Parallel Sparse Matrix-Matrix Multiplication Library[*]

Kadir Akbudak
Computer Engineering Department
Bilkent University
Ankara, 06800 Turkey
kadir@cs.bilkent.edu.tr
kadir.cs@gmail.com

Cevdet Aykanat
Computer Engineering Department
Bilkent University
Ankara, 06800 Turkey
aykanat@cs.bilkent.edu.tr

For source code and additional documents of the library:
http://sites.google.com/site/kadircs/

Bilkent University
Computer Engineering Department
Technical Report
BU-CE-1402, 2014

Bilkent University
Computer Engineering Department
Graduate School of Engineering and Science
06800 Ankara, Turkey
http://cs.bilkent.edu.tr

[*]This work was financially supported by the PRACE-1IP project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493 and FP7-261557. We acknowledge PRACE for the Preparatory Access Call Type B (resource) awards for our applications numbered 2010PA0930 and 2010PA2149. The library presented in this work has been developed and tested using these awarded resources, JUQUEEN at Jülich Supercomputing Centre and SuperMUC at Leibniz Supercomputing Center, all of which are based in Germany.

Contents

Abstract

This technical report provides brief information and usage details of our MPI-based outer-product–parallel sparse matrix-matrix multiplication library, which is developed for our research paper entitled "Simultaneous Input and Output Matrix Partitioning for Outer-Product-Parallel Sparse Matrix-Matrix Multiplication." The library makes use of the sparsity pattern of input and output matrices for distributed-memory parallelism. It performs the sparse matrix-matrix multiplication (SpGEMM) operation of the form $C = A \times B$ in two separate phases: First phase consists of communication-free local computation of outer products. Second phase consists of summation of results of the local outer products. The input matrices A and B are respectively partitioned one-dimensional (1D) columnwise and 1D rowwise conformably to enable communication-free local outer-product computations in the first phase. The output matrix C is partitioned on row, column or nonzero basis for mapping the local result summation tasks to processors. The actual SpGEMM computation can be represented by a novel hypergraph model and partitioning the input and output matrices can be established via partitioning this hypergraph model. The partitioning constraint of maintaining balance on part weights corresponds to obtaining computational load balance in the two phases, separately. The partitioning objective of minimizing cutsize corresponds to minimizing communication overhead during the second phase (summation phase).

Keywords: sparse matrix-matrix multiplication, SpGEMM, symbolic multiplication, distributed-memory parallelism, MPI, hypergraph, hypergraph partitioning

1 Introduction

Outer-product-parallel sparse matrix-matrix multiplication (SpGEMM) operation is based on one-dimensional (1D) columnwise and 1D rowwise partitioning of the input matrices A and B as follows:

$$\hat{A} = AQ = \begin{bmatrix} A_1^c & A_2^c & \dots & A_K^c \end{bmatrix} \qquad \text{and} \qquad \hat{B} = QB = \begin{bmatrix} B_1^r \\ B_2^r \\ \vdots \\ B_K^r \end{bmatrix} \tag{1}$$

Here, K denotes the number of parts and Q denotes the permutation matrix obtained from partitioning. The use of the same permutation matrix for column reordering of A and row reordering of B enables conformable columnwise and rowwise partitioning of matrices A and B. In this input partitioning, each outer product will be assigned to a processor. The nice property of the outer-product formulation using the same permutation matrix Q is that outer products are performed without any communication. However, multiple partial results will be needed to be summed to calculate the final value of each C-matrix nonzero. In other words, the output matrix is computed as follows in terms of the results of local outer-product computations:

$$C = C^1 + C^2 + \ldots + C^K \quad (\text{where } C^k = A_k^c \times B_k^r \text{ is performed by processor } P_k). \tag{2}$$

The summation of local C^k matrices incurs communication because multiple C^k matrices may have a nonzero at the same nonzero location of C. Since the input partitioning of A and B matrices does not enforce any output partitioning of the C matrix, the output partitioning can be performed in row, column or nonzero basis. The outer-product computations constitute the multiplication phase, whereas the summation of local C^k matrices constitutes the summation phase.

