ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.10, No.9, 1–7

Breaking the performance bottleneck of sparse matrix-vector multiplication on SIMD processors

Kai Zhang, Shuming Chen[a)], **Yaohua Wang, and Jianghua Wan**

School of Computer, National University of Defense Technology,

#109, Deya Road, Changsha, 410073, China

a) *smchen@nudt.edu.cn*

Abstract: The low utilization of SIMD units and memory bandwidth is the main performance bottleneck on SIMD processors for sparse matrix-vector multiplication (SpMV), which is one of the most important kernels in many scientific and engineering applications. This paper proposes a hybrid optimization method to break the performance bottleneck of SpMV on SIMD processors. The method includes a new sparse matrix compressed format, a block SpMV algorithm, and a vector write buffer. Experimental results show that our hybrid optimization method can achieve an average speedup of 2.09 over CSR vector kernel for all the matrices. The maximum speedup can go up to 3.24.

Keywords: SpMV, SIMD, CSR, stride-combination CSR with transpose

Classification: Integrated circuits

References

[1] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel, "Optimization of sparse matrix-vector multiplication on emerging multicore platforms," *SC'07*, Nevada, USA, pp. 1–7, 2007.

[2] N. Bell and M. Garland, "Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors," *SC'09*, Portland, Oregon, pp. 1–11, Nov. 2009.

[3] D. DuBois, A. DuBois, C. Connor, and S. Poole, "Sparse Matrix-Vector Multiplication on a Reconfigurable Supercomputer," *FCCM'08*, California, USA, pp. 239–247, April 2008.

[4] X. Feng, H. Jin, et al., "Optimization of Sparse Matrix-Vector Multiplication with Variant CSR on GPUs," *Proc. 17th Int. Conf. Parallel and Distributed Systems*, Taiwan, pp. 165–172, Dec. 2011.

[5] M. Woh, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner, "AnySP: Anytime Anywhere Anyway Signal Processing," *ISCA'09*, June 2009.

[6] S. Chen, et al., "YHFT-QDSP: High-Performance Heterogeneous Multi-Core DSP," *J. Comput. Sci. Technol.*, vol. 25, pp. 214–224, 2010.

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

1 Introduction

The SpMV arise in many scientific and engineering applications. It has proven to be of frequent bottleneck in scientific computing applications, and is notorious for the low utilization of peak processor performance [1]. The low peak performance utilization is mainly caused by the bound of memory bandwidth and low utilization of computation units. Massive researches improve the performance of SpMV by breaking the bound of memory bandwidth or improving the utilization of computation pipeline [2, 3, 4]. Most of which provide parallel computing resource to accelerate the SpMV.

The SpMV computation is usually represented by the computation of $y = Ax$, in which A is a sparse matrix and x is a dense vector. The number of nonzeors in A is usually little than 1% in proportion. Thus, the matrix A is generally stored in compressed format. These compressed formats include CSR, DIA, ELL, COO, HYB and so on. There is no affirmative answer for which storage format is best. And also, some of these formats can be only applied to a limited range of matrices and are only efficient for a specified architecture [4]. The CSR is the most common and flexible format. It is easy to transplant onto variant platforms. Many formats are based on the idea of CSR.

The single instruction multiple data (SIMD) technology has been a popular method to accelerate the parallel computation on many processors. When implementing the CSR-based SpMV on SIMD processors, there are three main problems which are the performance bottleneck. Firstly, the utilization of SIMD units is always low. Secondly, the performance losing caused by indirect memory access to x is worse. Lastly, as each iteration of SpMV produces a scalar result, the memory bandwidth is not fully utilized when storing the result.

To address these problems, this paper proposes a hybrid optimization method to break the performance bottleneck of SpMV on SIMD processors. A new sparse matrix compressed format, which is called stride-combination CSR with transpose (SCT), is proposed to increase the utilization of SIMD units. A block SpMV algorithm is proposed to increase the utilization of memory bandwidth for reading x. A vector write buffer (VWB) is designed to increase the utilization of memory bandwidth for storing the result.

