Modeling Modern GPU Applications in gem5

Matthew D. Sinclair

University of Wisconsin-Madison, AMD Research

sinclair@cs.wisc.edu

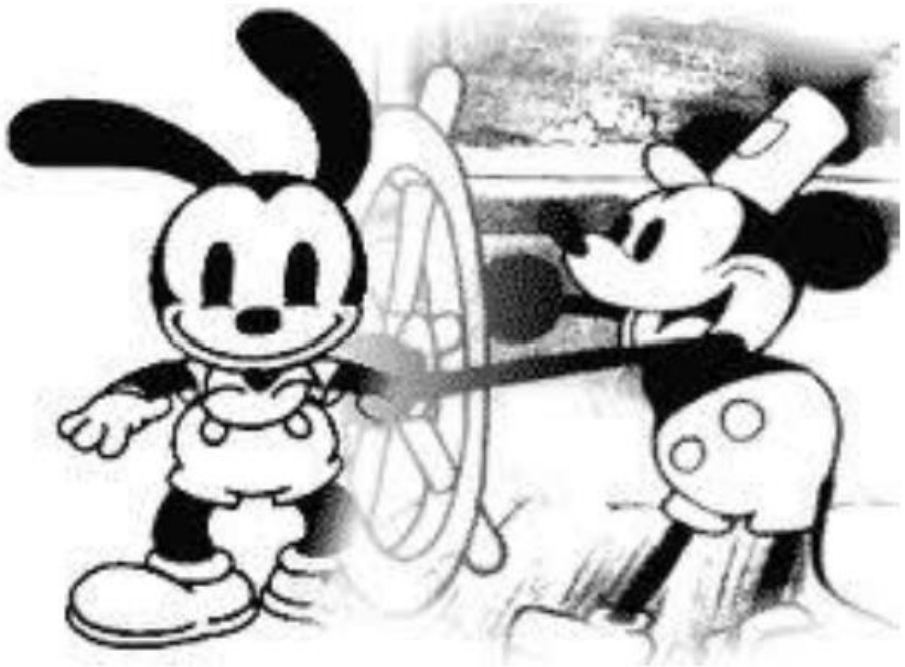
Collaborators: John Alsop (AMDR), Brad Beckmann (AMDR), Bobby Bruce (UCD), Alexandru Dutu (AMDR), Anthony Gutierrez (AMDR), Michael LeBeane (AMDR/Qualcomm), Jason Lowe-Power (UCD), Matthew Poremba (AMDR), Mike Swift (UW), and others

Students: Anuska Chandrashekar, Preyesh Dalmia, Gaurav Jain, Charles Jamieson, Suchita Pati, **Kyle Roarty**, Bobbi Yogatama, and others

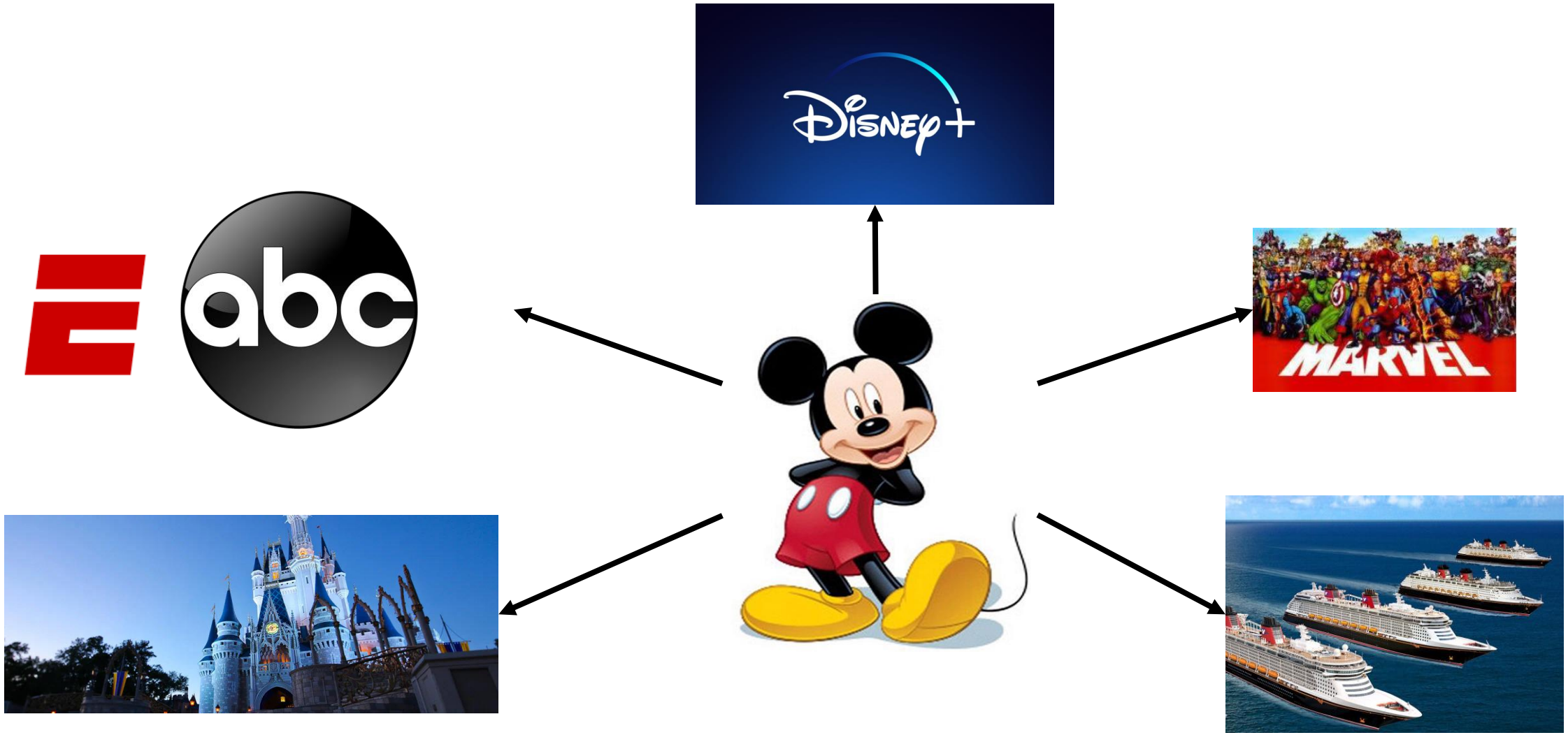
Let’s begin by thinking about a mouse

Walt Disney Co. in the beginning:

