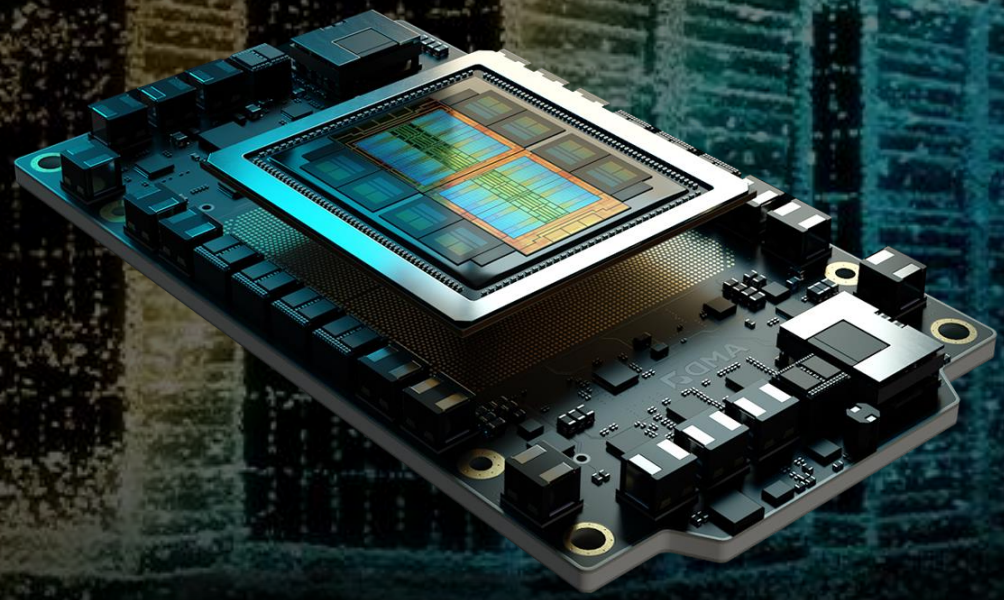
Lessons from a Quarter-Century of Computer Architecture Simulation



