CSPR: *Column* Only *SPAR*SE Matrix Representation for Performance Improvement on GPU Architecture

B. Neelima and Prakash S. Raghavendra

Department of Information Technology,
NITK Surathkal, Mangalore, Karnataka, India, 575 025
reddy_neelima@yahoo.com, srp@nitk.ac.in

Abstract. General purpose computation on graphics processing unit (GPU) is prominent in the high performance computing era of this time. Porting or accelerating the data parallel applications onto GPU gives the default performance improvement because of the increased computational units. Better performances can be seen if application specific fine tuning is done with respect to the architecture under consideration. One such very widely used computation intensive kernel is sparse matrix vector multiplication (SPMV) in sparse matrix based applications. Most of the existing data format representations of sparse matrix are developed with respect to the central processing unit (CPU) or multi cores. This paper gives a new format for sparse matrix representation with respect to graphics processor architecture that can give 2x to 5x performance improvement compared to CSR (compressed row format), 2x to 54x performance improvement with respect to COO (coordinate format) and 3x to 10 x improvement compared to CSR vector format for the class of application that fit for the proposed new format. It also gives 10% to 133% improvements in memory transfer (of only access information of sparse matrix) between CPU and GPU. This paper gives the details of the new format and its requirement with complete experimentation details and results of comparison.

Keywords: GPU, CPU, SPMV, CSR, COO, CSR-vector.

1 Introduction

Graphics processing unit was tricked by the programmer to do general purpose computation than doing only graphics related operations. The motivation behind the development of graphics processor evolution, to general purpose computation processor, is different than that of the CPU evolution, to multi core. Hence data formatting and optimizations designed with respect to CPU and its evolutions have to be tailored to GPU specific architectures. Even though GPU gives better performance of the accelerated applications than CPU and multi core, full utilization of the processor for much better performance is possible by tailor made data formatting and computations with respect to the architecture under consideration. Sparse matrix computations and usage is very large in most of the scientific and engineering applications. In sparse matrix, sparse vector multiplication is of singular importance in wide applications. This paper concentrates on sparse matrix vector multiplication aspect of compute intensive applications and

D. Nagamalai, E. Renault, and M. Dhanushkodi (Eds.): PDCTA 2011, CCIS 203, pp. 581–595, 2011.

through a new format shows the memory transfer and performance improvements than the existing data formats of sparse matrices. The results shown for proposed new data format are applicable to GPU in general but the results are particular to NVIDIA GPU analyzed on Geforce GT 525M.

Optimizing performance on GPU needs creation of thousands of threads, because it uses latency hiding by using thousands of threads and gives high throughput. Few of the existing methods like CSR, use row wise thread creation that cannot use global coalescing feature of GPU and GPU is underutilized if the number of non-zero elements per row is less than 32, the size of a warp. CSR vector is modified version of CSR that benefits from global coalescing by using fragmented reductions. The proposed CSPR (Column only SPaRse format) reduces the sparse matrix vector multiplication to constant time and threads can be launched continuously by parallelizing the outer loop for creating many threads. CSPR can be applied to any sparse matrix in general but better performances are seen for the matrices with large number of rows with minimum number of non-zero values per row and centrally distributed few dense rows as shown in Fig. 3. For such matrices, it can give 2x to 54x performance improvements compared to CSR, COO and CSR vector format. CSPR embeds the row information into column information and uses a single data structure; hence it can also optimize the memory transfer between CPU and GPU. CSPR format uses only one data structure to access the sparse matrix hence it is a good format for the internally bandwidth limited processors like GPU.

The paper is organized as follows. The next section gives the details of GPU architecture in general and CUDA in particular. Section III gives the sparse matrix introduction and its importance in scientific computation along with the introduction to data formats of sparse matrices. Section IV gives related work with respect to data formats and sparse matrices. Section V gives the working set up and introduction to sparse matrices considered for testing the new format. Section VI gives the experimental results and analysis. Section VII gives the conclusions and future work.

2 GPU Architecture

GPU is the co-processor on the desktop. It is connected to host environment via peripheral component interconnect (PCI Express 16E) to communicate with the CPU. The GPU used for the experimentation here is NVIDIA Geforce GT 525M, but the format proposed is in general applicable to all types of sparse matrices and all processor architectures including CPU. The proposed format is better suited and gives better performance on latency hiding based throughput oriented processors like GPU for specific class of sparse matrix structure. The third generation NVIDIA GPU has 32 CUDA cores in one SM (Streaming Multiprocessor). It supports double precision floating point operations. NVIDIA GPU has compute unified device architecture that uses the unified pipeline concept and the latest GPU supports up to 30000 co-resident threads at any point of time. GPU uses latency hiding to increase parallelism that is when active threads are running other threads will finish pending loads and become active to execute. It uses single instruction multiple threads concepts (SIMT) and executes the computation in warps that consists of 32 threads [1-3].