学 位 論 文 要 旨

題 目 A Study on Efficient GPU Implementations for Many Small Problems
(大量の小さな問題のための効率的な GPU 実装に関する研究)

氏 名　　戸倉 宏樹

A GPU (Graphics Processing Unit) is a specialized circuit designed to accelerate computation for building and manipulating images. Latest GPUs are designed for general purpose computing and can perform computation in applications traditionally handled by the CPU. So，GPUs have recently attracted the attention of many application developers. Some applications need to compute many instances. The computation of Summed Area Table(SAT) is needed many prefix-sum computations. We can obtain SAT by computing column-wise prefix-sums after computing row-wise prefix-sums. However each column or row has several thousands since the size of input image is fullHD or 4K. It is too small for the GPU to efficiently compute the prefix-sum because the GPU has over a thousand cores. Also, in the column-wise prefix-sum computation, each element in a column is stored in the non-consecutive location of the memory. So, the efficiency of memory access is low. In control design problems, the computation of large number of the small eigenvalue problem is necessary. However it is too small for the GPU to efficiently compute the small eigenvalue problem. Also, the algorithm of the eigenvalues computation is complicated. So, the efficiency of computation is low. Namely, GPU can not efficiently compute the column-wise prefix-sums or large number of the small eigenvalue problem due to programming issues.

In Chapter 1, we describe the introduction of the dissertation, the research background and our contributions. Chapter 2 describes GPU architecture and CUDA. CUDA which is provided by NVIDIA is a parallel computing architecture.

In chapter 3, we present *the Look-back Column-wise Prefix-sum (LCP) algorithm*, which computes the column-wise prefix-sums of a matrix very efficiently on the GPU. It partitions the matrix into small tiles and the column-wise sums and prefix-sums of every tile are computed using one CUDA block for each tile in parallel. The LCP algorithm does not perform stride access to the global memory, shared

memory access with bank conflicts, or separated kernel calls for global synchronization, which involve large overhead. Clearly, no GPU implementation of column-wise prefix-sum computation of an $n \times n$ matrix can be faster than matrix duplication, in which n^2 elements are read and written. Thus, we can say that a column-wise prefix-sum algorithm is optimal if the computing time is equal to matrix duplication. Quite surprisingly, the experimental results on NVIDIA TITAN X GPU show that our LCP algorithm runs only 2-6% slower than matrix duplication. Thus, our LCP algorithm is almost optimal.

In chapter 4, we propose a GPU implementation of bulk computation of eigenvalues of small, non-symmetric, real matrices of maximum size 30×30. In our GPU implementation, we considered programming issues of the GPU architecture including warp divergence, coalesced access of the global memory, utilization of the shared memory, and so forth. We focused on the thread assignment to obtain the optimal parallel execution with many threads on the GPU. In our GPU implementation, the optimal parameters have been obtained by evaluating the computation time for various parameters. Also, to improve the memory access efficiency, we introduce memory arrangements in the device memory on the GPU for each of the thread assignments. Furthermore, to hide CPU-GPU data transfer latency, overlapping computation on the GPU with the transfer is employed. We evaluated the performance of computing eigenvalues of 500000 matrices of size 5×5 to 30×30. The experimental results on NVIDIA TITAN X show that our GPU implementation attains a speed-up factor of up to 83.50 and 17.67 over the sequential CPU implementation and the parallel CPU implementation with eight threads on Intel Core i7-6700K, respectively.

In chapter 5, we conclude our works.