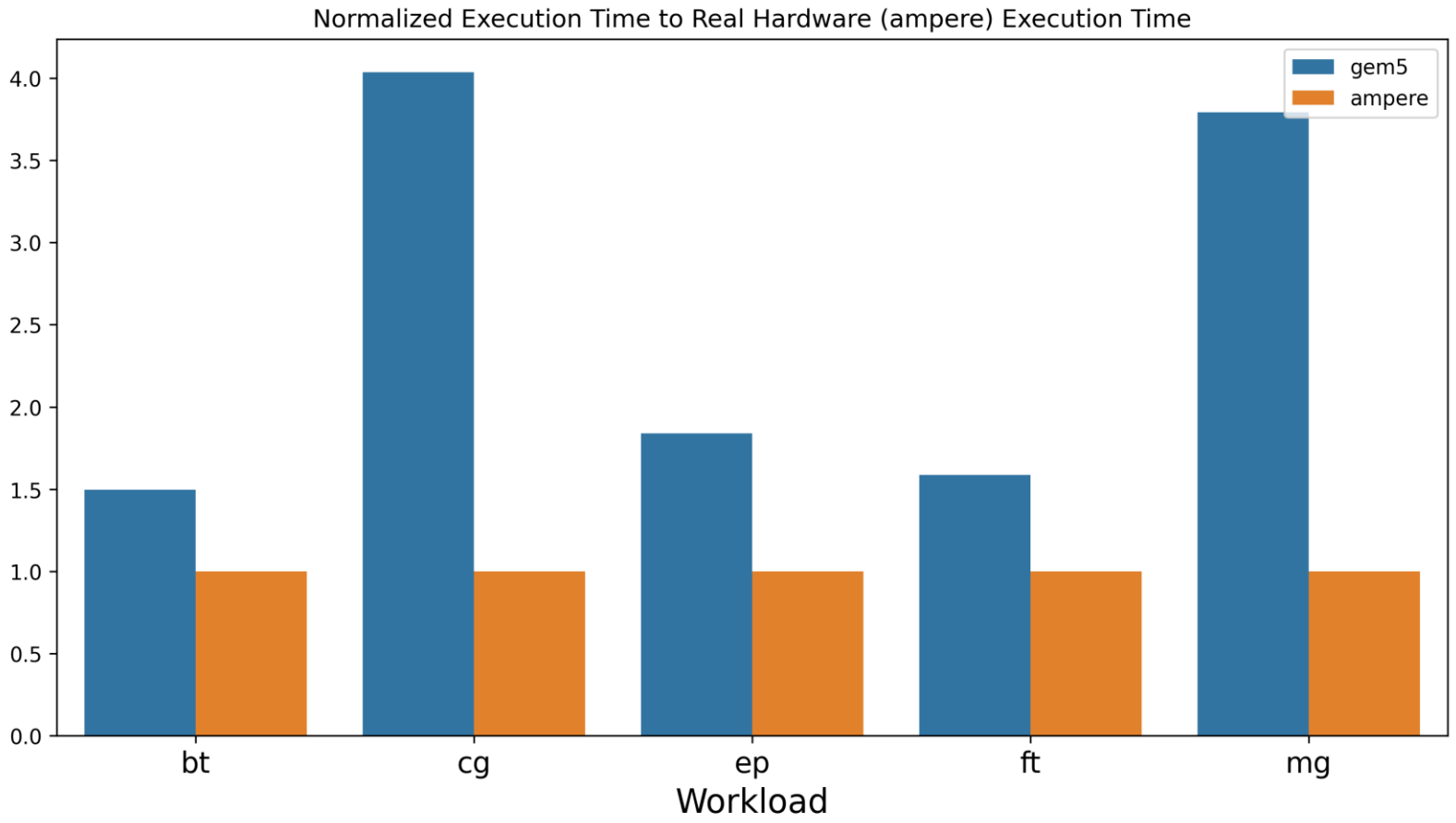


Spatter



Full applications

Unfortunately, the out-of-order CPU is not accurate



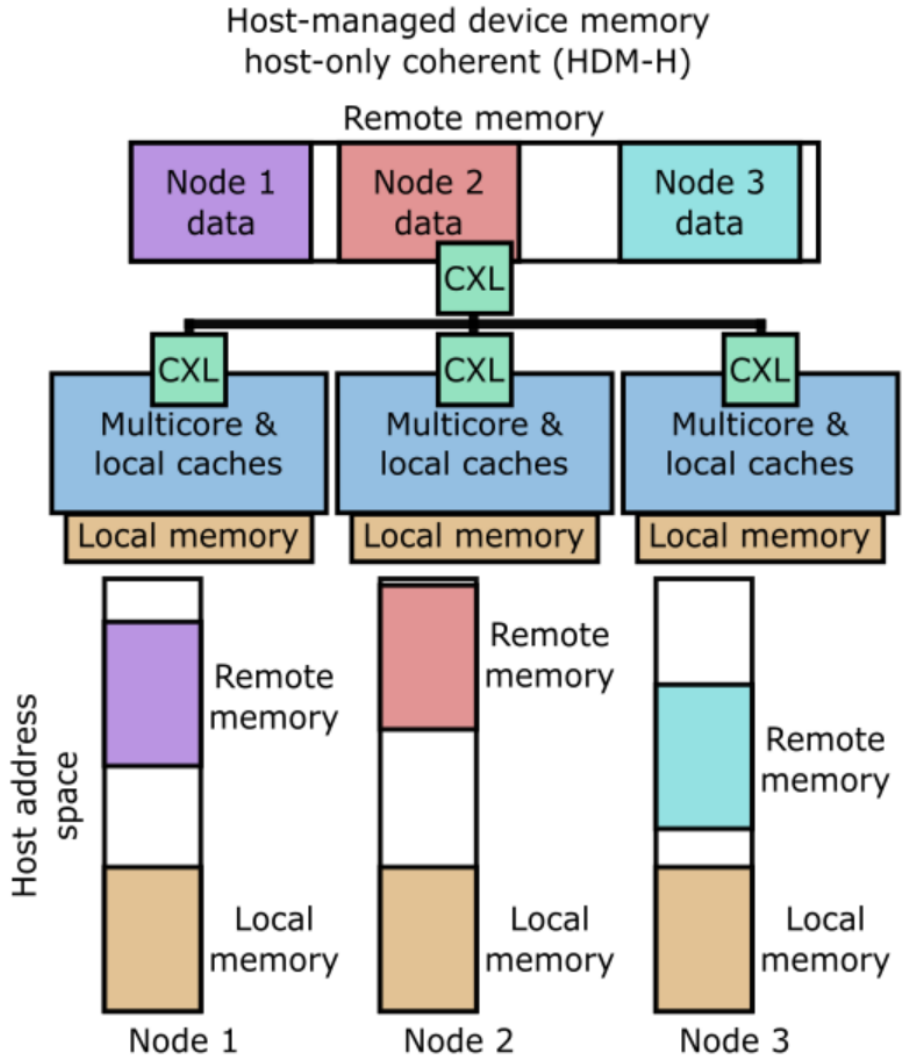
Modeling emerging large-scale systems

CXL is an emerging standard for remote memory

Many questions about performance, cost…

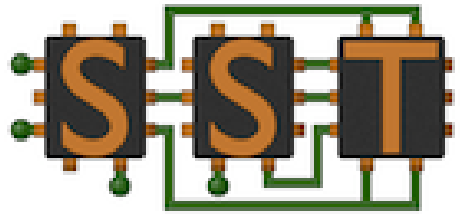
Full-system effects

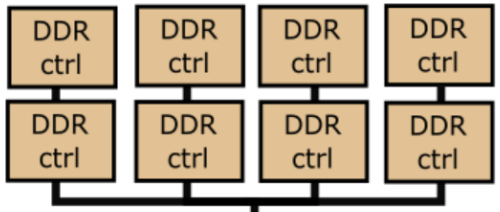
- NUMA, OS, DMA
- Software

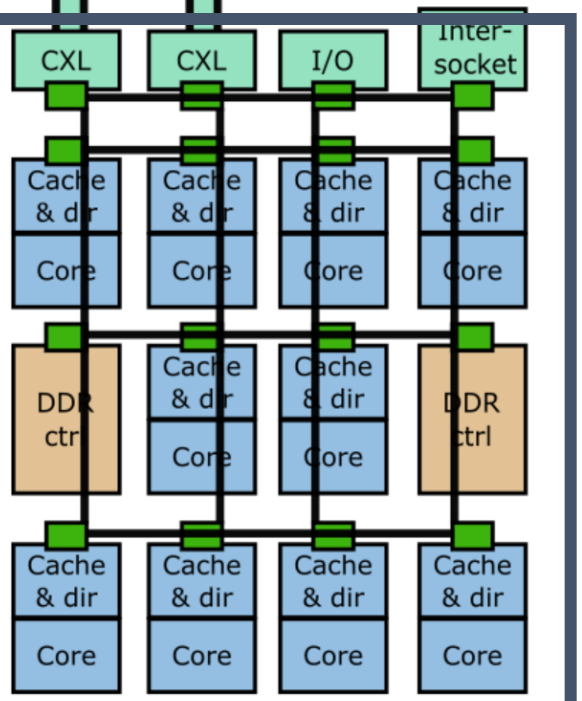


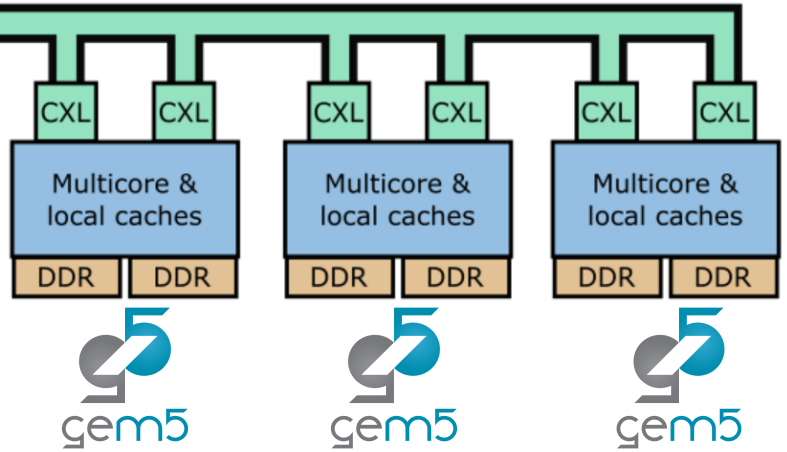
Lead: Kaustav Goswami

Remote memory with CXL









Main difficulties

"Functional" accesses

- gem5 likes to fake things, SST does **not**
- **Solution**: Mirror memory for gem5

gem5 is slow and full-system is a lot

- **Solution:** Checkpointing and fast-forwarding
- Only supported in "init" phase