2 Hypergraph Model for Outer-Product–Parallel SpGEMM Computation

The library's matrix partitioning scheme discussed in Section 1 requires full access to the actual computation that forms C. The actual SpGEMM computation can be represented by a novel hypergraph model. The outer product of each column of A with the respective row of B is represented by a vertex in this hypergraph model. This model also contains a vertex and a net for each nonzero of C matrix, which represents the summation operation and communication volume during the summation operation, respectively. Two weights are associated with each vertex. The first and second weights represent the amount of computation to be performed in the multiplication and summation phases, respectively. Then by partitioning this hypergraph model using two constraints, the library obtains partitions of the input and output matrices. The partitioning constraints of maintaining balance on part weights correspond to obtaining computational load balance in the two phases, separately. The partitioning objective of minimizing cutsize corresponds to minimizing communication overhead during the summation phase.

The library uses hypergraph partitioning (HP) to simultaneously find input and output partitionings. The objective in this partitioning is to allocate A-matrix columns and B-matrix rows that contribute to the same C-matrix entries into the same parts as much as possible. This in turn corresponds to forcing execution of the outer-product computations that contribute to the same output C-matrix entries on the same processor. The hypergraph model encodes this objective successfully thus enabling exploitation of locality and so it achieves to reduce communication overhead in the summation phase.

The library can perform three different partitionings of the C matrix. First one is nonzero based and it utilizes the above-mentioned hypergraph model, which will be referred to here as *elementary hypergraph model*. The second one is row based. The relevant hypergraph model can be obtained via amalgamating the vertices representing the C-matrix nonzeros at the same row of C. The third one is column based. The relevant hypergraph model can be obtained via amalgamating the vertices representing the C-matrix

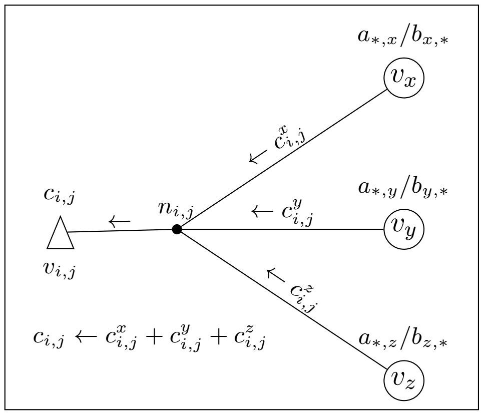

Figure 1: Elementary hypergraph model for nonzero-based output partitioning

nonzeros at the same column of C. All of these three hypergraph models achieve 1D input partitioning by conformably partitioning A and B matrices columnwise and rowwise, respectively, for outer-product–parallel SpGEMM.

Figure 1 illustrates the input and output dependency view of the elementary hypergraph model. As seen in this figure, net $n_{i,j}$, which connects vertices $v_x, v_y, v_z, v_{i,j}$, means that the outer products $a_{*,x} \times b_{x,*}$, $a_{*,y} \times b_{y,*}$, and $a_{*,z} \times b_{z,*}$ yield nonzero results $c^x_{i,j}$, $c^y_{i,j}$, and $c^z_{i,j}$, respectively. Hence, vertex $v_{i,j}$ represents the task of computing the final result for $c_{i,j}$ as $c_{i,j} = c^x_{i,j} + c^y_{i,j} + c^z_{i,j}$. Please refer to our paper [1] for more details of the hypergraph models used in the library.

3 Code Usage

3.1 Quick Start

1. Download the library from http://sites.google.com/site/kadircs/.

2. Decompress the downloaded library.

3. Set the variable `ROOT` located in `Makefile.inc` to the full path of the library.

4. Compile the source code:

   ```
   > make
   ```

5. This command runs the library for a small input matrix named `smallA`:

   ```
   > ./run.sh
   ```

3.2 Input Format for Sparse Matrices

The SpGEMM library uses binary format to reduce file I/O overhead during read and write operations of large sparse matrices. Following command can be used to convert a sparse matrix in Matrix Market format to the binary format used by the SpGEMM library:

```
> ./sspmxm/mtx2bintriplet in.mtx out.bin
```

For converting a sparse matrix in binary format to Matrix Market format:

```
> ./sspmxm/bintriplet2mtx in.bin out.mtx
```

3.3 Preprocessing Step for Partitioning Input and Output Matrices

Since the output matrix C is required to construct the hypergraph for the SpGEMM computation, C matrix must be computed prior to partitioning via the `./sspmxm/smult` program as follows:

```
> ./sspmxm/smult A.bin B.bin C.bin resultFile.txt CSR_SKIP_ZERO_ROWS
```

Partitioning for the parallel SpGEMM computation can be performed via the `./preprocess/preprocess` program, which takes the following sequence of parameters:

