HYPERGRAPH MODELS FOR PARALLEL SPARSE MATRIX-MATRIX MULTIPLICATION

A DISSERTATION SUBMITTED TO

THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER ENGINEERING

By

Kadir Akbudak

September, 2015

Hypergraph Models for Parallel Sparse Matrix-Matrix Multiplication
By Kadir Akbudak
September, 2015

We certify that we have read this dissertation and that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Cevdet Aykanat (Advisor)

Prof. Dr. Hakan Ferhatosmanoğlu

Assoc. Prof. Dr. Alper Şen

Assoc. Prof. Dr. Murat Manguoğlu

Assist. Prof. Dr. Tayfun Küçükyılmaz

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

HYPERGRAPH MODELS FOR PARALLEL SPARSE MATRIX-MATRIX MULTIPLICATION

Kadir Akbudak
Ph.D. in Computer Engineering
Advisor: Prof. Dr. Cevdet Aykanat
September, 2015

Multiplication of two sparse matrices (i.e., sparse matrix-matrix multiplication, which is abbreviated as SpGEMM) is a widely used kernel in many applications such as molecular dynamics simulations, graph operations, and linear programming. We identify parallel formulations of SpGEMM operation in the form of $C = AB$ for distributed-memory architectures. Using these formulations, we propose parallel SpGEMM algorithms that have the multiplication and communication phases: The multiplication phase consists of local SpGEMM computations without any communication and the communication phase consists of transferring required input/output matrices. For these algorithms, three hypergraph models are proposed. These models are used to partition input and output matrices simultaneously. The input matrices A and B are partitioned in one dimension in all of these hypergraph models. The output matrix C is partitioned in two dimensions, which is nonzero-based in the first hypergraph model, and it is partitioned in one dimension in the second and third models. In partitioning of these hypergraph models, the constraint on vertex weights corresponds to computational load balancing among processors for the multiplication phase of the proposed SpGEMM algorithms, and the objective, which is minimizing cutsize defined in terms of costs of the cut hyperedges, corresponds to minimizing the communication volume due to transferring required matrix entries in the communication phase of the SpGEMM algorithms. We also propose models for reducing the total number of messages while maintaining balance on communication volumes handled by processors during the communication phase of the SpGEMM algorithms. An SpGEMM library for distributed memory architectures is developed in order to verify the empirical validity of our models. The library uses MPI (Message Passing Interface) for performing communication in the parallel setting. The developed SpGEMM library is run on SpGEMM instances from various realistic applications and the experiments are carried out on a large parallel IBM BlueGene/Q system, named JUQUEEN. In the experimentation of the proposed hypergraph models, high speedup values are observed.

Keywords: sparse matrices, matrix partitioning, parallel computing, distributed memory parallelism, generalized matrix multiplication, GEMM, sparse matrix-matrix multiplication, SpGEMM, computational hypergraph model, hypergraph partitioning, BLAS (Basic Linear Algebra Subprograms) Level 3 operations, molecular dynamics simulations, graph operations, linear programming.

ÖZET

PARALEL SEYREK MATRİS-MATRİS ÇARPIMI İÇİN HİPERÇİZGE MODELLERİ

Kadir Akbudak
Bilgisayar Mühendisliği, Doktora
Tez Danışmanı: Prof. Dr. Cevdet Aykanat
Eylül, 2015

$C = AB$ şeklindeki genel seyrek matris-matris çarpımı (SyGEMM), moleküler dinamik benzetimi, çizge işlemleri, doğrusal programlama gibi pek çok uygulamada çekirdek işlem olarak kullanılmaktadır. SyGEMM işlemi için farklı paralelleştirme yöntemleri bulunmaktadır. Bu yöntemler için paralel SyGEMM algoritmaları önermekteyiz. Önerilen algoritmalar iki evreden oluşmaktadır. Evrelerden birisi yerel çarpma işlemleri içermekte olup, çarpma evresi olarak isimlendirilmektedir. Diğer evre ise, çarpma evresi için gerekli matris elemanlarının taşınması veya çarpma evresinde üretilen kısmi sonuçların aktarılarak toplanmasından oluşmakta olup, iletişim evresi olarak isimlendirilmektedir. Bu paralel algoritmalar için, girdi ve çıktı matrislerini aynı anda veri yinelemesiz olarak bölümleyebilen üç tane hiperçizge modeli önermekteyiz. Bu üç model, girdi A ve B matrislerini tek boyutlu (1D) olarak bölümlemekle beraber, ilk model çıktı C matrisini sıfır-dışı tabanlı olarak iki boyutlu (2D) ve geri kalan modeller ise çıktı C matrisini 1D olarak bölümlemektedir. Bu modellerde, köşe ağırlıkları üzerinde tanımlı olan bölümleme kısıtı, işlemcilerin işlemsel yüklerini dengelemeye karşılık gelmektedir. Keside kalan hiperkenarlar üzerinde tanımlanan kesi boyutunun azaltılması olan bölümleme amacı ise, iletişim evresinde yapılan toplam iletişim hacmini azaltmaya karşılık gelmektedir. Ayrıca, toplam mesaj sayısını azaltmakla beraber her bir işlemcinin yönettigi iletişimin hacmini dengelemeyi hedefleyen hiperçizge modelleri de önermekteyiz. Önerilen hiperçizge modellerinin geçerliliğini deneysel olarak da doğrulamak amacıyla, MPI (Message Passing Interface) tabanlı SyGEMM paket programı geliştirilmiştir. Çok çeşitli seyrek matrisler üzerinde bu program kullanılarak JUQUEEN isimli bir IBM BlueGene/Q sisteminde büyük ölçekli deneyler gerçekleştirilmiştir. Yapılan deneylerin sonucunda, önerilen hiperçizge modellerinin hesaplamarı önemli miktarda hızlandırdığı gözlemlenmiştir.

Anahtar sözcükler: seyrek matrisler, matris bölümleme, paralel hesaplama, dağıtık bellekte paralelleştirme, genel matris çarpımı, GEMM, seyrek matris-matris çarpımı, SpGEMM, bilişimsel hiperçizge modeli, hiperçizge bölümleme, BLAS (Basic Linear Algebra Subprograms) Seviye 3 işlemleri, moleküler dinamik benzetimi, çizge işlemleri, doğrusal programlama.

Acknowledgement

I would like to express my gratitude to my supervisor Professor Cevdet Aykanat for his suggestions, guidance, and encouragement to my research as being at the beginning steps of my academic life and while performing research to develop this thesis. His patience, motivation, lively discussions, cheerful laughter, and insightful directions provided a comfortable and invaluable environment for my research.

I must acknowledge efforts of members of Bilkent University for gathering valuable researchers in Ankara/Turkey.

I am thankful to members of Department of Computer Engineering for providing us a comfortable working environment and research facilities.

I am thankful to valuable professors, Assoc. Prof. Dr. Hakan Ferhatosmanoglu for his help throughout my research, Asst. Prof. Dr. Can Alkan for generously sharing his computing resources, especially Intel Xeon Phi coprocessors, and huge disk storage; Assoc. Prof. Dr. Alper Sen for being my jury member throughout my PhD studies, Assoc. Prof. Dr. Murat Manguoglu and Asst. Prof. Dr. Tayfun Kucukyilmaz for evaluating my PhD thesis.

I am grateful to my family, relatives and friends, especially Abdullah Bulbul, Hasan Baris Gecer, and Mustafa Urel for their support and help; and to my research group members: Mehmet Basaran, Vehbi Gunduz Demir, Mustafa Ozan Karsavuran, Enver Kayaaslan, Tayfun Kucukyilmaz, Erkan Okuyan, Reha Oguz Selvitopi, Fahrettin Sukru Torun, Ata Turk, and Volkan Yazici for being a good team.

I thank TÜBİTAK for supporting grant throughout my PhD and also throughout my MS studies.

This work is supported by the Partnership for Advanced Computing in Europe (PRACE) First Implementation Phase project, which is funded partially by the European Union's Seventh Framework Programme (FP7/2007-2013) having the number of agreement: FP7-261557 and RI-283493.

The experimental results presented in this thesis are achieved by the supports of PRACE Research Infrastructure. This infrastructure provided us awards on behalf of our applications to Access Calls. The results are achieved by using

these resources, which is accessing computing facilities at the large-scale parallel system named JUQUEEN. JUQUEEN is located at the Jülich Super-computing Centre (JSC), which is based in Germany.

Publications

- **K. Akbudak** and C. Aykanat, **Simultaneous Input and Output Matrix Partitioning for Outer-Product-Parallel Sparse Matrix-Matrix Multiplication**, **SIAM Journal on Scientific Computing**, vol. 36(5), pp. C568–C590, 2014, available at epubs.siam.org/doi/abs/10.1137/13092589X

- O. Karsavuran, **K. Akbudak** and C. Aykanat, **Locality-Aware Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication on Many-Core Architectures**, **IEEE Transactions on Parallel and Distributed Systems**, 2015, available at ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=7152923

- **K. Akbudak**, E. Kayaaslan, and C. Aykanat, **Hypergraph Partitioning Based Models and Methods for Exploiting Cache Locality in Sparse Matrix-Vector Multiplication**, **SIAM Journal on Scientific Computing**, vol. 35(3), pp. C237–C262, 2013, available at

  epubs.siam.org/doi/abs/10.1137/100813956

I dedicate this thesis to my beloved father,
who had always missed me.

Now, we are missing you...

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Sparse matrix-matrix multiplication (SpGEMM) in the form of $C = AB$ is an important computational kernel in various applications. Some of these applications are molecular dynamics (MD) [1, 2, 3, 4, 5, 6, 7, 8, 9], graph operations [10, 11, 12, 13, 14, 15, 16, 17], recommendation systems [18], and linear programming (LP) [19, 20, 21]. All of these applications definitely necessitate the use of parallel processing on large-scale systems in order to reduce their run times for large data.

There are several ways to formulate the general matrix-matrix multiplication. Four common formulations (see Section 13.2 of [22]) can be listed as follows:

- sum of outer products of columns of A and respective rows of B
- inner products of rows of A and columns of B
- pre-multiplication of rows of A with B
- post-multiplication of A with columns of B

Depending on these formulations, we propose parallel SpGEMM algorithms. All of these algorithms have two phases:

- multiplication phase, which consists of SpGEMM
- communication phase, which consists of transfer of required input matrix entries or transfer of produced partial results to the owner processor

The efficiency and scalability of the proposed parallel algorithms depends on the following quality criteria:

(*1*) balance among computational loads of processors

(*2*) total volume of communication

(*3*) total number of messages transferred over the interconnect network

(*4*) maximum volume of communication performed by a processor

(*5*) maximum number of messages handled by a processor

In this work, we first propose hypergraph partitioning (HP) based methods, which successfully and directly achieve the first two quality metrics, and indirectly achieve the third criterion. In the HP based methods, the partitioning constraint defined over the weights of the vertices corresponds to achieving the first criterion, whereas the partitioning objective of minimizing the cutsize defined over the cut nets corresponds to achieving the second criterion.

We also propose models for reducing the total number of messages (the third quality criterion) while maintaining balance on communication volumes (the fourth quality criterion) handled by processors during the communication phase of the SpGEMM algorithms. The communication hypergraph model is first proposed in [23] for the parallel sparse matrix-vector multiplication (SpMV) operation. The work [23] is further enhanced by [24]. The performance improvement by the proposed hypergraph models for reducing communication volume in parallel SpGEMM operations can be further enhanced by the use of the communication hypergraph models in a second preprocessing stage. In this second stage, the partitioning information of the first stage is preprocessed in order to reduce the total number of messages while maintaining balance on communication volumes handled by processors. This preprocessing step consists of construction of the respective communication hypergraph model and partitioning it. The partitioning objective of minimizing cutsize corresponds to minimizing the total number of messages transferred over the network. The partitioning constraint of balancing the part weights corresponds to maintaining balance on the volume of communication handled by processors.

The correctness of the proposed methods are shown both theoretically and empirically. For the empirical verification, we develop an SpGEMM library [25], which is based on MPI (Message Passing Interface) [26]. Extensive experimental analysis using our SpGEMM library are performed on a wide range of sparse matrices from different applications. A very large-scale supercomputer named JUQUEEN, which is an IBM BlueGene/Q system, is selected as an experimental testbed. Scalability of our SpGEMM library up to 1024 processors show the validity of the proposed HP based methods in practice.

This thesis is organized as follows: The background material on matrix multiplication and HP is given in Chapter 2. We review related work on SpGEMM in Chapter 3. The proposed parallel SpGEMM algorithms are presented in Chapter 4. We describe and discuss the proposed HP-based models and methods and their theoretical verification in Chapter 5. In Chapter 6, the communication hypergraph models are described. The empirical verification via presenting and discussing the experimental results is performed in Chapter 7. Thesis is concluded in Chapter 8, and some of the future research opportunities provided by this thesis are given in Chapter 9.

Chapter 2

Background

In this chapter, background material about matrix multiplication and hypergraph partitioning (HP) will be given. The matrix multiplication problem will be defined and its different formulations will be given in Section 2.1. Note that the problem definition and the formulations do not depend on sparsity of the involving matrices.

Section 2.2 will present the definition of hypergraph and the HP problem with the objective of cutsize minimization under the constraint of balancing part weights.

