B-SCT: Improve SpMV processing on SIMD architectures

Yaohua Wang^{1a)}, Dong Wang², and Xu Zhou¹

¹ College of Computer Science, National University of Defense Technology,
#109, Deya Road, Changsha, 410073, China

² Department of Science, National University of Defense Technology
a) nudyh@gmail.com

Abstract: Sparse matrix-vector multiplication (SpMV) represents the dominant cost in sparse linear algebra. However, sparse matrices exhibit inherent irregularity in both amount and distribution of none-zero values. This harnessed the tremendous potential of Single Instruction Multiple Data (SIMD) architectures, which is widely adopted in nowadays data-parallel processors. To improve the performance of SpMV, we proposed the Balanced SCT (B-SCT) method. The cornerstones are composed of the balanced-aware compression scheme and the on-the-fly data re-order structure. Our simulation results show that the B-SCT method provides an average speed-up of 130% over the commonly used CSR method, and 83% over the SIMD-oriented SCT method.

Keywords: SIMD, SpMV, B-SCT

Classification: Integrated circuits

References

- [1] Y. Wang, S. Chen, H. Chen, J. Wan, K. Zhang and S. Liu: IEICE Trans. Inf. & Syst. **E96-D** (2013) 365. DOI:10.1587/transinf.E96.D.365
- [2] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick and J. Demmel: Parallel Comput. **35** (2009) 178. DOI:10.1016/j.parco.2008.12.006
- [3] X. Feng, H. Jin, R. Zheng, K. Hu, J. Zeng and Z. Shao: IEEE 17th International Conference on Parallel and Distributed Systems (2011) 165. DOI:10.1109/ICPADS.2011.91
- [4] K. Zhang, S. Chen, Y. Wang and J. Wan: IEICE Electron. Express **10** (2013) 20130147. DOI:10.1587/elex.10.20130147
- [5] Y. Wang, S. Chen, K. Zhang, J. Wan, X. Chen, H. Chen and H. Wang: IEEE Comput. Archit. Lett. **11** (2012) 37. DOI:10.1109/L-CA.2011.34

1 Introduction

SpMV arises in numerous linear algebra applications, and is of particular importance in scientific and engineering computation. SIMD processors always contain multiple processing elements (PEs) running in a lock-step manner [1], providing efficient parallel data processing. Thus, the abundant data level parallelism in

SpMV makes SIMD processors become the most attracting processing tool. Unfortunately, none-zero values in sparse matrices are generally of small amount (less than 1%) and randomly distributed. This leads to under-utilization of hardware resources in SIMD processors, greatly harnessing the tremendous potential performance [2].

To improve the overall performance, a commonly used method is the CSR compression method [3]. As shown in Fig. 1, none-zero values in the sparse matrix are colored. The CSR method compressed the matrix in a horizontal manner. When executed on a 4-wide (number of PEs) SIMD architecture, there can be idle PEs (shown as blank arrows in Fig. 1) if the number of none-zero values in each matrix row is smaller than 4. In fact, PEs could be idle while the number of none-zero values in each row cannot be divided by the SIMD width. Thus, hardware utilization can be greatly reduced.

Fig. 1. None-zero values in sparse matrix are colored. The CSR the SCT method compressed the sparse matrix in horizontal and vertical manner respectively. Our proposed B-SCT method compressed matrix rows with similar number of none-zero values together in a vertical manner. The corresponding execution process on SIMD processors is also illustrated.

The SCT scheme [4] improved the CSR method with a vertical compression scheme. Thus, PEs process different matrix rows, and could be idle only in the processing of tail matrix rows. Moreover, the SCT scheme has also introduce a vector write buffer to fully use the memory bandwidth. However, as the number of none-zero values is different from row to row (as show in Fig. 1, the yellow row is 3, the green row is 4), it can happen that some PEs may finish computing current row and need to do the tail processing including: result value write-back and initialization for the next row, while other PEs still need to continue the computation. To maintain a single instruction across all PEs, tail processing is implemented through conditional operations. As shown in the execution process of the SCT method, these conditional operations lead to additional overhead. In fact, this overhead accounts for up to 50% of the overall execution time for typical SpMV applications, greatly reducing performance gains captured by the SCT method.