1. `P`	number of parts
2. `METRIC`	metric used for partitioning: • `CON`: connectivity-1 metric • `CUT`: cutnet metric
3. `IMBAL`	allowed imbalance ratio [0.0 ... 1.0]
4. `RCNETCOST`	use `RCNNZ`
5. `ZNETCOST`	cost type of nets in the hypergraph • `ZABVERT`: cost of a net is the number of input vertices connected by that net • `ZUNIT`: unit cost is associated with all nets
6. `MATRIX FORMAT`	• `MTX`: Matrix Market format • `TRIBIN`: Binary file that contains <row, column, value> triplets
7. `MATRIX A`	path to the input matrix A
8. `MATRIX B`	path to the input matrix B
9. `MATRIX C`	path to the output matrix C, it will be used to construct the hypergraph model of the SpGEMM computation
10. `RESULTFILE`	• `stdout`: standard out • `FILENAME`: filename to which results will be appended
11. `PART`	• `DOPART`: do partitioning and write partition vector to file • `LOADPARTVEC`: load partition vector from file • `CONSTRUCT_HYPERGRAPH_ONLY`: constructs the hypergraph of the SpGEMM computation
12. `PARTVECFILE`	name of file which will contain partition vector

13. `PARTMTX`	• `PARTMTX`: matrices will be partitioned • `PARTONLY`: matrices will not be partitioned
14. `HYPERGRAPH MODEL`	• `HPMODEL_C_NZ`: elementary hypergraph model (nonzero-based partitioning of C matrix) • `HPMODEL_C_ROW`: row-based partitioning of C matrix • `HPMODEL_C_COL`: column-based partitioning of C matrix
15. `STRMTXC`	use the same value of the parameter `MATRIX C`
16. `STRMTXCSS`	use `null` since this parameter is valid for other models and methods that are implemented for experimental purposes
17. `OUTPUT VERTEX WEIGHTS`	• `ZERO`: zero weights • `UNIT`: unit weights • `ABVERTICES_OF_ZNET`: weight of an output vertex is the number of input vertices connected by the net that connects this output vertex
18. `CSR SCHEME`	use `CSR_SKIP_ZERO_ROWS`
19. `INPUT VERTEX WEIGHTS`	• `ABMULT`: number of nets that connect the vertex • `ABMULTIROW`: another weighting scheme for testing purposes
20. `PARTREPEAT`	repeat partitioning count [1...]

3.4 Parallel SpGEMM Computation

Parallel SpGEMM computation can be performed via the `./pspmxm/pspmxm` program, which takes the following sequence of parameters:

1. `P`	number of processors
2. `STRMTXA`	use `null` since this parameter is valid for other models and methods that are implemented for experimental purposes
3. `STROUTPUTMTX`	use `null` since this parameter is valid for other models and methods that are implemented for experimental purposes
4. `MATRIX C`	path to the output C matrix that will be calculated in parallel
5. `COMMUNICATION PATTERN`	communication pattern that will be used in the second phase (summation phase). Use `NOCOMM`.
6. `MULTIPLICATION TYPE`	use `SYMBOLIC`
7. `PARTVEC`	path to partition vector

8. `CSR SCHEME`	• `CSR_NORMAL`: Gustavson's sequential SpGEMM algorithm is used • `CSR_SKIP_ZERO_ROWS`: Gustavson's sequential SpGEMM algorithm is used. But the outermost for-loop iterates over nonempty rows
9. `SEND MODE`	use `NORMAL`
10. `WRITE C MATRIX`	• `WRITEMTX`: write C matrix into file • `DONTWRITEMTX`: do not write C matrix, it is not printed either
11. `HYPERGRAPH MODEL`	hypergraph model used in partitioning of matrices
12. `MATRIX A`	path to input matrix A
13. `MATRIX B`	path to input matrix B
14. `MATRIX C`	path to output matrix C to be loaded for the parallel symbolic multiplication
15. `PERFORM NUMERICAL CHECK`	• `PERFORM_CHECK_IN_PREPROCESSING`: check equality of the loaded input matrix C and the matrix C calculated via parallel SpGEMM • `NO_CHECK_IN_PREPROCESSING`: no check
16. `PARTREPEAT`	repeat partitioning count [1...]

References

[1] K. Akbudak and C. Aykanat. Simultaneous Input and Output Matrix Partitioning for Outer-Product-Parallel Sparse Matrix-Matrix Multiplication. *Accepted for publication in SIAM Journal on Scientific Computing*, 2014.