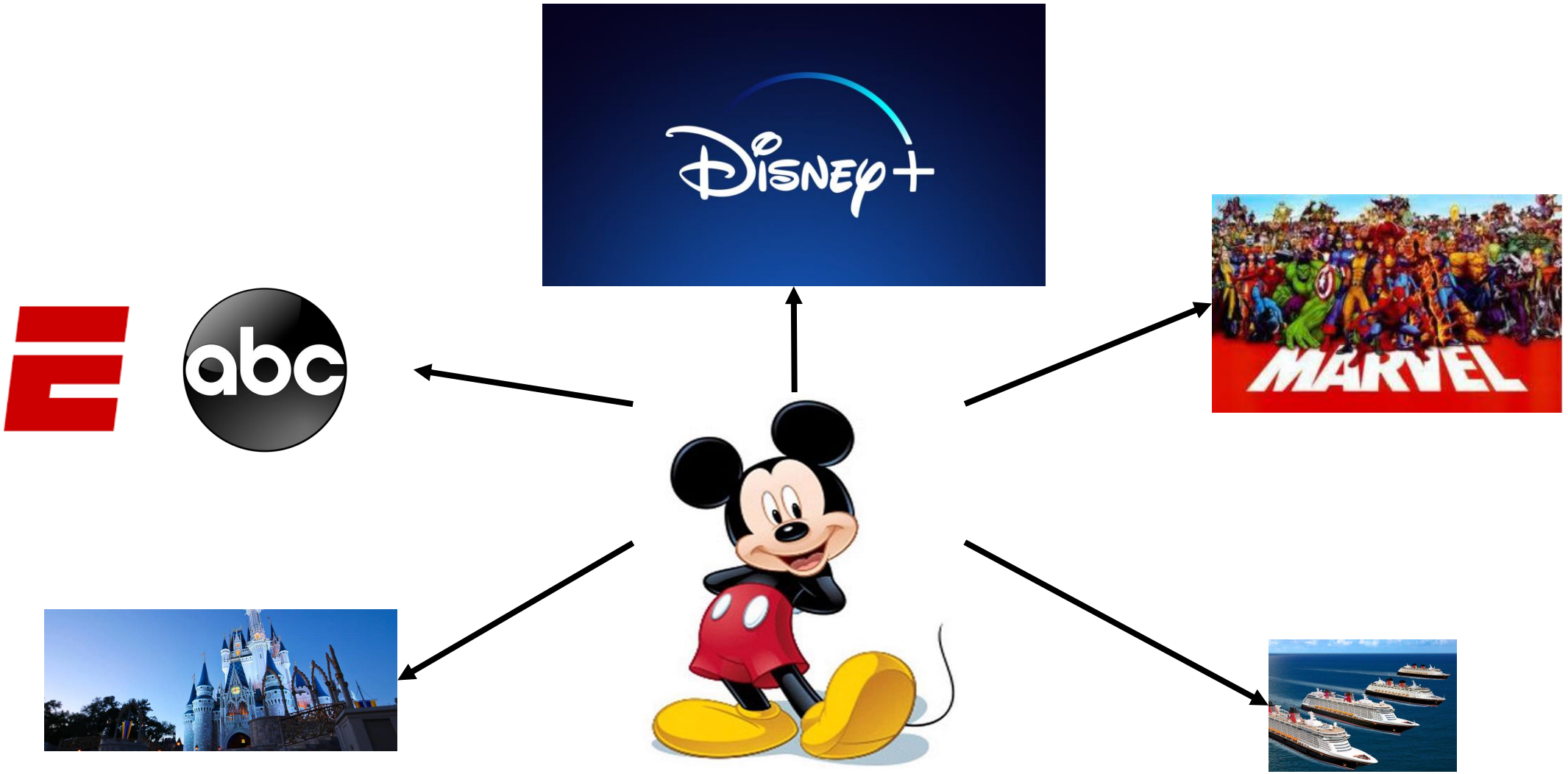
- Decided to be an animator
- Initial success in 1920's and 1930's
- Forced to expand into other areas



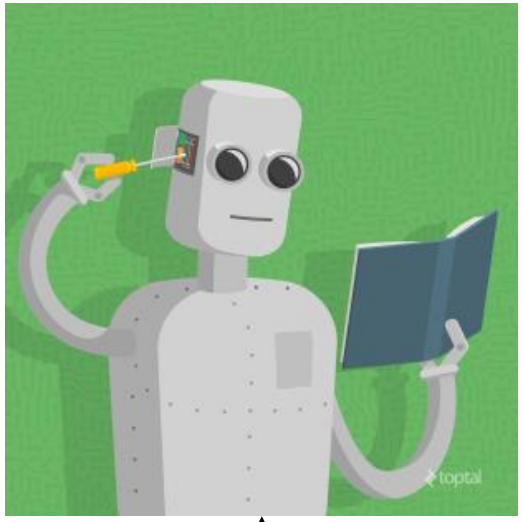
Walt Disney Co. as we know it today:



Walt Disney Co. as we know it today:

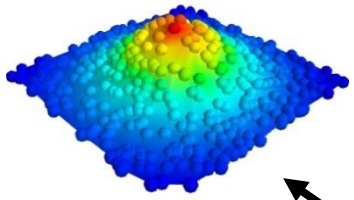


(GP)GPUs Also Need to Evolve



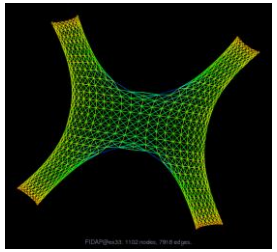
Machine Learning

Irregular Scientific Computing



Raytracing

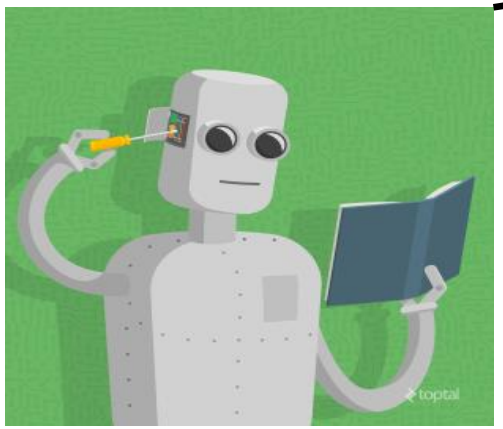
Graph Analytics



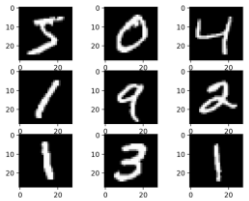
Personal Assistants

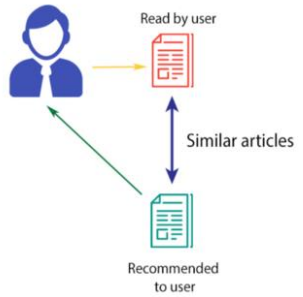
These Applications Drive Systems Reqs.

But they have different characteristics:



Machine Learning





...