2.1 Matrix Multiplication

Multiplication of two matrices A and B of sizes, respectively, M-by-N and N-by-R yields matrix C of size M-by-R as follows:

$$c_{i,j} = \sum_{k=1}^{k=N} a_{i,k} b_{k,j} \tag{2.1}$$

Here, the subscripts denote the index of matrix element, e.g., $a_{i,k}$ denotes element at row i and column k of matrix A. The matrix multiplication given in

Algorithm 1 Matrix multiplication algorithm based on inner-product formulation, i.e., $< i, j, k >$ loop order.

```
Require: $A$, $B$, and $C$
1: for $i \leftarrow 1$ to $M$ do
2:     for $j \leftarrow 1$ to $R$ do
3:         $c_{i,j} \leftarrow 0$
4:         for $k \leftarrow 1$ to $N$ do
5:             $c_{i,j} \leftarrow c_{i,j} + a_{i,k} b_{k,j}$
6:         end for
7:     end for
8: end for
9: return $C$
```

Equation (2.1) can be expressed in various ways. The most common ones are as follows:

2.1.1 Inner-Product Formulation

In this formulation, matrix multiplication is defined as follows:

$$c_{i,j} = a_{i,*} b_{*,j} \tag{2.2}$$

Here, $a_{i,*}$ denotes row i of matrix A and $b_{*,j}$ denotes column j of matrix B. The multiplication $a_{i,*} b_{*,j}$ is called as the *inner product* of row i of matrix A with column j of matrix B. Each inner product yields a scalar value, which is a nonzero element of C matrix. The pseudocode for this formulation is given in Algorithm 1. The loop order used in this algorithm is commonly known as $< i, j, k >$ [27].

2.1.2 Outer-Product Formulation

In this formulation, matrix multiplication is defined as follows:

$$C = \sum_{k=1}^{k=N} a_{*,k} b_{k,*} \tag{2.3}$$

Algorithm 2 Matrix multiplication algorithm based on outer-product formulation, i.e., $< k, i, j >$ loop order.

```
Require: $A$, $B$, and $C$
1: $C \leftarrow 0$
2: for $k \leftarrow 1$ to $N$ do
3:     for $i \leftarrow 1$ to $M$ do
4:         for $j \leftarrow 1$ to $R$ do
5:             $c_{i,j} \leftarrow c_{i,j} + a_{i,k}b_{k,j}$
6:         end for
7:     end for
8: end for
9: return $C$
```

Algorithm 3 Matrix multiplication algorithm based on row-by-row formulation, i.e., $< i, k, j >$ loop order.

```
Require: $A$, $B$, and $C$
1: for $i \leftarrow 1$ to $M$ do
2:     $c_{i,*} \leftarrow 0$
3:     for $k \leftarrow 1$ to $N$ do
4:         for $j \leftarrow 1$ to $R$ do
5:             $c_{i,j} \leftarrow c_{i,j} + a_{i,k}b_{k,j}$
6:         end for
7:     end for
8: end for
9: return $C$
```

Here, $a_{*,k}$ denotes column k of matrix A and $b_{k,*}$ denotes row k of matrix B. The multiplication $a_{*,k}b_{k,*}$ is called as the *outer product* of column k of matrix A with row k of matrix B. Multiplication of two such vectors, i.e., an outer product, yields a matrix, so final C matrix is summation of these partial result matrices. The pseudocode for this formulation is given in Algorithm 2. The loop order used in this algorithm is known as $< k, i, j >$.

2.1.3 Row-by-Row Formulation

In this formulation, matrix multiplication is defined as follows:

$$c_{i,*} = a_{i,*}B \tag{2.4}$$

Here, $c_{i,*}$ denotes row i of matrix C. The multiplication $a_{i,*}B$ is called as the *pre-multiply* of row i of matrix A with whole matrix B. Each pre-multiply yields a row of C matrix. The pseudocode for this formulation is given in Algorithm 3. The loop order used in this algorithm is known as $< i, k, j >$.

The column-by-column formulation, which is defined as the *post-multiply* of whole matrix A with column j of matrix B, is dual of row-by-row formulation.

2.2 Hypergraph Partitioning (HP)

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ consists of a vertex set $\mathcal{V}$ and a net (hyperedge) set $\mathcal{N}$ [28]. Every net $n \in \mathcal{N}$ connects a subset of vertices, i.e., $n \subseteq \mathcal{V}$. The vertices connected by a net n are named as *pins* of that net and $Pins(n)$ notation is used to denote these vertices. A net n can be associated with a cost $c(n)$. A net's degree is defined as the count of the net's pins. That is, for net n,

$$deg(n) = |Pins(n)|. \tag{2.5}$$

The nets connecting a vertex v are called the nets of the vertex and $Nets(v)$ notation is used to denote these nets. A vertex's degree is defined as the count of the vertex's nets. That is, for vertex v,

$$deg(v) = |Nets(v)|. \tag{2.6}$$

Three types of quantities is used to define the size of a given hypergraph: the vertex count ($|\mathcal{V}|$), the net count ($|\mathcal{N}|$), and the pin count:

$$\sum_{n \in \mathcal{N}} deg(n) = \sum_{v \in \mathcal{V}} deg(v). \tag{2.7}$$

When the partitioning involve more than one constraint, a vertex v is associated with T weights. Here, T denotes the constraint count. $w(v)$ is used to denote the weight of the vertex v. If multiple weights are assigned to vertex v, $w_t(v)$ is used to denote the tth weight of the vertex v.

For a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ is a K-way partitioning of the set of vertices $\mathcal{V}$ if the K parts are pairwise disjoint and mutually exhaustive. A K-way vertex partition of $\mathcal{H}$ is said to satisfy the partitioning constraint if

$$W_t(\mathcal{V}_k) \leq W_t^{avg}(1+\varepsilon), \quad \text{for } k=1,2,\ldots,K; \text{ and for } t=1,2,\ldots,T. \tag{2.8}$$

Here, part $\mathcal{V}_k$'s weight $W_t(\mathcal{V}_k)$ for the tth constraint is defined as the sum of the weights $w_t(v)$ of the vertices in that part, i.e.,

$$W_t(\mathcal{V}_k) = \sum_{v\in\mathcal{V}_k} w_t(v), \tag{2.9}$$

W_t^{avg} is the average part weight, i.e.,

$$W_t^{avg} = \frac{\sum_{v\in\mathcal{V}} w_t(v)}{K}, \tag{2.10}$$

and ε represents the predetermined, maximum allowable imbalance ratio.

In a partition $\Pi(\mathcal{V})$ of $\mathcal{H}$, a net that has at least one pin (vertex) in a part is said to *connect* that part. *Connectivity set* $\Lambda(n)$ of a net n is defined as the set of parts connected by n. *Connectivity* $\lambda(n)=|\Lambda(n)|$ of a net n denotes the number of parts connected by n. A net n is said to be *cut* (*external*) if it connects more than one part (i.e., $\lambda(n)>1$), and *uncut* (*internal*) otherwise (i.e., $\lambda(n)=1$). The set of cut nets of a partition Π is denoted as $\mathcal{N}_{\text{cut}}$. A vertex v is a *boundary* vertex if and only if a cut net connects the vertex v. If the vertex v is not connected by any cut net, this vertex v is an *internal* vertex.

In partitioning of a hypergraph, the objective of partitioning is minimizing the cutsize. The cutsize of a partition can be defined on the costs of the cut nets. One of the definitions for cutsize is [29]:

$$cutsize(\Pi(\mathcal{V})) = \sum_{n\in\mathcal{N}_{\text{cut}}} c(n)\Big(\lambda(n)-1\Big) \tag{2.11}$$

Here, each cut net n contributes the cost of $c(n)(\lambda(n)-1)$ to the cutsize. This cutsize definition is also known as "connectivity-1" metric.

Another definition for cutsize is [29]:

$$cutsize(\Pi(\mathcal{V})) = \sum_{n\in\mathcal{N}_{\text{cut}}} c(n) \tag{2.12}$$

Here, each cut net n contributes the cost of $c(n)$ to the cutsize irregardless of its connectivity set. This cutsize definition is also known as “cutnet” metric. The HP problem is shown to be in the set of NP-hard problems [30].

Chapter 3

Related Work

Some of the computational kernels in linear algebra are classified according to the BLAS (Basic Linear Algebra Subprograms) [31] standard as follows:

- Level 1: scalar, vector and vector-vector operations
- Level 2: matrix-vector operations
- Level 3: matrix-matrix operations

The classification is also valid for sparse vectors and matrices [32]. The focus of this thesis is sparse matrix multiplication, which is a sparse Level 3 operation.

Gustavson [33] propose an efficient sequential SpGEMM algorithm. In [33], different formulations of SpGEMM are analyzed in terms of the data accesses and the number of multiplications. The most efficient scheme is reported to be row-by-row formulation against the inner-product formulation, because all data accesses contribute to output in the row-by-row formulation, whereas all data access may not yield a result because of merge operations in the inner-product formulation. The algorithm in [33], which is based on row-by-row formulation, is also used in MATLAB as its dual form of column-by-column formulation [34]. Our SpGEMM library also uses this algorithm as sequential kernels of the library.

There are successful SpGEMM libraries for distributed memory architectures. These libraries do not perform symbolic SpGEMM prior to numerical SpGEMM so they perform dynamic memory allocation during multiplication. Here and hereafter, parallelization of the SpGEMM computation in the form of $C = AB$ on a system having K processors will be considered. Tpetra [35] package of Trilinos [36] uses one-dimensional rowwise partitioning of the input matrices A and B. The parallel algorithm used by Tpetra has K stages. At each stage, blocks of B matrix are shifted among neighboring processors on a virtual ring of processors so that each one of the processors has a block of rows of the output matrix C at the end. This parallelization scheme is based on the row-by-row formulation.

Combinatorial BLAS (CombBLAS) [37] adopts the SUMMA (Scalable Universal Matrix Multiplication Algorithm) [15]. The serial SpGEMM algorithm of CombBLAS uses hypersparse matrix multiplication kernel [16], which has a run time complexity proportional to the number of nonzeros of the input matrices. Since multiplication of irregularly sparse matrices incurs load imbalance in a parallel SpGEMM algorithm, the matrices that will be multiplied by CombBLAS are randomly permuted in order to balance computational loads of processors. SUMMA [38] uses 2D checker-board partitioning of the input matrices A and B. This algorithm runs on a mesh of $\sqrt{K} \times \sqrt{K}$ processors. It involves a consecutive series of $\sqrt{K}$ number of broadcasts along rows and columns of the processor grid. Rowwise broadcasts consist of the row blocks of the input matrix A and columnwise broadcasts consist of the column blocks of the input matrix B. Each processor is responsible for computing a block of the 2D checkerboard partitioned output matrix C. At each stage, local blocks of the output C matrix are updated via multiplying the received input matrices. At the end of $\sqrt{K}$ stages, the final output matrix C is obtained.

The work [39] investigate the cost of communication occurred during parallel SpGEMM operation involving sparse random matrices. Tighter lower bounds are provided by Ballard et. al. [39] for the expected costs of SpGEMM operation. Two recursive and iterative SpGEMM algorithms based on three-dimensional partitioning are proposed in [39] in order to achieve the provided bounds. These

SpGEMM algorithms are reported to be adaptation of previous matrix multiplication algorithms [40, 41] for dense matrices. These algorithms also do not benefit from the sparsity patterns of the input and output matrices.

3.1 Sample Applications that Utilize SpGEMM

Here, two sample applications, which use parallel SpGEMM operations, are discussed. In molecular dynamics simulations, CP2K program [9] utilizes SpGEMM operations in performing parallel atomistic and molecular simulations of biological systems, liquids, and solid state materials. Parallel SpGEMM operations in the form of $C = AA$ (i.e., $C = A^2$) are performed in iterations of Newton-Schulz method. This method is used to calculate the result of the sign function for an input matrix A. This kernel is reported to occupy at least half of the total simulation time [2]. CP2K uses Cannon's algorithm [42] for performing parallel SpGEMM operations [43].

In order to solve large linear programming (LP) problems, iterative interior point methods are generally utilized. At each iteration of the solvers of LP problems, these methods try to find solutions to the normal equations having the form of $(AD^2A^T)x = b$ to determine search directions. Here, A is a sparse matrix and it defines the constraints of the problem. D is a positive matrix, which has nonzeros in only diagonal entries. While the normal equations are being solved, direct solvers [19, 20, 21] based on Cholesky factorization and iterative solvers [21] that use preconditioners need explicitly-formed coefficient matrix. In every outer iteration, the coefficient matrix is formed through the SpGEMM operation. The nonzero structures of input matrices A and $B = D^2A^T$ remain the same throughout the iterations. Among the most time consuming parts of the LP solvers, i.e., SpGEMM and Cholesky factorization operations, for some LP problems, the SpGEMM computation may take significantly longer than Cholesky factorization [21].

The above-mentioned libraries and works do not benefit from the nonzero structures of the input or output matrices. In other words, they do not preprocess the input matrices A and B, or perform symbolic SpGEMM prior to numeric SpGEMM to obtain sparsity pattern of the output matrix C. However, the sparsity pattern of matrices can be used to reduce communication overhead during the parallel SpGEMM operation. This idea is widely used in sparse matrix-vector multiplication (SpMV) [29, 44].

To our knowledge, this thesis is the first attempt to preprocess the input matrices in order to obtain an efficient parallelization. So symbolic multiplication is performed for obtaining the computation pattern, which will be partitioned. This step is necessary in order to obtain a "good" partitioning of these matrices, so that the obtained matrix partitioning incurs less communication overhead and yields better balance on computational loads of processors during the SpGEMM operation. Depending on the nonzero structures of matrices, our aim is to devise sophisticated matrix partitioning methods that achieve reducing communication overhead and obtaining computational load balance during the parallel SpGEMM operations on distributed memory architectures.

Chapter 4

Parallel SpGEMM Algorithms

We consider the parallelization of the SpGEMM operation of the form $C = AB$: We propose four parallel algorithms based on 1D partitioning of the two input matrices A and B as follows:

- Column-by-Row-product parallel (a.k.a. outer-product parallel) (CRp)
- Row-by-Column-product parallel (a.k.a. inner-product parallel) (RCp)
- Row-by-Row-product parallel (RRp)
- Column-by-Column-product parallel (CCp)

Note that in the abbreviated names of the above-mentioned SpGEMM algorithms, the first capital letters stand for the rowwise or columnwise partitioning of input matrix A, whereas the second one stand for the rowwise or columnwise partitioning of input matrix B. Also note that these parallel algorithms depend on the matrix multiplication formulations described in Section 2.1.