Brad Beckmann – AMD Fellow

ASTRA-Sim Workshop – ISCA 2026

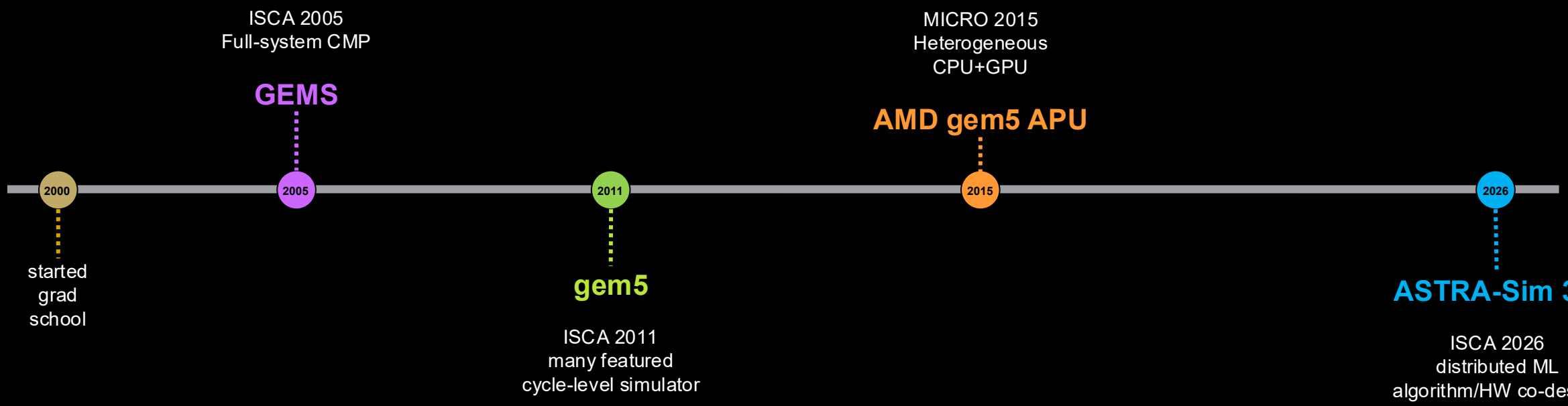
Raleigh, NC – June 27, 2026

AMD

together we advance_

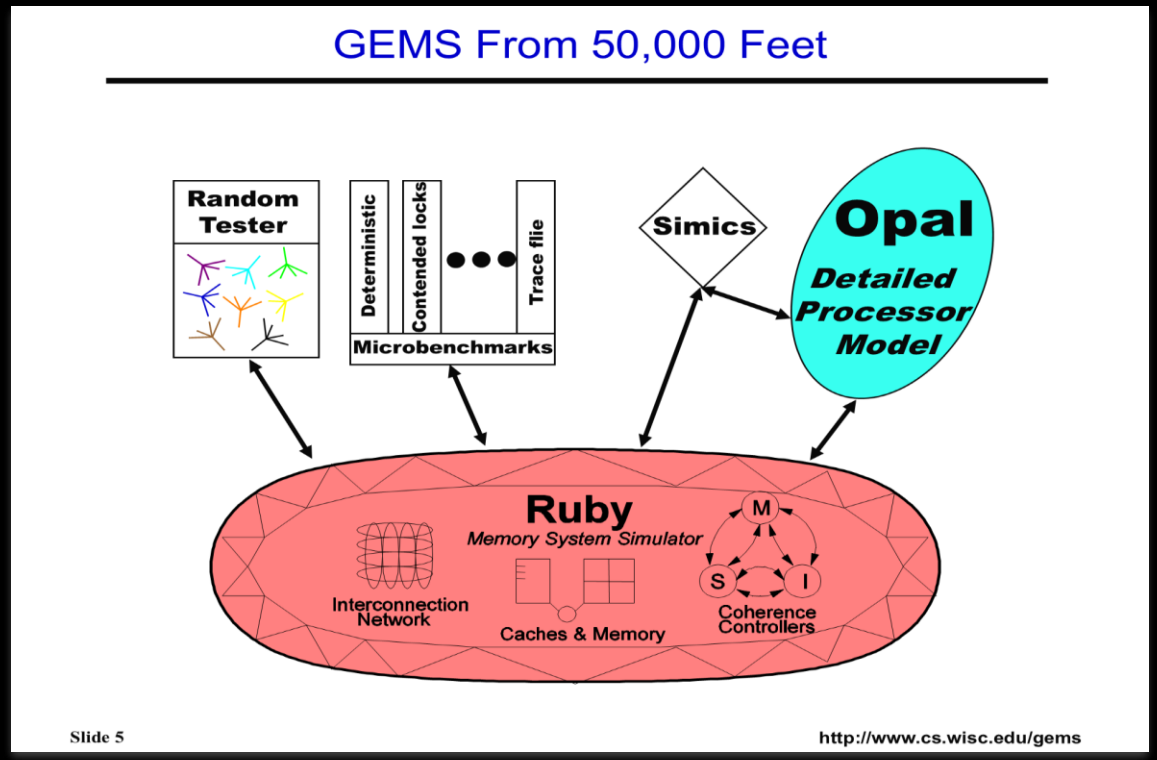
My Journey

- Started 2000 as a young graduate student at Wisconsin