Often reuse data, some utilize fine-grained synchronization

Widely varying utilization

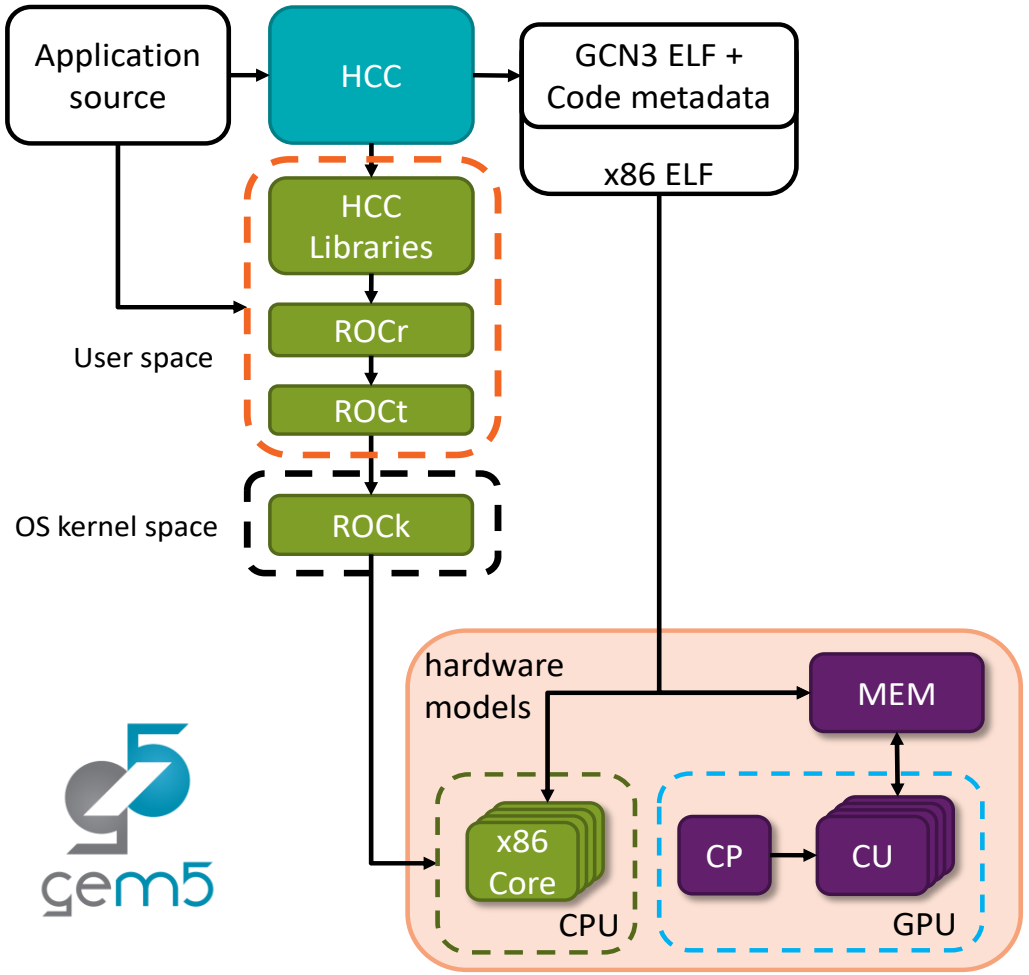
Some have tight real-time deadlines

Need high fidelity tools to model and prototype optimizations

Outline

- Motivation
- **Background**
- Improvements
- Conclusion & Future Work

Prior CPU-GPU Support in gem5



[Gutierrez et al., HPCA '18]

Execution-driven, cycle-level

- Accurately models complex CPUs & GPUs
- Rapid prototyping of new features
- Validate simulation with execute-in-execute

Prior work [Gutierrez et al. HPCA '18]

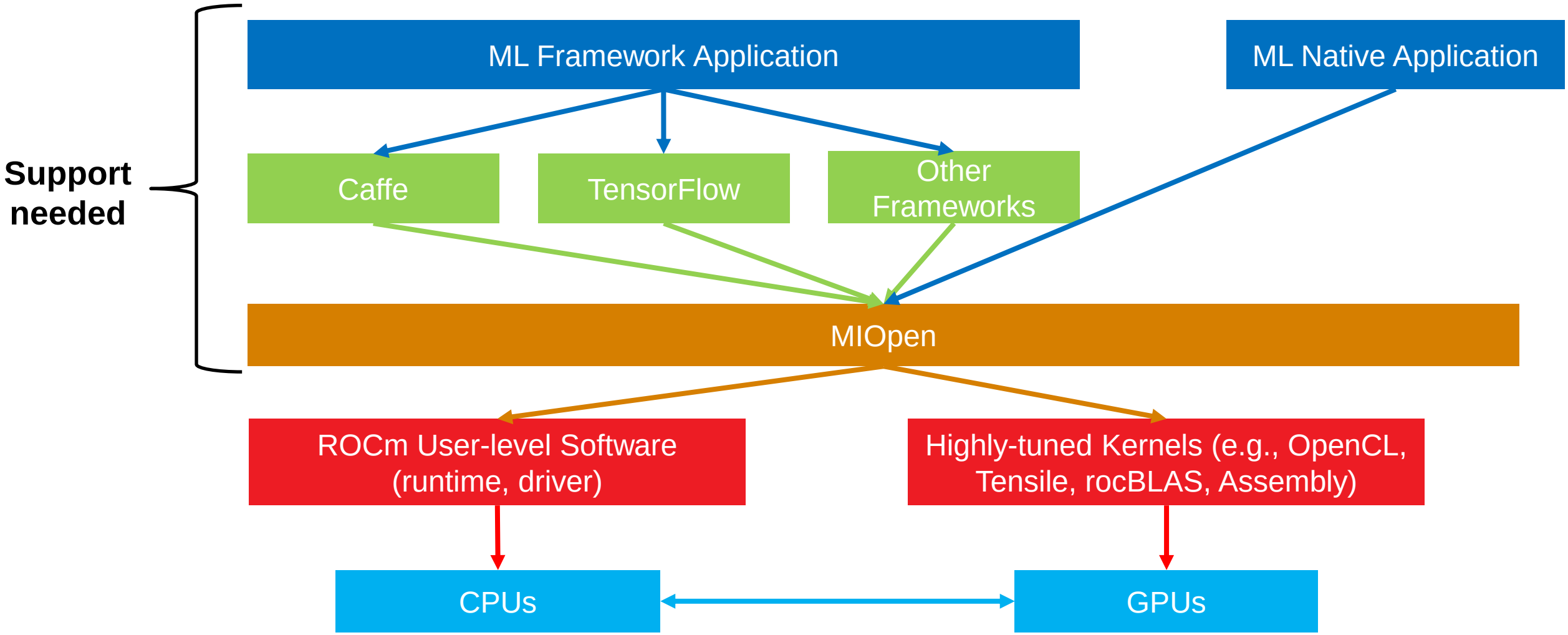
- Runs unmodified ROCm 1.6 user stack
- Simulates HIP and HCC applications