The CRp algorithm is based on conformable columnwise partitioning of A matrix and rowwise partitioning of B matrix. In the CRp algorithm, the outer product of column i of A with row i of B is defined as an atomic task, so that the

SpGEMM operation is split into concurrent outer-product computations. In this algorithm, entries of both input matrices are accessed only once, whereas output matrix entries are accessed multiple times.

The RCp algorithm is based on rowwise partitioning of A matrix and columnwise partitioning of B matrix. In the RCp algorithm, the inner product of a row of A matrix with a column of B matrix is an atomic task, so that the SpGEMM operation is split into concurrent inner-product computations. RCp has two variants: A-resident and B-resident. In A-resident RCp, A- and C-matrix entries are accessed only once, whereas B-matrix entries are accessed multiple times. In B-resident RCp, B- and C-matrix entries are accessed only once, whereas A-matrix entries are accessed multiple times. Since these two variants are dual, we will only consider A-resident RCp throughout this thesis. In other words, the B-resident RCp algorithm can easily be derived from the A-resident RCp algorithm.

The RRp algorithm is based on rowwise partitioning of both A and B matrices. In the RRp algorithm, pre-multiply of a row of A matrix with whole B matrix is defined as an atomic task, so that the SpGEMM operation is split into concurrent vector-matrix multiplications. A- and C-matrix entries are accessed only once, whereas B-matrix entries are accessed multiple times in this algorithm.

The CCp algorithm is based on columnwise partitioning of both A and B matrices. In the CCp algorithm, post-multiply of whole A matrix with a column of B is defined as an atomic task, so that the SpGEMM operation is split into concurrent matrix-vector multiplications. B- and C-matrix entries are accessed only once, whereas A-matrix entries are accessed multiple times in this algorithm. Since CCp is dual of RRp, CCp will not be discussed in the rest of the thesis. Note that the rows of the C matrix are accessed once in both RRp and CCp algorithms.

Figure 4.1 presents the data access requirements of the above-mentioned four algorithms with respect to the input and output matrices A, B, and C. As seen in this figure, all algorithms require multiple accesses of only one matrix, whereas the remaining two matrices are accessed only once. As also seen in the figure, only

CRp requires multiple accesses of the output matrix. Single accesses denoted by "s" do not incur communication because partitions of the respective matrix reside at the owner processor. Multiple accesses denoted by "m" incur communication of the respective input/output matrix because partitions of the matrix is required by processors other than the owner processor.

Table 4.1: Data access requirements of the four parallel SpGEMM algorithms.

Parallel SpGEMM algorithms			C	A	B
CRp	outer-product parallel		m	s	s
RCp	inner-product parallel	A-resident	s	s	m
		B-resident	s	m	s
RRp	row-by-row–product parallel		s	s	m
CCp	column-by-column–product parallel		s	m	s

"s" denotes single access, whereas "m" denotes multiple accesses to the rows/columns/nonzeros of matrices.

4.1 Outer-Product–Parallel SpGEMM Algorithm (CRp)

In outer-product–parallel SpGEMM algorithm (CRp), conformable 1D columnwise and 1D rowwise partitioning of the input matrices A and B are used as follows:

$$\hat{A} = AQ = \begin{bmatrix} A_1 & A_2 & \dots & A_K \end{bmatrix} \quad \text{and} \quad \hat{B} = QB = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_K \end{bmatrix} \tag{4.1}$$

Here, K denotes the number of parts and Q is the permutation matrix obtained from the partitioning. In Equation (4.1), the same permutation matrix Q is used to reorder columns of matrix A and rows of matrix B in order to achieve conformable columnwise and rowwise partitioning of matrices A and B. According to the input partitioning given in Equation (4.1), each processor P_k owns column slice A_k and row slice B_k of the permuted matrices. Consequently, as a result of the conformable partitioning of input matrices, each processor P_k performs the outer-product computation of $A_k B_k$ without any communication. Note that any row/column replication is not considered in the input data partitioning given in Equation (4.1).

According to the input matrix partitioning given in Equation (4.1), the output C matrix is calculated as follows using the results of the local SpGEMM computations:

$$C = C^1 + C^2 + \ldots + C^K \quad (C^k = A_k B_k \text{ is performed by processor } P_k). \tag{4.2}$$

Communication occurs in the summation of local C^k matrices. This is because many single-node-accumulation (SNAC) operations are needed to compute the final value of nonzero c_{ij} of C using the partial results generated by the local SpGEMM operations. In this CRp algorithm, the multiplication phase consists of local SpGEMM computations without any communication and the communication phase consists of many SNAC operations.

The input partitioning on matrices A and B does not yield a natural and inherent output partitioning on the C matrix. The processor that will be assigned the responsibility of summing all the partial results for each nonzero $c_{i,j}$ of C is determined by the output partitioning. Summation of all the partial results for each nonzero $c_{i,j}$ of C is defined as

$$c_{i,j} = \sum_{c_{i,j}^{(k)} \in C^k} c_{i,j}^{(k)}. \tag{4.3}$$

Here, $c_{i,j}^{(k)} \in C^k$ means that $c_{i,j}^{(k)}$ is a nonzero of C^k and so this value is a partial result for $c_{i,j}$ of C. The performance of the computation phase, which depends on the input partitioning, is directly related with the computational load balance among processors, whereas the overhead due to communication of partial results is the performance bottleneck in the communication phase, which depends on the output partitioning.

For output partitioning, 2D partitioning of the output matrix C will be considered. Here, the 2D output partitioning is based on partitioning nonzeros of matrix C so that the atomic tasks in communication phase is defined as the tasks of computing nonzeros of C. In this SpGEMM algorithm, the worst-case communication is $(K-1)nnz(C)$ words and $K(K-1)$ messages. Here, the number of nonzeros in a matrix is denoted by $nnz(\cdot)$. This worst case communication happens when a partial result for every nonzero of the output C matrix is generated by every local SpGEMM computation.

4.2 Inner-Product–Parallel SpGEMM Algorithm (RCp)

The inner-product–parallel SpGEMM algorithm (RCp) is based on 1D rowwise partitioning of A and C; and 1D columnwise partitioning of B as follows:

$$\hat{A} = PA = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_K \end{bmatrix}, \ \hat{B} = BQ = \begin{bmatrix} B_1 \ B_2 \cdots \ B_K \end{bmatrix}, \ \text{and}$$

$$\hat{C} = PCQ = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_K \end{bmatrix} \tag{4.4}$$

Here, K denotes the number of parts; and P and Q denote the permutation matrices obtained from partitioning. The use of the same permutation matrix for row reordering of A and row reordering of C shows that the rowwise partition on the input matrix A directly induces a rowwise partition on the output matrix C. In the input partitioning given in Equation (4.4), each processor P_k owns a row slice A_k and column slice B_k of the permuted matrices.

According to the input and output data partitioning given in Equation (4.4), the output matrix C is computed as follows:

$$C_k = A_k B \ \text{ for } \ k = 1, 2, \ldots, K. \tag{4.5}$$

Each submatrix-matrix multiplication $C_k = A_k B$ will be assigned to a processor of the parallel system. A nice property of this partitioning scheme is that A and C matrices are not communicated. However, columns of the input matrix B must be replicated to the processors that need these B-matrix columns for the computation of $C_k = A_k B$. The replication of B matrix constitutes the communication phase of the parallel SpGEMM algorithm. After the communication phase, $C_k = A_k B$ is performed without any communication in the multiplication phase. The worst case communication of this algorithm is $(K-1)nnz(B)$ words and $K(K-1)$ messages. This worst case communication happens when each $C_k = A_k B$ multiplication requires whole matrix B.

4.3 Row-by-Row–Product–Parallel SpGEMM (RRp)

The row-by-row–product–parallel SpGEMM algorithm (RRp) is based on 1D rowwise partitioning of A, B, and C matrices as follows:

$$\hat{A} = PAQ = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_K \end{bmatrix}, \quad \hat{B} = QB = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_K \end{bmatrix}, \text{and} \quad \hat{C} = PC = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_K \end{bmatrix} \tag{4.6}$$

Here, K denotes the number of parts; and P and Q denote the permutation matrices obtained from partitioning. The use of the same permutation matrix for row reordering of A and row reordering of C shows that the rowwise partition on the input matrix directly induces a rowwise partition on the output matrix.

According to the input and output data partitioning given in Equation (4.6), the output matrix C is computed as follows:

$$C_k = A_k B \quad \text{for} \quad k = 1, 2, \ldots, K. \tag{4.7}$$

Each submatrix-matrix multiplication $C_k = A_k B$ will be assigned to a processor of the parallel system. A nice property of this partitioning scheme is that A and

C matrices are not communicated. However, rows of the input matrix B must be replicated to processors that need the respective rows for the computation of $C_k = A_k B$. The replication of B matrix constitutes the communication phase of the parallel SpGEMM algorithm. After the communication phase, $C_k = A_k B$ is performed without any communication in the multiplication phase. The worst case communication of this algorithm is $(K-1)nnz(B)$ words and $K(K-1)$ messages. This worst case communication happens when each $C_k = A_k B$ multiplication requires whole matrix B.

Note that the definitions of local SpGEMM operations in RCp and RRp algorithms are same. They only differ in the partitioning of B matrix, i.e. B matrix is partitioned columnwise in RCp and rowwise in RRp. The use of the same definition of local SpGEMM operations is because only two variants of RCp algorithm are given as depicted in Table 4.1 in order to obey the owner computes rule. Hence, the owner computes rule enables avoiding unnecessary communication of either one of the input matrices in these variants.

Chapter 5

Hypergraph Models for Parallel SpGEMM Algorithms

In this thesis, we propose one hypergraph model for each one of the three parallel SpGEMM algorithms proposed in Chapter 4. These three hypergraphs contain a vertex to represent each atomic task of local SpGEMM operation. These hypergraphs also contain a vertex for each entity of the communicated matrix to enable simultaneous input and output partitioning. These hypergraphs contain a net (hyperedge) for each communicated entity (row/column/nonzero) of the corresponding matrix in order to represent the total volume of communication that occur in the communication phase of the SpGEMM algorithms. By using these hypergraph models, hypergraph partitioning (HP) methods are proposed to simultaneously partition the input and output matrices in one step. That is, partitioning of input and output matrices are performed in a single partitioning process. The constraint in the partitioning of these hypergraph models corresponds to computational load balancing among processors for the multiplication phase of the parallel SpGEMM algorithms. The objective of minimizing the cutsize corresponds to minimizing the total volume of communication that occurs in the communication phase of the parallel algorithms.

Note that the proposed models do not use any of the previously proposed hypergraph models (e.g., column-net, row-net, and row-column-net (finegrain) hypergraph models [45, 29]) for partitioning sparse matrices for SpMV. The aim of the hypergraph models proposed in this thesis is representing the SpGEMM computations.

The proposed hypergraph models and HP-based methods are tested on many SpGEMM instances arising in various applications. The well-known HP tool PaToH [46] is used the partition the hypergraph models of the SpGEMM instances. In order to show that the theoretical improvements due to the hypergraph models are also valid in practice, an MPI (Message Passing Interface)-based [26] SpGEMM library [25] is designed and developed using the C programming language. Parallel SpGEMM runs on large-scale distributed-memory IBM BlueGene/Q system, named JUQUEEN, show that the proposed models and methods achieve good scalability and high speedup.

Note that the constructions of the hypergraph models for CRp and RCp require apriori knowledge of the pattern of computation that yields the output matrix C. Through performing symbolic SpGEMM, this pattern of computation is obtained from the nonzero structures of the input matrices A and B. This requirement of symbolic multiplication before partitioning is an important difference, when the proposed partitioning methods are compared against the partitioning methods for parallel SpMV. This is because the computational structure of the SpMV operation directly and solely depends on the sparsity pattern of the input matrix A.

5.1 The Hypergraph Model $\mathcal{H}_{\mathrm{cr}}$ for CRp

In this model, an SpGEMM computation of $C = AB$ is represented as a hypergraph $\mathcal{H}_{\mathrm{cr}} = (\mathcal{V} = \mathcal{V}^{AB} \cup \mathcal{V}^{C}, \mathcal{N})$ for 1D conformable columnwise partitioning of A and rowwise partitioning of B, and the nonzero-based 2D partitioning of the output C matrix. The vertices in $\mathcal{V}^{AB}$ are referred to as input vertices and the vertices in $\mathcal{V}^{C}$ are referred to as output vertices. There exists an input vertex v_x in $\mathcal{V}^{AB}$ for each column x of A and row x of B. There exist both an output vertex $v_{i,j}$ in $\mathcal{V}^{C}$ and a net $n_{i,j}$ in $\mathcal{N}$ for each nonzero $c_{i,j}$ of the output C matrix. $n_{i,j}$ connects v_x if and only if there exists a nonzero at row i and column x of A; and there exists a nonzero at row x and and column j of B. That is, $n_{i,j}$ connects v_x if the outer-product computation of column x of A with row x of B yields a partial result for $c_{i,j}$ of C. $n_{i,j}$ also connects $v_{i,j}$. So $n_{i,j}$ is defined as follows:

$$Pins(n_{i,j}) = \{v_x : a_{i,x} \in A \wedge b_{x,j} \in B\} \cup \{v_{i,j}\}. \tag{5.1}$$

As seen in Equation (5.1), each net $n_{i,j}$ connects at least one input vertex v_x and exactly one output vertex $v_{i,j}$. Note that, the definitions of input vertices contain single subscript, whereas the definitions of nets and output vertices contain double subscript.