2 CSR SpMV on SIMD processors

The CSR format is one of the most popular and universal sparse matrix representations. Fig. 1 (a) shows an example in which the sparse matrix is stored in CSR format. Given a sparse matrix A, CSR stores A into a compressed format with three one-dimensional arrays: **val**, **col_ind**, and **row_ptr**. The array **val** stores the nonzeros of A. The array **col_ind** stores the column indices of nonzeros. The array **row_ptr** is used to represent rows of varying length. It stores pointers to the first nonzero of each matrix row.

Two approaches have been proposed to parallelize the CSR SpMV on the SIMD architecture. One approach is referred as the scalar CSR kernel

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

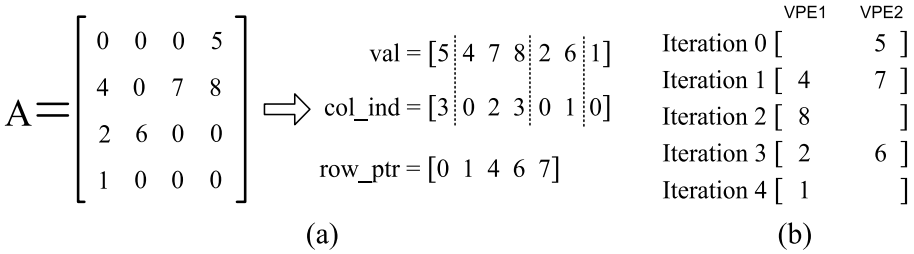

Fig. 1. (a) CSR fromat for a simple example matrix A. (b) Iteration process of vector CSR SpMV kernel.

and the other is the vector kernel. The experimental results show that the performance of the vector kernel is always better than that of the scalar kernel [2]. The performance benefit is mainly caused by accessing indices and data contiguously, and which therefore overcomes the principal deficiency of the scalar approach.

A SIMD processor generally consists of multiple Vector Processing Elements (VPEs) to accelerate the parallel computation [5]. The number of VPEs is referred as SIMD width. We employ an example of vector CSR SpMV on the SIMD processor with 2 VPEs. Fig. 1 (b) shows the data processed pattern in each iteration. The **array val** and **col_ind** is compressed from the matrix which is shown in Fig. 1 (a). It is likely that many SIMD units within one instruction controlled will remain idle while the number of nonzeros in each matrix row is not the integer times than that of SIMD width.

Indirect and non-contiguous memory accesses reduce bandwidth efficiency and therefore the performance of memory-bound kernels. On SIMD processors, each indirect parallel access to vector **x** is possible to bring many memory access conflicts, which make the above problem more serious. When mapping CSR-based SpMV onto the SIMD architecture, the vector kernel performs a dot product operation between each matrix row and the vector **x**. As only one scalar result is produced in each access to the parallel memory of SIMD processors, the powerful memory bandwidth is usually wasted.

To address the above problems, a hybrid optimization of SpMV based on a SCT compressed format is proposed in the following section.

3 Hybrid optimization of SpMV

3.1 SCT-based SpMV

The SCT format consists of three arrays which are **val**, **col_ind**, and **row_cnt**. Given a matrix A with n rows, SCT format compresses it into S rows which are then stored in column order. The matrix is firstly compressed into CSR format. Then the array **row_ptr** is transformed into the array **row_cnt**, which directly shows the number of nonzeros in each row. In the stride-combination step, a row is chosen from the first S CSR rows in sequence. The chosen row is taken as the head of combined-row and is combined with a series of rows with stride S. The new S combined-rows are then transposed to be stored in column order. After the transpose step, the array **val** can

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

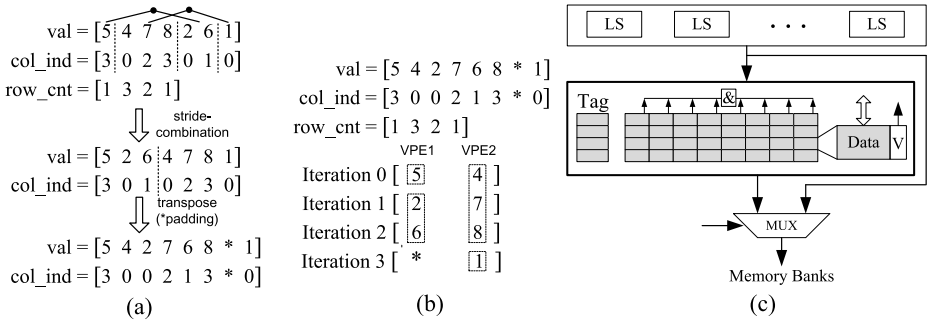

Fig. 2. (a) An example of CSR format to SCT format with $S = 2$. (b) Iteration process of SCT SpMV kernel. (c) The organization of VWB.