Solid foundation, but does not support ML workloads

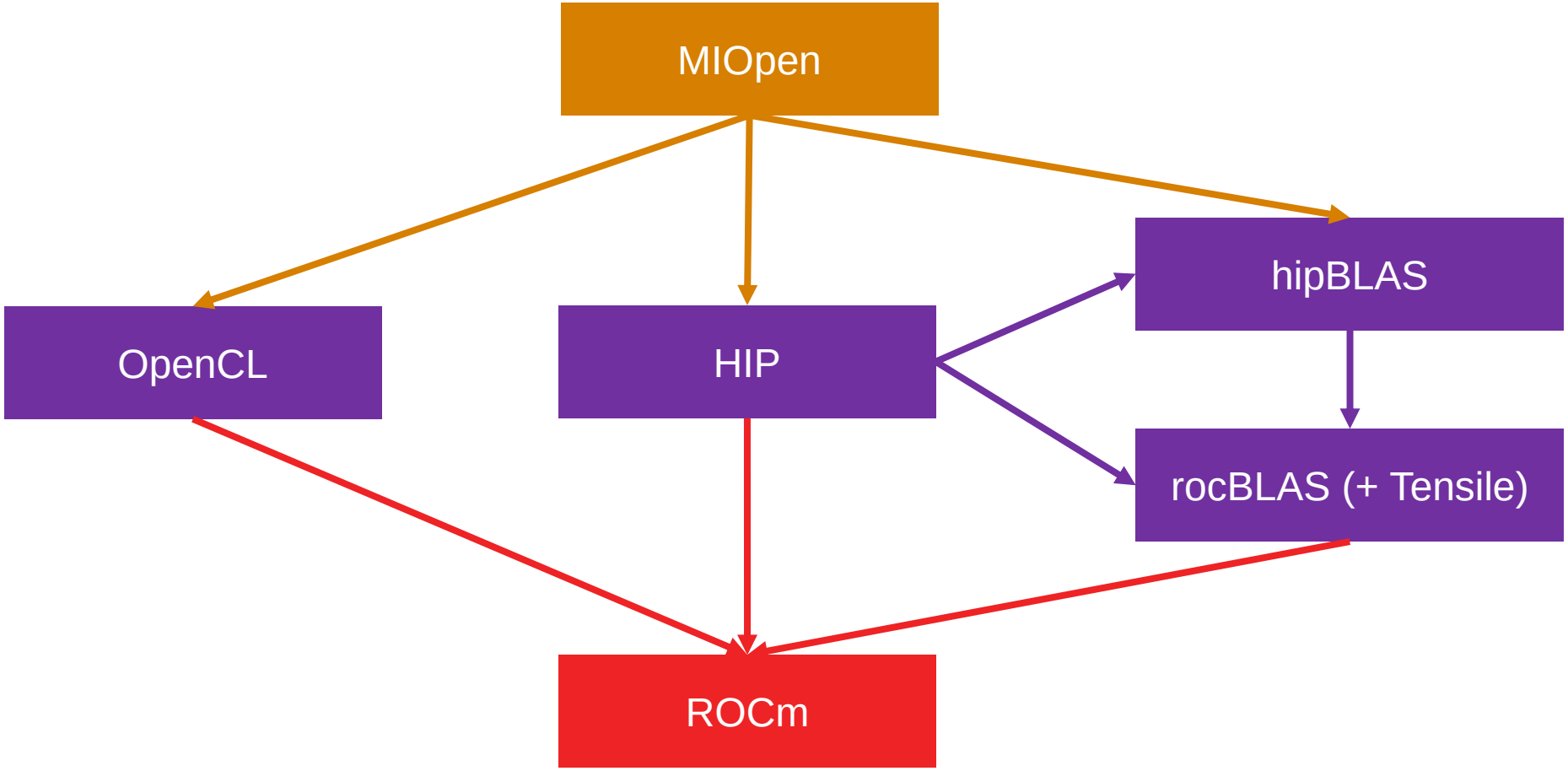
Outline

- Motivation
- Background
- **Improvements**
  - **Enabling Support for ML Workloads in gem5**
  - Making gem5 GPU models easier to use
  - Improved, higher fidelity support for gem5 GPU
- Future Work & Conclusions

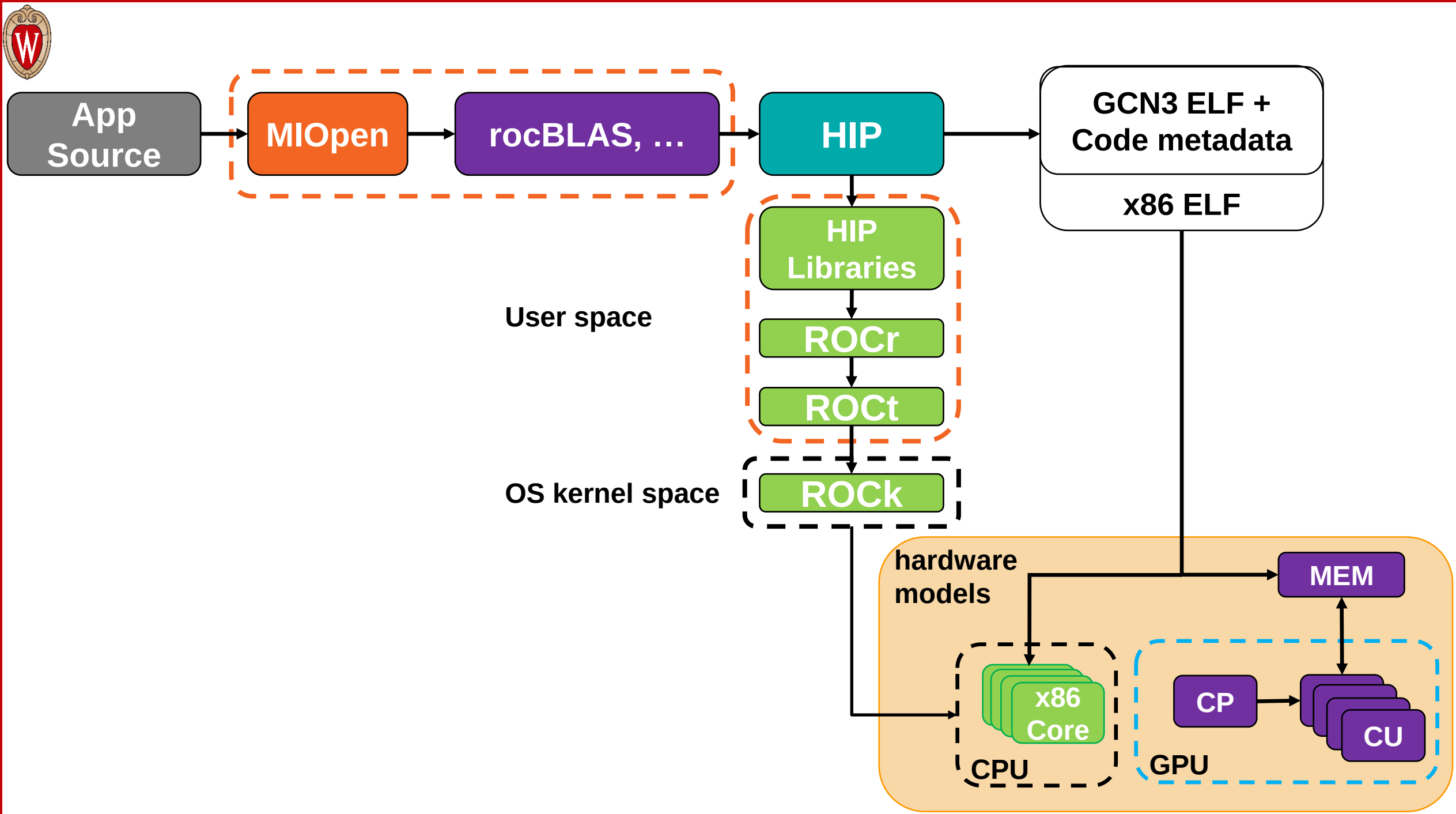
Flow of a ML Application



MIOpen Also Requires Additional Libraries



Build on prior work to add support for ML workloads [Alsop, et al. IISWC '19]


