Journal of the Korea Industrial Information Systems Research Vol. 19, No. 2, Apr. 2014
http://dx.doi.org/10.9723/jksiis.2014.19.2.015

비x86 플랫폼 상에서의 CUDA 컴퓨팅을 위한 QEMU 및 GPGPU-Sim 기반 시뮬레이션 프레임워크 개발†

(A Simulation Framework for CUDA Computing on Non-x86 Platforms based on QEMU and GPGPU-Sim)

황 재 민[1], 최 종 욱[2], 최 성 림[3], 남 병 규[4]
(Jaemin Hwang, Jong-Wook Choi, Seongrim Choi, and Byeong-Gyu Nam)

요 약 본 논문에서는 QEMU와 GPGPU-Sim에 기반하여 비x86 플랫폼을 위한 CUDA 시뮬레이션 프레임워크를 제안한다. 기존 CPU-GPU 이종 컴퓨팅 시뮬레이터는 x86 CPU 모델만을 지원하거나 CUDA를 지원하지 않는 한계를 가진다. 제안된 시뮬레이터는 이러한 문제를 해결하기 위해 x86을 포함하여 비x86 CPU 모델을 지원 가능한 QEMU와 CUDA를 지원하는 GPU 시뮬레이터인 GPGPU-Sim을 통합하였다. 이를 통해 비x86 기반의 CUDA 컴퓨팅 환경을 시뮬레이션할 수 있도록 하였다.

핵심주제어 : 비x86, CUDA, 시뮬레이터, QEMU, GPGPU-Sim

Abstract This paper proposes a CUDA simulation framework for non-x86 computing platforms based on QEMU and GPGPU-sim. Previous simulators for heterogeneous computing platforms did not support for non-x86 CPU models or CUDA computing platform. In this work, we combined the QEMU and the GPGPU-Sim to support the non-x86 CPU models and the CUDA platform, respectively. This approach provides a simulation framework for CUDA computing on non-x86 CPU models.

Key Words : Non-x86, CUDA, Simulator, QEMU, GPGPU-Sim

1. 서 론

GPU는 많은 수의 연산 유닛을 가지고 있어 데이터 병렬성이 높은 작업을 처리하는데 유리한 구조를 가지고 있다[1]. 이러한 GPU의 특징을 활용하여 프로그램에서 병렬 처리가 가능한 부분을 GPU가 처리하고, 나머지 부분을 CPU가 담당함으로써 성능향상을 꾀하는 CPU-GPU 이종 컴퓨팅 (CPU-GPU heterogeneous computing)이 최근 들어 각광받고 있다. 이러한 CPU-GPU 이종 컴퓨팅 플랫폼으로는 크게 NVIDIA의 CUDA 컴퓨팅[2]과 Khronos Group의 OpenCL[3]이 존

† 이 연구는 2010년도 충남대학교 학술연구비에 의해 지원되었음
1) 충남대학교 컴퓨터공학과
2) 항공우주연구원 위성비행소프트웨어팀
3) 충남대학교 컴퓨터공학과
4) 충남대학교 컴퓨터공학과, 교신저자(bgnam@cnu.ac.kr)

재한다. 현재 Khronos의 OpenCL이 이종 컴퓨팅에 대한 표준 플랫폼으로 재정되어 있지만 OpenCL의 API들이 CUDA에 비해 저수준에서 정의되어 있어 보다 사용이 편리한 CUDA가 널리 사용되고 있는 실정이다. 이러한 이종 컴퓨팅 동향에 맞물려 현재 다양한 분야에서 CPU와 GPU를 활용한 이종 컴퓨팅이 각광받고 있으며, 그에 따라 다양한 형태의 이종 컴퓨팅 시스템을 개발할 필요성이 나타나고 있다. 특히, 모바일 시스템에서는 저전력 ARM CPU에 기반한 CUDA 컴퓨팅이 연구되고 있고[4], 최근에는 저전력 서버에서도 ARM CPU에 기반한 CUDA 컴퓨팅이 시도되는 것으로 알려지고 있다. 이러한 새로운 시스템들에 대한 개발 시간을 단축하기 위하여 시뮬레이터 상에서 소프트웨어를 미리 개발하고 테스트하기 위한 목적으로 이종 컴퓨팅 시뮬레이터에 대한 연구도 활발히 진행 중이다[5-11]. 그러나 기존의 이종 컴퓨팅 시뮬레이터들은 x86 CPU 모델만을 지원하거나 비x86 CPU 모델을 지원하더라도 GPU 컴퓨팅을 위한 CUDA를 지원하지 못하는 한계를 지니고 있다[5-8].