- Participated in three prior tutorials that advanced open-source simulation
- Applying lessons learned to ASTRA-Sim and preparing for the age of agentic AI development



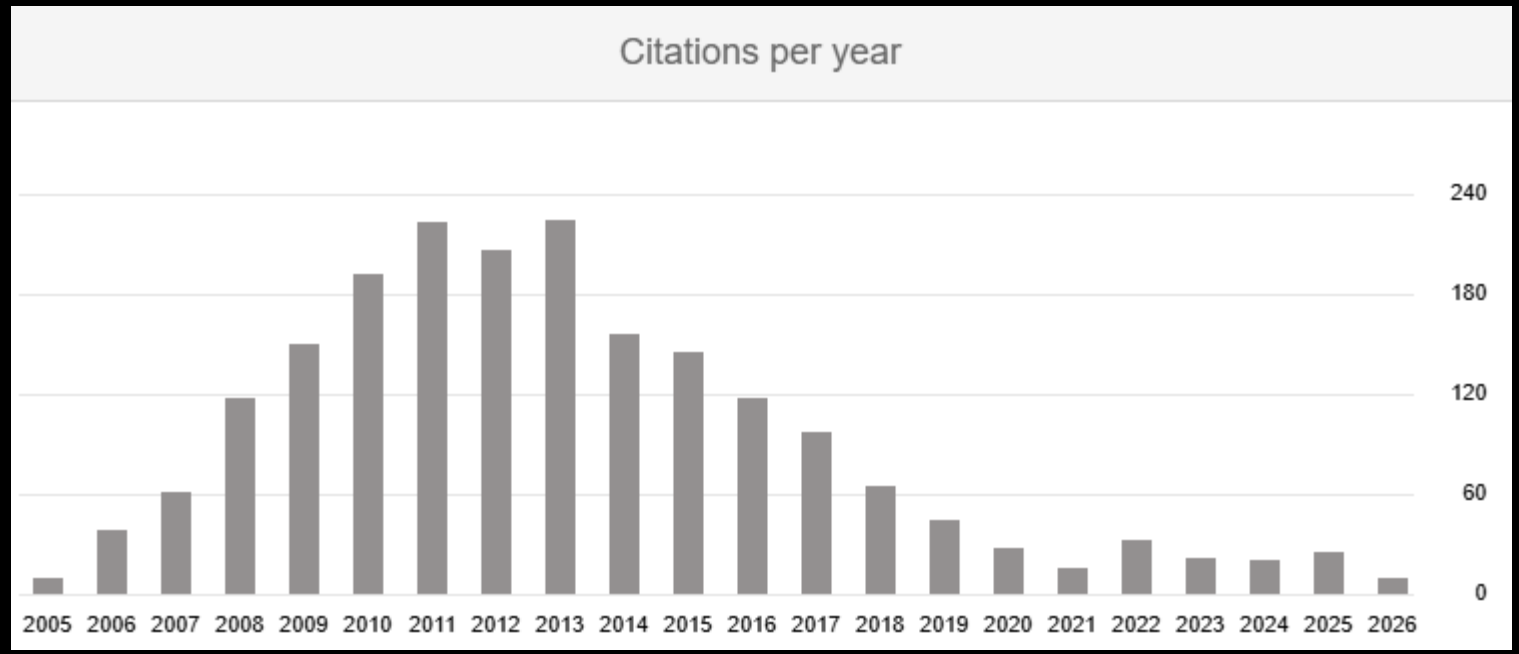
GEMS @ ISCA 2005 (Madison, WI)

