Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A parallel computing method using blocked format with optimal partitioning for SpMV on GPU

Wangdong Yang [a,b,*], Kenli Li [a,c,*], Keqin Li [a,d]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China
[b] College of Information Science and Engineering, Hunan City University, Yiyang, Hunan 413000, China
[c] The National Supercomputing Center in Changsha, Changsha, Hunan 410082, China
[d] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 3 January 2016
Received in revised form 9 June 2017
Accepted 26 September 2017
Available online xxxx

Keywords:
Blocked format
CPU/GPU
Dynamic programming
Heterogeneous parallel computing
Partitioning
Reordering
Sparse matrix–vector multiplication

ABSTRACT

For large-scale sparse matrices, SpMV cannot be processed on GPU using the common storage formats because of the memory limitation. In addition, the parallel effect is poor using general formats for the sparse matrices with extremely uneven distribution of non-zero elements, which leads to performance deterioration. This paper presents an optimal partitioning strategy based on the distribution of non-zero elements in a sparse matrix to improve the performance of SpMV, and uses a hybrid format, which mixes CSR and ELL formats, to store the blocks partitioned from the sparse matrix. The hybrid blocked format has better compression effect and more uniform distribution of non-zero elements, which can be suitable for more types of sparse matrices. Our partitioning strategy is proven to be optimal, which can yield the minimum parallel execution time on GPU.

1. Introduction

1.1. Motivation

In recent years, accelerator-based computing using accelerators, such as the IBM Cell synergistic processing unit (SPU), field programmable gate array (FPGA), graphics processing unit (GPU), and application specific integrated circuit (ASIC), has achieved clear performance gains compared to CPUs. Among the accelerators, GPUs have occupied a prominent place due to their low cost and high performance-per-watt ratio along with powerful programming models. However, as CPU architectures also evolve and address challenges such as the power wall and the memory wall, and compete with these accelerators, it is imperative that CPUs should also be included in computations. It is further observed by [1] that several irregular applications such as sparse matrix–vector multiplication (SpMV) can benefit from heterogeneous algorithms that run on a CPU and GPU based heterogeneous computing platform.

SpMV is an essential operation in solving linear systems and partial differential equations. SpMV faces two challenges, which are large scales and irregular distributions of non-zero elements. With the increasing scale of sparse matrices, the sparse matrix of SpMV cannot be loaded into the GPU once to be computed. So a large-scale sparse matrix must be split into some submatrices to be computed separately. It is a challenging issue to adopt an appropriate method to split a

* Corresponding authors.
E-mail addresses: yangwangdong@163.com (W. Yang), lkl@hnu.edu.cn (K. Li).

https://doi.org/10.1016/j.jcss.2017.09.010
0022-0000/

Please cite this article in press as: W. Yang et al., A parallel computing method using blocked format with optimal partitioning for SpMV on GPU, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2017.09.010

sparse matrix. In addition, load imbalance of parallel computing on GPU are generated, because of irregular distribution of non-zero elements in a sparse matrix, which leads to parallel efficiency decrease. The ELL format is a regular compression format for a sparse matrix, which has the same length of rows, and can avoid load imbalance. But the significant difference between the numbers of non-zero elements in rows will lead to more filling of zeroes. So those rows with similar numbers of non-zero elements are grouped into the same block from a sparse matrix to improve the effect of compression of ELL. It is also a challenging issue to determine the size of a block. If the size of the block is too large, there are more zeros to fill in the block. On the contrary, if the size of the block is too small, the number of blocks is too much, leading to an increased number of computing tasks.

How to make full use of computing resources to maximize parallel computing ability is the key to improve the performance of SpMV. Firstly, load balancing between the threads is the basis of improving performance for streaming multiprocessor (SM) on GPU. Secondly, improving the efficiency of data access is very important to improve the parallel efficiency of GPU.

1.2. Our contributions

The present paper makes the following unique contributions to parallel computation of SpMV on GPU and CPU.

- We develop an optimal partitioning strategy based on dynamic programming and a distribution function (DF) of non-zero elements to improve the performance of SpMV.
- We present a reordering algorithm in which the time complexity is only $O(N \log_2 k)$, where k is the number of partitions and far less than the number of rows. However, the time complexity of a general method is $O(N \log_2 N)$.
- We employ a hybrid format to store a blocked sparse matrix partitioned by our optimal partitioning strategy.

Our partitioning strategy is proven to be optimal, which can yield the minimum parallel execution time on GPU. Our partitioning strategy is based on the DF, which characterizes the distribution of non-zero elements in a sparse matrix. Our partitioning strategy consists of three steps, i.e., building the DF of a sparse matrix, partitioning the rows using dynamic programming, and reordering the rows. Firstly, the DF of a target sparse matrix is constructed according to the analysis of the distribution of non-zero elements per row. Secondly, the intervals of partitioning are determined by our partitioning algorithm based on the DF. Thirdly, these blocks for SpMV are segmented from the sparse matrix by a reordering algorithm. Furthermore, these blocks are stored in a hybrid format to obtain further performance gain.

In this paper, 20 sparse matrices, which are obtained from [2], are tested on NVIDIA K20c GPU and AMD Opteron 6376 CPU. The performance improvement of our algorithm is very effective according to our experiments. Our partitioning strategy has the best performance, which can partition for sparse matrices according to the minimum parallel processing time. According to our experiments on 20 test cases, the performance of SpMV is significantly improved when a sparse matrix is partitioned into blocks by our method, and improvement of our reordering algorithm is also effective.

This paper extends our previous work [3,4]. The current paper presents a new partitioning strategy based on processing time and using DF, and proposes a hybrid storage format and a kernel function for SpMV on GPU.

The remainder of the paper is organized as follows. In Section 2, we review related research on SpMV. In Section 3, we review the programming model of GPU and the storage formats of sparse matrices. In Section 4, we present the DF for sparse matrices. In Section 5, we analyze the parallel processing time of SpMV to develop our optimal partitioning strategy. In Section 6, we describe the implementation of SpMV in parallel using our method on GPU. In Section 7, we demonstrate our extensive experimental performance comparison results. In Section 8, we conclude the paper.

2. Related work

2.1. Parallel implementation of SpMV on GPU and partitioning strategies

Bolz et al. [5] proposed one of the first SpMV CUDA kernel implementations. [6] designed a new HYB format for SpMV in CUDA, representing the matrix in ELLPACK format (ELL) and coordinate format (COO) portions, to combine the speed of ELL and the flexibility of COO. Lee et al. [1] discussed optimization techniques for both CPU and GPU, and analyzed what architecture features contribute to performance differences between the two architectures. Stanimire et al. [7] presented a set of techniques that can be used to develop efficient dense linear algebra algorithms for hybrid multicore + GPU systems, and used asynchronous techniques to reduce the amount of communication between the hybrid components.

In large-scale scientific and engineering calculations, some very big sparse matrices are produced. These sparse matrices are too big to compute by one GPU once. So they should be partitioned into small blocks to be processed multiple times on GPU. But due to various distributions of non-zero elements in sparse matrices, there is no general partitioning method to adapt to all kinds of sparse matrices.

For the blocked compressed sparse row (BCSR) format [8] and the row-grouped CSR (GCSR) format [9], which the rows in a sparse matrix are split into blocks. For the blocked ELLPACK (BELLPACK) format [10] and the sliced ELLPACK (SELLPACK) format [11], a sparse matrix is partitioned into blocks after it is compressed by the ELL format. For segmented interleave

Please cite this article in press as: W. Yang et al., A parallel computing method using blocked format with optimal partitioning for SpMV on GPU, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2017.09.010