본 논문에서는 이러한 한계점들을 극복하고 x86 CPU를 비롯한 비x86 CPU 모델을 지원함과 동시에 CUDA를 지원하기 위하여 오픈소스 기반의 QEMU와 GPGPU-Sim을 통합한 CPU-GPU 이종 컴퓨팅 시뮬레이터를 개발하였다. 여기에 사용된 QEMU는 x86뿐만 아니라 비x86 CPU에 대한 폭넓은 모델을 제공하는 CPU 시뮬레이터이며[12], GPGPU-Sim은 CUDA를 지원하는 가장 널리 사용되고 있는 NVIDIA GPU 시뮬레이터이다[13].

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구에 대해 소개하며, 3장에서는 제안된 CPU-GPU 통합 시뮬레이터를 설명한다. 그리고 4장에서는 제안된 시뮬레이터를 이용한 결과를 보여주며, 5장에서 결론을 맺는다.

2. 관련 연구

현재까지 연구된 CPU-GPU 이종 컴퓨팅 시뮬레이터 및 그 특징을 <표 1>에 나타내었다. Multi2Sim[5]는 x86뿐만 아니라 비x86 CPU 모델을 가지지만 유저 레벨 시뮬레이션만 가능하므로 운영체제를 구동할 수 없으며, 특히 GPU 모델과의 연동은 x86 CPU 모델에서만 가능하다. FusionSim[6]의 경우 CUDA를 지원하는 GPGPU-Sim을 이용하지만, PLTSim[9]을 이용하므로 비x86 CPU 모델을 지원하지 못하는 제한을 가진다. integrated gem5+GPGPU-Sim[7]에서는 ARM과 Alpha 두 가지 모델의 비x86 CPU를 지원하며 GPGPU-Sim과 연동하여 CUDA 컴퓨팅을 수행 할 수 있다. 하지만 완성된 바이너리를 사용하여 아키텍처 탐색을 목적으로 개발되

<표 1> CPU-GPU 시뮬레이터 비교
<Table 1> Comparison of CPU-GPU simulators

Simulator	Developer	CPU Models	GPU Models	Simulator Type	Non-x86 +CUDA
QEMU+GPGPU-Sim	Chungnam National University	**QEMU** (ARM, x86, Alpha, SPARC, etc.)	**GPGPU-Sim** (NVIDIA GT200 and Fermi)	Full System Simulator	Yes
Multi2Sim[5]	Northeastern University	x86	NVIDIA Fermi, AMD Radeon 5870	User Level Simulator	No
FusionSim[6]	University of Toronto	**PLTSim** (x86)	**GPGPU-Sim** (NVIDIA GT200 and Fermi)	Full System Simulator	No
integrated gem5+GPGPU-Sim[7]	University of Wisconsin - Madison	**gem5** (ARM, Alpha)	**GPGPU-Sim** (NVIDIA GT200 and Fermi)	Full System Simulator	Yes (only for architecture analysis)
QEMU+GPUSim[8]	University of Bologna	**QEMU** (ARM, x86, Alpha, SPARC, etc.)	Many-core Simulator	Full System Simulator	No

어 있어 소프트웨어 개발의 목적으로 사용하기에 적합하지 않다. QEMU+ GPUSim[8]은 QEMU를 이용하여 x86과 비x86 CPU 모델을 모두 지원하며 ARM CPU에서 GPU 모델과의 연동이 가능하다. 하지만 독자적인 GPU 모델을 대상으로 한 시뮬레이터인 관계로 CUDA를 지원하지 않는다.

