Cassiano Kleinert Casagrande
Dr. B. Mora

Prifysgol Abertawe
Swansea University

Implementing a Scheduler on GPU using OpenCL

Introduction

Graphics Processors can now be used as a computational platform. The parallel nature of modern Graphics Processing Units added to the fact that almost every modern computer has a GPU make them very useful on solving data-parallel problems. GPUs nowadays have more transistors then CPUs and most of those transistors are employed on more Single Instruction Multiple Data(SIMD) units, while on CPUs almost two thirds of those transistors are used to implement cache memory and cache logic.

The goal of this project was to implement a scheduler on the GPU memory, each GPU working group will retrieve tasks from the scheduler and will process them. By using using this task scheduler there will be less need to transfer data between the CPU and the GPU, the number of kernel calls and idle working groups will also be lower.

Scheduler

The scheduler stores user defined tasks in the GPU memory. Instead of retrieving the tasks from deterministic places, like a stack or queue, the scheduler tries to retrieve tasks from a random place on the array queue. This way, it avoids busy waiting caused by several working groups trying to access the same memory region to retrieve a task.

The insertion of tasks is also made randomically searching for a empty place on the task array to ensure that all idle working groups do not try to retrieve tasks from the same place.

Besides storing the task data in the array, the scheduler also has fields to store, for each task, a mutex to control access to this task, the current state of the task and the index to the position of the parent task in the array.

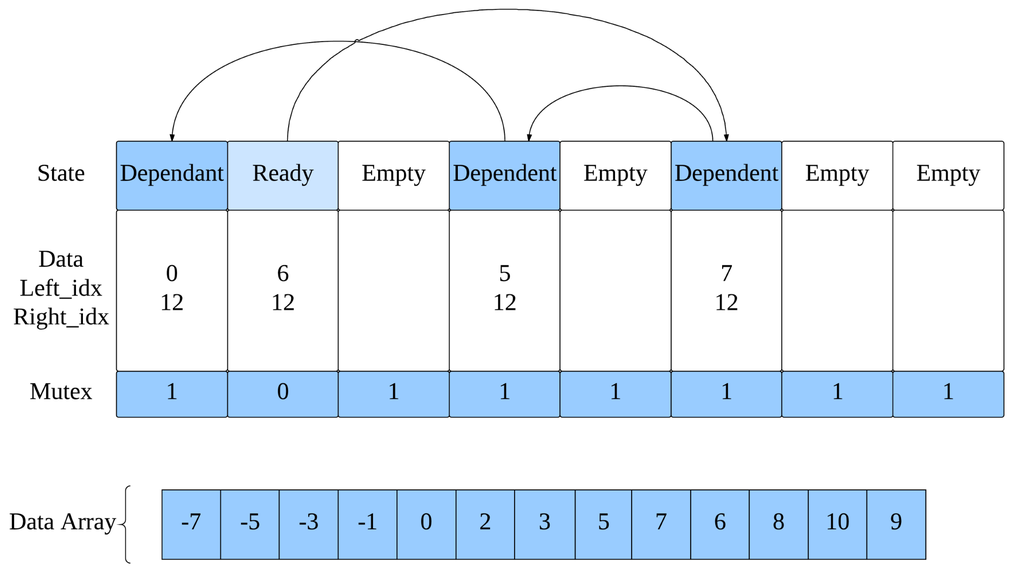

Figure 1: The scheduler task array applied to sorting an array.

Results

After the implementation, the library was used to sort an array using the quick sort algorithm as a proof of concept. In this use, each task contained data of the left-most and right-most elements of sub-arrays that need to be partitioned, the sub-arrays that are on the right and left of the pivot position are then added as tasks to be processed.

If a sub-array retrieved from the task pool is smaller then a given limit, the array is sorted using the method insertion sort.

Bibliography

[1] M. Vinkler, J. Bittner, V. Havran, and M. Hapala, "Massively Parallel Hierarchical Scene Processing with Applications in Rendering", presented at Comput. Graph. Forum, 2013, pp. 13-25.

[2] Ryoji Tsuchiyama, Takashi Nakamura, Takuro Iizuka, Akihiro Asahara, Jeongdo Son, Satoshi Miki (2012). The OpenLC Programming Book. Japan: Fixstars.

[3] KHRONOS Group. (2007). OpenCL 2.1 Reference Pages. Available at: http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/. Last accessed 31th Aug 2014

September 4, 2014

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico