Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 22 (2017) 409–418

Transportation Research Procedia

www.elsevier.com/locate/procedia

19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey

Accelerating local search algorithms for the travelling salesman problem through the effective use of GPU

Gizem Ermiş[*], Bülent Çatay

Sabanci University, Faculty of Engineering and Natural Sciences, Tuzla, Istanbul 34956, Turkey

Abstract

Graphics processor units (GPUs) are many-core processors that perform better than central processing units (CPUs) on data parallel, throughput-oriented applications with intense arithmetic operations. Thus, they can considerably reduce the execution time of the algorithms by performing a wide range of calculations in a parallel manner. On the other hand, imprecise usage of GPU may cause significant loss in the performance. This study examines the impact of GPU resource allocations on the GPU performance. Our aim is to provide insights about parallelization strategies in CUDA and to propose strategies for utilizing GPU resources effectively. We investigate the parallelization of 2-opt and 3-opt local search heuristics for solving the travelling salesman problem. We perform an extensive experimental study on different instances of various sizes and attempt to determine an effective setting which accelerates the computation time the most. We also compare the performance of the GPU against that of the CPU. In addition, we revise the 3-opt implementation strategy presented in the literature for parallelization.

Keywords: GPU computing; parallelization; optimization; GPU architecture; travelling salesperson problem.

1. Introduction

With their highly parallel structure, graphics processor units (GPUs) are many-core processors that are specifically designed to perform data-parallel computation. Because of the architectural differences between the

[*] Corresponding author. Tel.: +90-537-981-3498
*E-mail address:*ermisgizem@sabanciuniv.edu

central processing units (CPUs) and GPUs, the GPU performance is better on data parallel, throughput-oriented applications with intense arithmetic operations. GPUs have the capability to accelerate algorithms that require high computational power. Thus, they can considerably reduce the execution time of such algorithms by performing a wide range of calculations in a parallel manner.

Modern GPUs are many-core processors that are specifically designed to perform data-parallel computation. Data parallelism means that each processor performs the same task on different pieces of distributed data (Brodtkorb et al., 2013). Before the evolution of today's advanced GPUs, traditional, single-core processors were exploited. Computationally hard tasks were taking a great deal of time when they were solved by the help of single-core processors. Faster single-core processors were developed by the computer industry but they were still insufficient for peak performances. Around the year 2000, by fitting more cores in the same chip, single-core processors evolved to multi-core processors (Fig. 1), which work together to process instructions, and thus have higher total theoretical performance (Brodtkorb et al., 2013).

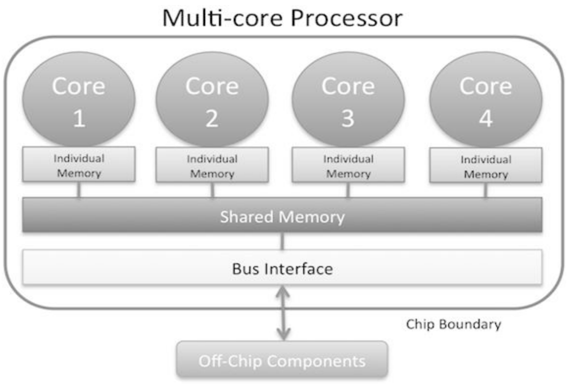

Fig 1. A basic block diagram of a generic multi-core processor

Because of gaming industry needs, GPUs which actually were the normal component in common PCs, developed quickly in terms of computational performance. Multi-core GPU processors evolved to massive multi-core or many-core processors which work as massively parallel stream processing accelerators or data parallel accelerators. Because of the rapid advancements in the GPU technology, they became common as accelerators in general purpose programming. Although both multi-core CPUs and GPUs can implement parallel algorithms, the architectural differences between CPUs and GPUs created different usage areas depending upon the nature of the problem. While multi-core CPUs are designed for task parallel implementations, many-core processors are specifically designed for data parallel implementations.

The efficiency is as much critical factor as the solution quality when an algorithm is applied to an optimization problem such as traveling salesman problem (TSP), a well-known NP-Hard combinatorial optimization problem. Local search algorithms such as 2-opt or 3-opt are computationally difficult when they are implemented on the CPU. Because these techniques evaluate all edge exchanges on the tour to determine the exchange that reduces the tour length the most, they require a large number of computations and comparisons. So, parallel implementation can accelerate these computations significantly. GPUs which have data parallel structure can perform these simple computations in parallel. On the other hand, the design of the parallel implementation plays a crucial role in achieving effective utilization of the GPU resources, thus optimizing the system performance.

CUDA is a parallel computing platform and programming model introduced by NVIDIA. It enables programmers to use GPUs for general purpose processing (Wikipedia, 2016). Van Luong et al. (2009) used GPU as a coprocessor for extensive computations where the solutions of the TSP from a given 2-exchange neighborhood are evaluated in parallel. The remaining computations are performed on the CPU.

A local search has four main steps: neighborhood generation, evaluation, move selection, and solution update. The simplest method is to create the neighborhood on the CPU and transfer it to GPU each time. Van Luong et al. (2013) applied this technique; however, it requires copying of a lot of information from the CPU to the GPU. To prevent this drawback Rocki and Suda (2012) and Schulz (2013) utilized an explicit formula to explore the