Available soon™ at https://arch.cs.ucdavis.edu/projects/gem5-cxl

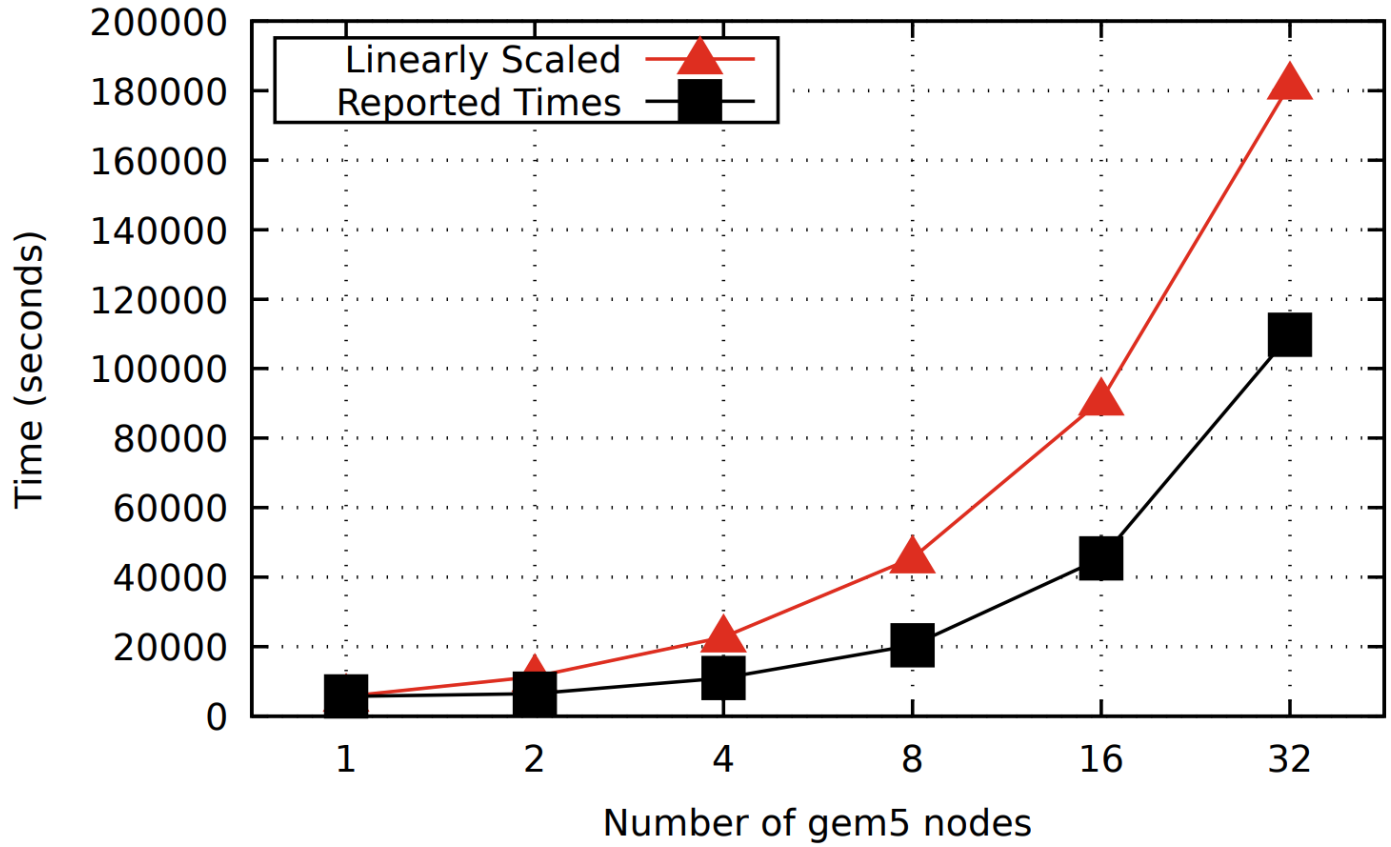
Current status of infrastructure

Working

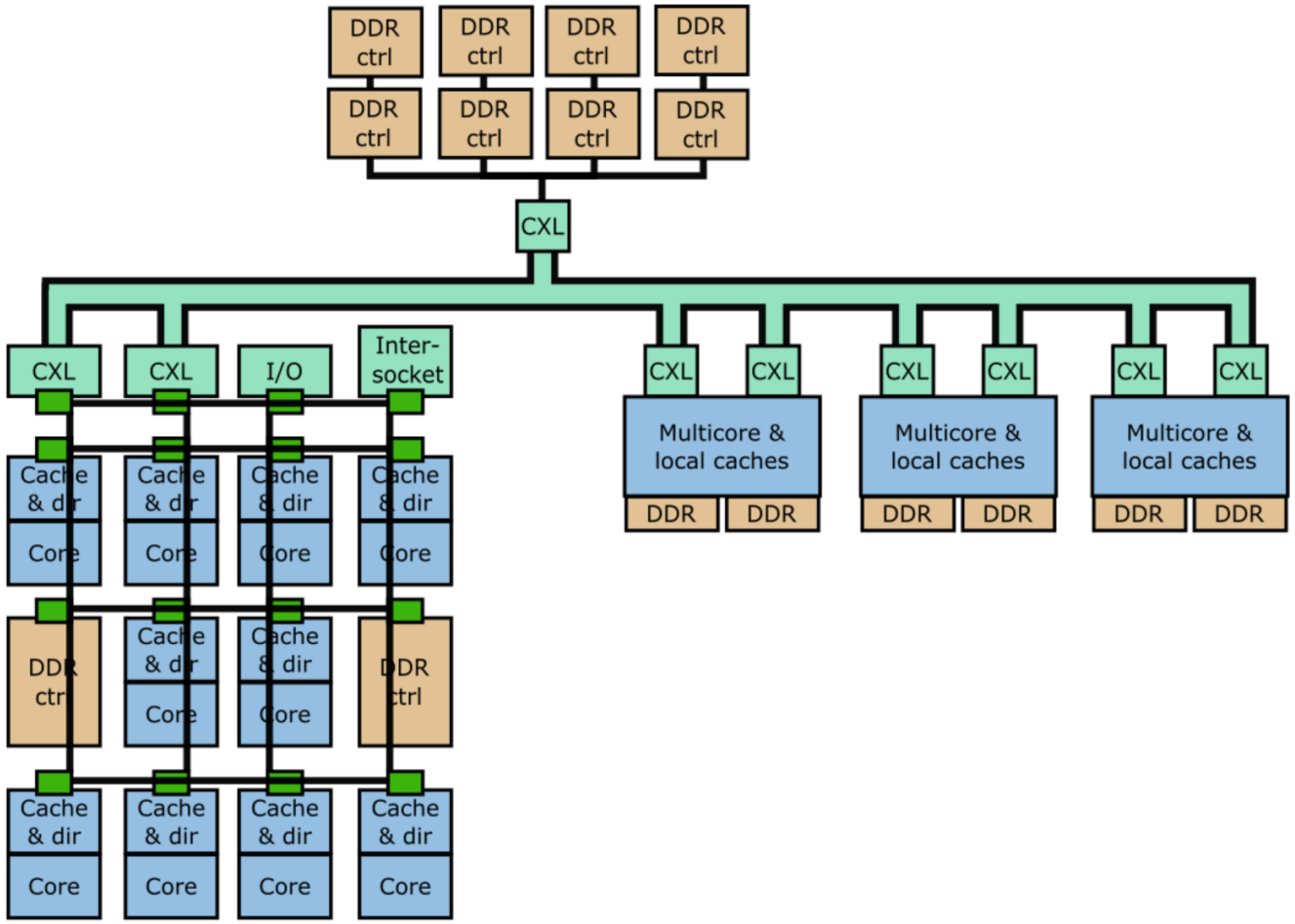
- Full system boot
- Checkpointing
- Arm
- NUMA

In progress

- Multi-ISA
- Sharing
- Real workloads



