CUDA-to-x86 명령어 변환도구의 성능측정 및 분석

박용, 신동군
성균관대학교 정보통신공학부
e-mail : *pangol@skku.edu, dongkun@skku.edu*

Evaluation and Analysis of the translation tool converting CUDA code to x86 instructions

Yong Park, DongKun Shin
School of Information and Communication Engineering
Sungkyunkwan University

Abstract

We evaluate the translation tool, Ocelot, which converts CUDA code to x86 instructions. The performance of the translated program is compare with that of the program written in OpenMP. The reason for low performance of the translated program results from unequal distribution of task and initialization.

I. 서론

오늘날 싱글코어 프로세서의 한계를 극복하기 위해서 멀티코어 프로세서의 개발이 지속되고 있으며 이러한 멀티코어 프로세서를 이용하기 위해서 여러 종류의 프로그래밍 모델이 제시되고 있다.

하지만 병렬 프로그래밍을 위해서 제공되고 있는 프로그래밍 모델들은 하드웨어에 종속적이기 때문에 서로 다른 시스템에서는 다른 프로그래밍 모델을 사용해야 하는 문제점이 있다.

이러한 문제점을 개선하고자 서로 다른 프로그래밍 모델로 작성이 된 프로그램을 변환하여 시스템에 독립적으로 실행 시킬 수 있는 기법이 제시되고 있다.

본 논문에서는 이러한 기법 중에서 Ocelot[1]을 통해서 변환된 프로그램의 성능을 OpenMP[2]와 비교하고 분석한다.

II. 본론

2.1 OpenMP

OpenMP는 공유메모리 환경에서 병렬 프로그래밍을 지원하는 API (Application Program Interface) 이다.

2.2 CUDA

CUDA[3]는 컴퓨터 그래픽스를 위한 계산만 다루는 그래픽 처리 장치를 CPU가 전통적으로 취급했던 응용 프로그램들의 계산을 수행할 수 있게 하는 프로그래밍 모델이다.

2.3 Ocelot

Ocelot은 이기종 시스템 환경을 위한 동적 컴파일 프레임워크이다. Ocelot은 현재 CUDA 프로그램을 그래픽 처리 장치와 x86 CPU에서 재 컴파일 필요 없이 실행 가능하게 변환을 해준다.

Ocelot에서는 프로그램 변환 시에 하드웨어특징에 따른 초기화 작업과 쓰레드의 수를 조절하는 쓰레드

융합, 그리고 쓰레드별로 일을 분배하는 작업을 수행한다.

Ⅲ. 실험

3.1 실험환경

실험에는 Intel Core i5-750을 사용했으며 벤치마크는 Rodinia[4]를 사용하였고 그 중에서 Back Propagation, HeartWall, Needleman-Wunsh, SRAD을 사용하였다.

실험은 OpenMP 프로그램의 실행시간과 Ocelot을 통해서 변환된 프로그램의 실행시간을 비교하였다.

3.2 실험결과

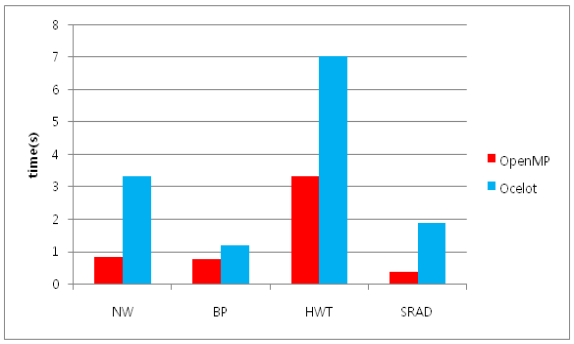

그림 1. OpenMP 와 Ocelot의 실행시간 측정

그림 1을 통해서 확인할 수 있듯이 Ocelot을 통해서 변환된 프로그램을 실행시켰을 경우에 OpenMP로 작성된 프로그램보다 실행시간이 적게는 1.5배 많게는 4배 오래 걸리는 것을 확인할 수가 있다.

더 자세한 분석을 위해 프로그램의 실행구간을 순차적인 구간과 병렬적인 구간으로 나눠서 시간을 측정하였다.

벤치마크	순차구간		병렬구간	
	OpenMP	Ocelot	OpenMP	Ocelot
BP	0.42	0.1	0.345	1.086
HWT	0.1	2.93	3.22	4.081
NW	0.3	1.1	0.527	2.221
SRAD	0.62	0.75	0.075	1.137

표1. 구간 별 OpenMP와 Ocelot 실행시간 (단위 :초)

표 1에서 보면 Back Propagation (BP) 을 제외한 프로그램들의 두 구간에서 성능이 악화되는 것을 확인할 수 있다. 순차적인 구간에서 성능저하의 이유는 프로그램을 처음 실행 할 때, 메인 쓰레드가 각 쓰레드별로 일을 분배하는 작업으로 인해 생기는 시간과 CUDA에서 설정된 쓰레드의 수와 CPU에서 사용할 수 있는 쓰레드의 수의 차이로 인해 생기는 문제점을 해결하려고 할 때 발생되는 것으로 OpenMP에서 생기지 않는 쓰레드들의 조인 때문이다.

병렬구간에서 성능저하의 이유는 쓰레드별로 할당이 균등하게 일어나지 않아 이미 수행을 끝낸 쓰레드가 다른 쓰레드들을 기다리게 되는 시간 때문에 발생한다. BP의 경우에는 순차적인 구간에서 걸리는 시간이 적은데 그 이유는 초기화에 드는 시간을 절약할 수 있었기 때문이며 따라서 성능저하가 적게 일어날 수 있었다.

Ⅳ. 결론 및 향후 연구 방향

OpenMP로 작성된 프로그램과 Ocelot을 이용하여 변형한 프로그램의 성능을 비교하였으며 최대 4배의 성능저하를 확인할 수 있었다.

이러한 성능저하의 원인은 변환된 프로그램에서 초기화에 드는 시간과 OpenMP에서는 없었던 쓰레드들의 조인현상으로 발생하였으며 각 쓰레드별 할당이 불균형적으로 분배된 것에 기인한다.

CUDA에서 x86프로그램으로 변환할 때는 쓰레드별로 일을 균등하게 분배하는 것과 초기화에 걸리는 시간을 줄이는 것이 필요하다는 것을 확인할 수 있었다.

참고문헌

[1] Gregory Diamos et al., Ocelot: A Dynamic Optimization Framework for Bulk-Synchronous Applications in Heterogeneous Systems, PACT'10, September, 2010.

[2] Leonardo Dagun et al., OpenMP: An Industry-Standard API for Shared-Memory Programming, IEEE Computational Science & Engineering, v.5 n.1.p.46-55, January 1998.

[3] John Nickolls et al., Scalable Parallel Programming with CUDA, Queue, v.6 n.2, March/April 2008.

[4] Shuai Che et al., Rodinia: A Benchmark Suite for Heterogeneous Computing, IEEE IISWC, October, 2009