- Multifacet's General Execution Model Simulator (GEMS)
- Full-system SMP/CMP simulation on Virtutech Simics
- Timing-first Opal OOO core + Ruby memory system (SLICC)
- Enabled detailed study of commercial multithreaded workloads
- Presenters:
  - Brad Beckmann
  - Mike Marty
  - Luke Yen
  - Alaa Alameldeen
  - Kevin Moore



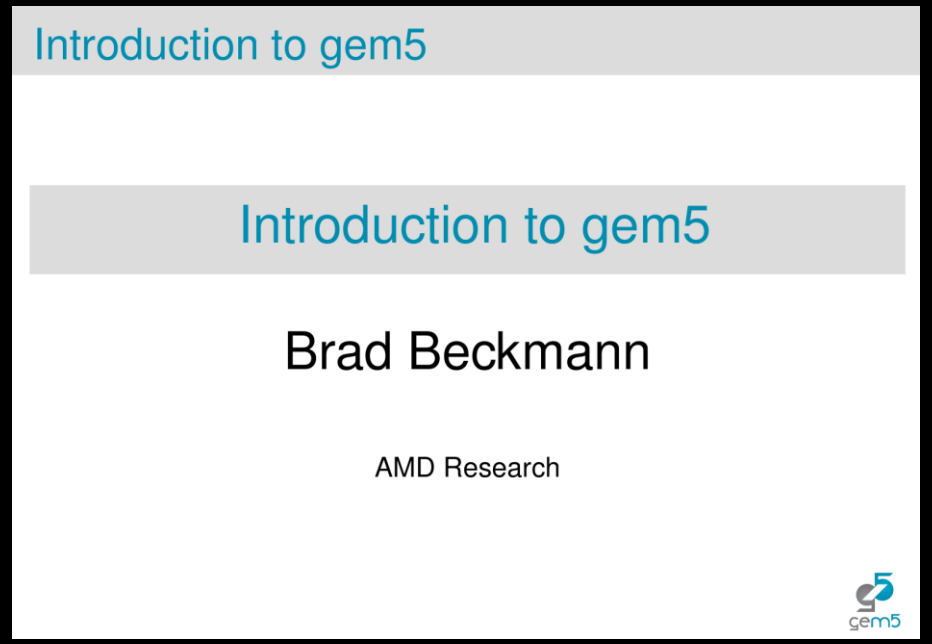
GEMS: Successful, But Short-Lived

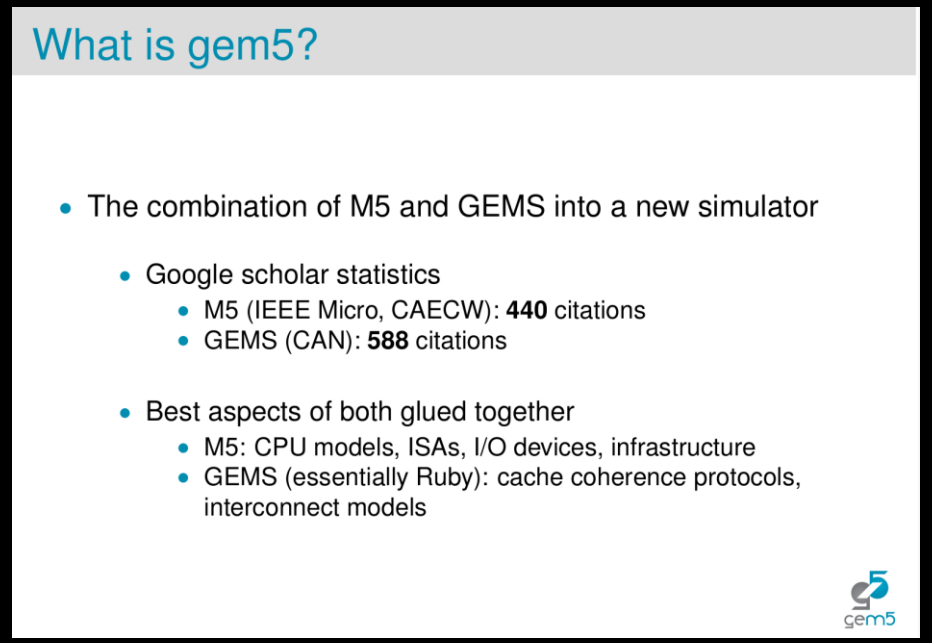
- Loved for Opal / Ruby / SLICC — but built entirely on proprietary Simics
  - Intel acquired Virtutech (2007) → public Simics use restricted → GEMS doomed
  - Lesson: Longevity needs open foundations



gem5 @ ISCA 2011 — The Beginning

- Community-driven successor to GEMS and M5
  - Fully open source
  - No external dependencies
- Unified framework: CPU, memory, I/O
  - Full-system, runs unmodified OS
- Released before the 2011 tutorial
  - Adoption exploded!
- Now the most cited computer architecture paper from the last 40 years — by far!!!
  - 6000+ citations
- A powerful platform for academia and industry worldwide