The number of vertices, nets, and pins of the proposed hypergraph model $\mathcal{H}_{\mathrm{cr}}$ can be expressed using the properties of input A and B matrices, and the output C matrix:

$$|\mathcal{V}| = nnz(C) + cols(A) = nnz(C) + rows(B), \tag{5.2}$$

$$|\mathcal{N}| = nnz(C), \tag{5.3}$$

$$\#\ of\ pins = \sum_{x=1}^{cols(A)} \Big(nnz(a_{*,x}) \cdot nnz(b_{x,*}) \Big) + nnz(C). \tag{5.4}$$

Here, $rows(\cdot)$ and $cols(\cdot)$ respectively represent the number of rows and the number of columns of a given matrix. The summation term in Equation (5.4) given for calculating the number of pins of $\mathcal{H}_{\mathrm{cr}}$ is equal to the number of multiply-and-add operations to be executed in an SpGEMM computation $C = AB$.

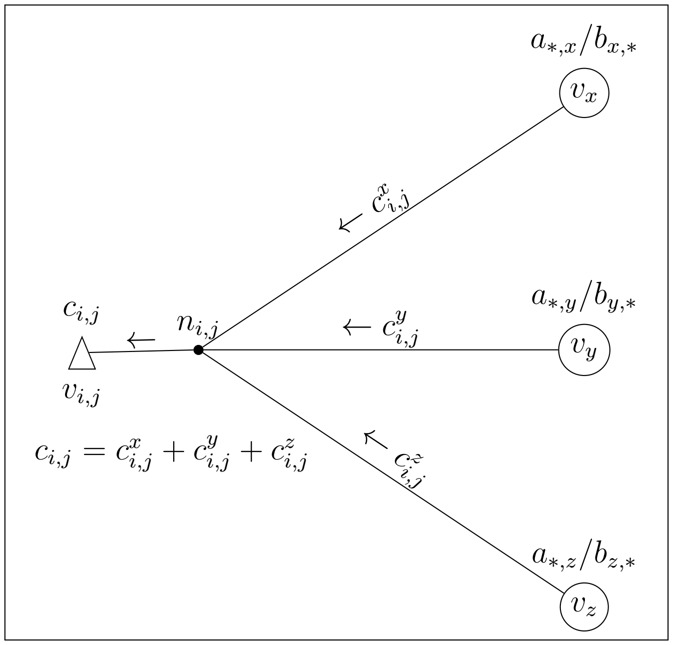


Figure 5.1: The proposed hypergraph model $\mathcal{H}_{\text{cr}}$ for CRp

Two weights are assigned to each vertex in order to encode the computational costs of the multiply-and-add operations in the multiplication phase and the summation operations in the communication phase of CRp. In other words, the first and second weights of a vertex correspond to the computational loads of the atomic task represented by that vertex for the multiplication and communication phases, respectively. A vertex v_x in $\mathcal{V}^{AB}$ corresponds to the atomic task of computing the outer product of column x of A with row x of B. The outer-product computation $a_{*,x}b_{x,*}$ yields $nnz(a_{*,x}) \cdot nnz(b_{x,*})$ multiply-and-add operations to obtain $nnz(a_{*,x}) \cdot nnz(b_{x,*})$ number of partial results. Hence, vertex v_x is assigned the following two weights:

$$w_1(v_x) = nnz(a_{*,x}) \cdot nnz(b_{x,*}), \qquad\qquad w_2(v_x) = 0. \tag{5.5}$$

Note that $w_1(v_x)$ also corresponds to the number of nets that connect the input vertex v_x, i.e.,

$$deg(v_x) = w_1(v_x). \tag{5.6}$$

Each vertex $v_{i,j}$ in $\mathcal{V}^C$ corresponds to the the atomic task of computing $c_{i,j}$ via summing the partial results generated by the outer-product computations, i.e.,

$$c_{i,j} = \sum_{v_x \in Pins(n_{i,j})} c_{i,j}^x, \tag{5.7}$$

and storing the final result $c_{i,j}$. Each net $n_{i,j}$ corresponds to the dependency of the calculation of $c_{i,j}$ to the outer-product operations, i.e., the vertices in $Pins(n_{i,j}) - \{v_{i,j}\}$ correspond to the set of partial outer-product results that are required for calculating the final value of $c_{i,j}$.

Figure 5.1 shows the hypergraph model $\mathcal{H}_{\text{cr}}$ and encoding of the input and output dependencies. As shown in the figure, the net $n_{i,j}$ with $Pins(n_{i,j}) = \{v_x, v_y, v_z, v_{i,j}\}$ corresponds to the outer-product computations $a_{*,x}b_{x,*}$, $a_{*,y}b_{y,*}$, and $a_{*,z}b_{z,*}$, which yield partial results $c_{i,j}^x$, $c_{i,j}^y$, and $c_{i,j}^z$, respectively. Hence, vertex $v_{i,j}$ corresponds to the atomic task of computing the final value of $c_{i,j}$ via the summation $c_{i,j} = c_{i,j}^x + c_{i,j}^y + c_{i,j}^z$. Here, $c_{i,j}^x$ denotes the partial result for $c_{i,j}$ generated by the outer product $a_{*,x}b_{x,*}$. Hence, the following two weights are assigned to vertex $v_{i,j}$:

$$w_1(v_{i,j}) = 0, \qquad w_2(v_{i,j}) = |Pins(n_{i,j})| - 1. \tag{5.8}$$

As seen in Equations (5.5) and (5.8), the first and second vertex weights correspond the computational loads of the atomic tasks associated with these vertices in the multiplication and communication phases of CRp, respectively. Hence, the second weights of the input vertices are set to be equal to zero (i.e., $w_2(v_x) = 0$ in Equation (5.5)) because the atomic tasks associated with the input vertices do not incur any computation in the communication phase. Similarly, the first weights of the output vertices are set to be equal to (i.e., $w_1(v_{i,j}) = 0$ in Equation (5.8)) because the atomic tasks associated with the output vertices do not incur any computation in the multiplication phase.

Each net $n_{i,j}$ represents the dependency related with only one nonzero $c_{i,j}$, cost $c(n_{i,j})$ is set to be equal to one, i.e.,

$$c(n_{i,j}) = 1. \tag{5.9}$$

A K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ on $\mathcal{V}$ inherently induces a partition $\Pi(\mathcal{V}^{AB})$ on $\mathcal{V}^{AB} \subseteq \mathcal{V}$ and a partition $\Pi(\mathcal{V}^{C})$ on $\mathcal{V}^{C} \subseteq \mathcal{V}$. Note that, that part $\mathcal{V}_k$ is assumed to be assigned to processor P_k for $k = 1, 2, \ldots, K$ without loss of generality. $\Pi(\mathcal{V}^{AB})$ is decoded as an input partition on the columns of A and rows of B; and $\Pi(\mathcal{V}^{C})$ is decoded as an output partition on the nonzeros of matrix C. That is, v_x in $\mathcal{V}_k$ means that column $a_{*,x}$ of A and row $b_{x,*}$ of B are stored by only processor P_k and the responsibility of performing the outer-product computation $a_{*,x}b_{x,*}$ without any communication is assigned to P_k in accordance with the owner computes rule. $v_{i,j}$ in $\mathcal{V}_k$ denotes that responsibility of summing the partial results for computing the final result of $c_{i,j}$ and storing $c_{i,j}$ is assigned to processor P_k.

5.1.1 Model Correctness

The correctness of the proposed hypergraph model $\mathcal{H}_{\text{cr}}$ can be proved by showing the following:

(a) The two partitioning constraints on part weights correspond to balancing computational loads of processors during the two phases of CRp.

(b) The partitioning objective of minimizing cutsize corresponds to the minimization of the total volume of communication during the communication phase of CRp.

Consider a K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ of vertices of $\mathcal{H}_{\text{cr}}$ for both (a) and (b).

For (a), $\Pi(\mathcal{V})$ is assumed to satisfy balance constraints given in Equation (2.8) for $T = 2$. Considering the first weights given in Equations (5.5) and (5.8), the first partitioning constraint correctly encodes balancing the computational loads in terms of number of multiply-and-add operations in local outer products to be performed by processors in the multiplication phase. Considering the second weights given in Equations (5.5) and (5.8), the second partitioning constraint correctly encodes balancing the number of local summation operations on the partial-results to be performed by processors in the communication phase.

The above-mentioned correctness of the second partitioning constraint depends on a naive implementation. In this naive implementation scheme, each processor obtains separate output C matrix for each local outer-product computation rather than accumulating on a single local output matrix C. In an efficient implementation scheme, each processor P_k accumulates results of its outer-product computations on a single local output C matrix just after every local outer-product computation as follows

$$C^k = C^k + a_{*,x} b_{x,*}, \quad \text{where} \quad v_x \in \mathcal{V}_k. \tag{5.10}$$

The correctness of the first partitioning constraint for the multiplication phase is not disturbed in this efficient implementation scheme because each scalar multiply operation incurs a scalar addition operation as follows

$$c^x_{i,j} = c^x_{i,j} + a_{i,x} \cdot b_{x,j}. \tag{5.11}$$

However, the correctness of the second partitioning constraint is disturbed for the communication phase. Anyway, the second partitioning constraint may still be used to enforce balancing the computational loads of the local summations during the communication phase of this efficient implementation scheme, because such errors are expected to happen for the second weights of the vertices in all parts of a partition.

For showing (b), consider an output vertex $v_{i,j}$, which is assigned to $\mathcal{V}_k$ (i.e., $v_{i,j} \in \mathcal{V}_k$). Recall that each net $n_{i,j}$ connects only one output vertex, which is $v_{i,j} \in \mathcal{V}_k$. Then, each part $\mathcal{V}_m \in \Lambda(n_{i,j}) - \{\mathcal{V}_k\}$ has at least one input vertex corresponding to an outer-product computation that yields a partial results for $c_{i,j}$. So, for each part $\mathcal{V}_m \in \Lambda(n_{i,j}) - \{\mathcal{V}_k\}$, processor P_m computes a partial result

$$c^{(m)}_{i,j} = \sum_{v_x \in \mathcal{V}_m} c^x_{i,j} \tag{5.12}$$

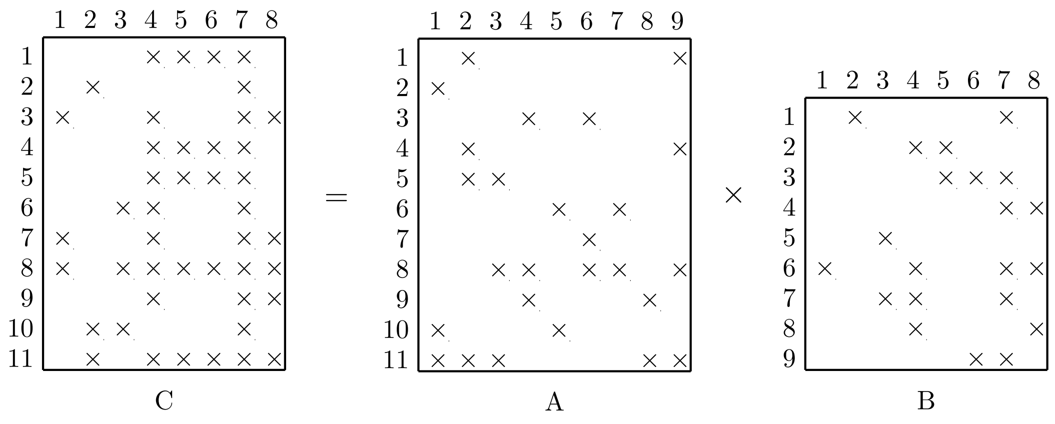


Figure 5.2: A sample SpGEMM computation of the form $C = AB$

from the results of its local outer-product computations. Here, $c_{i,j}^{(m)}$ denotes the sum of local partial results obtained from outer-product computations corresponding to vertices $v_x \in \mathcal{V}_m$. After the outer-product computations, processor P_m sends $c_{i,j}^{(m)}$ to processor P_k. Hence, $v_{i,j} \in \mathcal{V}_k$ means that processor P_k will receive only one partial result from each of the $\lambda(n_{i,j}) - 1$ processors in $\Lambda(n_{i,j}) - \{\mathcal{V}_k\}$. After receiving these partial results, processor P_k will accumulate them for computing the final value of $c_{i,j}$. As seen in Equation (2.11), the contribution of net $n_{i,j}$ to cutsize is $\lambda(n_{i,j}) - 1$. As a result, the equivalence between $\lambda(n_{i,j}) - 1$ and the communication volume occurred in the summation of $c_{i,j}$ in the communication phase is shown. Consequently, the total communication volume in this communication phase is correctly encoded by the cutsize given in Equation (2.11).

In order to illustrate the proposed hypergraph model $\mathcal{H}_{\mathrm{cr}}$, a sample SpGEMM computation and its hypergraph model are included. Figure 5.2 displays a sample SpGEMM operation. The input matrices are A and B are 11×9 and 9×8 matrices that have 26 and 21 nonzeros, respectively. The output matrix C is an 11×8 matrix that has 44 nonzeros. There are 9 outer-product computations in the multiplication of these A and B matrices.

Figure 5.3 illustrates the hypergraph model $\mathcal{H}_{\mathrm{cr}}$ that is used for modeling the SpGEMM operation shown in Figure 5.2. The input and output vertices are respectively represented by circles and triangles in Figure 5.3. There are $9+44 = 53$ vertices in $\mathcal{H}_{\text{cr}}$ as seen in the figure. There are also $\sum_{x=1}^{9} deg(v_x) = 61$ pins in $\mathcal{H}_{\text{cr}}$. As also seen in the figure, $deg(v_4) = 6$ since $nnz(a_{*,4}) \cdot nnz(b_{4,*}) = 3 \cdot 2 = 6$.