be viewed as an matrix with S columns. Each column is the combination of multiple CSR rows with stride S. In the above steps, the **col_ind** array must refresh to fit the new array **val**. The parameter S is usually set to the SIMD width of the target architecture. Fig. 2 (a) displays an example of transforming CSR format to SCT format with $S = 2$.

The padding step is employed to align all the columns. We propose a heuristic padding algorithm for the **col_ind** array to ensure that the padding value will not bring additional conflicted memory access to **x**. Given one position for padding, the index is decreased until it is divisible by S. We assume that the new index is i. Then the next S elements of the **col_ind** array starting from index i form an exemption set. The padding value should be different from all the elements within the exemption set.

We employ an example of SCT SpMV on the SIMD processor with 2 VPEs. Fig. 2 (b) shows the data processed pattern in each iteration. As it can be seen, the required number of iteration in SCT SpMV is less than CSR. In the SCT SpMV algorithm, the SIMD units load contiguous elements of **val** from multiple memory banks of the SIMD processor, which is a vector access to memory without conflict. The loaded elements of **val** comes from different matrix rows. After obtaining **val** and **x**, the MAC unit is employed to execute the multiplication and accumulating operation.

The **row_cnt** array is used on each VPE to segment the computation of variant matrix rows with the help of conditional execution. The **row_cnt** is loaded to local register of each VPE. The register decreases in each iteration to determine whether the current matrix row is completed. Thus, variant matrix rows can be successively processed on each VPE, which greatly increases the utilization of SIMD units and memory access.

3.2 Block SpMV algorithm

The SIMD access to **x**, which are indexed by **col_ind**, may generates conflicts. The conflicts are caused by the random distribution of nonzeros in matrix row. To address this issue, we propose a block SpMV algorithm. The matrix is divided into several blocks by columns. The vector **x** is divided into several sub-vectors by rows. We use SCT-based SpMV to process a block of data, perform significant amount of computation, and then transfer a new block.

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

For each block of data, the sub-vector $\mathbf{x}$ is duplicated among all the memory banks. Thus, the indirect and SIMD access to $\mathbf{x}$ can be performed without any conflicts.

3.3 VWB for multiple memory banks

In SIMD SpMV algorithms, only one scalar result is produced in each access to the parallel memory of SIMD processors. Thus, the powerful memory bandwidth is wasted when storing the result. The VWB is proposed to address this problem. The organization of VWB is as shown in Fig. 2 (c). It consists of several buffer lines. The number of buffer lines is referred as buffer depth. Each buffer line contains multiple buffer registers and corresponds to one tag field. The number of buffer registers equals to the SIMD width. Each buffer register consists of data field and one additional bit to record the data is valid or invalid. The valid state means the data has been modified by Load/Store (LS) unit. The tag field is used to record the high bits of destination address for the store operation. The recorded high bits index the start address of one contiguous memory access. There is one bit in tag field which is referred as dirty field to indicate whether the corresponding buffer line has been written.

We transplant the idea of cache into VWB. While a store operation requires write data to memory banks, VWB employs the fully associative method to determine that the data be placed in which buffer line. We adopt FIFO strategy for selecting which line to replace. The VWB can be powered off when accesses to memory banks are always contiguous. Then the data is directly bypassed to parallel memory banks without access to VWB. When the width of each data is larger than that of each buffer register, buffer registers in the same position of neighbor buffer lines is combined together to buffer each data.