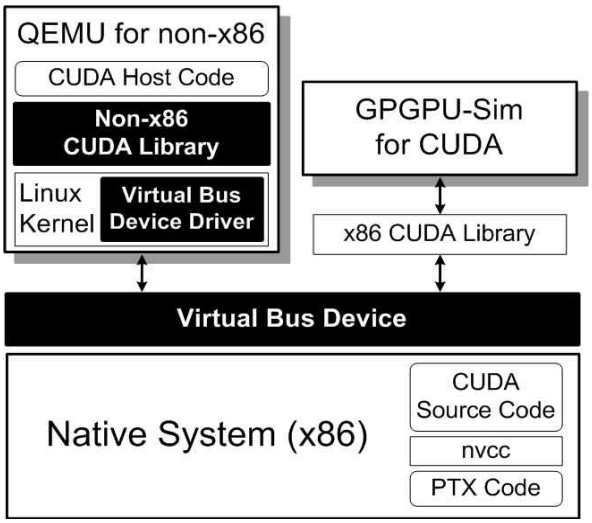
본 논문에서 제안한 QEMU+GPGPU-Sim은 기존의 연구가 지닌 한계들을 극복하기 위하여 x86뿐만 아니라 비x86 CPU 모델을 지원하는 QEMU를 사용하였고, 이에 CUDA를 지원하기 위해 GPGPU-Sim을 연동하여 비x86상에서도 CUDA 개발이 가능한 시뮬레이션 프레임워크를 구현하였다.

3. QEMU 및 GPGPU-Sim 기반의 시뮬레이션 프레임워크 개발

3.1 QEMU+GPGPU-Sim 구조

NVIDIA에서는 자사의 CUDA 플랫폼을 x86 환경에 대해서만 지원하고 있으며, 비x86 환경에 대해서는 CUDA 프로그래밍을 위한 nvcc 컴파일러 및 CUDA 라이브러리를 제공하지 않고 있다. 따라서 본 논문에서는 <그림 1>에 보인 구조를 가지는 비x86 환경에 대한 CUDA 시뮬레이션 프레임워크를 개발하였다. <그림 1>에서는 제안된 프레임워크를 위해 개발된 비x86용 CUDA 라이브러리, 가상버스 드라이버 및 가상버스를 검은색 상자로 표시하였다.

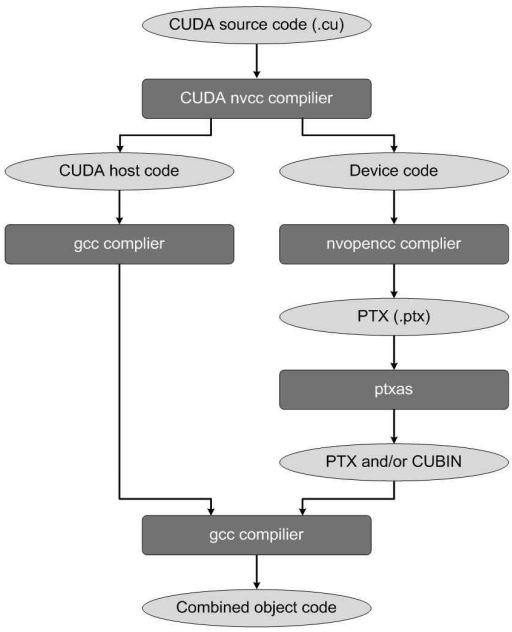
본 논문에서 제안된 시뮬레이션 프레임워크 x86 CPU를 사용하는 기반시스템 상에 QEMU와 GPGPU- Sim을 올리고 가상의 버스를 통해 이 둘이 통신하도록 하였다. QEMU와 GPGPU-Sim은 각각 가상의 CPU와 GPU를 시뮬레이션하며, 비x86 환경에서 CUDA 호스트 코드 수행에 필요한 CUDA 라이브러리를 새로이 개발하여 제공한다. 또한, QEMU와 GPGPU- Sim간의 통신 수단인 가상 버스의 드라이버가 QEMU에 등록 되어 있으며, CUDA 소스 코드는 기반 x86 시스템에 서 nvcc 컴파일러를 통해 GPU에서 수행될 PTX 코드로 컴파일된다.