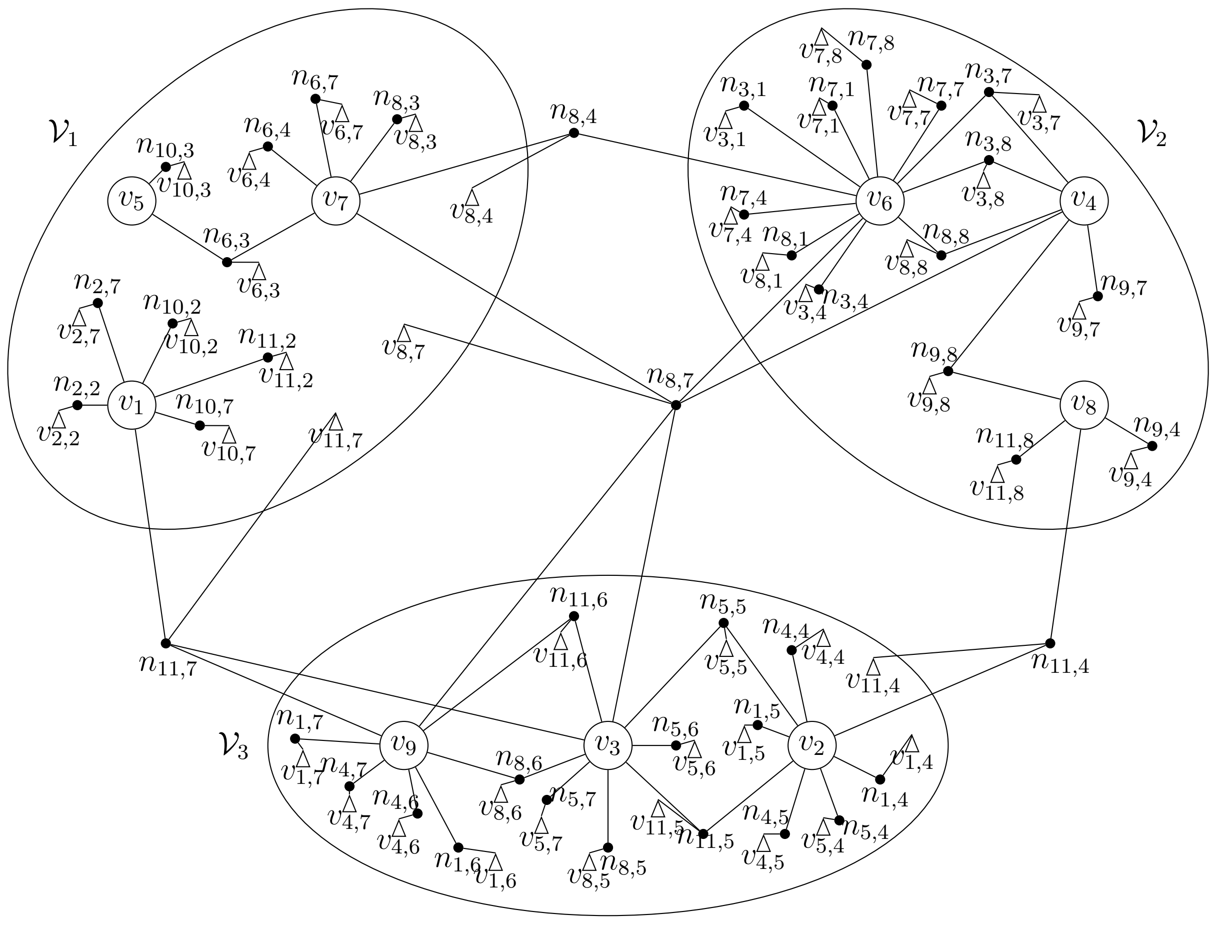


Figure 5.3: Hypergraph model $\mathcal{H}_{\text{cr}}$ for representing the SpGEMM operation shown in Figure 5.2 and three-way partition $\Pi(\mathcal{V})$ of this hypergraph. Each round vertex v_x shown in the figure corresponds to the atomic task of performing the $a_{*,x}b_{x,*}$ outer product. Each triangular vertex $v_{i,j}$ corresponds to the atomic task of computing final value of nonzero $c_{i,j}$ of matrix C. Each net $n_{i,j}$ corresponds to the dependency between the task of summing partial result for $c_{i,j}$ and the outer product computations that yields a partial result for $c_{i,j}$.

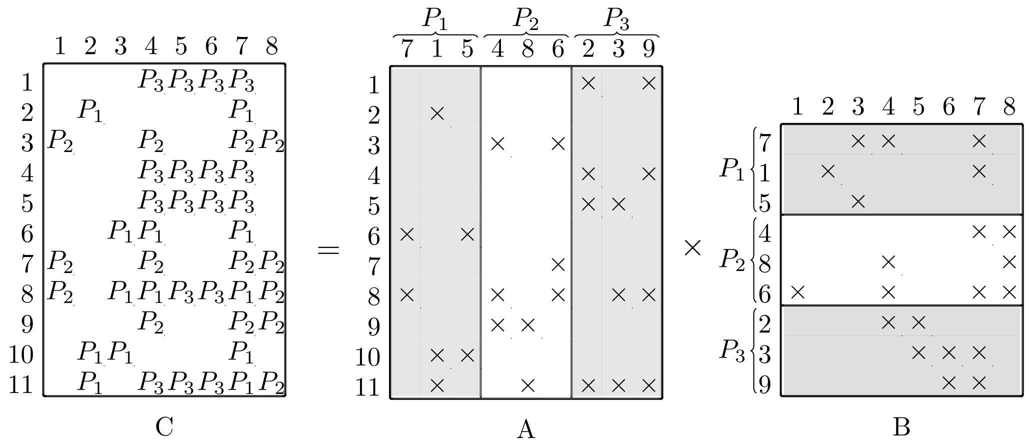


Figure 5.4: Matrices A, B, and C that are partitioned according to the partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\text{cr}}$ given in Figure 5.3

A three-way partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\text{cr}}$ is also shown in Figure 5.3. The three-way partition of the sample input and output matrices induced by this $\Pi(\mathcal{V})$ are shown in Figure 5.4. In $\Pi(\mathcal{V})$ of $\mathcal{H}_{\text{cr}}$, $W_1(\mathcal{V}_2) = w_1(v_4) + w_1(v_6) + w_1(v_8) = deg(v_4) + deg(v_6) + deg(v_8) = 6 + 12 + 4 = 22$. Similarly, $W_1(\mathcal{V}_1) = 14$ and $W_1(\mathcal{V}_3) = 25$. As a result, a percent load imbalance value of 23% on the first vertex weights is incurred by this $\Pi(\mathcal{V})$. In $\Pi(\mathcal{V})$, $W_2(\mathcal{V}_1) = 13$, $W_2(\mathcal{V}_2) = 14$, and $W_2(\mathcal{V}_3) = 17$ since parts $\mathcal{V}_1$, $\mathcal{V}_2$, and $\mathcal{V}_3$ contain 13, 14, and 17 output vertices as shown by triangles. As a result, considering this partition $\Pi(\mathcal{V})$, load imbalance on the second weights of vertices is equal to 16%.

In the three-way partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\text{cr}}$ shown in Figure 5.3, these four nets $n_{8,7}$, $n_{8,4}$, $n_{11,7}$, and $n_{11,4}$ are cut. All of the other 40 nets are uncut, in other words they are internal to a part. As shown in the figure, net $n_{8,7}$ has two pins in each part so $\lambda(n_{8,7}) = 3$. As a result, cut net $n_{8,7}$ incurs a cost of $\lambda(n_{8,7}) - 1 = 3 - 1 = 2$ to the cutsize. $v_{8,7}$ is in part $\mathcal{V}_1$ means that the responsibility of summing the partial nonzero results obtained from the local outer-product computations is assigned to processor P_1. P_1 will receive the partial result $c^{(2)}_{8,7} = c^4_{8,7} + c^6_{8,7}$ from P_2 and the partial result $c^{(3)}_{8,7} = c^3_{8,7} + c^9_{8,7}$ to P_1 from P_3. So, only two words will be communicated during the calculation of final value for $c_{8,7}$ by P_1. Hence, the equivalence between $\lambda(n_{8,7}) - 1$ and the communication volume occurred during the calculation of $c_{8,7}$ in the communication phase is shown. In a similar way, since $\lambda(n_{8,4}) - 1 = 1$, $\lambda(n_{11,7}) - 1 = 1$, and $\lambda(n_{11,4}) - 1 = 1$ for the remaining cut nets, total cutsize is equal to five. As a result, the total volume of communication is equal to five words. Therefore, the total volume of communication occurred in this phase is correctly encoded by the cutsize given in Equation (2.11).

5.1.2 Model Construction

Algorithm 4 shows the pseudocode for constructing the hypergraph model $\mathcal{H}_{\mathrm{cr}}$ for outer-product based multiplication of a given pair of A and B matrices stored in CSC and CSR formats [47], respectively. Here, CSC refers to compressed sparse columns and CSR refers to compressed sparse rows. For the sake of efficiency in constructing $\mathcal{H}_{\mathrm{cr}}$, Algorithm 4 utilizes the net-list representation instead of pin-list representation of hypergraphs. That is, net-list definition given below is utilized instead of pin-list definition given in Equation (5.1):

$$\begin{aligned} Nets(v_x) &= \{n_{i,j} : a_{i,x} \in A \wedge b_{x,j} \in B\}, && (5.13) \\ Nets(v_{i,j}) &= \{n_{i,j}\}. && (5.14) \end{aligned}$$

This net-list definition enables the proper allocation and construction of the net lists of the vertices in successive locations of a net lists array. For the current vertex v_x, a net list allocation of appropriate size and weight assignments are performed at lines 6 and 7, respectively. The test at line 10 is performed to identify a new net $n_{i,j}$ and hence an output vertex $v_{i,j}$ to assign the next available indices to $n_{i,j}$ and $v_{i,j}$ at lines 11 and 12, respectively, during the construction. The two weights for a new output vertex $v_{i,j}$ are initialized at line 14 and the second weight of an existing output vertex is incremented at line 16. Net $n_{i,j}$–irregardless of being a new or an existing net–is appended to the end of the net list of vertex v_x at line 18.

Algorithm 4 Construction of the hypergraph model $\mathcal{H}_{\text{cr}}$

```
Require: $A$ matrix in CSC format and $B$ matrix in CSR format
1: $\mathcal{V}^{AB} \leftarrow \varnothing$
2: $\mathcal{V}^{C} \leftarrow \varnothing$
3: $\mathcal{N} \leftarrow \varnothing$
4: for each column $x$ of $A$ do
5:     $\mathcal{V}^{AB} \leftarrow \mathcal{V}^{AB} \cup \{v_x\}$
6:     allocate net list of size $nnz(a_{*,x}) \cdot nnz(b_{x,*})$ for vertex $v_x$
7:     $w_1(v_x) \leftarrow nnz(a_{*,x}) \cdot nnz(b_{x,*})$;   $w_2(v_x) \leftarrow 0$
8:     for each nonzero $a_{\imath,x}$ in column $x$ of $A$ do
9:         for each nonzero $b_{x,j}$ in row $x$ of $B$ do
10:            if $n_{i,j} \notin \mathcal{N}$ then
11:                $\mathcal{N} \leftarrow \mathcal{N} \cup \{n_{i,j}\}$
12:                $\mathcal{V}^{C} \leftarrow \mathcal{V}^{C} \cup \{v_{i,j}\}$
13:                $Nets(v_{i,j}) \leftarrow \{n_{i,j}\}$
14:                $w_1(v_{i,j}) \leftarrow 0$;   $w_2(v_{i,j}) \leftarrow 1$
15:            else
16:                $w_2(v_{i,j}) \leftarrow w_2(v_{i,j}) + 1$
17:            end if
18:            $Nets(v_x) \leftarrow Nets(v_x) \cup \{n_{i,j}\}$
19:        end for
20:    end for
21: end for
22: return $\mathcal{H}_{\text{cr}}(\mathcal{V} = \mathcal{V}^{AB} \cup \mathcal{V}^{C}, \mathcal{N})$
```

5.2 The Hypergraph Model $\mathcal{H}_{\text{rc}}$ for RCp

In this model, an SpGEMM computation $C = AB$ is represented as a hypergraph $\mathcal{H}_{\text{rc}} = (\mathcal{V} = \mathcal{V}^A \cup \mathcal{V}^B, \mathcal{N})$ for 1D rowwise partitioning of A and C matrices; and 1D columnwise partitioning of B matrix. There exists a vertex $v_x \in \mathcal{V}^A$ for each row x of A. There exists a vertex $v_j \in \mathcal{V}^B$ for each column j of B. There exists a net $n_{i,j} \in \mathcal{N}$ for each nonzero $b_{i,j}$ of B. Net $n_{i,j}$ connects vertices corresponding to the rows that have nonzeros at ith column of A as well as v_j. That is, $n_{i,j}$ connects v_x if and only if the pre-multiply of row x of A with B requires nonzero $b_{i,j}$ of matrix B. So $n_{i,j}$ is defined as follows:

$$Pins(n_{i,j}) = \{v_x : a_{x,i} \in A \wedge b_{i,j} \in B\} \cup \{v_j\}. \tag{5.15}$$

As seen in Equation (5.15), each net $n_{i,j}$ connects at least one vertex that represents a column of matrix B. Note that single subscript is used for both types of vertices. When x is used in subscript, the vertex that represents the atomic task of computing pre-multiply is intended, whereas when j is used, the vertex that represents a column of matrix B is intended.

The number of vertices, nets, and pins of the proposed hypergraph model $\mathcal{H}_{\text{rc}}$ can be expressed using the attributes of input matrices A and B:

$$|\mathcal{V}| = rows(A) + cols(B), \tag{5.16}$$

$$|\mathcal{N}| = nnz(B), \tag{5.17}$$

$$\#\ of\ pins = |\{(x,i,j) : a_{x,i} \in A \wedge b_{i,j} \in B\}| + nnz(B). \tag{5.18}$$

In Equation (5.18), which is for calculating the number of pins of $\mathcal{H}_{\text{cr}}$, the term for the size of the set corresponds to the total number of scalar multiply-and-add operations to be performed in an SpGEMM computation $C = AB$.

