Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 63 (2017) 125–135 www.elsevier.com/locate/endm

Utilization of OpenCL for Large Graph Problems on Graphics Processing Unit

Vinod Kumar Mishra[1] Pankaj Singh Sammal[2]

Department of Computer Science and Engineering
Bipin Tripathi Kumaon Institute of Technology
Dwarahat, Almora, Uttarakhand, India

Abstract

During recent years graphics processing unit (GPU) has become an inexpensive high-performance parallel computing units and general purpose computation on graphics processing unit (GPGPU) become an alternative of conventional CPU computation, which provide massive parallel computation capability to a normal computer over a system which uses only CPU. This paper investigates the performance of the algorithm for the solution of single source shortest path (SSSP) problem, all pair shortest path (APSP) problem and graph coloring problem for large graph on GPU regardless of a specific vendor. The application programming interface (API) used for programming in graphic processing unit is open compute language (OpenCL), it is a specification for heterogeneous computing. The performance of solutions on GPU is compared with the solution on CPU in term of their execution time. The result indicates the significant improvement in the performance of the solutions on GPU over solutions on CPU.

Keywords: Graphics processing unit (GPU); OpenCL; Single source shortest path; All pair shortest path; Graph coloring.

[1] Email: vkmishra2005@gmail.com
[2] Email: pankajsammal29@gmail.com

https://doi.org/10.1016/j.endm.2017.11.007

1 Introduction

Graph representation is common for representing many applications using graphs in various domains including scientific and engineering domain. Basic graph operations such as breath-first search, depth-first search, shortest path etc. are used frequently in these domains. Larger problems need to be mapped over a very large graph consisting of millions of vertices. For example, chip layout, data mining and network analysis representation can consist of a graph with millions of vertices. There are many more applications of graph in the field of physics, chemistry, communication science, computer science, and apart from these areas, many practical problems can also be solved with the help of the graph theoretic algorithm. Graphs can have millions of nodes depending on their uses, which can be hard to calculate using normal methods over CPU. There is a need to optimize the solutions of these graph problems so that they can perform better and execute faster. Basic algorithms are available to solve these problems but these are impractical for use over very large graph consisting of millions of vertices. Parallel version of these algorithms can be used to solve these problems in a practical time but the hardware cost of parallel system is too high. GPU has become a cost-effective platform for parallel computation and can be used to solve many general problems including in the areas of image processing, signal processing, computer vision, dynamic programming etc. Initially GPUs were optimized for graphic processing only and had a very restricted programming model but this all changed in 2006 when GPUs with Shader Model 4.0 was launched, these GPU had a unified architecture for all processors which allowed them to be used in more flexible ways than its predecessors. Nvidia launched its Compute Unified Device Architecture (CUDA) to allow programmers access to underlying parallel processor. AMD also launched its own language of its own GPUs known as AMD Steam. Both CUDA and AMD Steam are incompatible with each other and have to be programmed in their proprietary language. There exists an open source alternative to these proprietary languages in form of Open Compute Language (OpenCL). OpenCL provides an open computing API providing GPU and other co-processors to work in tandem with the CPU. The major advantage of OpenCL for GPGPU computation is substantial acceleration in parallel processing. OpenCL takes advantage of all supported resources such as multi-core CPUs and GPUs and other OpenCL compatible devices and divide the workload across all devices. OpenCL also support cross vendor software portability as there is a low lever layer which separates the hardware and upper software layer. Execution time of basic algorithm to solve