<그림 1> QEMU+GPGPU-Sim 시뮬레이션 프레임워크의 구조

<Fig. 1> Structure for QEMU+GPGPU-Sim simulation framework

3.2 QEMU+GPGPU-Sim 동작



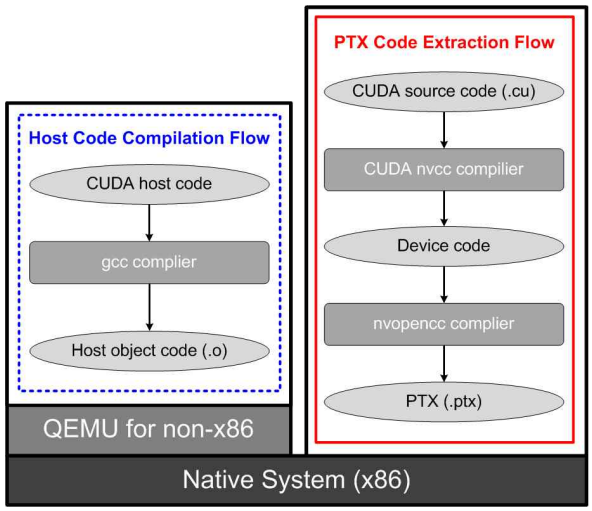
<그림 2> nvcc를 이용한 컴파일 과정

<Fig. 2> Compilation flow using nvcc

기존의 CUDA 컴파일 과정은 <그림 2>에 나타

난 바와 같이 nvcc 컴파일러를 통해 소스 코드를 CUDA 호스트 코드와 CUDA 디바이스 코드로 분리한다. 이렇게 분리된 CUDA 호스트 코드는 gcc를 통해 컴파일되며, CUDA 디바이스 코드는 nvopencc를 통해 PTX 코드로 컴파일된 후 다시 ptxas를 통해 어셈블된다. 이렇게 생성된 두 파일은 최종적으로 gcc를 통해 오브젝트 코드로 합쳐진다. 그러나 CUDA에서는 비x86 환경을 위한 nvcc 컴파일러가 제공되지 않으므로 <그림 2>에 나타난 CUDA 컴파일 과정은 x86 환경에서만 가능하여, 본 연구에서는 <그림 3>과 같은 방법을 제안하여 nvcc 컴파일러 문제를 해결하였다. CUDA 소스 코드 중 CUDA 호스트 코드는 gcc로 컴파일 가능하므로 <그림 3>의 왼쪽 점선상자에 보인 바와 같이 gcc를 지원하는 비x86 환경의 QEMU 상에서 gcc를 이용하여 컴파일된다. 반면, GPU에서 동작하는 CUDA 디바이스 코드는 비x86용 nvcc 컴파일러가 제공되지 않으므로 <그림 3>의 오른쪽 실선상자와 같이 기반 x86 시스템에서 컴파일하며 이 과정에서 nvopencc가 생성하는 PTX 코드만을 추출해 낸다. 이를 QEMU 상의 CUDA 호스트 코드가 GPGPU-Sim에게 전달함으로써 비x86 환경에서의 CUDA 컴파일 문제를 해결하였다.

한편, QEMU 상의 비x86 환경에서 CUDA 호스트 코드는 CUDA 라이브러리를 통하여 GPGPU-Sim을 제어해야 하지만 비x86용 CUDA 라이브러리가 제공되지 않으므로 본 연구에서는 해당 라이브러리를 새롭게 개발하였다. 또한 비



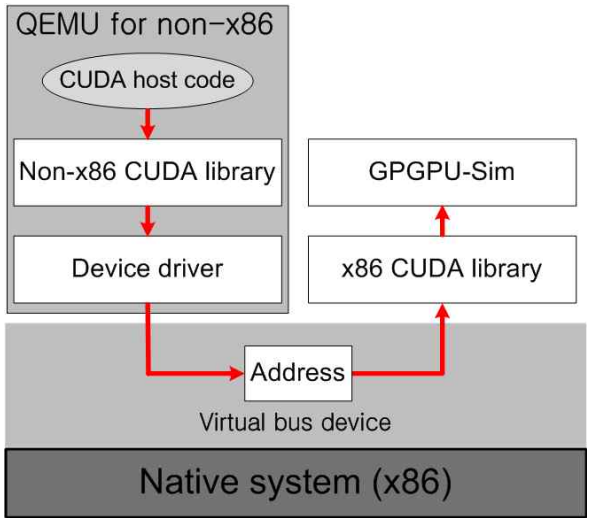
<그림 3> 제안된 PTX 추출 과정
<Fig. 3> Proposed PTX extraction flow

x86용 CUDA 라이브러리에서 GPGPU-Sim을 제어하기 위해서는 QEMU와 GPGPU-Sim간 통신할 수 있는 채널이 필요하므로, 제안된 시뮬레이터에서는 QEMU 상의 특정 I/O 영역에 매핑되는 가상 버스를 구현하여 이러한 채널을 마련하였다.

개발된 CUDA 라이브러리 및 가상 버스를 이용하여 CUDA 호스트 코드가 GPGPU-Sim을 제어하는 과정은 <그림 4>에 나타나 있다. 먼저 CUDA 호스트 코드가 비x86용 CUDA 라이브러리의 API를 호출하면 디바이스 드라이버를 통해 가상 버스의 특정 주소를 액세스한다. 이러면 가상 버스는 해당 주소에 대응하는 x86용 CUDA

<표 2> 개발된 비x86용 CUDA 라이브러리의 API
<Table 2> Developed non-x86 CUDA library API

Non-x86 CUDA Library API	Description
cuda_register_fatbinary()	- initializes GPGPU-Sim - loads PTX code
cuda_register_function()	- generates a map between PTX entry point and application function address
cuda_malloc()	- allocates memory of the device
cuda_memcpy_htod()	- copies data to device from host
cuda_launch_kernel()	- sets grid and block - sets up the arguments for the PTX code - launches the PTX code
cuda_memcpy_dtoh()	- copies data to host from device
cuda_free()	- releases memory of the device



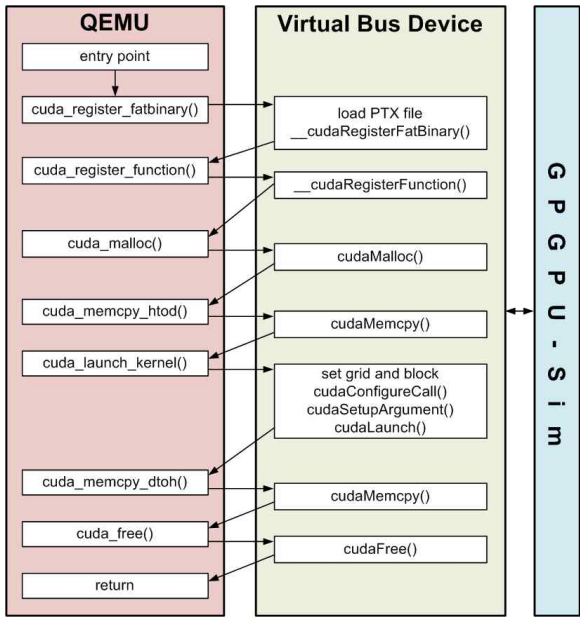
<그림 4> GPGPU-Sim 제어 과정
<Fig. 4> GPGPU-Sim control flow

라이브러리의 API를 호출함으로써 CUDA 호스트 코드가 GPGPU-Sim을 제어할 수 있도록 하였다.