Cool research directions



Investigate full system

Node-level coherence?

Local DDR as cache?

Memory tiering

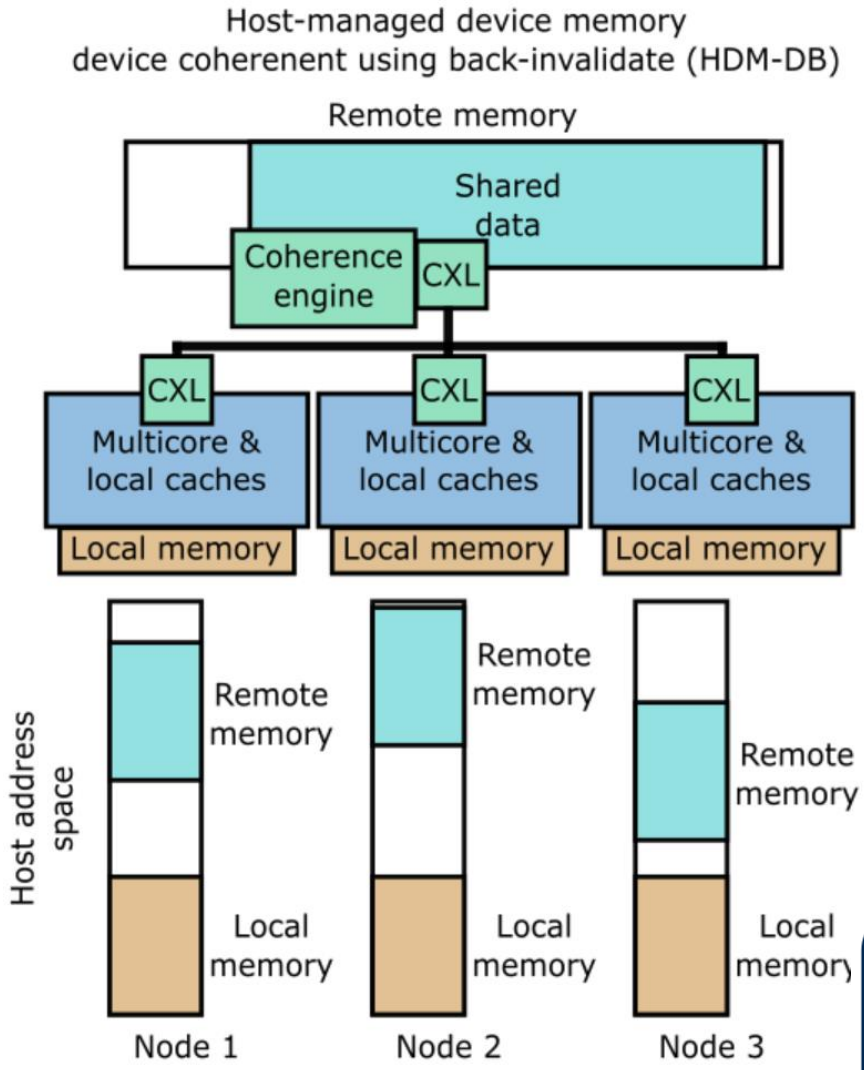
Really cool research directions

CXL 3.0/3.1 defines sharing semantics

How to manage coherence?

Local memory as a cache?

Memory tiering with coherence?



