Designing and Implementing a Single-Phase Row-Column-Parallel Sparse Matrix Vector Multiplication Algorithm based on 2D Matrix Partitioning

Enver Kayaaslan[a], Bora Ucar[b], Ozcan Ozturk[a] ,Cevdet Aykanat[a]

[a]*Bilkent University, Computer Engineering Department, 06800 Ankara, Turkey*
[b]*CNRS, Lyon, France*

Abstract

One-dimensional (1D) partitioning of sparse matrices results in lower quality partitioning than two-dimensional (2D) partitionings in the context of parallel sparse matrix vector multiply (SpMxV) operations. However, 2D partitioning schemes incur two communication phases. We propose a novel sparse matrix partitioning scheme which achieves a single communication phase as in 1D schemes and partitions the nonzeros with a flexibility close to that in 2D schemes. In the proposed partitioning scheme, a nonzero is assigned to either the receiver or the sender processor associated with the related input- or output-vector entries. We observe that a fine-grain partitioning should satisfy this constraint for most of the nonzeros. Based on this observation, we propose a simple yet effective heuristic to obtain such a partition in two steps. In the first step, we obtain partitions on the rows, columns, and nonzeros of the given matrix using some known methods. In the second step, we refine the nonzero partition such that the aforementioned constraint is met while keeping the row and column partitions obtained in the first step intact. We demonstrate that the proposed partitioning scheme improves the performance of parallel SpMxV operations both in theory and practice with respect to 1D and 2D partitionings.

1. Introduction

Sparse matrix vector multiplication (SpMxV) is a kernel operation repeatedly performed in iterative linear system solvers. There are mainly three types of parallel SpMxV algorithms used in the scientific community: row-parallel, column-parallel and row-column-parallel. The row-parallel algorithm involves expand-type point-to-point communication operations on the local input vector entries before the local SpMxV operations, whereas column-parallel algorithm involves fold-type point-to-point communication operations on the local output vector results after the local SpMxV operations. The row-column-parallel algorithm necesitates two-phase communication: expand operation before local SpMxVs and fold operation after the local SpMxVs. 1D rowwise and columnwise partitioning of the coefficiant matrix are used for row-parallel and column-parallel SpMxV algorithms, respectively, whereas 2D-nonzero partitioning of the coefficiant matrix is used for row-column-parallel SpMxV algorithms. Several hypergraph partitioning models and methods have been successfully used for sparse matrix partitioning for

efficient row-parallel, column-parallel and row-column-parallel SpMxV operations. In all these models the partitioning objective is to minimize the total volume of communication whereas the partitioning constraint is to minimize the computational load balance. 2D nonzero based partitioning models are both more scalable and perform considerably better than the 1D partitioning models in terms of communication volume metric. However, 1D models perform considerably better than 2D models in terms of speedup values due to the increased number of messages in the row-column-parallel SpMxV algorithm. In this project, a one-phase row-column-parallel SpMxV algorithm to address this bottleneck of the row-column-parallel SpMxV operation so that nonzero-based matrix partitioning models and methods can be successfully utilized.

2. Work Done

Given a sparse matrix A, we obtain a K-way nonzero decomposition **A=A^r + A^c + A^d** , where and **A^r**, **A^c** and **A^d** are mutually disjoint,and **A^r** and **A^c** are partitioned rowwise and columnwise, respectively,

$$A^r = \begin{bmatrix} A^r_{1,*} \\ A^r_{2,*} \\ \vdots \\ A^r_{K,*} \end{bmatrix} = \begin{bmatrix} 0 & A^r_{1,2} & \dots & A^r_{1,K} \\ A^r_{2,1} & 0 & \dots & A^r_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ A^r_{K,1} & A^r_{K,2} & \dots & 0 \end{bmatrix}, \quad A^d = \begin{bmatrix} A^d_1 & & & \\ & A^d_2 & & \\ & & \ddots & \\ & & & A^d_K \end{bmatrix},$$

$$A^c = \begin{bmatrix} A^c_{*,1}, A^c_{*,2}, \dots, A^c_{*,K} \end{bmatrix} = \begin{bmatrix} 0 & A^c_{1,2} & \dots & A^c_{1,K} \\ A^c_{2,1} & 0 & \dots & A^c_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ A^c_{K,1} & A^c_{K,2} & \dots & 0 \end{bmatrix}. \qquad (1.1)$$

