Chapter 9

Efficient Prefix Scan for the GPU-Based Implementation of Random Forest

Bojan Novak
UM FERI, Slovenia

ABSTRACT

The random forest ensemble learning with the Graphics Processing Unit (GPU) version of prefix scan method is presented. The efficiency of the implementation of the random forest algorithm depends critically on the scan (prefix sum) algorithm. The prefix scan is used in the depth-first implementation of optimal split point computation. Described are different implementations of the prefix scan algorithms. The speeds of the algorithms depend on three factors: the algorithm itself, which could be improved, the programming skills, and the compiler. In parallel environments, things are even more complicated and depend on the programmer’s knowledge of the Central Processing Unit (CPU) or the GPU architecture. An efficient parallel scan algorithm that avoids bank conflicts is crucial for the prefix scan implementation. In our tests, multicore CPU and GPU implementation based on NVIDIA’s CUDA is compared.

INTRODUCTION

In the first chapter the evolution of implementations of random forest algorithms that can be executed on the graphics processing unit (GPU) are described. Random forests are ensemble learning methods for classification that constructs a set of decision trees at training process (Breiman, 2001). The output class is calculated from the classes output by individual trees. Advantages of training algorithms that can produce compact random forests composed of many, small trees rather than fewer, deep trees form a solid foundation for the GPU implementation. Pros and contras of depth-first and breadth-first search are presented.

For better understanding of next chapters a short description of the GPU architecture is given. The GPU Architecture allows the possibility to program GPUs using a master-slave computing model (Knuth, 1974). PC’s central processing unit (CPU) works as the master and the GPU processors act as the slaves. The host invokes kernels (C...
functions) that run on slave processors. Kernels can be executed as threads.

In the third chapter theory and implementations of the prefix scan algorithm on the GPU is presented. The performances of prefix scan algorithms are measured by different criteria: computational complexity, memory usage, stability, recursion etc. (Cormen, Leiserson, Rivest, & Stein, 2001) and (Knuth, 1973). The speeds of the algorithms depend on three factors: the algorithm itself, which can be improved, the programming skills, and the compiler. In parallel environments things are even more complicated and depend on the programmer’s knowledge of the CPU or GPU unit (Govindaraju, Gray, Kumar, & Manocha, 2006). Particularly when the architecture of GPU changes, this happens almost every year, as programs are no longer optimal.

Follows the chapter about random forests ensemble learning with the GPU version of the prefix scan method. The efficiency of the implementation of the random forest algorithm depends critically on the scan (prefix sum) algorithm. The prefix scan is used in the depth-first implementation of optimal split point computation. Described are different implementations of the prefix scan algorithms. Follows some performance tests for multicore CPU and GPU implementations based on NVIDIA’s CUDA platform on the medium size data set MNIST and the large Poker hand dataset. The experiments were performed on the PC with Intel I7 920, 2.66GHz CPU, and NVIDIA GTX TITAN GPU.

In the next chapter some future direction of development of the proposed algorithm for the multi GPU environment is discussed when the new NVLink, a high-speed interconnect will be available.

In the last chapter a short summary of theory and benchmark results are presented. Follows some additional explanation of achieved results. Given are some advices of the proper use of the GPUs version of the algorithms.

**RELATED WORK**

Random forest classification is a machine learning technique that generates classification of an input sample by the majority classification by the ensemble of decision trees named as “forest”. One of the earliest paper on this subject was written by Sharp where a method for implementing the evaluation and training of decision trees and forests entirely on a GPU, and it was shown how this method can be used in the context of object recognition in the computer vision (Sharp, 2008). Traditional random forest classifiers can be highly effective, but classification using a random forest is memory bound and not typically suitable for acceleration using field-programmable gate array FPGAs or GP-GPUs due to the need to traverse large, possibly irregular decision trees. At Lawrence Livermore National Laboratory researchers had developed several variants of random forest classifiers, including the Compact Random Forest (CRF), that can generate decision trees more suitable for acceleration than traditional decision trees (Van Essen, Macaraeg, Gokhale, & Prenger, 2012). They made a comparison of the effectiveness of FPGAs, GP-GPUs, and multi-core CPUs for accelerating classification, using models generated by compact random forest machine learning classifiers. Taking advantage of training algorithms that can produce compact random forests composed of many, small trees rather than fewer, deep trees. But the depth-first algorithm is efficient during the early stages of tree construction, when large numbers of examples are being processed. As trees become deeper, however, the overhead of invoking GPU kernels to evaluate small numbers of samples becomes dominant (Yisheng & Rubinsteyn, 2013). Breadth-first construction is less efficient at the top of a tree and decreases the kernel launch overhead significantly by processing many sites on a tree at the same time. The strengths of both strategies are combined into a hybrid tree construction algorithm. This strategy starts tree