Analyzing OpenCL 2.0 Workloads Using a Heterogeneous CPU-GPU Simulator

Li Wang*, Ren-Wei Tsai†, Shao-Chung Wang‡, Kun-Chih Chen§, Po-Han Wang*,
Hsiang-Yun Cheng¶, Yi-Chung Lee*, Sheng-Jie Shu†, Chun-Chieh Yang‡, Min-Yih Hsu‡,
Li-Chen Kan‡, Chao-Lin Lee‡, Tzu-Chieh Yu*, Rih-Ding Peng*, Chia-Lin Yang*,
Yuan-Shin Hwang‖, Jenq-Kuen Lee‡, Shiao-Li Tsao† and Ming Ouhyoung*
*National Taiwan University, †National Chiao Tung University, ‡National Tsing Hua University,
§National Sun Yat-Sen University, ¶Academia Sinica, ‖National Taiwan University of Science and Technology
*{r03922025,f96922002,b00902051,r03922157,r03944033,yangc,ming}@csie.ntu.edu.tw,
†{yamato1219.cs03g,wind8.cs03g,sltsao}@cs.nctu.edu.tw, ‡{scwang,jet,myhsu,lckan,clli}@pllab.cs.nthu.edu.tw,
‡jklee@cs.nthu.edu.tw, §kcchen@mail.cse.nsysu.edu.tw, ¶hycheng@citi.sinica.edu.tw, ‖shin@csie.ntust.edu.tw

***Abstract*—Heterogeneous CPU-GPU systems have recently emerged as an energy-efficient computing platform. A robust integrated CPU-GPU simulator is essential to facilitate researches in this direction. While few integrated CPU-GPU simulators are available, similar tools that support OpenCL 2.0, a widely used new standard with promising heterogeneous computing features, are currently missing. In this paper, we extend the existing integrated CPU-GPU simulator, gem5-gpu, to support OpenCL 2.0. In addition, we conduct experiments on the extended simulator to see the impact of new features introduced by OpenCL 2.0. Our OpenCL 2.0 compatible simulator is successfully validated against a state-of-the-art commercial product, and is expected to help boost future studies in heterogeneous CPU-GPU systems.**

I. Introduction

Heterogeneous systems that tightly integrate CPU and GPU together to provide immense computing capability are becoming the recent trend in microprocessor designs. While heterogeneous CPU-GPU computing quickly gains momentum, lack of robust and comprehensive simulators for such systems hinders advanced progress in this research direction. Although several tools are currently available for simulating heterogeneous CPU-GPU systems [1], [2], [3], none of them supports the newly introduced heterogeneous computing standard, OpenCL 2.0 [4].

In this paper, we extend gem5-gpu [1] to support OpenCL 2.0. We choose gem5-gpu among several CPU-GPU simulators, as CPU and GPU in gem5-gpu share the same address space and the underlying GPU architecture is capable of running OpenCL 1.2 kernels. Compared to OpenCL 1.2, OpenCL 2.0 is widely adopted by industry and it better represents future heterogeneous systems, as OpenCL 2.0 enables (1) shared virtual memory, (2) dynamic parallelism, (3) platform atomics, and (4) work-group built-in functions. We modify the OpenCL interface and the GPU model to support these new features. In addition, we conduct experiments on the developed simulator to analyze the performance impact of these new features.

II. Heterogeneous Systems and OpenCL 2.0

The heterogeneous system we discussed in this paper is based on the integrated CPU-GPU simulator, gem5-gpu. The system consists of two parts: a CPU cluster modeled by gem5 [5] and

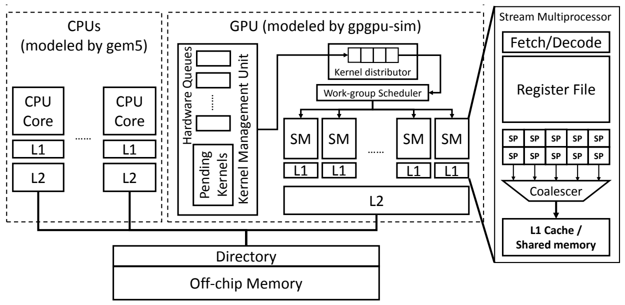

Fig. 1: Overview of gem5-gpu architecture.