4. 수행 결과

제안된 QEMU+GPGPU-Sim 시뮬레이터를 테스트하기 위해 구축한 환경은 다음과 같다. 기반 x86 시스템의 운영체제는 Ubuntu 12.04 LTS 32-bit를 사용하였으며, PTX 파일을 추출하기 위한 컴파일러는 Linux용 NVIDIA CUDA Toolkit v3.1의 nvcc을 이용하였다. 제안된 QEMU+GPGPU-Sim 프레임워크를 구성하기 위하여 QEMU v1.6.2 및 GPGPU-Sim v3.2.2을 이용하였으며, CPU 및 GPU 모델은 각각 SPARC 및 GTX 480으로 선택하였고, 덧셈 프로그램으로 테스트를 수행하였다.

QEMU+GPGPU-Sim 시뮬레이터를 검증하기 위해 QEMU가 비x86용 CUDA 라이브러리를 이용하여 GPGPU-Sim을 제어하는 과정을 확인하였다. 비x86용 CUDA 라이브러리는 <표 2>에 나타낸 바와 같이 GPGPU-Sim을 제어하기 위한 API로 구성되며, 이 API들을 이용하여 QEMU가 GPGPU-Sim을 제어하는 과정을 <그림 5>에 보였다. 이 과정을 살펴보면, 먼저 QEMU 상의