To solve the above problems, we proposed the Balance SCT (B-SCT) method. B-SCT largely eliminates the overhead of conditional operations, while maintaining the key benefits of SCT in PEs and memory bandwidth utilization. The cornerstone of SCT method is composed of balanced-aware compression (BAC) and the on-the-fly data re-order structure. As shown in Fig. 1, the key idea of BAC is to compress matrix rows with similar number of none-zero values together in a vertical manner. Thus, tail processing is scheduled at the end of the longest row rather than done in each iteration through conditional operations. The on-the-fly reorder structure regulates the computation result to its original order. Our experiment results show that the B-SCT method can achieve an average speed-up of 130% over the CSR method, and 83% over the SIMD-oriented SCT method, while the hardware overhead is equal to the SCT method.

2 The balanced-aware compression (BAC)

The BAC scheme compressed a sparse matrix into four arrays which are `val`, `col_ind`, `Bdl_cnt` and `Bdl_ind`. Given a sparse matrix *A*, matrix rows are firstly grouped into bundles of *N* rows, each row in the bundle corresponds to a PE of SIMD processors. Secondly, matrix rows of different bundles are exchanged, such that the newly formed bundles contain rows with similar length (the number of none-zero values in each row). To ease the recover process, we proposed the PE-aware exchanging protocol. This protocol exchanges rows within *M* consecutive bundles, and only rows corresponding to the same PE can be exchanged with each other. *M* is the exchange scope, which can be well tuned with respect to both SIMD width and the characteristics of applications. At last, the bundles are compressed in a vertical manner with padding zeros inserted to form equal row length.

`Val` contains the none-zero values, and `col_ind` is the column index of these values in the original matrix. `Bdl_cnt` stores the largest row length of each bundle. `Bdl_cnt` is used as the loop counter for the computation of each row bundle. `Bdl_ind` indicates the original bundle number which the compressed matrix row belongs to, and is used to recover the order of computation results.

As shown in Fig. 2a, a sparse matrix is firstly grouped into two bundles *B0* and *B1*, and then matrix rows of different bundles that correspond to the same PE are exchanged, here the exchange scope *M* is 2. Row 2, 3 are exchanged with row 6, 7 respectively. After the exchanging, both *B0* and *B1* contain matrix rows with similar length. At last *B0* and *B1* is compressed vertically with padding zeros inserted. The final result for `Val`, `Col_ind`, `Bdl_cnd` and `Bdl_ind` is also listed. The `Col_ind` for padding zeros is listed as ‘*’, which can be statically set to achieve confliction-free memory accesses. Compared to the SCT method, we have added extra padding zeros and the information of `Bdl_ind`, at the same time, we have also reduced the per-matrix row information `row_cnt` by a factor of the SIMD width. Thus, the total compression rate is more or less the same with the SCT method. Moreover, for modern processing of SpMV, the main concern is the computation efficiency rather than the compression rate.

Fig. 2. The BAC scheme groups sparse matrix into row bundles, each row in a bundle corresponding to a PE of SIMD processors, and then exchanges rows corresponding to the same PE, such that each row bundle contains rows with similar length. Row bundles are then compressed vertically. The ORB structure recovers the order of results in a on-the-fly manner.