App Source
MIOpen
rocBLAS, …
HIP
GCN3 ELF + Code metadata
x86 ELF
HIP Libraries
ROCr
ROCt
ROCk
User space
OS kernel space
hardware models
MEM
x86 Core
CPU
CP
CU
GPU

Challenges Running (ML in) GPU Model

- Specific software is required
  - Initially ROCm 1.6.x & gcc 5.4 [Gutierrez et al. HPCA '18]
  - We updated to ROCm 4.0 & gcc 7 [Roarty et al. gem5 Wkshp '20]
- Barriers to entry
  - Challenging to install the libraries correctly
  - Lack of support and documentation
  - Need to support and model newer GPUs, GPU features
  - Improve accuracy of publicly available GPU models

Our work removes/reduces these barriers to entry

Outline

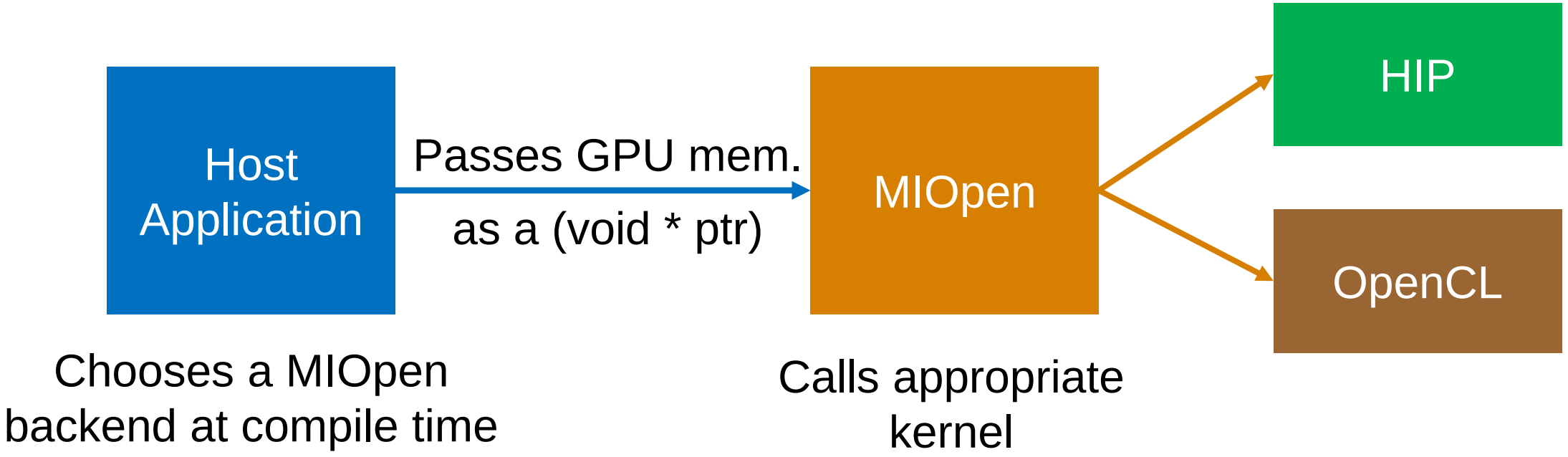
- Motivation
- Background
- **Improvements**
  - Enabling Support for ML Workloads in gem5
  - **Making gem5 GPU models easier to use**
  - Improved, higher fidelity support for gem5 GPU
- Future Work & Conclusions

Creating Portable Resources

- Docker container
  - Properly installs ROCm software stack

- **Publicly Available!**
  - Integrated into gem5 repo: https://gem5.googlesource.com/
  - Added bmks & doc. in gem5-resources *[Bruce ISPASS '20 Best Paper Nom.]*
  - Used in continuous integration to ensure GPU support is stable
  - Suggest building applications requiring ROCm with docker

MIOpen Programming Model



Running ML Applications in gem5

- Codebase and simulator aren't optimized for each other
- Example: MIOpen uses OpenCL kernels
  - Requires online compilation → slow, unsupported in gem5

- Solution:
  - Generate kernel files ahead of time for desired GPU
  - Link kernel files into Docker container
    - Avoid re-generating kernels
    - Reduces simulation time

Mostly automated process eases adoption by new users

Ex. gem5-resources README (DNNMark)

resources.gem5.org/resources/dnn-mark

DNNMark is a benchmark framework used to characterize the performance of deep neural network (DNN) primitive workloads.

The gem5 DNNMark tests can be used to test the GCN3-GPU model.

Compiling DNNMark, compiling the GCN3_X86 gem5, and running DNNMark on gem5 is dependent on the gcn-gpu docker image, built from the `util/dockerfiles/gcn-gpu/Dockerfile` on the gem5 stable branch.

Compilation and Running

To build DNNMark: **NOTE**: Due to DNNMark building a library, it's important to mount gem5-resources to the same directory within the docker container when building and running, as otherwise the benchmarks won't be able to link against the library. The example commands do this by using `-v ${PWD}:${PWD}` in the docker run commands

```
cd src/gpu/DNNMark
docker run --rm -v ${PWD}:${PWD} -w ${PWD} -u $UID:$GID gcr.io/gem5-test/gcn-gpu ./setup.sh HIP
docker run --rm -v ${PWD}:${PWD} -w ${PWD}/build -u $UID:$GID gcr.io/gem5-test/gcn-gpu make
```

DNNMark uses MIOpen kernels, which are unable to be compiled on-the-fly in gem5. We have provided a python script to generate these kernels for a subset of the benchmarks for a gfx801 GPU with 4 CUs by default

To generate the MIOpen kernels:

```
cd src/gpu/DNNMark
docker run --rm -v ${PWD}:${PWD} -v${PWD}/cachefiles:/root/.cache/miopen/2.9.0 -w ${PWD} gcr.io/gem5-test/gcn-gpu python3 generate_cachefiles.py cachefiles.csv [--gfx-version={gfx801,gfx803
```

Due to the large amounts of memory that need to be set up for DNNMark, we have added in the ability to MMAP a file to reduce setup time, as well as added a program that can generate a 2GB file of floats.

To make the MMAP file:

```
cd src/gpu/DNNMark
g++ -std=c++0x generate_rand_data.cpp -o generate_rand_data
./generate_rand_data
```

DNNMark is a GPU application, which requires that gem5 is built with the GCN3_X86 architecture. To build GCN3_X86:

```
# Working directory is your gem5 directory
docker run --rm -v ${PWD}:${PWD} -w ${PWD} -u $UID:$GID gcr.io/gem5-test/gcn-gpu scons -sQ -j$(nproc) build/GCN3_X86/gem5.opt
```

To run one of the benchmarks (fwd softmax) in gem5:

```
# Assuming gem5 and gem5-resources are sub-directories of the current directory
docker run --rm -v ${PWD}:${PWD} -v ${PWD}/gem5-resources/src/gpu/DNNMark/cachefiles:/root/.cache/miopen/2.9.0 -w ${PWD} gcr.io/gem5-test/gcn-gpu gem5/build/GCN3_X86/gem5.opt gem5/configs/e
```

Outline

- Motivation
- Background
- **Improvements**
  - Enabling Support for ML Workloads in gem5
  - Making gem5 GPU models easier to use
  - **Improved, Higher Fidelity gem5 GPU support**
- Future Work & Conclusions