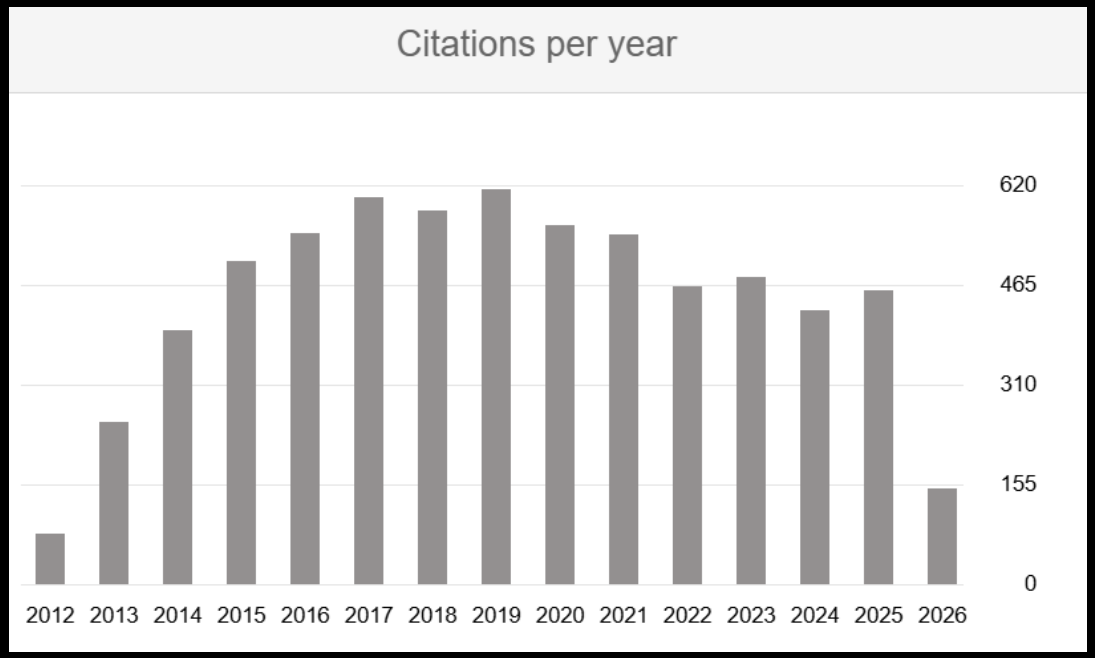




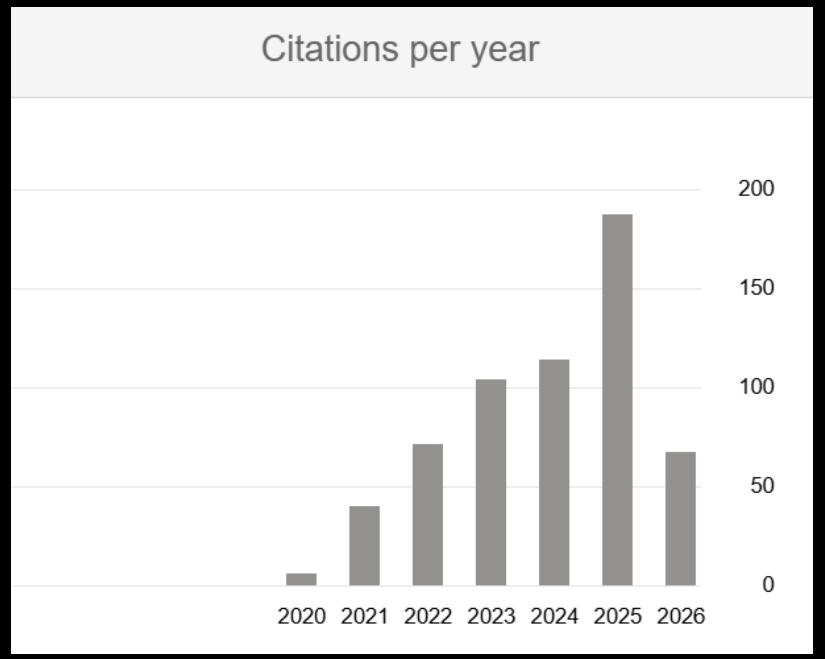
gem5: Unprecedented Impact

- Citation growth unlike any prior simulator
- Jason L-P and gem5 v20.0 sustained the momentum
- ASTRA-Sim can only hope for similar impact!

Binkert et al. The gem5 Simulator. August 2011.



Lowe-Power et al. The gem5 Simulator: Version 20.0+. July 2020.



gem5: Powerful — But Does Not Play Well with Others

- Stats system and configuration: Extremely powerful
- Did not integrate well with other tools and workflows
- At AMD: Internal tools need to own the configuration process
- gem5's interface with AMD workflows was brittle
- Still used at AMD today — but separate from other flows
- Lesson: Integration flexibility matters as much as features

"Nasty cat!" by Hannibal Poenaru, CC BY-SA 2.0, via Wikimedia Commons

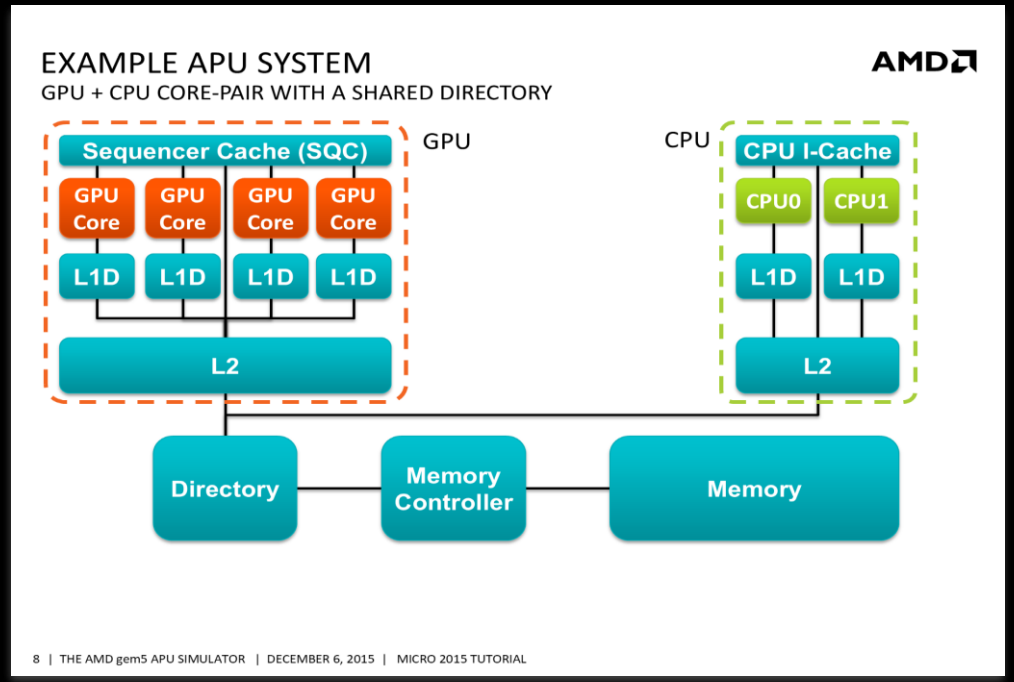
AMD gem5 APU @ MICRO 2015 — Waikiki, Hawaii

Cumulus Clouds, CC BY-SA 3.0, via Wikimedia Commons

- December in Hawaii — what a great excuse to attend a conference!
  - June in North Carolina…not so much ☺
- AMD Research extended gem5 with a full APU timing model
- Showcased gem5's strong software engineering infrastructure
  - GPU executed HSA Intermediate Language (HSAIL), later the real machine ISA
  - A different accelerator + separate ISA — integrated seamlessly