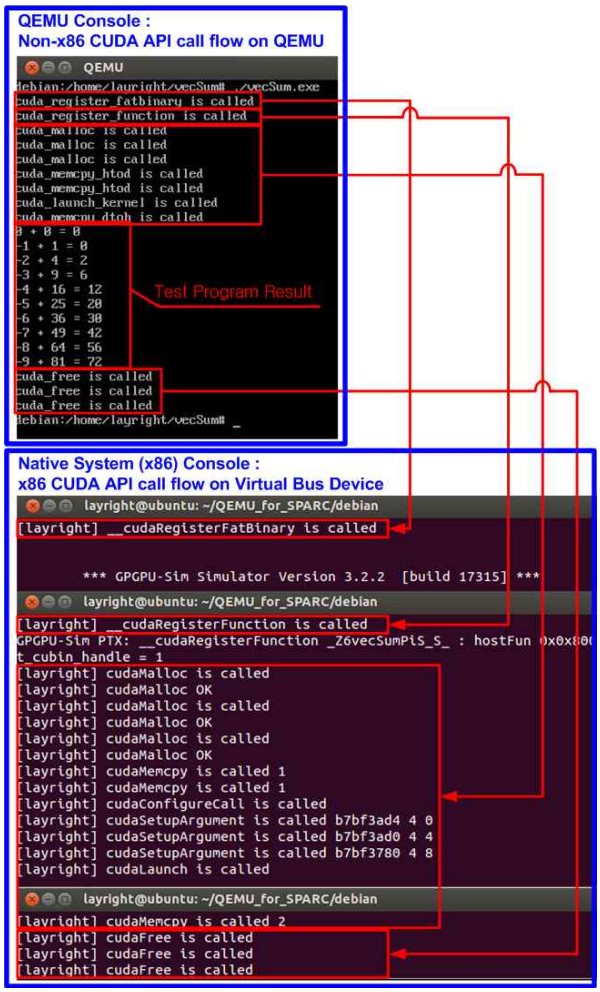


<그림 5> 실행 과정
<Fig. 5> Execution trace

CUDA 호스트 코드는 비x86용 CUDA 라이브러리의 cuda_register_fatbinary() API를 호출하여 기반 x86 시스템으로부터 PTX 코드를 로드하며, 가상 버스의 __cudaRegisterFatBinary() API를 통해 GPGPU-Sim으로 PTX 코드를 전달한다. 이후 CUDA 호스트 코드가 비x86용 CUDA 라이브러리의 cuda_register_function() API를 통해 GPGPU-Sim에게 PTX 코드의 시작점을 알려주고, cuda_malloc() API를 호출하여 GPGPU-Sim이 필요한 메모리를 할당받게 한다. 그리고 cuda_memcpy_htod() API가 CUDA 호스트 코드의 파라미터를 GPGPU-Sim으로 복사해주면, cuda_launch_kernel() API는 GPGPU-Sim에서 그리드 및 블록을 설정하고 복사된 파라미터를 GPGPU-Sim에서 실행될 커널에게 전달해주어 PTX 파일의 시작점부터 명령어를 수행할 수 있도록 한다. QEMU는 GPGPU-Sim이 CUDA 코드 실행을 마치면 cuda_memcpy_dtoh() API를 통해 결과를 받아오며, cuda_free() API를 통해 GPGPU-Sim에서 작업을 위해 할당된 메모리를 모두 해제하고 종료하게 된다.

<그림 6>에서는 <그림 5>에 보인 과정이 본

연구에서 구현한 시뮬레이터 상에서 제대로 수행되는 것을 보이고 있다. <그림 6>의 QEMU 콘솔(上)에서 비x86용 CUDA API를 호출하면 기반 x86 콘솔(下)에서 이에 대응하는 x86용 CUDA API가 호출되는 것을 화살표로 나타내고 있으며, 테스트 프로그램이 정상 수행된 결과가 출력되는 것을 볼 수 있다. 결과적으로 QEMU와 GPGPU-Sim이 제대로 연동되고, 비x86 CPU인 SPARC 플랫폼 상에서 CUDA 컴퓨팅이 성공적으로 수행되는 것을 확인할 수 있다.



<그림 6> 예제 프로그램 수행 결과
<Fig. 6> Sample program running result

5. 결 론

본 논문에서는 기존의 CPU-GPU 이종 컴퓨팅 시뮬레이터들이 비x86 CPU에 대한 CUDA 컴퓨팅이 불가능한 한계를 극복하기 위해, 비x86 환경을 위한 CUDA 컴퓨팅 시뮬레이션 프레임워크를 제안하였다. 이를 위해, 비x86 CPU를 지원하는 QEMU와 CUDA를 지원하는 GPU 시뮬레이터인 GPGPU-Sim을 통합하였고, 이 때 비x86 환경을 위한 nvcc 컴파일러 및 CUDA 라이브러리가 지원되지 않는 문제를 해결하기 위해 비x86용 CUDA 라이브러리를 개발하고 QEMU와 GPGPU-Sim간 통신 채널을 구성하는 가상 버스를 구현하였다. 이를 통해 비x86 CPU인 SPARC 플랫폼 상에서도 CUDA 컴퓨팅을 수행하는 프로그램을 개발하고 디버깅할 수 있는 시뮬레이션 프레임워크를 구현할 수 있음을 보였다.

References

1] B.-G. Nam and H.-J Yoo, “An Embedded Stream Processor Core Based on Logarithmic Arithmetic for a Low-Power 3-D Graphics SoC”, IEEE J. Solid-State Circuits, Vol. 44, No. 5, pp. 1554 - 1570, 2009.

[2] J. Nickolls and W. J. Dally, “The GPU Computing Era”, IEEE Micro, Vol. 30, No. 2, pp. 56-69, 2010.

[3] J. E. Stone, D. Gohara, and G. Shi, “OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems”, IEEE Computing in Science & Engineering, Vol. 12, No. 3, pp. 66-73, 2010.