Single weight is assigned to each vertex v_x in order to encode the computational costs of the multiply-and-add operations in the multiplication phase of RCp. Each vertex $v_x \in \mathcal{V}^A$ represents the atomic task

$$c_{x,*} = a_{x,*} \times B \tag{5.19}$$

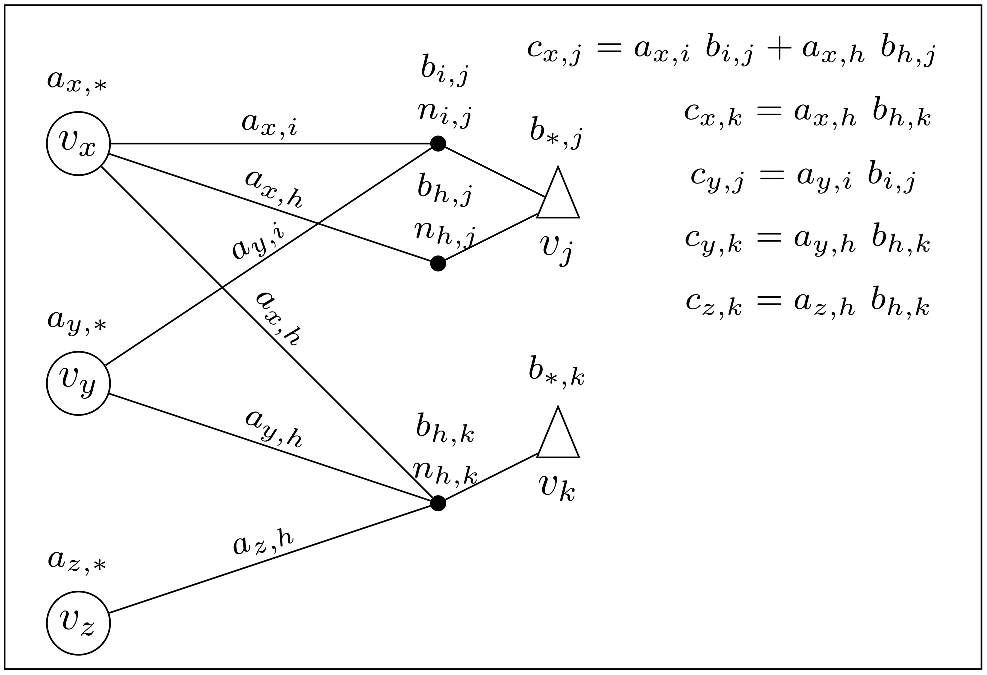


Figure 5.5: The proposed hypergraph model $\mathcal{H}_{\mathrm{rc}}$ for RCp

of pre-multiplying row x ($a_{x,*}$) of A with matrix B to compute the xth row of C ($c_{x,*}$). So vertex $v_x \in \mathcal{V}^A$ is associated with a weight $w(v_x)$ proportional to the computational load of this vector-matrix product in terms of scalar multipy-and-add operations. That is,

$$w(v_x) = \sum_{a_{x,i} \in a_{x,*}} nnz(b_{i,*}). \tag{5.20}$$

The weight $w(v_j)$ of vertex $v_j \in \mathcal{V}^B$ is set to be equal to zero since it does not represent any computation. That is,

$$w(v_j) = 0. \tag{5.21}$$

The existence of v_j vertices with zero weights is to enforce columnwise partitioning of matrix B. Each net $n_{i,j} \in \mathcal{N}$ represents the requirement of B-matrix nonzero $b_{i,j}$ in order to compute $c_{x,j} = a_{x,i}b_{i,j}$. So cost $c(n_x)$ of net n_x is set to be equal to one. That is,

$$c(n_{i,j}) = 1. \tag{5.22}$$

Figure 5.5 shows the hypergraph model $\mathcal{H}_{\mathrm{rc}}$ and the dependencies in scalar multiplications of input matrix nonzeros. As seen in this figure, net $n_{i,j}$ with $Pins(n_{i,j}) = \{v_x, v_y, v_j\}$ corresponds to the need of multiplications $a_{x,i}b_{i,j}$ and $a_{y,i}b_{i,j}$ for B-matrix nonzero $b_{i,j}$, which is in column j of B.

A K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ on $\mathcal{V}$ inherently induces a partition $\Pi(\mathcal{V}^A)$ on $\mathcal{V}^A \subseteq \mathcal{V}$ and a partition $\Pi(\mathcal{V}^B)$ on $\mathcal{V}^B \subseteq \mathcal{V}$. $\Pi(\mathcal{V}^A)$ is decoded as a partition on the rows of A and rows of C; and $\Pi(\mathcal{V}^B)$ is decoded as a partition on the columns of B. That is, v_x in $\mathcal{V}_k$ denotes that row x of A is stored by only processor P_k and P_k is held responsible for computing the pre-multiply $c_{x,*} = a_{x,*}B$. v_j in $\mathcal{V}_k$ denotes that responsibility of storing column j of matrix B and sending the nonzeros of this column to other processors is assigned to processor P_k.

5.2.1 Model Correctness

The correctness of the proposed hypergraph model $\mathcal{H}_{\mathrm{rc}}$ can be proved by showing the following:

(a) The partitioning constraint on part weights corresponds to the balancing computational loads of the processors during the multiplication phase of RCp.

(b) The partitioning objective of minimizing cutsize corresponds to the minimization of the total volume of communication occurred during the communication phase of RCp.

Consider a K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ of vertices of $\mathcal{H}_{\mathrm{rc}}$ for both (a) and (b).

For (a), $\Pi(\mathcal{V})$ is assumed to satisfy the balance constraint given in Equation (2.8) for $T = 1$. Considering the weights given in Equations (5.20) and (5.21), the partitioning constraint correctly encodes balancing the computational loads in terms of number of multiply-and-add operations in local pre-multiply operations to be performed by processors during the multiplication phase.

For showing (b), consider a vertex v_j, which is assigned to $\mathcal{V}_k$ (i.e., $v_j \in \mathcal{V}_k$). Recall that each net $n_{i,j}$ connects only one vertex v_j, which represents column j of B. Assume that $|\Lambda(n_{i,j}) - \{\mathcal{V}_k\}| \geq 1$. Then, each part $\mathcal{V}_m \in \Lambda(n_{i,j}) - \{\mathcal{V}_k\}$ has at least one vertex corresponding to a pre-multiply operation that requires nonzero $b_{i,j}$ of matrix B. So, for each part $\mathcal{V}_m \in \Lambda(n_{i,j}) - \{\mathcal{V}_k\}$, processor P_m receives $b_{i,j}$ from P_k. Hence, $v_j \in \mathcal{V}_k$ means that processor P_k will send only one nonzero to each of the $\lambda(n_{i,j}) - 1$ processors corresponding parts $\mathcal{V}_k$ in $\Lambda(n_{i,j}) - \{\mathcal{V}_k\}$. After the receive of required nonzeros of matrix B, the pre-multiply operations are performed. As seen in Equation (2.11), the contribution of net $n_{i,j}$ to cutsize is $\lambda(n_{i,j}) - 1$. As a result, the equivalence between $\lambda(n_{i,j}) - 1$ and the communication volume regarding the transfer of nonzeros of matrix B in the communication phase is shown. Consequently, the total communication volume during this communication phase is correctly encoded by the cutsize given in Equation (2.11).

5.2.2 Model Construction

Algorithm 5 shows the pseudocode for constructing the hypergraph model $\mathcal{H}_{\mathrm{rc}}$ for inner-product based multiplication of a given pair of A and B matrices. This algorithm requires matrix A to be in CSC format, whereas matrix B can be in COO format. Here, COO refers to coordinate format, in which nonzero elements stored as row index, column index, and nonzero value tuples. Algorithm 5 utilizes the pin-list representation of hypergraphs. For the current net $n_{i,j}$, net cost assignment and pin list allocation of appropriate size are performed at lines 7 and 8, respectively. Vertex v_x is appended to the end of the pin list of net $n_{i,j}$ at line 10. The weight of vertex v_x is incrementally updated at line 11. At line 13, the vertex v_j representing column j of B is appended to the end of the pin list of net $n_{i,j}$.

Algorithm 5 Construction of the hypergraph model $\mathcal{H}_{\text{rc}}$

```
Require: $A$ matrix in CSC format and $B$ matrix in COO format
 1: $\mathcal{V}^A \leftarrow \{v_x : a_{x,*} \in A\}$
 2: $w(v_x) \leftarrow 0, \forall v_x \in \mathcal{V}^A$
 3: $\mathcal{V}^B \leftarrow \{v_{j+rows(A)} : b_{i,j} \in B\}$
 4: $\mathcal{N} \leftarrow \varnothing$
 5: for each nonzero $b_{i,j}$ of $B$ do
 6:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{n_{i,j}\}$
 7:     $c(n_{i,j}) \leftarrow 1$
 8:     allocate pin list of size $(nnz(a_{*,i}) + 1)$ for net $n_{i,j}$
 9:     for each nonzero $a_{x,i}$ in column $i$ of $A$ do
10:         $Pins(n_{i,j}) \leftarrow Pins(n_{i,j}) \cup \{v_x\}$
11:         $w(v_x) \leftarrow w(v_x) + 1$
12:     end for
13:     $Pins(n_{i,j}) \leftarrow Pins(n_{i,j}) \cup \{v_{j+rows(A)}\}$
14: end for
15: return $\mathcal{H}_{\text{rc}}(\mathcal{V} = \mathcal{V}^A \cup \mathcal{V}^B, \mathcal{N})$
```

5.3 The Hypergraph Model $\mathcal{H}_{\text{rr}}$ for RRp

In this model, an SpGEMM computation $C = AB$ is represented as a hypergraph $\mathcal{H}_{\text{rr}} = (\mathcal{V} = \mathcal{V}^A \cup \mathcal{V}^B, \mathcal{N})$ for 1D rowwise partitioning of A, B and C. There exists a vertex $v_x \in \mathcal{V}^A$ for each row x of A. There exist both a vertex $v_i \in \mathcal{V}^B$ and a net $n_i \in \mathcal{N}$ for each row i of B. Net n_i connects vertices corresponding to the rows that have nonzeros at ith column of A as well as v_i. That is, n_i connects v_x if and only if the pre-multiply of row x of A with B requires row i of matrix B. So n_i is defined as follows:

$$Pins(n_i) = \{v_x : a_{x,i} \in A\} \cup \{v_i\}. \tag{5.23}$$

As seen in Equation (5.23), each net n_i connects at least one vertex that represents a row of matrix B. Note that single subscript is used for both types of vertices. When x is used in subscript, the vertex that represents the atomic task of computing pre-multiply is intended, whereas when i is used, the vertex that represents a row of matrix B is intended.

The number of vertices, nets, and pins of the proposed hypergraph model $\mathcal{H}_{\text{rr}}$ can be expressed using the attributes of input matrices A and B:

$$|\mathcal{V}| = rows(A) + rows(B), \tag{5.24}$$

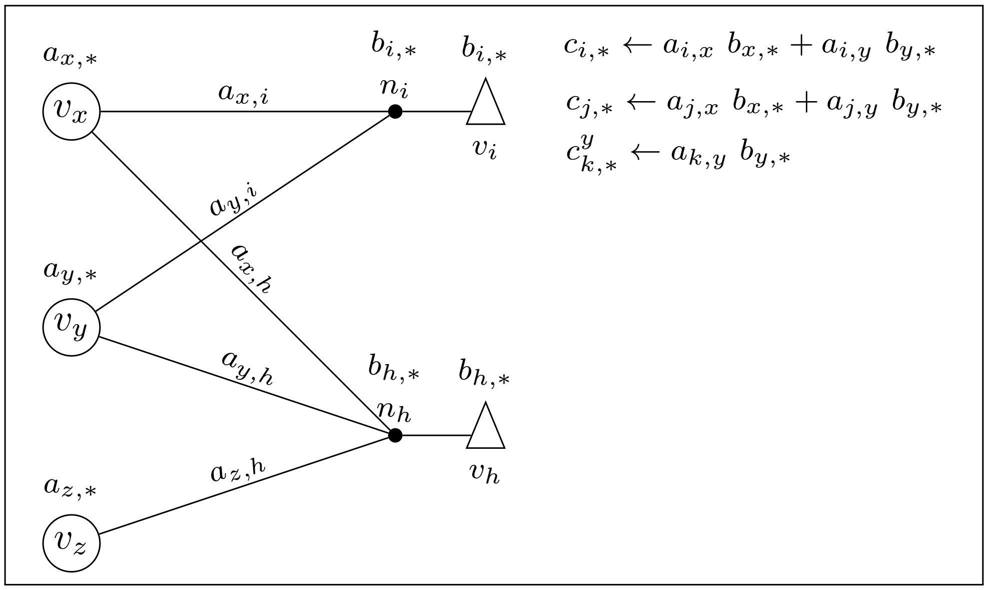
$$|\mathcal{N}| = rows(B), \tag{5.25}$$

$$\#\ of\ pins = |\{(x,i) : a_{x,i} \in A \wedge b_{i,*} \in B\}| + rows(B). \tag{5.26}$$

In Equation (5.26), which is for calculating the number of pins of $\mathcal{H}_{\text{rr}}$, the term for the size of the set corresponds to the total number of operations of scaling a row of B matrix by a nonzero of A matrix to be performed in an SpGEMM computation $C = AB$.