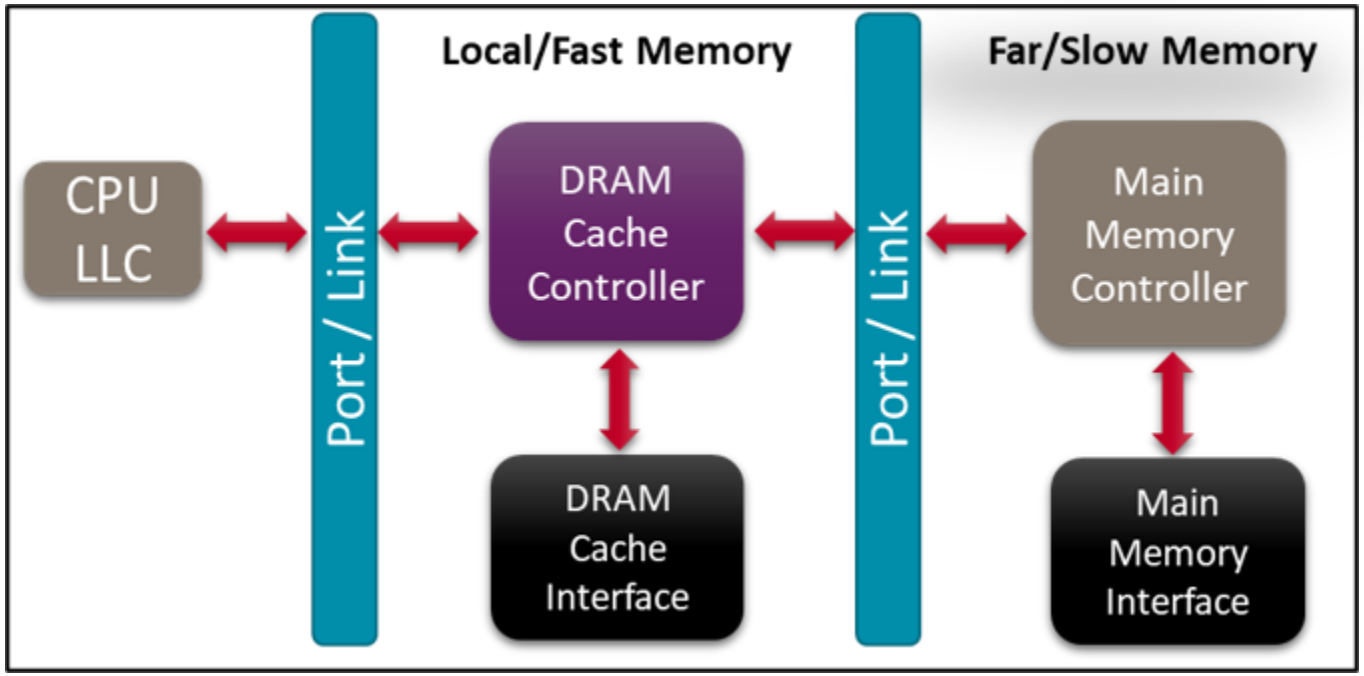
Lead: Maryam Babaie

Full system with DRAM cache

Key capability:

- Modify DRAM microarch.
- Investigate performance

High confidence of “reasonable” design

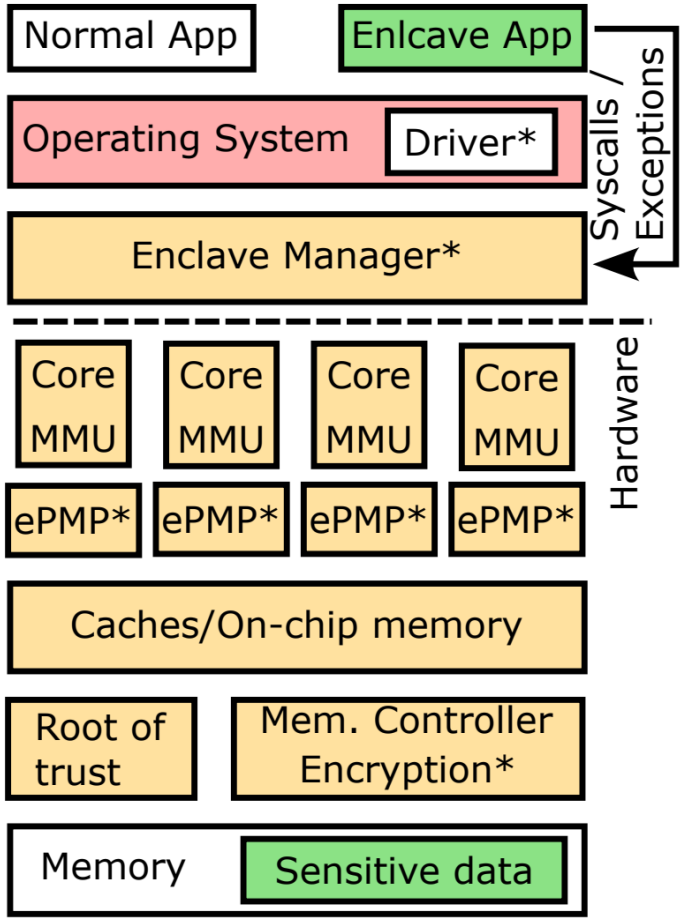


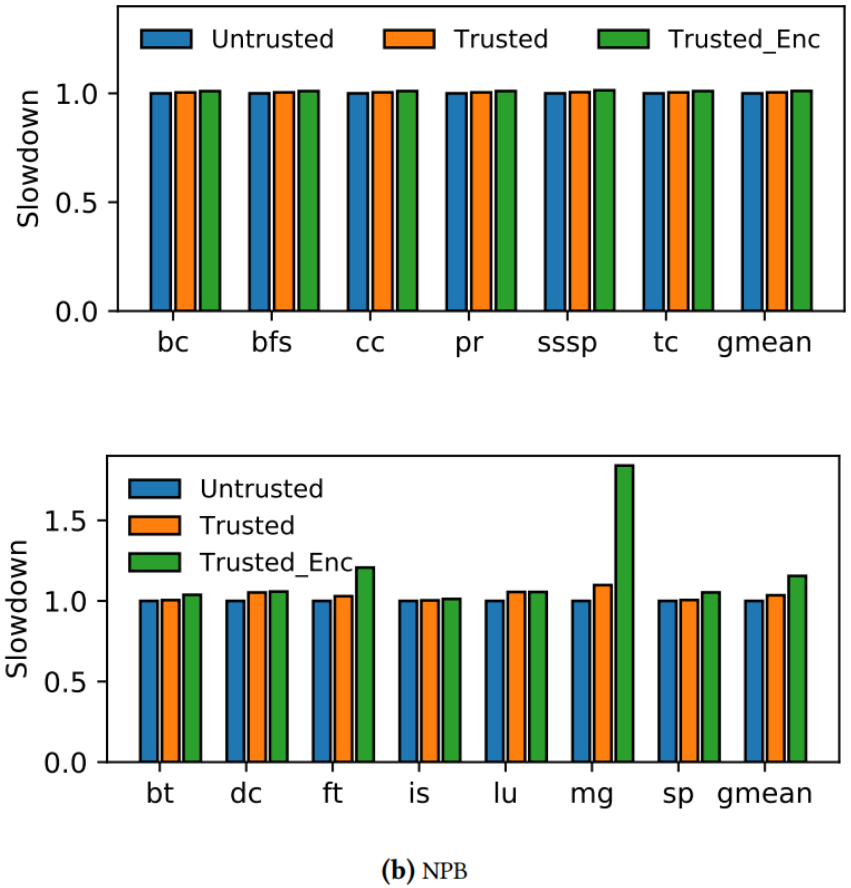
https://arxiv.org/abs/2404.14617

Lead: Dr. Ayaz Akram

H/W S/W co-design for security

Problem: How to do scientific computing on sensitive data?