[4] W. Sheng, P. Szymanski, R. Leupers, and G. Ascheid, “Software Migration for Parallel Execution on a Multicore Tablet: A Case Study”, The IEEE 7th International Symposium on Embedded Multicore SoC, pp. 26-28, Sep. 2012.

[5] R. Ubal, J. Sahuquillo, S. Petit, and P. Lòpez “Multi2Sim: a Simulation Framework for CPU-GPU Computing”, in Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques, pp. 335-344, Sep.

2012.

[6] V. Zakharenko, T. Aamodt, and A. Moshovos, “Characterizing the performance benefits of fused CPU/GPU systems using FusionSim”, IEEE in Proc. DATE, pp. 685-688, Mar. 2013.

[7] H. Wang, V. Sathish, R. Singh, M. Schulte, and N. Kim, “Workload and Power Budget Partitioning for Single-Chip Heterogeneous Processors”, IEEE/ACM Int. Conf. on Parallel Architecture and Compilation Techniques (PACT), Sep. 2012.

[8] S. Raghav, C. Pinto, M. Ruggiero, A. Marongiu, D. Atienza, and L. Benini, “Full System Simulation of Many-Core Heterogeneous SoCs using GPU and QEMU Semihosting”, In Proceedings of ACM Workshop on General Purpose Processing with Graphics Processing Units, pp. 101-109, Mar. 2012.

[9] M. T. Yourst, "PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator", IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 23-34, 2007.

[10] S.-T. Shen, S.-Y. Lee, and C.-H. Chen, “Full System Simulation with QEMU: an Approach to Multi-view 3D GPU Design”, IEEE Int. Symp. on Circuits and Systems (ISCAS), pp. 3877-3880, May, 2010.

[11] S. Collange, M. Daumas, D. Defour, and D. Parello, “Barra: a Parallel Functional Simulator for GPGPU”, IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 351-360, Aug. 2010.

[12] F. Bellard, “QEMU, a Fast and Portable Dynamic Translator”, in Proceedings of USENIX Annual Technical Conference, pp. 41-46, June, 2005.

[13] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong and T. M. Aamodt, “Analyzing CUDA Workloads Using a Detailed GPU Simulator”, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 163-174, Apr. 2009.

황 재 민 (Jaemin Hwang)

- 2014년 충남대학교 메카트로닉스공학 학사 졸업 (컴퓨터공학 복수전공)
- 2014년～현재 충남대학교 컴퓨터공학 석사과정
- 관심분야 : 플랫폼 소프트웨어, SoC 설계

최 종 욱 (Jong-Wook Choi)

- 1999년 경북대학교 전자공학 학사 졸업
- 2001년 경북대학교 전자공학 석사 졸업
- 2000년～현재 한국항공우주 연구원 위성비행소프트웨어 팀
- 관심분야 : 시뮬레이터, 실시간운영체제

최 성 림 (Seongrim Choi)

- 2011년 충남대학교 컴퓨터공학 학사 졸업
- 2013년 충남대학교 컴퓨터공학 석사 졸업
- 2013년～현재 충남대학교 컴퓨터공학 박사과정
- 관심분야 : 모바일 GPU, 디지털 연산기, SoC 설계

남 병 규 (Byeong-Gyu Nam)

- 1999년 경북대학교 컴퓨터공학 학사 졸업
- 2001년 KAIST 전자전산학 석사 졸업
- 2007년 KAIST 전자전산학 박사 졸업
- 2001년～2002년 ETRI 컴퓨터시스템 연구부 연구원
- 2007년～2010년 삼성전자 SystemLSI 사업부 책임연구원
- 2010년～현재 충남대학교 컴퓨터공학과 교수
- 관심분야 : 모바일 GPU, 임베디드 CPU, 저전력 SoC 설계, 임베디드 SW 플랫폼

논 문 접 수 일 : 2014년 03월 07일
1 차 수 정 완 료 일 : 2014년 03월 28일
2 차 수 정 완 료 일 : 2014년 04월 11일
게 재 확 정 일 : 2014년 04월 15일