APU Model: Seamless Heterogeneous Integration

- The gem5 infrastructure absorbed a completely different accelerator
  - Separate GPU ISA (HSAIL) integrated without rewriting the framework
  - Strong software engineering was the enabler
- Integrating tools into broader workflows enables unique research
  - Enabled heterogeneous CPU-GPU coherence evaluation
    - E.g., Lowe-Power et al., MICRO 2013
  - Lesson: infrastructure + integration = unique insights



ASTRA-Sim Applying Lessons Learned

- Solid, consistent software engineering for long-term success
- Open foundations — never depend on close software
- Must work well with others — does not need to own the event queue, configuration, and statistic collection
- Integration flexibility is a first-class design goal
- Shared-library design with pluggable network backends
- Chakra traces + plug-and-play compute / network / memory

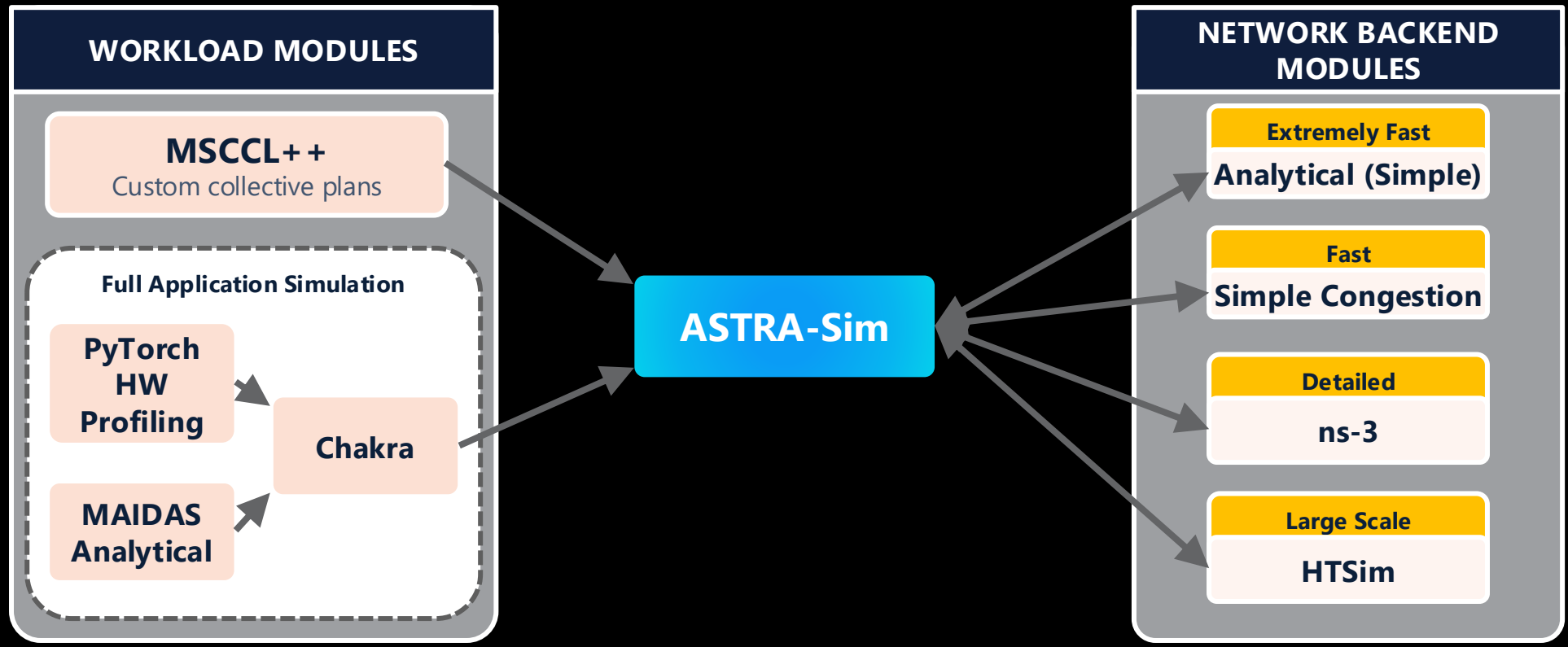
A Lesson from Sparky Anderson

- Sparky Anderson — legendary manager of the Cincinnati Reds
- Took a clubhouse of diverse personalities and big egos to win
- He never said 'Listen to me, I'll get you there'
- Instead: 'I'm here to win and I want you to help me'
- Uniquely crafted how he managed every single player
- Treated all with respect — but dealt with each one differently
- Greatness comes from adapting to others, not commanding them