(b) NPB

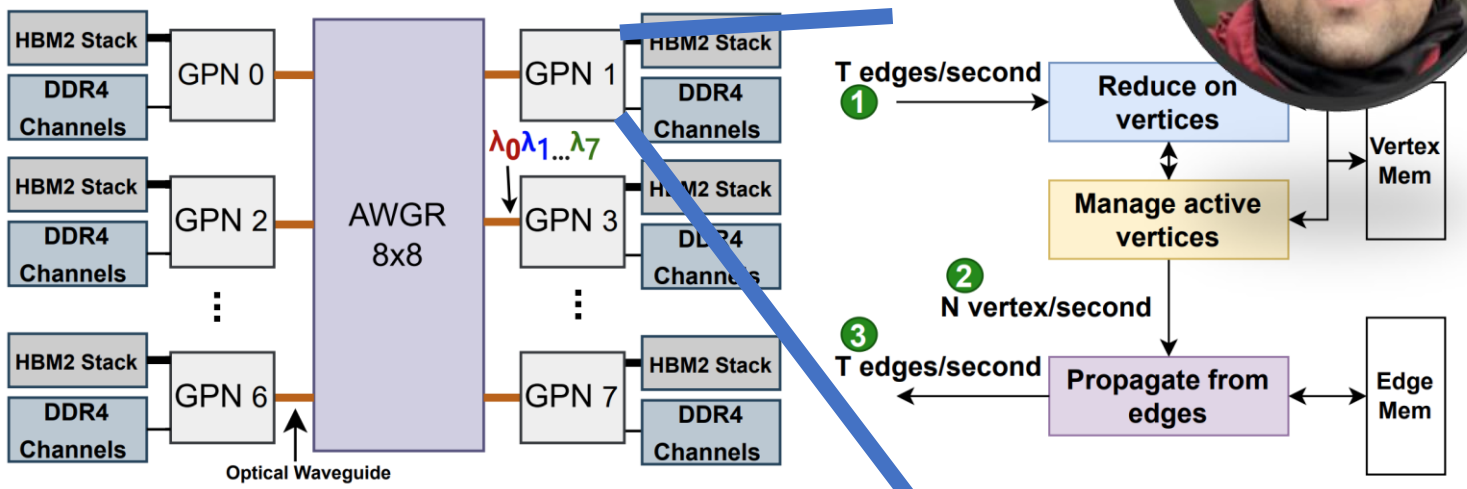
Lead: Mahyar Samani

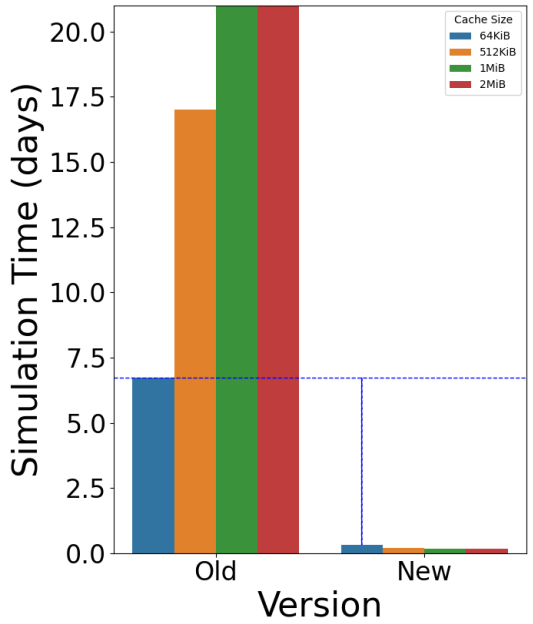
SEGA: Scaling graph accelerators

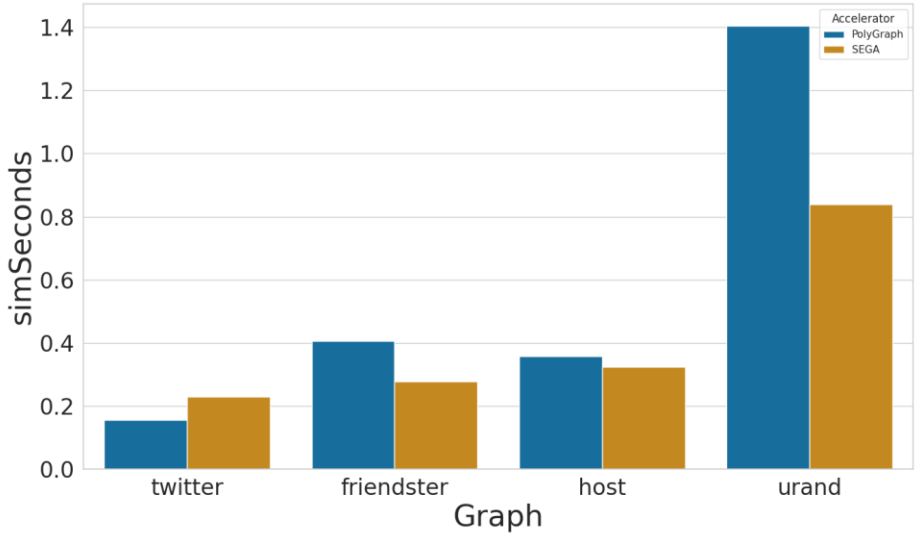
Goal: Run graph analytics on largest data