Single weight is assigned to each vertex v_x in order to encode the computational costs of the multiply-and-add operations in the multiplication phase of RRp. Each vertex $v_x \in \mathcal{V}^A$ represents the atomic task

$$c_{x,*} = a_{x,*} \times B \tag{5.27}$$



$a_{x,*}$
v_x
$a_{x,i}$
$b_{i,*}$
n_i
$b_{i,*}$
v_i
$c_{i,*} \leftarrow a_{i,x}\ b_{x,*} + a_{i,y}\ b_{y,*}$
$c_{j,*} \leftarrow a_{j,x}\ b_{x,*} + a_{j,y}\ b_{y,*}$
$c^y_{k,*} \leftarrow a_{k,y}\ b_{y,*}$
$a_{y,i}$
$a_{x,h}$
$a_{y,*}$
v_y
$a_{y,h}$
$b_{h,*}$
n_h
$b_{h,*}$
v_h
$a_{z,h}$
$a_{z,*}$
v_z

Figure 5.6: The proposed hypergraph model $\mathcal{H}_{\mathrm{rr}}$ for RRp

of pre-multiplying row x ($a_{x,*}$) of A with matrix B to compute the xth row of C ($c_{x,*}$). So vertex $v_x \in \mathcal{V}^A$ is associated with a weight $w(v_x)$ proportional to the computational load of this vector-matrix product in terms of scalar multipy-and-add operations. That is,

$$w(v_x) = \sum_{a_{x,i} \in a_{x,*}} nnz(b_{i,*}). \tag{5.28}$$

The weight $w(v_i)$ of vertex $v_i \in \mathcal{V}^B$ is set to be equal to zero since it does not represent any computation. That is,

$$w(v_i) = 0. \tag{5.29}$$

The existence of v_i vertices with zero weights is to encode rowwise partitioning of matrix B as a vertex partition. Each net $n_i \in \mathcal{N}$ represents the requirement of B-matrix row $b_{i,*}$ by the computation of $c_{x,*} = a_{x,*}b_{i,*}$. So cost $c(n_x)$ of net n_x must be proportional to the number of nonzeros in row i of B matrix. That is,

$$c(n_i) = nnz(b_{i,*}). \tag{5.30}$$

Figure 5.6 shows the hypergraph model $\mathcal{H}_{\mathrm{rr}}$ and the dependencies in the multiplications of A-matrix nonzeros and B-matrix rows. As seen in this figure, net n_i

with $Pins(n_i) = \{v_x, v_y, v_z, v_i\}$ corresponds to the need of multiplications $a_{x,i}b_{i,*}$, $a_{y,i}b_{i,*}$, and $a_{z,i}b_{i,*}$ for B-matrix row $b_{i,*}$.

A K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ on $\mathcal{V}$ inherently induces a partition $\Pi(\mathcal{V}^A)$ on $\mathcal{V}^A \subseteq \mathcal{V}$ and a partition $\Pi(\mathcal{V}^B)$ on $\mathcal{V}^B \subseteq \mathcal{V}$. $\Pi(\mathcal{V}^A)$ is decoded as a partition on the rows of A and C matrices; and $\Pi(\mathcal{V}^B)$ is decoded as a partition on the rows of B matrix. That is, v_x in $\mathcal{V}_k$ denotes that row x of A is stored by only processor P_k and P_k is held responsible for computing the pre-multiply $c_{x,*} = a_{x,*}B$. v_i in $\mathcal{V}_k$ denotes that responsibility of storing row i of matrix B and sending this row to other processors is assigned to processor P_k.

5.3.1 Model Correctness

The correctness of the proposed hypergraph model $\mathcal{H}_{\text{rr}}$ can be proved by showing the following:

(a) The partitioning constraint on the part weights corresponds to balancing the computational loads of processors during the multiplication phase of RRp.

(b) The partitioning objective of minimizing the cutsize corresponds to the minimization of the total volume of communication occurred during the communication phase of RRp.

Consider a K-way partition $\Pi(\mathcal{V}) = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_K\}$ of vertices of $\mathcal{H}_{\text{rr}}$ for both (a) and (b).

For (a), $\Pi(\mathcal{V})$ is assumed to satisfy balance constraint given in Equation (2.8) for $T = 1$. Considering the weights given in Equations (5.28) and (5.29), the partitioning constraint correctly encodes balancing the computational loads in terms of number of multiply-and-add operations in local pre-multiply operations to be performed by processors during the multiplication phase.

For showing (b), consider a vertex v_i, which is assigned to $\mathcal{V}_k$ (i.e., $v_i \in \mathcal{V}_k$). Recall that each net n_i connects only one vertex v_i, which represents a row of B. Then, each part $\mathcal{V}_m \in \Lambda(n_i) - \{\mathcal{V}_k\}$ has at least one vertex corresponding to a pre-multiply operation that requires row i $(b_{i,*})$ of matrix B. So, for each part $\mathcal{V}_m \in \Lambda(n_i) - \{\mathcal{V}_k\}$, processor P_m receives $b_{i,*}$ from P_k. Hence, $v_i \in \mathcal{V}_k$ means that processor P_k will send only one row to each of the $\lambda(n_j) - 1$ processors in $\Lambda(n_j) - \{\mathcal{V}_k\}$. After the receive of required rows of matrix B, the pre-multiply operations are performed. As seen in Equation (2.11), the contribution of net n_i to cutsize is $c(n_i)(\lambda(n_i)-1)$. As a result, the equivalence between $c(n_i)(\lambda(n_i)-1)$ and the communication volume regarding the transfer of rows of matrix B in the communication phase is shown. Consequently, the total communication volume during this communication phase is correctly encoded by the cutsize given in Equation (2.11).

5.3.2 Model Construction

Algorithm 6 Construction of the hypergraph model $\mathcal{H}_{\text{rr}}$

```
Require: $A$ matrix in CSC format and $B$ matrix in COO format
 1: $\mathcal{V}^A \leftarrow \{v_x : a_{x,*} \in A\}$
 2: $w(v_x) \leftarrow 0, \forall v_x \in \mathcal{V}^A$
 3: $\mathcal{V}^B \leftarrow \{v_{i+rows(A)} : b_{i,*} \in B\}$
 4: $\mathcal{N} \leftarrow \varnothing$
 5: for each row $b_{i,*}$ of $B$ do
 6:     $\mathcal{N} \leftarrow \mathcal{N} \cup \{n_i\}$
 7:     $c(n_i) \leftarrow nnz(b_{i,*})$
 8:     allocate pin list of size $(nnz(a_{*,i}) + 1)$ for net $n_i$
 9:     for each nonzero $a_{x,i}$ in column $i$ of $A$ do
10:         $Pins(n_i) \leftarrow Pins(n_i) \cup \{v_x\}$
11:         $w(v_x) \leftarrow w(v_x) + c(n_i)$
12:     end for
13:     $Pins(n_i) \leftarrow Pins(n_i) \cup \{v_{i+rows(A)}\}$
14: end for
15: return $\mathcal{H}_{\text{rr}}(\mathcal{V} = \mathcal{V}^A \cup \mathcal{V}^B, \mathcal{N})$
```

Algorithm 6 shows the pseudocode for constructing the hypergraph model $\mathcal{H}_{\text{rr}}$

for row-by-row parallel multiplication of a given pair of A and B matrices. This algorithm requires matrix A to be in CSC format, whereas matrix B can be in COO format. Algorithm 6 utilizes the pin-list representation of hypergraphs. For the current net n_i, net cost assignment and pin list allocation of appropriate size are performed at lines 7 and 8, respectively. Vertex v_x is appended to the end of the pin list of net n_i at line 10. The weight of vertex v_x is incrementally updated at line 11. At line 13, the vertex v_i representing row i of B is appended to the end of the pin list of net n_i.

Chapter 6

Communication Hypergraph Models for Parallel SpGEMM Algorithms

In this chapter, we propose communication hypergraph models for further improving the performance of the SpGEMM algorithms proposed in Chapter 4. The hypergraph models proposed in Chapter 5 aim at reducing total communication volume while maintaining computational load balance among processors. The communication hypergraph models proposed in this chapter aim at reducing total number of messages while maintaining balance on communication volumes handled by processors. In the overall framework, the hypergraph models proposed in Chapter 5 are used in the first stage of preprocessing step and the communication hypergraph models are used in the second stage of the preprocessing step prior to parallel SpGEMM operation. Note that the notion of communication hypergraph model is first proposed in [23] in order to improve parallel performance of sparse matrix-vector multiplication (SpMV) in the form of $y = Ax$.

In this thesis, we propose how to generate communication hypergraph models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ for the three hypergraph models $\mathcal{H}_{\mathrm{cr}}$, $\mathcal{H}_{\mathrm{rc}}$, and $\mathcal{H}_{\mathrm{rr}}$, respectively. In partitioning of $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$, the objective of minimizing cutsize corresponds to minimizing the total number of messages transferred over network and the constraint of balancing part weights corresponds to maintaining balance on communication volumes handled by processors.

6.1 The Communication Hypergraph Models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$

Recall that, in the hypergraph models $\mathcal{H}_{\mathrm{cr}}$, $\mathcal{H}_{\mathrm{rc}}$, and $\mathcal{H}_{\mathrm{rr}}$, there are two types of vertices:

- vertices corresponding to SpGEMM computations in the multiplication phase,
- vertices corresponding to matrix entries to be transferred in the communication phase.

The communication hypergraph aims at modeling only the communication, hence $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ use the part information of second type of vertices in the $\mathcal{H}_{\mathrm{cr}}$, $\mathcal{H}_{\mathrm{rc}}$, and $\mathcal{H}_{\mathrm{rr}}$ models, respectively. The part information is obtained after partitioning the corresponding hypergraph models, $\mathcal{H}_{\mathrm{cr}}$, $\mathcal{H}_{\mathrm{rc}}$, and $\mathcal{H}_{\mathrm{rr}}$. Recall that the part information of vertices is decoded as task to processor assignment since part k is assumed to be assigned to processor P_k for $k = 1, 2, \ldots, K$ without loss of generality.

In the communication hypergraph model, communication tasks are represented with vertices and the processors are represented with nets. Considering the SpGEMM algorithms, there are two kinds of communication tasks:

- Gathering operation: the gathering of partial results from processors in order to calculate the final value of $c_{i,j}$ in the CRp algorithm

- Replication operation: the replication of B-matrix nonzeros in RCp and B-matrix rows in RRp to processors that need the corresponding B-matrix entities.

In the communication hypergraph model, there exists a net for each processor and there exists a vertex for each communicated entry. A net connects a vertex if and only if the communicated entry, which is represented by that vertex, is sent to or received by the processor, which is represented by that net. Each such net also connects a vertex fixed to a part.

Each net is associated with a unit weight. Vertices corresponding to the communicated entries are associated with a weight proportional to the volume of communication that occurs in sending these communicated entries. This weighting scheme will be used to maintain balance on the volumes of messages sent by processors.

6.1.1 Obtaining $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$ from $\mathcal{H}_{\mathrm{cr}}$

Given a partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\mathrm{cr}}$, the boundary vertices of $\mathcal{H}_{\mathrm{cr}}$ corresponding to C-matrix nonzeros are vertices of $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$. The processors that involve in communication correspond to nets. A net connects a vertex if and only if the processor corresponding to that net generates a partial result or gathers the partial results.

6.1.2 Obtaining $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$ from $\mathcal{H}_{\mathrm{rc}}$

Given a partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\mathrm{rc}}$, the boundary vertices of $\mathcal{H}_{\mathrm{rc}}$ corresponding to B-matrix columns are vertices of $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$. The processors that involve in communication correspond to nets. A net connects a vertex if and only if the processor corresponding to that net needs at least one nonzero of B-matrix column represented by that vertex or owns this B-matrix column.

6.1.3 Obtaining $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ from $\mathcal{H}_{\mathrm{rr}}$

Given a partition $\Pi(\mathcal{V})$ of $\mathcal{H}_{\mathrm{rr}}$, the boundary vertices of $\mathcal{H}_{\mathrm{rr}}$ corresponding to B-matrix rows are vertices of $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$. The processors that involve in communication correspond to nets. A net connects a vertex if and only if the processor corresponding to that net needs the B-matrix row represented by that vertex or owns this B-matrix row.

6.2 Decoding a Partition of the Communication Hypergraph Model

A K-way partition of vertices of the communication hypergraph can be decoded as follows: Note that, that part k is assumed to be assigned to processor P_k for $k = 1, 2, \ldots, K$ without loss of generality. A vertex in part k is decoded as the communication task related with that vertex is assigned to processor P_k. That is, P_k is held responsible for gathering all the partial results from other processors in the CRp algorithm or replicating the B-matrix entities to other processors in the RCp and RRp algorithms. In this partitioning, minimizing the cutsize (2.11) encodes minimization of total number of messages transferred over network. Maintaining balance (2.8) corresponds to balancing volumes of messages sent by processors. For details of the communication hypergraph, please refer to [24].

Chapter 7

Experiments

In this chapter, empirical verification of the parallel algorithms proposed in Chapter 4, the hypergraph models proposed in Chapters 5 and 6 will be presented and discussed.

7.1 Experimental Dataset

Matrices arising in real applications that involve SpGEMM operations and matrices obtained from the the University of Florida Sparse Matrix Collection [48] are included in the test dataset. For the application of molecular dynamics simulations, we simulate H_2O molecules by executing CP2K [9]. During the simulation, two SpGEMM instances, which are used in calculations related with Kohn-Sham Density Functional Theory, are dumped. For cut-off values of 0.5 10^{-7} and 10^{-6}, the matrices `cp2k-h2o-.5e7` and `cp2k-h2o-e6` are respectively obtained. For the application of linear programming (LP), LP constraint matrices are downloaded from the University of Florida Sparse Matrix Collection. For the application of recommendation systems [18], two matrices, `amazon0312` and `amazon0302`, are included. These matrices represent the relation between similar items. User preference matrices, `amazon0312-user` and `amazon0302-user`, are randomly generated according to Zipf distribution.

The test dataset are divided into three categories according to the multiplication types of SpGEMM instances. The multiplication types are: $C = AA^T$, $C = AA$, and $C = AB$. In each category, matrices are displayed in alphabetical order of their names.

Tables 7.1 and 7.2 respectively show the properties of input and output matrices. In the tables, the number of rows, columns, and nonzeros are given first. Then, the average and maximum number nonzeros per row and column are given in “avg” and “max” columns, respectively.