Improving gem5's GPU Support

- Support for ROCm 4.0 in SE mode
- Support for Carrizo-class dGPUs (gfx803)
- Support for Vega-class dGPUs & APUs (gfx900, gfx902)
- Improved concurrency via dynamic register allocation
- Improved concurrency via better dependency management
- CU masking support to run kernels on a subset of GPU resources
- Extend BLAS support to use kernels that run on real devices [Roarty et al. gem5 Wkshp'20]
- Multi-chiplet support – more representative of future GPUs [Yogatama et al. gem5 Wkshp'20]
- Scripts to compare gem5's fidelity against modern GPUs
- Properly report device properties for (ML) programs that dynamically select kernels
- …

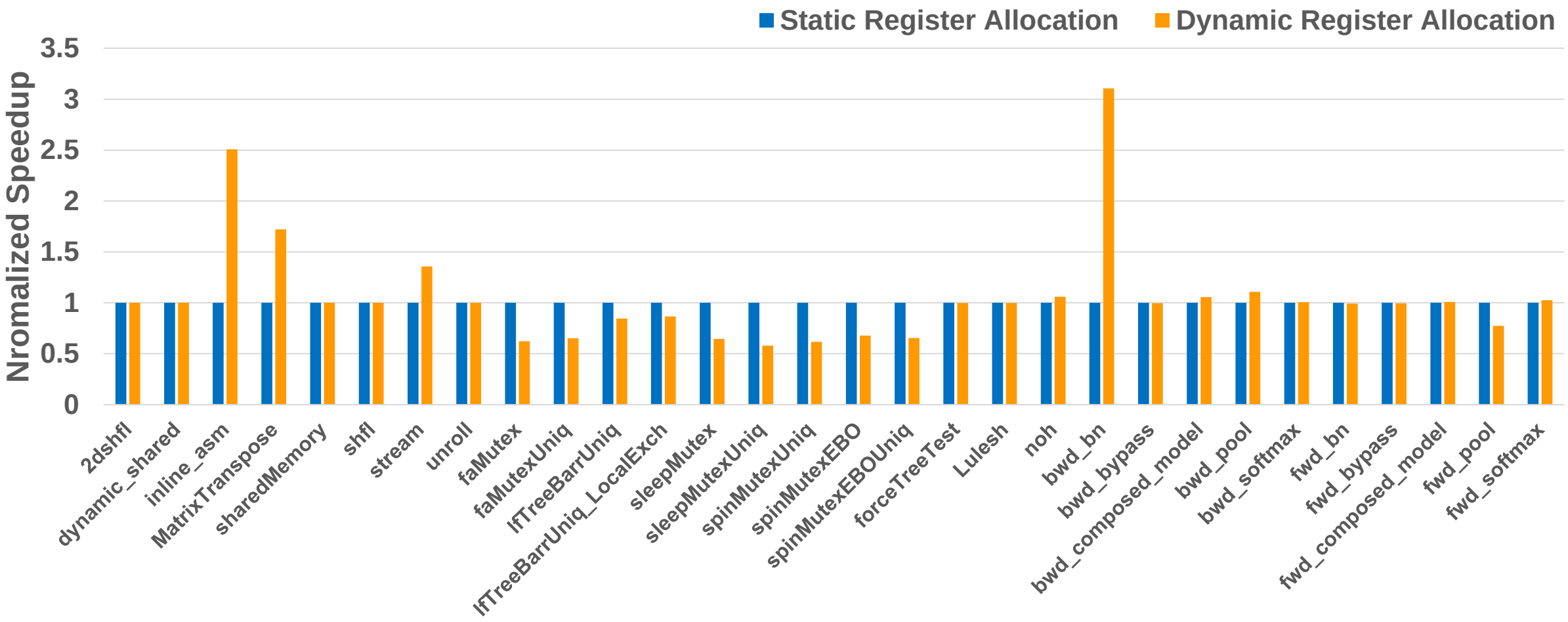
Today's focus

Improving Register Allocation Support

- Simple dependence tracking – only 1 wavefront/CU at a time
  - Even if sufficient registers are available for more WFs
- Issue: unrealistic relative to real GPUs
- Solution: add dynamic register allocator support
  - If enough registers available, schedule additional WFs concurrently/CU
  - Potentially can utilize all WF slots depending on register requirements
  - More complex, higher performance designs possible

Intuition: Dynamic allocator significantly improves accuracy

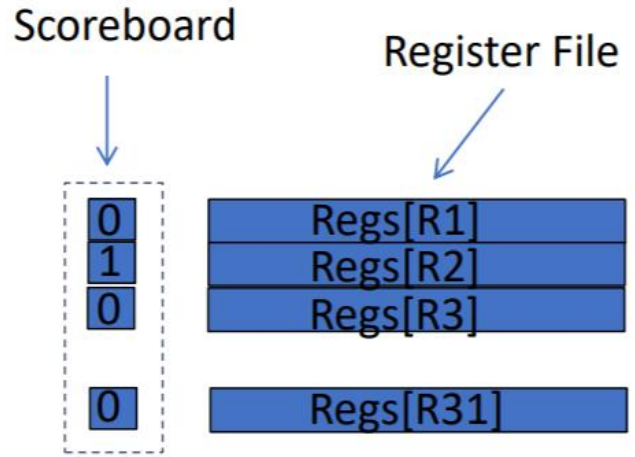
Dynamic Register Allocator Performance



Reality: dynamic register allocator 6% worse than simple – why?

