Using Inter-Block Synchronization to Improve the Knapsack Problem on GPUs

Xue Sun, College of Urban Rail Transit and Logistics, Beijing Union University, Beijing, China & Department of Electrical Engineering, National Changhua University of Education, Changhua, Taiwan

Chao-Chin Wu, Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua, Taiwan

Liang-Rui Chen, Department of Electrical Engineering, National Changhua University of Education, Changhua, Taiwan

Jian-You Lin, Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua, Taiwan

ABSTRACT

This article describes how as one of the hot parallel processors, the general-purpose graphics processing unit (GPU) has been widely adopted to accelerate various time-consuming algorithms. Dynamic programming (DP) optimization is a popular method to solve a particular class of complex problems. This article focuses on serial-monadic DP problems onto NVIDIA GPUs. As 0/1 knapsack is one of the most representational problems in this category and it often arises in many other fields of applications. The previous work proposed the compression method to reduce the amount of data transferred, but data in shared memory cannot be reused. This article demonstrates how to apply a more condensed data structure and the inter-block synchronization to efficiently map the serial-monadic DP onto GPUs. Computational experiments reveal that the best performance improvement of the approach is about 100% comparing with the previous work.

KEYWORDS

0/1 Knapsack Problem, Barrier, CUDA, Dynamic Programming, GPU, Inter-Block Synchronization, Parallel Computing, Synchronization

INTRODUCTION

Graphics Processing Units (GPUs) have become a powerful tool for high-performance computing in recent years. As specialized computer processors, GPUs can manipulate large amounts of data efficiently by highly parallel multi-core system. For providing low-level control of GPUs, NVIDIA has introduced CUDA (Compute Unified Device Architecture) to ease programming for various types of parallel applications (NVIDIA GPU, 2015). The CUDA platform, which can be designed to write programs with C, C++, FORTRAN, OpenCL and other languages, can access to virtual instruction set of GPUs and parallel computational elements directly with the execution of compute shaders (Abi-Chahla, 2015). Using CUDA GPUs to solve large-scale or complex optimization problems is a big challenge and hot issues in research.

Dynamic programming (DP), also known as dynamic optimization, is a popular method to solve a particular class of complex problems. DP method breaks up a problem into a collection of smaller sub-problems, then solves each of them only once and saves the results for future reference so as to avoid recomputation. DP can be classified into four categories: serial-monadic, non-serial-monadic,
serial-polyadic and non-serial-polyadic (Grama, Gupta, Karypis, Kumar, 2003). In our previously research, we have studied how to use CUDA GPUs to accelerate the solutions of non-serial-polyadic DP optimization problems, where we chose Optimal Matrix Parenthesization (OMP) problem as an example to study (Wu, Ke, Lin, and Feng, 2011). In this work, we focus on serial-monadic DP problems onto NVIDIA GPUs. As 0/1 knapsack is one of the most representational problems in this category and it often arises in many other fields of applications, we adopted 0/1 knapsack problem for this study.

0/1 knapsack problem is known as a NP-hard problem. For high time complexity, speeding up the solution of the problems has attracted concern of researchers (Boyer, El Baz, and Elkhel, 2012; Goldman and Trystram, 2004; El Baz and Elkhel, 2005; El Baz and M. E. D., 2006; Li, and Liu, 2015; Hajarian, Shahbahrani, and Hoseini, 2016). To parallel DP approach for 0/1 knapsack problem, recently Boyer and his colleagues proposed a compression mechanism to implement it on CUDA GPUs (Boyer et al., 2012). The method mainly used the item selection table, which is saved as a 2-dimentional (2D) array, to record if an item is selected or not for each capacity. Assuming the maximum capacity of the knapsack is c, and there are n items to be prepared for putting into the knapsack, so the dimension of the 2D array is \( n \times c \). If \( n \) and \( c \) are large integers, the 2D array needs a large amount of global memory space. To address the problem, they used a data compression technique which is called the row-compression method to reduce significantly memory occupancy and limit the communication between the CPU and the GPU.

However, we observed that the method of Boyer has two disadvantages. First, although the compression method can reduce the amount of data transmission and the memory occupancy effectively, if the item table is one-dimensional (1D) array instead of 2D, the data can be minimized more significantly. Second, that Boyer’s method invokes the same kernel one time for each stage of the DP problem that cannot reuse the data in shared memory. To address these problems, we applied a more condensed data structure and provided the inter-block synchronization to improve the parallelism degree.

The paper is organized as follows. Section 2 introduces the related work about the 0/1 knapsack problem and GPU parallelization. Section 3 proposes our GPU solution includes 1D matrix solution and Wait and Signal synchronization. Section 4 demonstrates and analyzes the results of the experiment. Finally, conclusions are given in Section 5.

RELATED WORK

The 0/1 Knapsack Problem

The knapsack problem belonging to NP-hard combinational optimization is one of the simplest integer linear programming problems, which obtains extensive research. 0/1 knapsack problem as one type of knapsack problem has applied to plentiful applications such as capital budgeting (Taha, 1997), cryptographic schemes (Diffie and Hellman, 1997), stock cutting (Gilmore and Gomory, 1961), cargo loading (Kleywegt and Papastavrou, 2011), crew scheduling and assignment (Barnhart et al., 1998), batch processor scheduling ().

The description of 0/1 knapsack problem is as follows. Given a set of \( n \) items \( i \) to be packed into a knapsack of capacity \( c \), where \( c \) is a positive integer and \( 1 \leq i \leq n \). If item \( i \) have a weight \( w_i \in \mathbb{N}^+ \) and a profit \( p_i \in \mathbb{N}^+ \). The goal of the 0/1 knapsack problem is to choose a subset of items and maximize accumulated profit of the chosen items, moreover, the total (Papastavrou, Rajagopalan, and Kleywegt, 1996) weight cannot exceed the capacity of the knapsack. Let decision variables \( x_i (i = 1, 2, \ldots, n) \) represent whether the item \( i \) is selected or not, that is each item can appear at most once, and \( x_i \) is restricted to be the integer 0 or 1. The 0/1 knapsack problem is mathematically formulated the following integer linear programming:
A Trusted Data Storage Infrastructure for Grid-Based Medical Applications
www.igi-global.com/article/trusted-data-storage-infrastructure-grid/3962?camid=4v1a

Collaboration Support for Activity Management in a Personal Cloud Environment
www.igi-global.com/chapter/collaboration-support-activity-management-personal/72254?camid=4v1a