Problem: Simulation time of weeks-years

Careful optimizations and careful modeling enabled new innovations







Other cool things

HammerSim

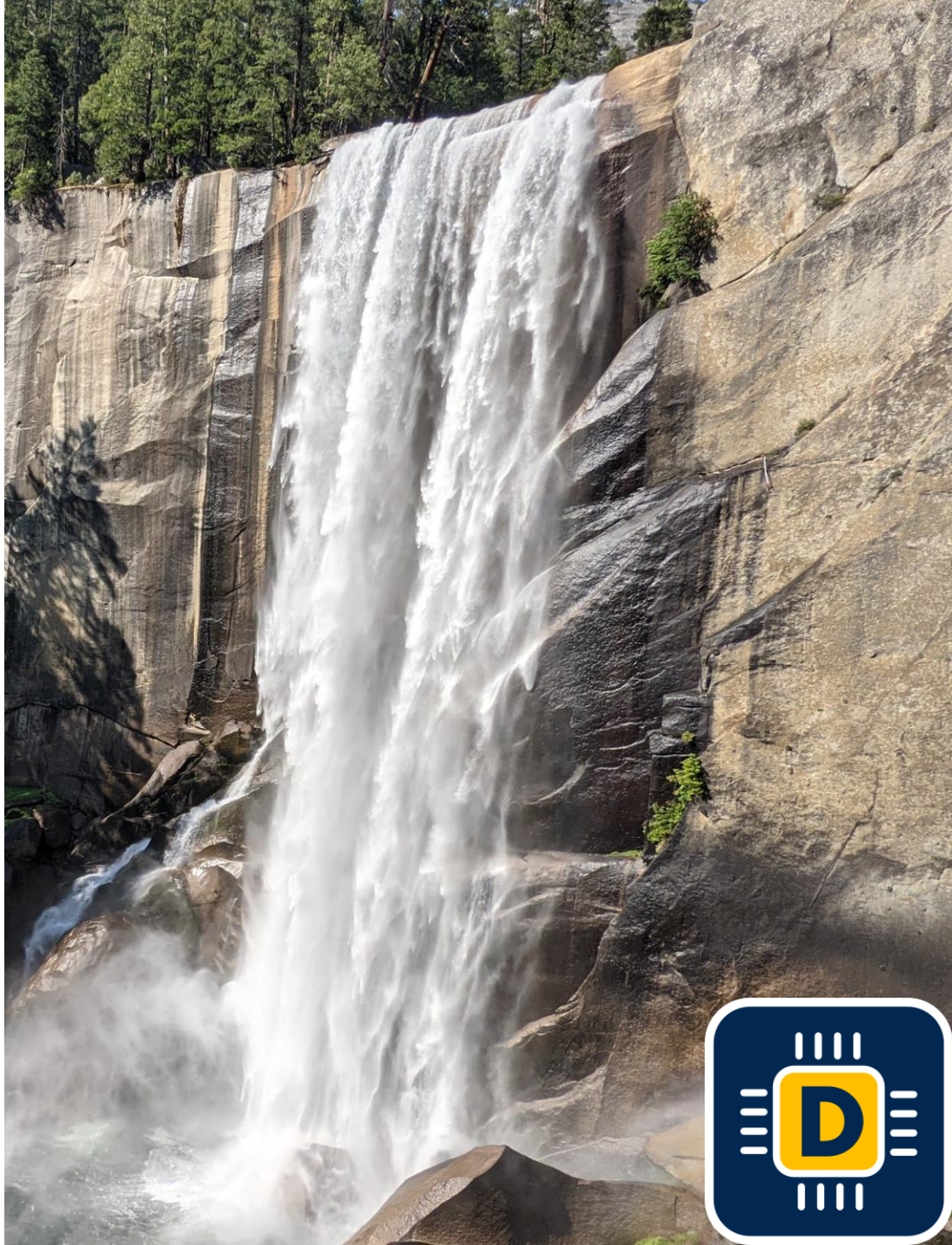
- Modeling reliability: RowHammer attacks
- What are the application-level effects?
- What are some OS-level solutions?

gem5 validation

- Is computer architecture simulation valid?
- Can we create “absolutely” and “relatively” accurate models?

Superconducting circuits

- Simulating 100+ GHz
- Effects of 4K->77K->300K



Future directions

Goal: Model *full-**system** full-**scale*** workloads on supercomputer

How do we scale to a supercomputer? (In reasonable time)

- ~200K inst/sec modeling, 3 hours is ~10,000 seconds
- 10^9 or 1 billion instructions
- Exaflop machines run 10^{18} instructions per second!
- 32,000 years to simulate 1 second

How do we get 9+ orders of magnitude speedup?

9 orders of magnitude: **Mixed fidelity**

10^3: Simulate a small fraction of the time

Use either targeted and/or statistical sampling

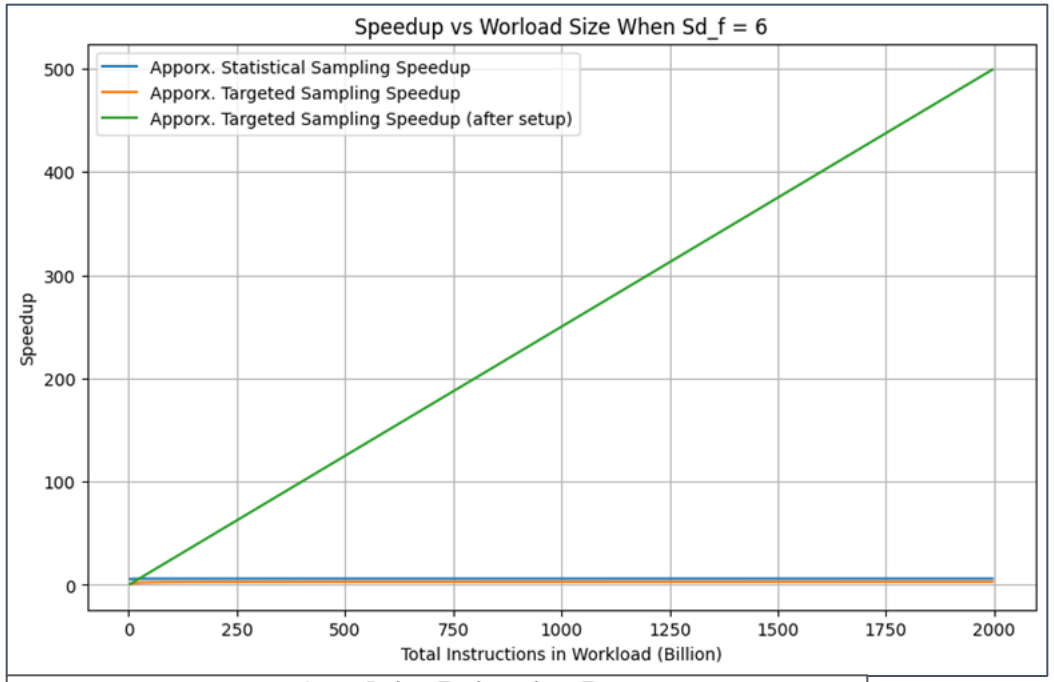
Sampling

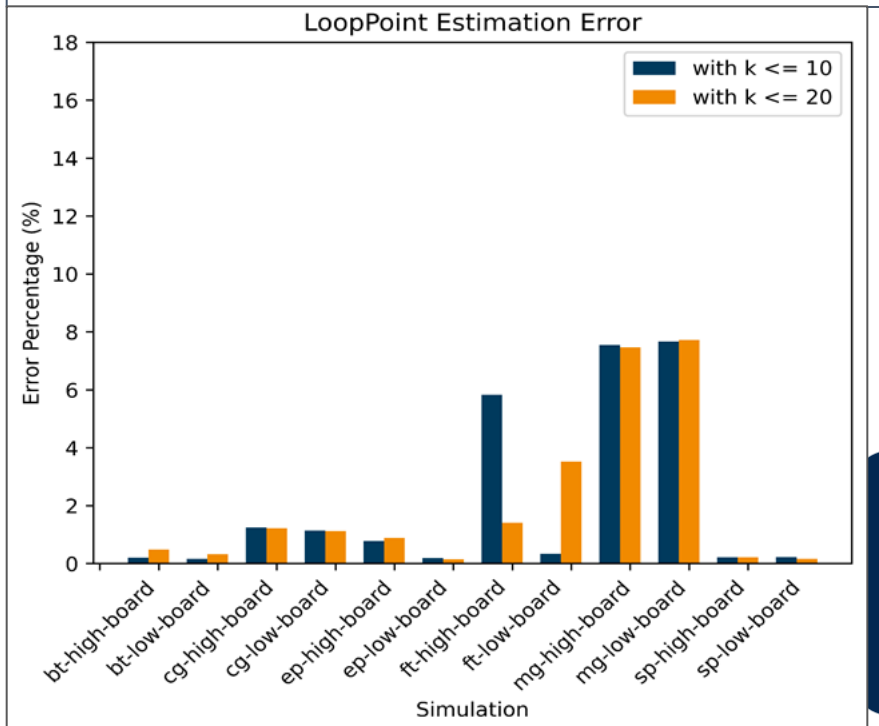
As workloads grow in size, more similarity in regions

