MGPUSim: Enabling Multi-GPU Performance Modeling and Optimization

Yifan Sun[1], Trinayan Baruah[1], Saiful A. Mojumder[2], Shi Dong[1], Xiang Gong[1], Shane Treadway[1], Yuhui Bao[1], Spencer Hance[1], Carter McCardwell[1], Vincent Zhao[1], Harrison Barclay[1], Amir Kavyan Ziabari[3], Zhongliang Chen[3], Rafael Ubal[1], José L. Abellán[4], John Kim[5], Ajay Joshi[2], and David Kaeli[1]

[1]Northeastern University, [2]Boston University, [3]AMD, [4]Universidad Católica San Antonio Murcia, [5]KAIST

Multi-GPU Systems

- **Deliver high computing power**
- **Commonly used for:**
  - Machine learning training and inference
  - Large scale physics simulation
- **Exploit new communication features**
  - Shared Virtual Memory / Unified Memory
  - Demand paging / peer-to-peer memory access
  - System-level atomic operations
- **Require complex modeling capabilities**
  - Capture complex interactions between GPUs
- **Performance analysis for multi-GPU systems is necessary and difficult**

Current GPU Simulators

- **Model dated GPU architectures**
  - New GPU architectures have new multi-GPU communication features
- **Lack flexibility**
- **Have limited simulation scalability**
- **A systematic solution and a new GPU architecture simulator are needed**

Simulator Requirements

- **No magic**
- **Parallel simulation**
- **Traking data w/ timing**
- **No busy ticking**
- **Extension w/o modification**

Akita Simulator Framework

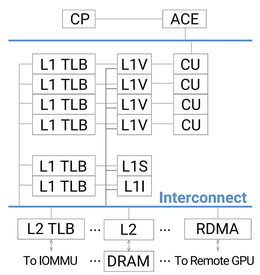
AKITA

https://gitlab.com/akita/akita

- **Implemented in the Go programming language**
- **Highly flexible**
  - Easy to replace components since they follow strong software engineering principles
- **Enables high performance simulation**
  - Pure event-driven simulation
  - Skips redundant simulated state updates
  - Parallel simulation w/o accuracy loss
- **Akita: a high performance simulator framework that can generate a number of high performance CPU/GPU simulators**

MGPUSim

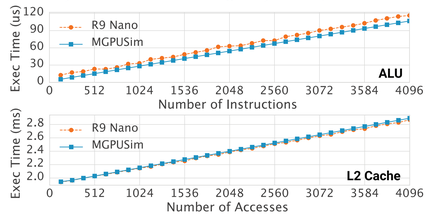
- **Models AMD GCN3 ISA**
  - 6-stage pipeline, w/ concurrent instruction issue
  - Write-through L1 cache, write-back L2 cache
  - Address translation with TLBs and IOMMUs
  - Peer-to-peer memory access using RDMA engines



- **MGPUSim is designed for multi-GPU simulations**
  - To be demonstrated with 2 design studies (Lcality API and Progressive Page-Splitting Migration)

Simulator Validation

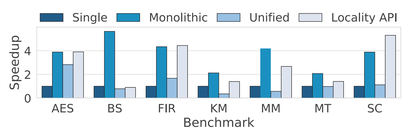
- **Microbenchmark validation**
  - 57 microbenchmarks (all show similar trends as the *ALU* and *L2 Cache* microbenchmark below)
  - High fidelity when simulating key components



- **Full Benchmark validation**
  - 5.5% average error in terms of execution time
- **Parallel simulation perfomance**
  - 3.5X speedup in functional emulation (4-core)
  - 2.5X speedup in detailed timing simulation (4-core)
- **MGPUSim is fast and accurate**

Locality API

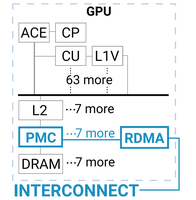
- **Unified GPU model**
  - Easy to program, but low performance
  - Lack of control on data/compute placement
- **Memory Placement API**
  - Remaps pages in consecutive virtual memory address spaces for each GPU
- **Compute Placement API**
  - Launches sub-kernels to each individual GPU



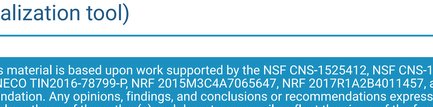
- **Locality API can reduce inter-GPU traffic and improve performance by 1.6X**

PASI (Progressive Page-Splitting Migration)

- **Page migration**
  - Employs Page Migration Controllers (PMC) to migrate pages
  - Better utilizes interconnect bandwidth and data locality



- **ESI coherency protocol**
  - Forms a Cache-Only Memory Architecture for multi-GPU systems
  - Allows GPUs to share pages with a "Shared" (S) state
  - Uses the "Exclusive" (E) state for managing writes



- **Page splitting**
  - Exploits a large initial page size to improve TLB coverage
  - Progressively splits shared pages, up to a minimum size
  - Adaptively finds an efficient page size and placement
- **PASI: 2.6X speedup compared to an unified 4-GPU system**

Takeaways

- **Akita framework**
  - A highly flexible simulator framework
  - High performance parallel simulation
- **MGPUSim**
  - Extensively validated AMD GCN3 Simulator
  - Multi-GPU simulation made easy
- **Locality API & PASI**
  - Design studies that demonstrate the flexibility and utility of MGPUSim
  - Explores effective multi-GPU memory management
- **Future work**
  - Modeling RISC-V based CPU architectures
  - Adding useful features to Akita (e.g. checkpointing, a visualization tool)

For more information, please visit **bit.ly/mgpusim**

This material is based upon work supported by the NSF CNS-1525412, NSF CNS-1525474, MINECO TIN2016-78799-P, NRF 2015M3C4A7065647, NRF 2017R1A2B4011457, and HSA Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding sources.