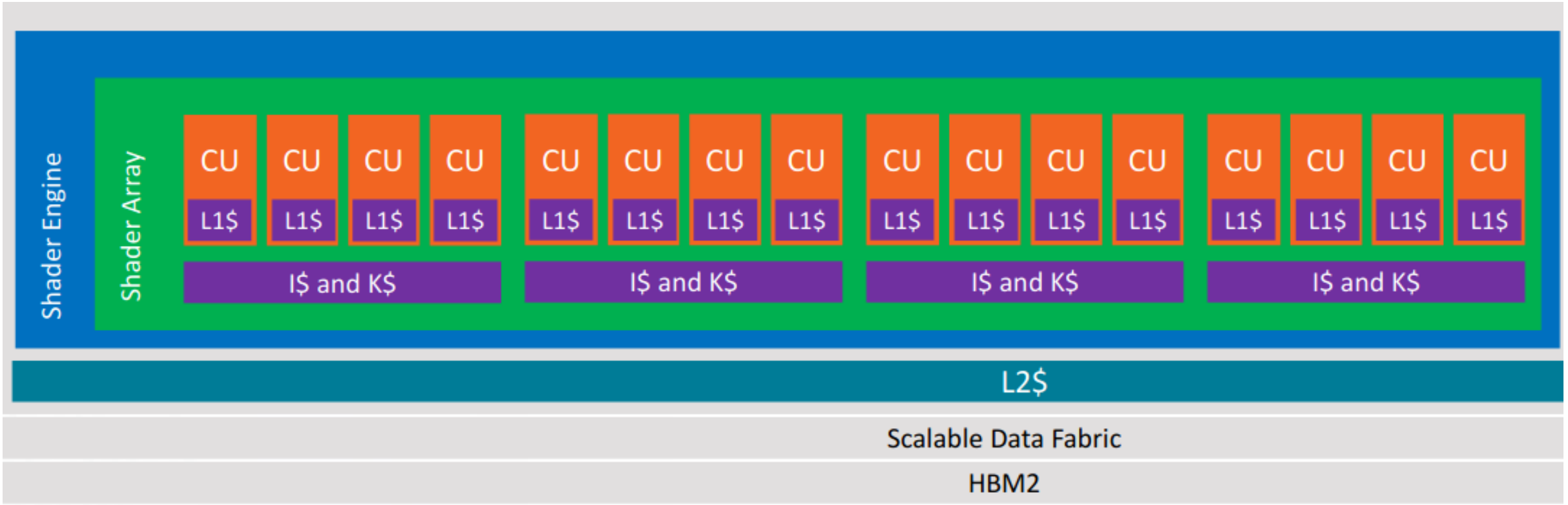
Issue: Dependence Tracking

- GPU model did not track dependencies well → many stalls
  - Result: optimizing register allocation in isolation was insufficient
- Issue: Proprietary GPU dependence checking sols unknown
- Solution: simple, in-order scoreboard
  - Bit per register to track use status
  - Cleared on instruction completion
  - Checks for RAW/WAW hazards



Result: up to 44% reduction in stalls

Vega GPU Support

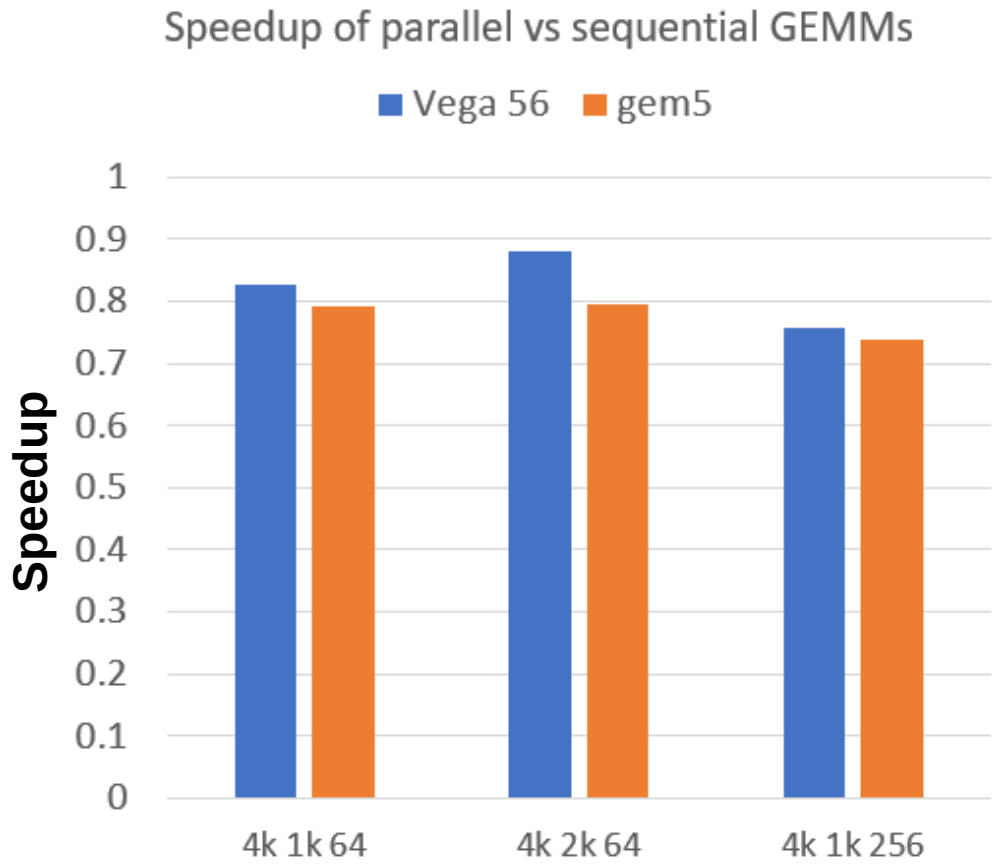


Source: AMD

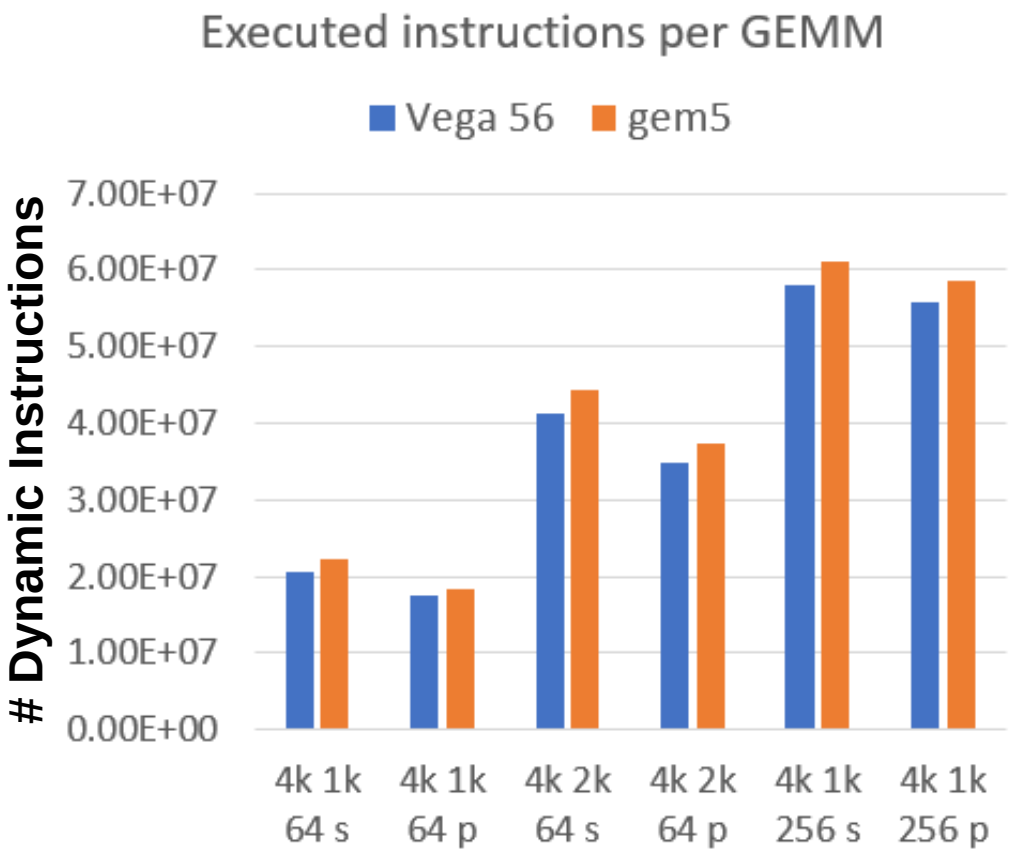
Added new HW support and configurations to model Vega

Validating gem5 Vega GPU configuration

- Validated Vega GPU configuration against Vega 56 GPU
- Ran sets of GEMMs sequentially and concurrently with CU masking & streams