Can have high accuracy compared to simulating the whole application

Next step: enable real time analysis

Built-in support in gem5 resources

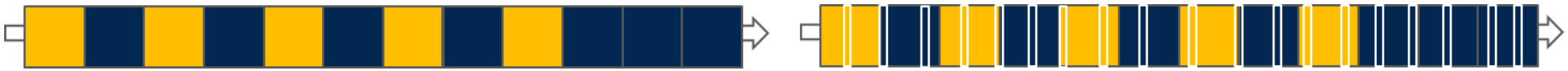




9 orders of magnitude: **Mixed fidelity**

10^3: Simulate a small fraction of the time

Use either targeted and/or statistical sampling



10^3: Simulate a small fraction in space

Only have a small fraction of nodes running at high fidelity

If gem5 is 100,000x slower, then you can **emulate** 100,000 nodes in same time it takes to **simulate**

10^3: Parallelize on a supercomputer

Use SST to run many gem5 nodes

Simulate a supercomputer on a supercomputer

The gem5 project

How has the project management evolved?

Why do I think it's been so successful?

What can we do to continue success?

A brief history

Multifacet GEMS
General Execution-driven Multiprocessor Simulator

2000

2011

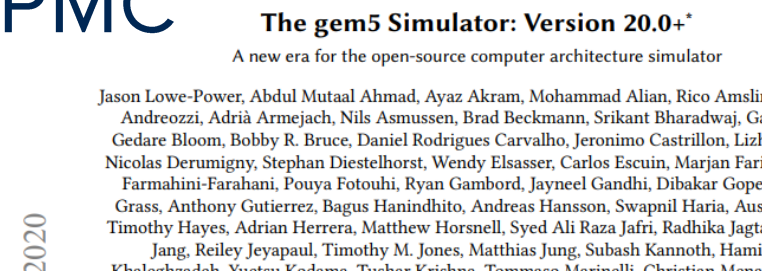
Jason Power
UW-Madison

2015

2017

Governance and PMC

2020



The gem5 Simulator: Version 20.0+*

A new era for the open-source computer architecture simulator

Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amsli
Andreozzi, Adrià Armejach, Nils Asmussen, Brad Beckmann, Srikant Bharadwaj, G
Gedare Bloom, Bobby R. Bruce, Daniel Rodrigues Carvalho, Jeronimo Castrillon, Liz
Nicolas Derumigny, Stephan Diestelhorst, Wendy Elsasser, Carlos Escuin, Marjan Fa
Farmahini-Farahani, Pouya Fotouhi, Ryan Gambord, Jayneel Gandhi, Dibakar Gope
Grass, Anthony Gutierrez, Bagus Hanindhito, Andreas Hansson, Swapnil Haria, Aus
Timothy Hayes, Adrian Herrera, Matthew Horsnell, Syed Ali Raza Jafri, Radhika Jagt
Jang, Reiley Jeyapaul, Timothy M. Jones, Matthias Jung, Subash Kannoth, Hami

2020

The first 15 years, no formal funding

Past 6 or 7 years, **huge thanks**

CCRI
(now CIRC)

CSSI

gem5's open-source ethos

Most of the things in this talk aren't that new

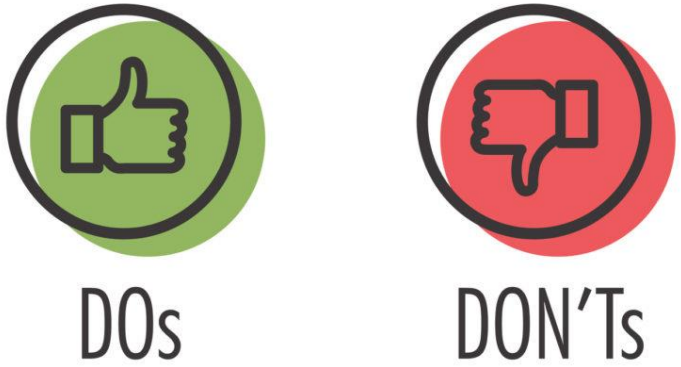
What has made gem5 a ***success*** is **O**pen **S**ource

Next few slides are an excerpt

from a NOPE talk https://arch.cs.ucdavis.edu/assets/papers/nope-bad-research-tools.pdf

How **NOT** to develop a **bad** tool

- **Do** share your tool: **Let others use and develop**
  - **Do** share your tool as widely and easily as possible!
  - **Do** make your tool open source
  - **Market** your tool anywhere and everywhere
    Websites, tutorials, books, videos, etc.
- **Do** follow software best practices: **Make it easy for others to use your tool**
  - **Do** use git, good design practices, …
  - **Do** use agile development practices, code review, …
  - **Do** use the most popular tools for your tool
- **Do** support the tool: **Help others use your tool**
  - **Do** provide documentation and support
  - **Do** continue development after initial release



Who is the gem5 community?

Do volunteers make up most of the contributors?

No!

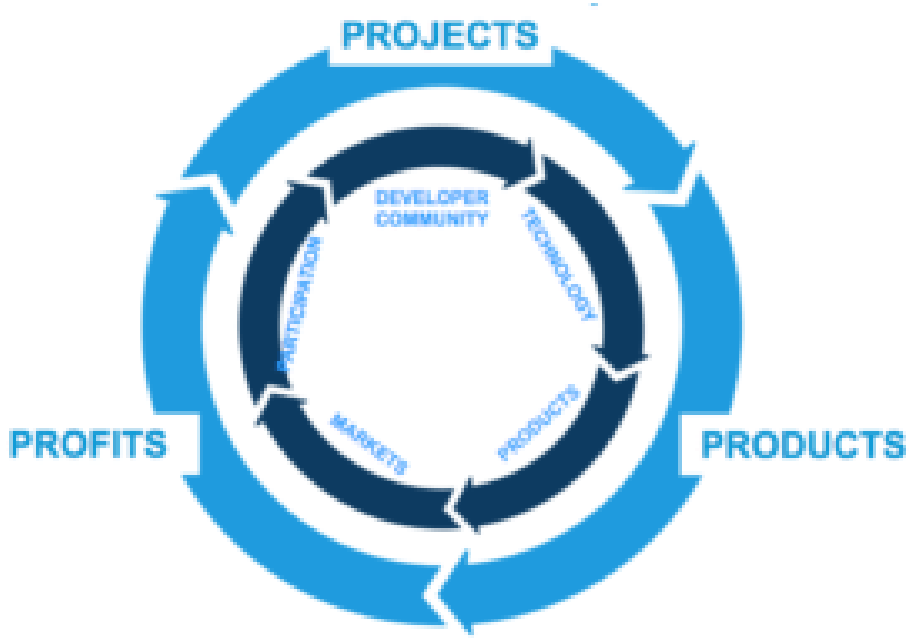
Mostly people paid to contribute

Industry contributors

Academic contributors (mostly grad students)

Industry contributors

(Highly) paid to *complete employer's tasks*



Why contributing to open source?