Courtesy Gerald R. Ford Presidential Library

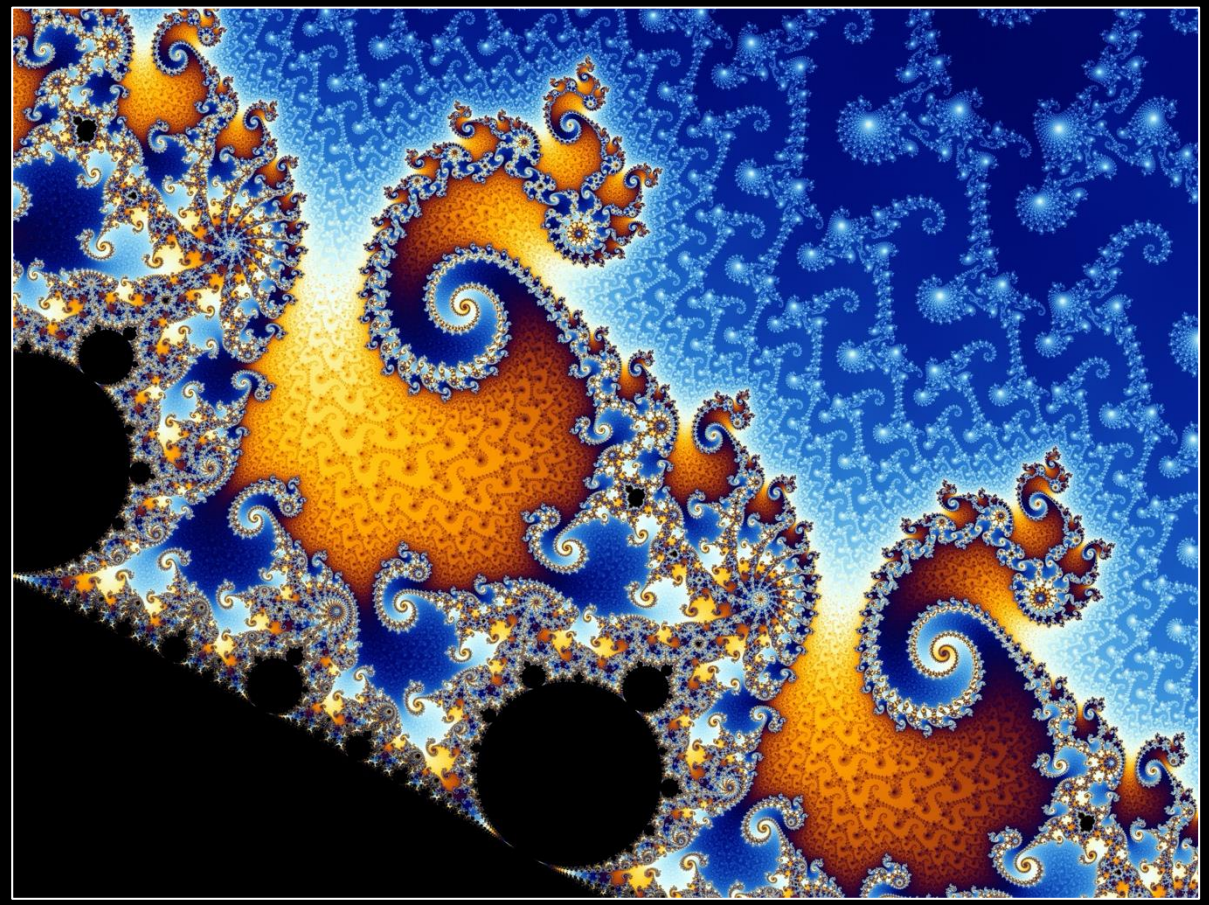
ASTRA-Sim Manages Its Roster Like Sparky

- 'I'm here to win — help me': a shared library that adapts to each backend's lead — their own event queues and exotic configs — with a lightweight config of its own



Agentic Software Development: The Wave of the Future

- **Today**: AI agents write, refactor, and extend the code
- Iterative trial-and-test loops replacing purely hand-crafted dev
- ASTRA-Sim's modular, library-based design is agent-friendly
- Autonomous agents are only as good as the feedback loop they receive
- The bottleneck shifts from writing code to verifying correctness



Wolfgang Beyer, created with Ultra Fractal 3, CC BY-SA 3.0, via Wikimedia Commons

Testing Provides the Guardrails

- Agentic flows require strong, automated tests — no exceptions
- Without tests, agents iterate blindly and fail silently
- GEMS, gem5, and ASTRA-Sim all invested heavily in testing
- Speed, coverage, extensibility — hard problems worth solving
- Academia must invest in testing — not dismiss it as 'industry work'
- Graduate students: Solving testing problems is real research