4 Experimental results

We build a cycle-accurate simulator based on our previous single-core simulator [6] for the SIMD architecture. The SIMD pipeline simulation is combined with a cycle-driven memory system simulation that models the multi-banked memory, which is organized as scratchpad memory. The number of VPE is 16, each having a 32-entry register file. Each VPE contains four function units: ALU, two L/S and MAC. The parallel memory has 16 banks, each corresponding to a SIMD unit. The buffer depth of VWB is 8. Each buffer register is 64-bit wide. The above parameters are the basic configuration of our simulator. Some parameters of the simulator can be dynamically configured. Manually optimized assemble code is used as the input of the simulator.

The experimented sparse matrices are from Tim Davis's sparse matrix collection, which are used by Williams et al. [1] in their multicore benchmarking study. They represent different kind of real applications often used in scientific computing area, including economics, epidemiology, FEM based

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

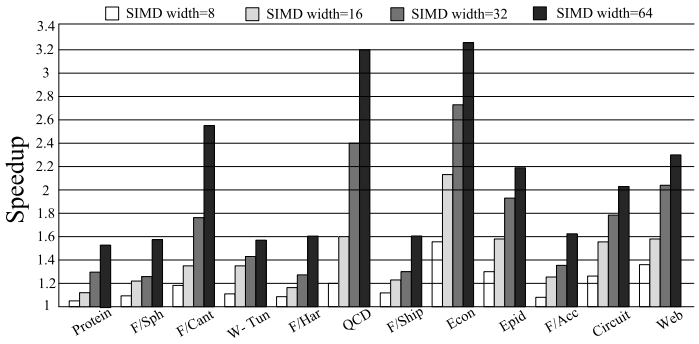

Fig. 3. The speedup of SCT-based SpMV over vector CSR kernel with variant SIMD width.

modeling, protein, webbase, and so on.

4.1 Experimental results

Fig. 3 shows the speedup of SCT-based SpMV over vector CSR kernel with variant SIMD width. As shown, all the SCT implementations outperform the vector CSR-based SpMV. The maximum speedup can be up to 3.24 in Economics matrix. The average speedup is 2.09 with the buffer depth of 8. We can see that the speedup for matrices with little number of nonzeros per matrix row, such as Economics, Epidemiology, Circuit, and Webbase, always outperform the others. In the CSR implementations for the above matrices, SIMD units are not fully utilized in each iteration, while this deficiency can be well overcome in the SCT implementations. It obviously shows that the proposed SCT format, block algorithm and VWB are efficient for improving the performance of SpMV by increasing the utilization of SIMD units and memory bandwidth.

The parameter S in SCT format is set to the SIMD width of the target architecture. So the performance of SCT implementations is influenced by the SIMD width. For implementations with the SIMD width of 16, the maximum speedup can be up to 2.15, and the average speedup is 1.43. The average speedup is 1.2 with the SIMD width of 8. For the SIMD width of 32, which equals to the width of modern GPUs, the average speedup is 1.72.

4.2 Hardware cost

To evaluate the hardware cost of VWB, we implement the VWB with 8 buffer lines in Verilog HDL. Each buffer line has 32 buffer registers. Each buffer register is 64-bit wide. The RTL implementation has been synthesized with Synopsys Design Complier under TSMC 45 nm technology. The clock frequency is set to 1 GHz. The area and power of VWB is 0.07 mm^2 and 5.2703 mW respectively. Thus, we can obtain efficient performance improvement with the help of VWB by spending little hardware cost.

5 Conclusions

This paper has proposed a hybrid optimization method for SpMV on SIMD processors. The new sparse matrix compressed format SCT can increase the

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013

utilization of SIMD units. The block SCT-based SpMV algorithm can eliminate the memory access conflicts caused by the indirect and SIMD access to **x**. The VWB combines several divergence write accesses into one contiguous access, which further increase the utilization of memory bandwidth by efficiently reducing the memory access. Our hybrid optimization method well overcomes the deficiency of vector CSR-based SpMV whose performance is sensitive to the distribution of nonzeros in each row.

Acknowledgments

This work is supported by the NSF of China with Grant No.61070036.

DOI: 10.1587/elex.10.20130147
Received February 27, 2013
Accepted April 12, 2013
Published May 10, 2013