Each processor **P_k** holds the row stripe **A^r_{k,*}**, the column stripe **A^c_{*,k}**, and the diagonal **A^d_k**. In parallel matrix vector multiplication, each processor **P_k** executes the following steps:

1. For each nonzero off-diagonal block $A^r_{\ell,k}$, form sparse vector $\hat{x}^\ell_k$, which contains only those entries of x_k corresponding to the nonzero columns in $A^r_{\ell,k}$.
2. For each nonzero off-diagonal block $A^c_{\ell,k}$, form sparse vector $\hat{y}^k_\ell$, which contains only those results of $y^k_\ell = A^c_{\ell,k} \times x_k$ corresponding to the nonzero rows in $A_{\ell,k}$.
3. Send $[\hat{x}^\ell_k, \hat{y}^k_\ell]$ to processor P_ℓ.
4. Compute the diagonal block product $y^k_k = A^d_k \times x_k$, and set $y_k = y^k_k$.
5. For each nonzero off-diagonal block $(A^r_{k,\ell}+A^c_{k,\ell})$, receive $[\hat{x}^k_\ell, \hat{y}^\ell_k]$ from processor P_ℓ, then compute $y^\ell_k = \hat{y}^\ell_k + A^r_{k,\ell} \times \hat{x}^k_\ell$, and update $y_k = y_k + y^\ell_k$.

Note that in the first item, **P_k** does not need to hold **A^r_{l,k}** as it only needs to know which x-vector entires are required by **P_\ell** from **P_k**. Also note that in the second item, **P_k** holds **A^c_{l,k}**.

We propose a two-step approach to obtain a row/column and nonzero partition of a given sparse square matrix. In the former step, we have a K-way row/column partition and initial nonzero partition, either using one-dimensional (coarse-grain) [1] or two-dimensional (fine-grain) [3] partitioining approaches. As a result of this step, for each ordered pair of parts we get a submatrix whose nonzeros are to be determined to be hold by either receiver or sender processor. To achieve this goal, the latter step refines the nonzero partition by using dulmage-mendhelson decomposition of those submatrices. In case of one-dimensional partitioning in the former step, the off-diagonal submatrices are used, whereas in case of two-dimensional partitioning, in order to keep load balance, for each ordered pair of parts, the subset of the off-diagonal submatrix that is comprised of only nonzeros assigned to neither reciever nor sender processor is used. In the refinement step, the nonzeros that lie inside the horizontal blocks are assigned to sender, and the remaining nonzeros of the submatrices are assigned to receiver processor.

3. Results Obtained

MTX	ROW	COL	TWO	ONE1D	ONE2D
bundle1	8.32629	8.00294	5.73851	5.91815	8.34232
cbuckle	9.54464	10.6223	8.61245	9.81922	10.6047
finan512	13.5746	13.5038	11.6399	13.6765	13.6823
laminar_duct3	13.2674	13.1951	12.0709	12.9872	12.7951
poisson3Da	6.89166	6.90723	4.15417	7.17906	6.71825
rgg_n_2_17_s	12.7009	13.2403	11.1373	13.1798	12.8681
shuttle_eddy	6.07852	5.99124	3.74284	6.17699	6.18047
tube1	11.263	11.2328	9.50504	11.1489	8.66773
vibrobox	5.18068	6.01728	3.41504	6.12348	4.93331
ASIC_320ks	11.5035	11.578	10.4217	12.1037	12.2934

The following table presents the speedup results of different partitioning and parallel matrix vector multiplication approaches with 16 processors in a single-core multi-processor system connected with a Gigabit Ethernet Switch. ROW and COL respectively show the row-parallel and column-parallel matrix vector multiplication results. TWO represents the traditional row-col-parallel[3] matrix vector multiplication. Lastly ONE1D and ONE2D refers to the proposed matrix vector multiplication results where the matrix partition is obtained via using a one-dimensional and two-dimensional partition, respectively. The matrices are obtained from Florida Matrix Collection [2].

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557.

References

[1] Umit V. Catalyurek and Cevdet Aykanat, “Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplication”, IEEE Transactions on Parallel and Distributed Systems, vol. 10, no. 7, pp. 673-693, 1999

[2] Timothy A. Davis, “University of Florida sparse matrix collection”, NA Digest, 1997.

[3] Umit V. Çatalyürek, Cevdet Aykanat, and Bora Uçar, “On two-dimensional sparse matrix partitioning: Models, methods, and a recipe”, SIAM Journal on Scientific Computing, Vol. 32, pp. 656--683, 2010.