Technical Challenges ASTRA-Sim Must Address

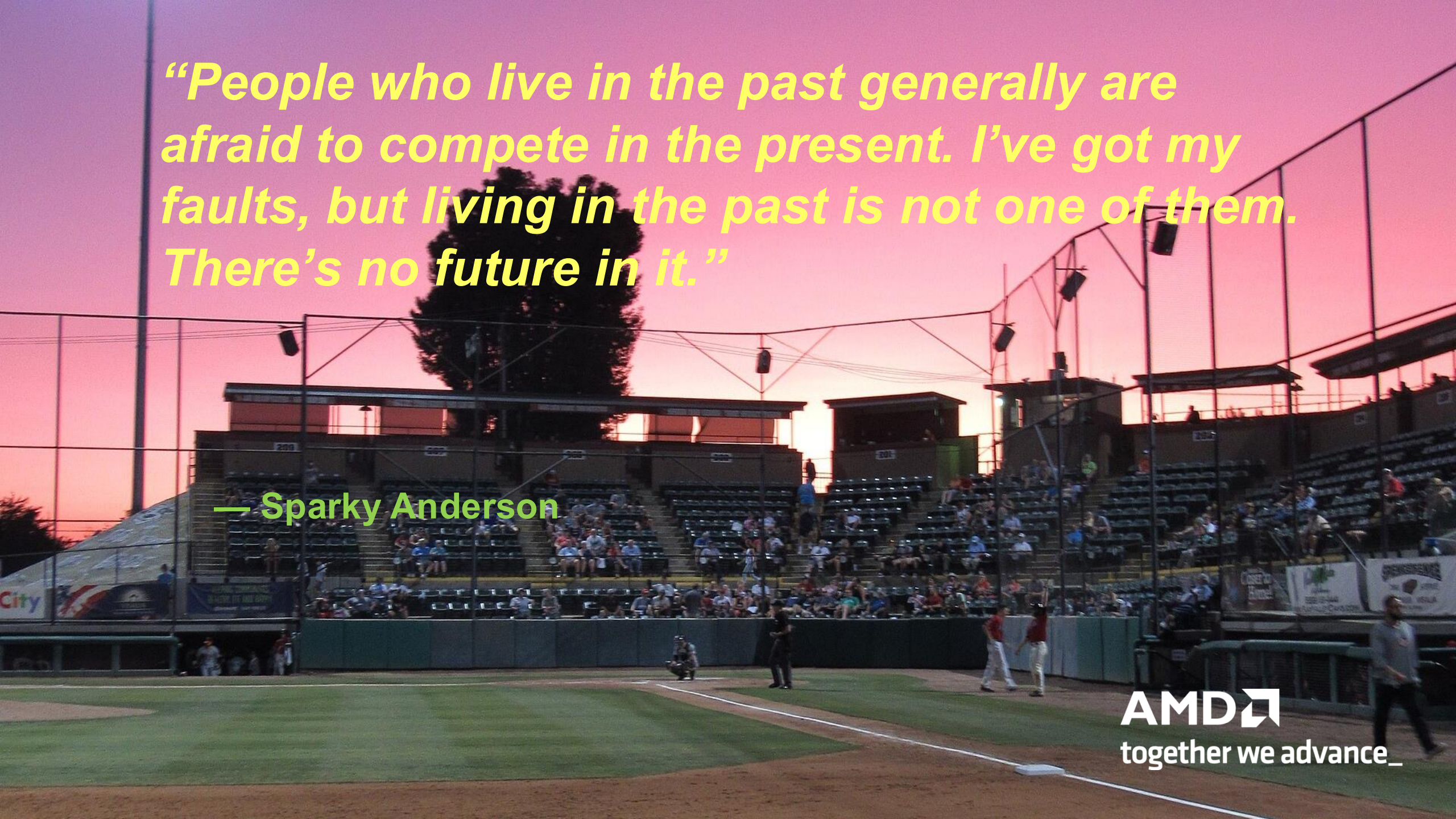
- GPU load-store collectives are not optimal
  - X thread blocks with Y warps assigned to move data within single device or across scale-up domain
  - Unrolling load-store loop bloats register budget for collective kernels
  - Access patterns change due to data fabric differences between architectures and network topologies

- Collective kernels concurrently run with GEMM kernels considered harmful
  - Memory bandwidth competition between concurrent kernels
  - Performance brittleness caused by GEMM quantization
  - Fusing GEMM with communication into a single kernel is difficult as well
  - Power budgeting actively fights GEMMs MAFs and lowers clock for memory transfers

- DMA engines are preferred data movers
  - Tensor Memory Acceleration within the CU
  - SDMA / Copy Engines
  - RDMA

Challenge to the Community

- Embrace significant testing flows for ASTRA-Sim and beyond
- Build regression suites that agents (and humans) can trust
- Create an open testing infrastructure alongside open simulators
- Let's build a simulator that last — together!


“People who live in the past generally are afraid to compete in the present. I’ve got my faults, but living in the past is not one of them. There’s no future in it.”
— Sparky Anderson
AMD
together we advance_

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION


AMD