3 The on-the-fly recovery buffer (ORB)

The ORB serves as a write cache between PEs and multi-banked vector memory. ORB can recover the result order, which can be disturbed during the BAC process. As shown in Fig. 2b, ORB comprises a vector buffer and a control unit. The vector buffer is M-depth and N-width, M is equal to the exchange scope in BAC (M is set to 2 in the figure). Each vector buffer line corresponds to the result of a row bundle, and contains a start address register (S_addr) and N data elements, where N equals to the SIMD width. The control unit stores each element of the vector result to the corresponding vector buffer line, according to the Bdl_ind information. The PE-aware exchange protocol guarantees a conflict free storing process. After M consecutive stores, the result order within the same exchange scope is recovered on-the-fly. Then, the entire vector buffer will write-back the result to vector memory, and get ready for the result of the next M bundles. The write-back memory efficiency of the SCT method well maintained. A bypass data-path between PEs and the vector memory is also added to maintain the traditional memory access efficiency.

4 Methodology

We used the cycle-accurate simulator of SIMD processor [5]. The PE simulation is combined with a cycle-driven memory system simulation that models the multi-banked scratchpad memory. The number of PEs is 16, each PE contains four function units: ALU, L/S and two MACs, and has its private 32-entry register file. The vector memory contains 16 memory banks. The vector buffer depth of the ORB is set to be 4.

To show the performance gain of the B-SCT method, we set the baseline simulator for CSR method as the one without ORB. Moreover, to compare with the SCT method, we have also implemented the VWB structure of SCT on the baseline simulator. The buffer depth of VWB is also set to be 4. Manually optimized assemble code is used as the input of the simulator. The experimented sparse matrices are from Tim Davis's sparse matrix collection, mentioned in [2], which covered a broad range of applications including economics, FEM based modeling, web-base, protein, epidemiology, and so on.

5 Experimental results

5.1 Performance analysis

The performance gain of SCT and the proposed B-SCT over the CSR method is shown in Fig. 3. The B-SCT can achieve an average performance gain of about 130% and 83% over the CSR and SCT methods respectively. The B-SCT method well maintained the benefits of the SCT method in vertical compression, which can greatly increase the utilization of PEs. At the same time, the ORB helps to fully use the vector memory bandwidth in the result write-back process.

Fig. 3. The overall performance gain.

The row exchange efficiency plays an important role in the performance gain of B-SCT method. This efficiency is defined as the variance of mean among row length values in each row bundle, and can be greatly affected by the row length distribution in the sparse matrices. As shown in Fig. 3, applications of QCD, Epid, Circuit and Web have a relatively uniform distribution of row length, leading to a higher performance gain. For QCD, as all of the matrix rows have the same length, the performance gain more or less doubled that of the SCT method. Other applications have various row length, leading to more padding zeros inserted after the row exchanging process. Thus, the performance gain is relatively small. With larger exchange scope, the performance gain of B-SCT can be improved, while the hardware cost of ORB can also be increased as the ORB depth should be equal to that scope. So that proper exchange scope depends on trade-offs between performance gain and hardware cost.

5.2 Hardware cost

We implement the ORB in Verilog HDL with buffer depth set to 4. Each buffer line contain 16 32-bits data element and 1 64-bit start address. The RTL implementation

has been synthesized with Synopsys Design Compiler under TSMC 45 nm technology at 1 GHz. The area and power of ORB is 0.009 mm^2 and 0.669 mW respectively. With an equal buffer depth, the overall hardware overhead is relatively the same with the SCT method.

6 Conclusion

In this paper, we proposed a novel mechanism, Balanced SCT (B-SCT), which maintains the main benefit of SCT [4] in PE and memory bandwidth utilization, while eliminating the unnecessary conditional operations. The corner stone of B-SCT includes both the balanced-aware compression (BAC) scheme and order recovery buffer (ORB). Our simulation qualifies that the B-SCT achieves an overall 130% and 83% performance gain over the CSR [3] and SCT method respectively. The area cost of the ORB is about 0.009 mm^2 .

Acknowledgments

This work is supported by the NSF of China in with Grant No. 61402493 and the Research Project of National University of Defense Technology with Grant No. GC14-06-02.