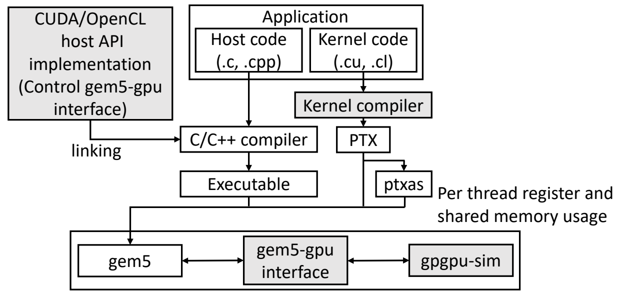

Fig. 2: gem5-gpu simulation framework and our extensions.

a GPU modeled by gpgpu-sim [6]. Figure 1 shows a high level overview of the heterogeneous system.

OpenCL 2.0 introduces some new features to enable efficient heterogeneous computing. These features give GPU more control over data management and kernel execution, leading to flexible CPU-GPU cooperation.

A. Shared Virtual Memory (SVM)

The SVM feature in OpenCL 2.0 allows CPU and GPU to share a common virtual address region. This feature makes the data transfer between CPU and GPU more efficient and enables flexible usage of the programming model, such as the usage of pointer-based data structures.

B. Dynamic Parallelism

OpenCL 2.0 allows a GPU to launch kernels to itself, enabling more flexible algorithm design. Nested parallelism and data-dependent parallelism are two use cases that could benefit from dynamic parallelism. With dynamic parallelism,

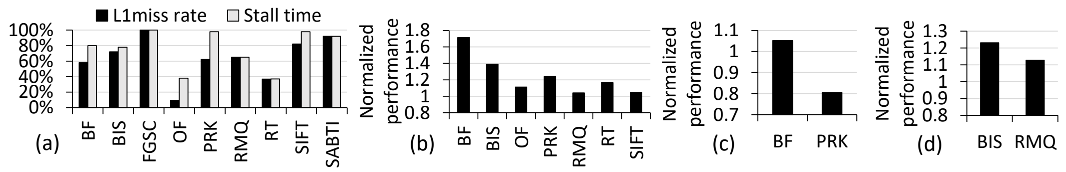

Fig. 3: (a) L1 miss rate and portion of stall time; Performance of OpenCL 2.0 workloads using (b) SVM, (c) dynamic parallelism, and (d) work-group built-in functions normalized to OpenCL 1.2 version of applications.

the delay caused by load imbalance and data-dependency in the kernel processing of these two use cases can be alleviated.

C. Platform Atomics

Atomic operations provide efficient ways to achieve inter-thread communications in heterogeneous systems. OpenCL 2.0 defines three platform-layer atomic functions (scopes) to guarantee the write atomicity for all threads (a) in the same work-group (i.e., a group of threads dispatched to the same stream multi-processor), (b) in the same computing device (i.e., a GPU), or (c) in different devices connected to the SVM.

D. Work-Group Built-in Functions

The work-group built-in functions introduced in OpenCL 2.0 provide programmers a high-level interface to allow threads within a work-group to perform simple arithmetic operations, including reduction, broadcasting, and scanning. These functions improve the programmability of OpenCL.

III. Implementation of the Simulation Framework

We extend gem5-gpu to support the simulation of OpenCL 2.0 workloads. Figure 2 shows the simulation framework. The application's host codes and kernel codes are compiled separately into host executable files and PTX instructions (GPU assembly codes). During the execution time, gem5 executes our customized host API implementation and sends commands to gpgpu-sim through the gem5-gpu interface. When there is a memory request during PTX execution, gpgpu-sim send the memory request through gem5-gpu interface to the memory system modeled by gem5. We modify several components, marked in gray in Figure 2, to support OpenCL 2.0.