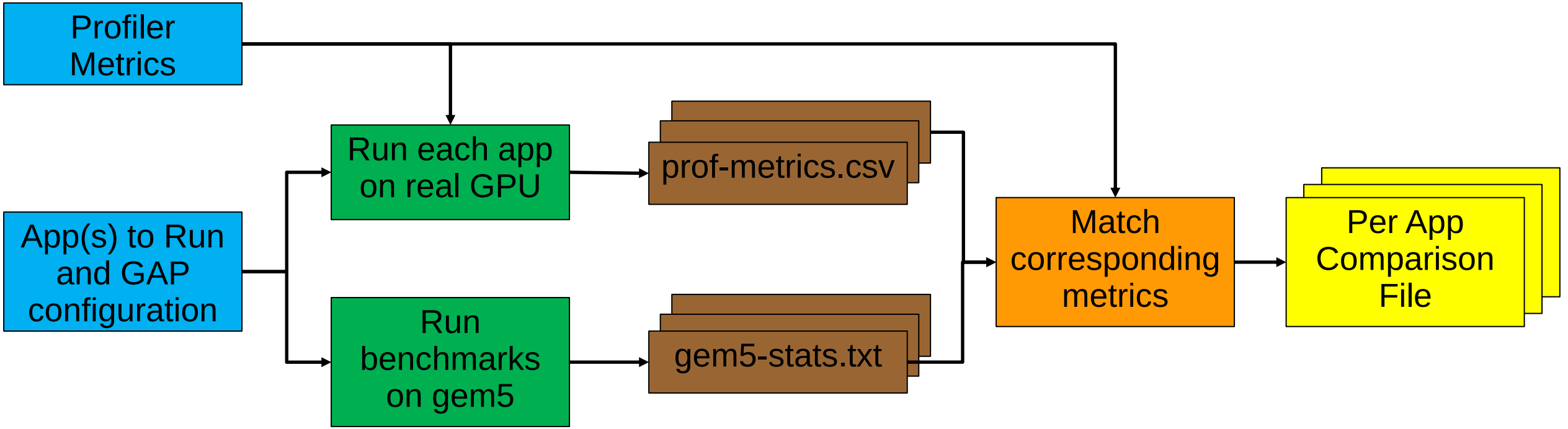
Average: 6% (Worst case: 10%)



Average: 6% (Worst case: 7%)

How Does GAP Work?

- Issue: need standard approach for evaluating new configurations
- Solution: gem5 GPU Accuracy Profiler (**GAP**)



Using GAP to iteratively refine gem5 GPU models

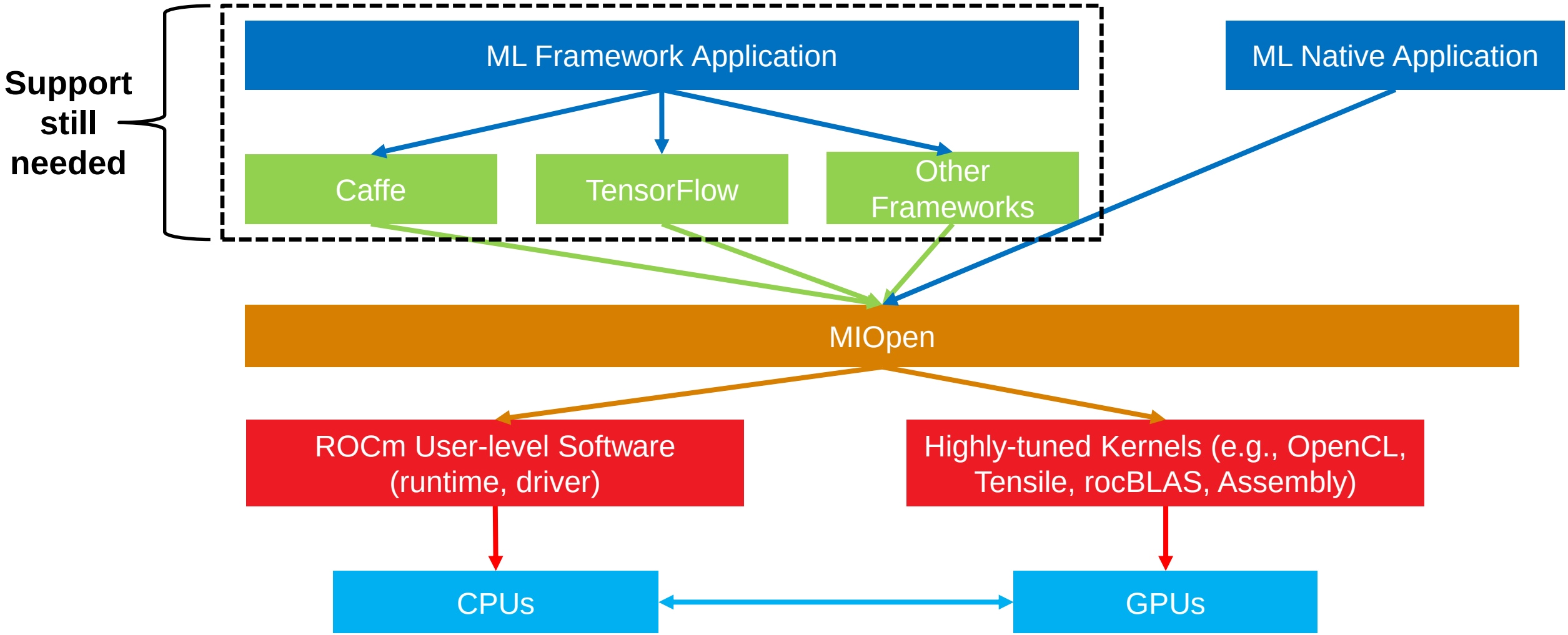
gem5 blog post and performance regression testing coming soon!

Outline

- Motivation
- Background
- Improvements
- **Future Work & Conclusions**

But what about TensorFlow?!

Next Steps



Modeling High-Level ML Frameworks in SE Mode is Challenging

Adding Full-System GPU Mode Support

- Additional Challenge:
  - Updating SE mode for new ROCm versions is time consuming
  - Ideally, avoid updating gem5 every time ROCm is updated
- Solution: add GPU FS mode support
  - Can checkpoint and fast-forward to focus on simulating ROIs
  - Newer ROCm versions more easily supported
  - Enables studying full impact of drivers on ML applications
  - Current status: able to run simple TF apps in gem5 on GPU!

Goal: support GPU FS mode in next gem5 release

Conclusions & Future Work

- Our updates enable gem5 to run native ML applications
- Significant updates to improve both usability and accuracy
- **Reduced barriers to entry for simulation**
  - gem5.org/documentation/general_docs/gpu_models/GCN3

- Further enhancements on the way:
  - FS and SE mode support – checkpointing and fast-forwarding
  - Additional publicly available applications and resources
  - Performance regression testing
  - Improved ability to run at different fidelity levels

RE-gem5: Building Sustainable Research Infrastructure

by Jason Lowe-Power and Matt Sinclair on Sep 12, 2019 | Tags: Measurements, Methodology, Simulators

ENS-1925485