- For fun or internally driven
- For business reasons

Limited time/energy for volunteering

- Only do what they enjoy

Business is profit-driven, not community or long-term driven

Researchers

(Poorly) paid graduate students

Professional researchers, professors, undergrads, etc.

Why contributing to open source?

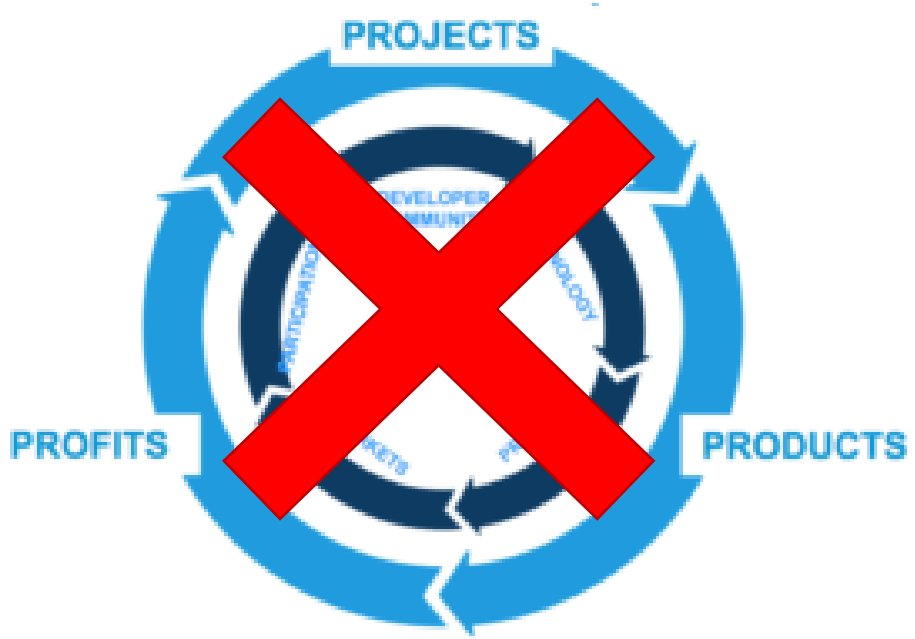
- For fun or internally driven
- Do better research, write better papers
- *To have impact*

Limited time/energy for volunteering

- Only do what they enjoy

No virtuous cycle

No (good) mechanisms for crediting the impact



But how to create good tools?

(Research) Incentive structure pushes us towards bad tools

- Bean counters, not fertile soil counters
- Need more recognition:
  - Infrastructure papers?
  - Awards? Artifact badges are a great start!
  - Count commits? Code reviews? Stackoverflow posts?

Funding is for *research* not infrastructure

- 3-year grant: papers published in years 2&3
- The experts (students) graduate or end their internships
- Need more **research software engineers** in systems research
  - Software developers to provide continuity
  - Continuing infrastructure development funds

Next steps for gem5

Solved community
Solved usability

Verification
Known-good models
Out-of-order core
Reproducibility

Simulation speed

Adoption

Announcement!

gem5 bootcamp

July 29th to Aug 2nd: Five days, in-person at UC Davis

Focus is new gem5 users (industry and academia)

Will have travel grants for 1st and 2nd year students

Conclusions

gem5 has been successful

- Many cool use cases
- Many great new things happening

The future is bright

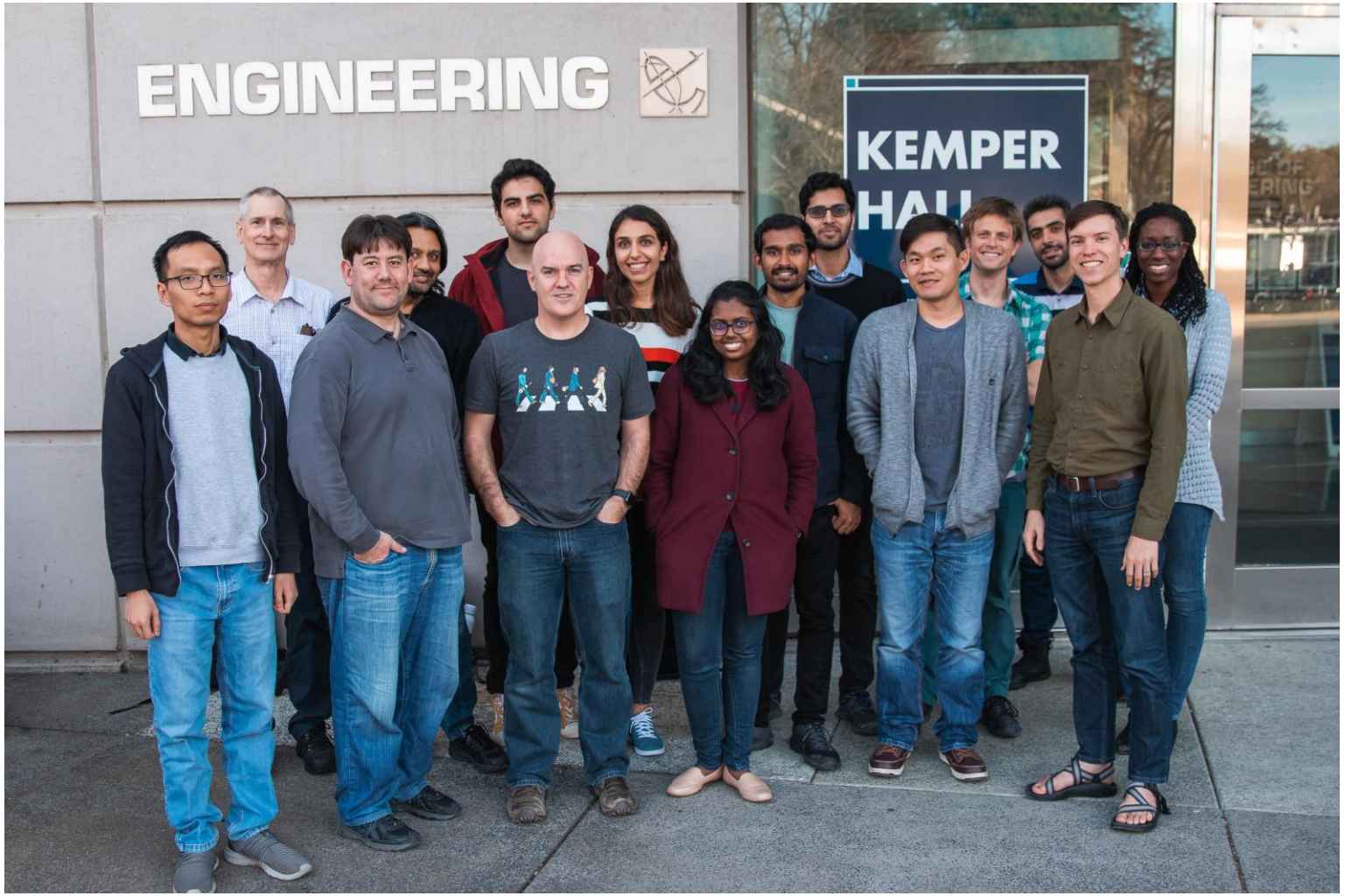
- Demonstrations of gem5 for supercomputers coming soon

This success has been despite our current research system

- We need improved incentive structures

Thanks!

DArchR
DAVIS ARCHITECTURE RESEARCH



jlowepower@ucdavis.edu

https://arch.cs.ucdavis.edu/