A. Customized OpenCL Host API

We leverage current implementation of gem5-gpu to add OpenCL APIs. For the APIs that have different parameters between OpenCL and gem5-gpu, we gather necessary information from OpenCL APIs and send the commands to gem5-gpu. For memory-related APIs, we use *cudaMallocHelper* and *cudaMemcpy* in original gem5-gpu to emulate.

B. OpenCL to PTX Compiler

We use LLVM [7] to implement a new compiler to compile OpenCL kernel code. The compiler front end (*Clang*) is modified to support SVM and the OpenCL C built-in function library (*libclc*) is extended to support work-group built-in functions and atomic operations. Our compiler also supports dynamic parallelism by leveraging device-launch APIs.

C. Supporting New Version of PTX

To exploit new features in OpenCL 2.0, we need to support new version of PTX (PTX 3.1) by modifying gem5-gpu interface and gpgpu-sim. We create a data path from stream multi-processors (SMs) to the kernel management unit to enable the implementation of dynamic parallelism. New options (i.e., scopes) are added on the existing atomic instructions in gem5-gpu to support platform atomics, and *warp shuffle* in PTX 3.1 is utilized to implement work-group built-in functions.

IV. Evaluation

We validate our simulator against a AMD Kaveri A10-7850k APU. Nine benchmarks from AMD's APP SDK v3.0, Pannotia Benchmark Suite, and computer vision-related applications are selected for the evaluation. Results show that the normalized simulation runtime of our simulator is strongly correlated with the normalized runtime of real hardware, with 0.96 correlation coefficient on average. The performance of each benchmark is closely related to the portion of stall time, which greatly matches the L1 miss rate, as shown in Figure 3(a).

In addition to performance validation, we also analyze the impact of new features introduced by OpenCL 2.0. Figure 3(b) shows that SVM can improve the performance of traditional OpenCL 1.2 applications by eliminating their memory copy time. Dynamic parallelism incurs performance overhead when launching fine-grained kernels, so slowdown is observed in some benchmarks, such as PRK in Figure 3(c). On the other hand, dynamic parallelism enables pipeline execution between producer and consumers (child kernels) in BF, and the performance gain of producer-consumer pipelining outweighs the performance overhead. Using OpenCL 2.0 work-group built-in functions also provides performance benefits compared to using OpenCL 1.2 functionally-equivalent implementation, as shown in Figure 3(d). With the new PTX *warp shuffle* instruction, the synchronization overhead during data exchange can be eliminated, leading to significant performance improvement.

Acknowledgment

This work is supported in part by research grants from the Ministry of Science and Technology of Taiwan (MOST-105-2221-E-002-156-MY2 and MOST-105-2622-8-002-002), and sponsored by MediaTek Inc., Hsin-chu, Taiwan.

References

[1] J. Power *et al.*, "gem5-gpu: A heterogeneous CPU-GPU simulator," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 34–36, Jan 2015.

[2] R. Ubal *et al.*, "Multi2Sim: A simulation framework for CPU-GPU computing," in *PACT*, 2012, pp. 335–344.

[3] P. H. Wang *et al.*, "Full system simulation framework for integrated CPU/GPU architecture," in *VLSI-DAT*, 2014, pp. 1–4.

[4] J. E. Stone *et al.*, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science Engineering*, vol. 12, no. 3, pp. 66–73, May 2010.

[5] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, May 2011.

[6] A. Bakhoda *et al.*, "Analyzing CUDA workloads using a detailed GPU simulator," in *ISPASS*, 2009, pp. 163–174.

[7] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *CGO*, 2004, p. 75.