Table 7.1: Input matrix properties

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
			$C = AA^T$					
AAT1	cont11_l	1,468,599	1,961,394	5,382,999	4	5	3	7
AAT2	fome13	48,568	97,840	285,056	6	228	3	14
AAT3	fome21	67,748	216,350	465,294	7	96	2	3
AAT4	fxm3_16	41,340	85,575	392,252	9	57	5	36
AAT5	fxm4_6	22,400	47,185	265,442	12	57	6	24
AAT6	lp_pds_20	33,874	108,175	232,647	7	96	2	3
AAT7	pds-30	49,944	158,489	340,635	7	96	2	3
AAT8	pds-40	66,844	217,531	466,800	7	96	2	3
AAT9	pds-50	83,060	275,814	590,833	7	96	2	3
AAT10	pds-60	99,431	336,421	719,557	7	96	2	3
AAT11	pds-90	142,823	475,448	1,014,136	7	96	2	3
AAT12	sgpf5y6	246,077	312,540	831,976	3	61	3	12
AAT13	watson_1	201,155	386,992	1,055,093	5	93	3	9
AAT14	watson_2	352,013	677,224	1,846,391	5	93	3	15
			$C = AA$					

Table 7.1: Input matrix properties (continued)

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
AA1	144	144,649	144,649	2,148,786	15	26	15	26
AA2	2cubes_sphere	101,492	101,492	1,647,264	16	31	16	31
AA3	Chevron4	711,450	711,450	6,376,412	9	9	9	9
AA4	cp2k-h2o-.5e7	279,936	279,936	3,816,315	14	24	14	27
AA5	cp2k-h2o-e6	279,936	279,936	2,349,567	8	20	8	20
AA6	mac_econ_fwd500	206,500	206,500	1,273,389	6	44	6	47
AA7	majorbasis	160,000	160,000	1,750,416	11	11	11	18
AA8	mario002	389,874	389,874	2,101,242	5	7	5	7
AA9	mc2depi	525,825	525,825	2,100,225	4	4	4	4
AA10	poisson3Da	13,514	13,514	352,762	26	110	26	110
AA11	scircuit	170,998	170,998	958,936	6	353	6	353
AA12	t2em	921,632	921,632	4,590,832	5	5	5	5
AA13	thermomech_dK	204,316	204,316	2,846,228	14	20	14	20
AA14	tmt_sym	726,713	726,713	5,080,961	7	9	7	9
AA15	torso2	115,967	115,967	1,033,473	9	10	9	10
AA16	xenon2	157,464	157,464	3,866,688	25	27	25	27

Table 7.1: Input matrix properties (continued)

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
				$C = AB$				
AB1	amazon0302	262,111	262,111	1,234,877	5	5	5	420
	amazon0302-user	262,111	1,000	2,111,519	8	911	2,112	2,278
AB2	amazon0312	400,727	400,727	3,200,440	8	10	8	2,747
	amazon0312-user	400,727	1,000	3,226,140	8	461	3,226	3,405
AB3	crashbasis	160,000	160,000	1,750,416	11	11	11	18
	majorbasis	160,000	160,000	1,750,416	11	11	11	18
AB4	darcy003	389,874	389,874	2,101,242	5	7	5	7
	mario002	389,874	389,874	2,101,242	5	7	5	7
AB5	thermomech_dK	204,316	204,316	2,846,228	14	20	14	20
	thermomech_dM	204,316	204,316	1,423,116	7	10	7	10

Table 7.2: Output matrix properties

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
				$C = AA^T$				
AAT1	cont11_lcont11_l-T	1,468,599	1,468,599	18,064,261	12	23	12	23
AAT2	fome13fome13-T	48,568	48,568	658,136	14	568	14	568
AAT3	fome21fome21-T	67,748	67,748	640,240	9	97	9	97
AAT4	fxm3_16fxm3_16-T	41,340	41,340	765,526	19	158	19	158
AAT5	fxm4_6fxm4_6-T	22,400	22,400	526,536	24	98	24	98
AAT6	lp_pds_20lp_pds_20-T	33,874	33,874	320,120	9	97	9	97
AAT7	pds-30pds-30-T	49,944	49,944	468,266	9	97	9	97
AAT8	pds-40pds-40-T	66,844	66,844	637,867	10	97	10	97
AAT9	pds-50pds-50-T	83,060	83,060	802,503	10	97	10	97
AAT10	pds-60pds-60-T	99,431	99,431	972,220	10	97	10	97
AAT11	pds-90pds-90-T	142,823	142,823	1,363,698	10	97	10	97
AAT12	sgpf5y6sgpf5y6-T	246,077	246,077	2,776,645	11	367	11	367
AAT13	watson_1watson_1-T	201,155	201,155	1,937,163	10	123	10	123
AAT14	watson_2watson_2-T	352,013	352,013	3,390,279	10	123	10	123

$C = AA$

Table 7.2: Output matrix properties (continued)

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
AA1	144144	144,649	144,649	10,416,087	72	116	72	116
AA2	2cubes_sphere2cubes_sphere	101,492	101,492	8,974,526	88	180	88	180
AA3	Chevron4Chevron4	711,450	711,450	17,706,328	25	25	25	25
AA4	cp2k-h2o-.5e7cp2k-h2o-.5e7	279,936	279,936	17,052,039	61	99	61	103
AA5	cp2k-h2o-e6cp2k-h2o-e6	279,936	279,936	7,846,956	28	50	28	50
AA6	mac_econ_fwd500mac_econ_fwd500	206,500	206,500	6,704,899	32	215	32	157
AA7	majorbasismajorbasis	160,000	160,000	8,243,392	52	52	52	68
AA8	mario002mario002	389,874	389,874	6,449,598	17	19	17	19
AA9	mc2depimc2depi	525,825	525,825	5,245,952	10	10	10	10
AA10	poisson3Dapoisson3Da	13,514	13,514	2,957,530	219	584	219	584
AA11	scircuitscircuit	170,998	170,998	5,222,525	31	1,885	31	1,885
AA12	t2emt2em	921,632	921,632	11,924,892	13	13	13	13
AA13	thermomech_dKthermomech_dK	204,316	204,316	7,874,152	39	52	39	52
AA14	tmt_symtmt_sym	726,713	726,713	14,503,181	20	25	20	25
AA15	torso2torso2	115,967	115,967	2,858,293	25	27	25	28
AA16	xenon2xenon2	157,464	157,464	14,037,210	89	99	89	99

Table 7.2: Output matrix properties (continued)

		Number of			nnz in row		nnz in column	
Instance	Matrix	rows	columns	nonzeros	avg	max	avg	max
				$C = AB$				
AB1	amazon0302amazon0302-user	262,111	1,000	9,787,258	37	913	9,787	11,024
AB2	amazon0312amazon0312-user	400,727	1,000	24,951,760	62	500	24,952	29,392
AB3	crashbasismajorbasis	160,000	160,000	8,243,392	52	52	52	68
AB4	darcy003mario002	389,874	389,874	6,449,598	17	19	17	19
AB5	thermomech_dKthermomech_dM	204,316	204,316	7,874,148	39	52	39	52

7.2 Experimental Setup

The partitioning tool used in partitioning the proposed hypergraph models of the test SpGEMM instances will be given in Section 7.2.1. In Section 7.2.2, overview of our SpGEMM library will be given. Some of the important properties of the parallel platform, on which the experiments are carried out, will be mentioned in Section 7.2.3.

7.2.1 Partitioning Tool

In order to partition the proposed hypergraph models of the SpGEMM instances, the successful state-of-the-art serial hypergraph partitioning (HP) tool PaToH [46] is used. Recursive bipartitioning scheme is used in PaToH for obtaining multiway partitions of a given hypergraph. For bipartitioning a given hypergraph, PaToH uses multilevel approach. Each level consists of coarsening, initial bipartitioning and uncoarsening phases [46, 29].

In the experiments, PaToH's `PATOH_SUGPARAM_SPEED` parameter is used. This parameter enables finding reasonably good bipartitions faster than the default parameter as mentioned in the manual [46]. A trade-off between the bipartitioning time and the solution quality is established through using absorption clustering using pins in the coarsening phase and using boundary FM for faster refinement in the uncoarsening phase. The use of absorption clustering using pins in the coarsening phase also leads to less number of levels.

The results are reported by averaging the values obtained in three different runs because PaToH uses randomized algorithms. Each of these three runs is randomly seeded. The maximum allowed imbalance ratio ϵ is set to be equal to 10%. The hypergraphs of the SpGEMM instances are partitioned for each value of $K = 256, 512$ and 1024. Here, K is the number of used processors of the parallel system.

7.2.2 The SpGEMM Library

In order to show the actual performances of the proposed SpGEMM algorithms and hypergraph models, a SpGEMM library is developed. This library is based on the MPI (Message Passing Interface) library [26] so it is designed for running on distributed memory architectures. The library is developed using the C programming language. This library can perform the following operations:

- Conversion of sparse matrices stored as text to binary format in order to decrease time spent in file I/O.
- Sequentially multiply two sparse matrices using Gustavson's SpGEMM algorithm [33]
- Partition given two input matrices according to the models proposed in Chapter 5
- Perform parallel multiplication of given two matrices according to a given partitioning information using the parallel SpGEMM algorithms proposed in Chapter 4
- Verify numerical correctness of the multiplication results

The manual of this library can be found in Appendix Chapter A. The library can be downloaded from [49].

7.2.3 The BlueGene/Q System

Our SpGEMM library is tested on an IBM BlueGene/Q system, named JUQUEEN. JUQUUEN is located at the Jülich Supercomputing Centre in Germany. A compute node of the BlueGene/Q system has 16 PowerPC A2 cores. These cores are clocked at 1.6 GHz. A compute node has 16 GB of RAM. The compute nodes of BlueGene/Q system are connected to each other via a five dimensional torus network.

The BlueGene/Q system has special compiler named IBM XL compiler suite. Our SpGEMM library is compiled with the C compiler of this suit using `O2` flag. The BlueGene/Q system uses MPI implementation based on MPICH2 [50]. Eight processes per node is used on JUQUEEN during the parallel runs.

Each parallel SpGEMM operation is repeated 10 times and average of these 10 runs are reported. Before these runs, three SpGEMM operations are performed in order to alleviate the cold start problem. The sequential algorithm of proposed by Gustavson [33] is used to measure sequential run time on a single core of JUQUEEN. Speedup values on JUQUEEN are computed according to these sequential run times.

Table 7.3: Results of $\mathcal{H}_{\mathrm{cr}}$, $\mathcal{H}_{\mathrm{rc}}$, and $\mathcal{H}_{\mathrm{rr}}$

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$
						$C = AA^T$											
		256	215	215	215	5	4	3	67	41	40	0.003	0.042	0.042	0.005	0.060	0.082
AAT1	cont11_lcont11_l-T	512	409	409	410	6	5	4	76	42	45	0.002	0.030	0.030	0.004	0.043	0.062
		1024	621	735	749	6	6	6	222	40	37	0.002	0.021	0.022	0.003	0.036	0.048
		256	131	114	106	20	13	14	56	28	32	0.020	0.905	0.916	0.031	1.153	1.717
AAT2	fome13fome13-T	512	172	135	140	26	21	21	58	40	34	0.012	0.537	0.545	0.023	0.750	1.105
		1024	190	140	175	32	27	26	44	49	51	0.008	0.325	0.324	0.027	1.078	0.827
		256	131	99	104	11	18	18	54	64	54	0.013	0.492	0.492	0.021	0.893	0.988
AAT3	fome21fome21-T	512	165	123	136	13	20	19	59	67	77	0.008	0.319	0.323	0.015	0.574	0.740
		1024	180	148	191	17	23	21	67	73	59	0.006	0.213	0.212	0.010	0.353	0.484
		256	117	80	88	52	79	87	170	137	118	0.004	0.106	0.112	0.020	0.279	0.444
AAT4	fxm3_16fxm3_16-T	512	157	100	139	80	97	98	160	159	107	0.006	0.115	0.113	0.026	0.328	0.432
		1024	187	116	177	96	104	101	137	183	99	0.007	0.107	0.106	0.029	0.444	0.499
		256	97	74	80	54	85	81	122	83	90	0.007	0.133	0.146	0.023	0.342	0.615
AAT5	fxm4_6fxm4_6-T																

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		512	130	100	111	63	95	98	107	73	83	0.009	0.161	0.161	0.031	0.441	0.610
		1024	157	106	141	68	107	113	98	108	83	0.010	0.145	0.142	0.030	0.693	0.864
		256	88	67	75	11	20	19	59	59	60	0.016	0.635	0.638	0.029	1.128	1.444
AAT6	lp_pds_20lp_pds_20-T	512	96	82	101	17	23	23	78	63	60	0.011	0.422	0.419	0.020	0.688	0.923
		1024	110	85	128	22	28	28	67	87	59	0.007	0.279	0.273	0.013	1.050	0.728
		256	104	84	91	12	18	19	65	51	59	0.014	0.555	0.553	0.026	0.879	1.217
AAT7	pds-30pds-30-T	512	131	107	118	13	19	20	67	57	68	0.009	0.372	0.370	0.018	0.605	0.801
		1024	143	107	154	19	26	25	70	92	74	0.006	0.237	0.236	0.011	0.376	0.601
		256	129	99	103	11	16	16	51	57	50	0.013	0.503	0.506	0.023	0.837	1.156
AAT8	pds-40pds-40-T	512	169	132	134	12	20	20	56	56	72	0.008	0.322	0.326	0.015	0.565	0.750
		1024	184	143	181	17	23	23	57	71	68	0.006	0.216	0.214	0.010	0.354	0.560
		256	141	104	110	10	16	16	52	60	61	0.012	0.452	0.459	0.019	0.859	1.099
AAT9	pds-50pds-50-T	512	189	144	154	12	20	20	58	63	64	0.008	0.307	0.310	0.015	0.527	0.697
		1024	209	171	209	16	23	22	60	66	62	0.005	0.209	0.208	0.009	0.329	0.463

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		256	149	118	112	11	18	17	59	52	70	0.011	0.415	0.413	0.018	0.758	0.949
AAT10	pds-60pds-60-T	512	211	155	164	12	19	20	58	65	73	0.007	0.294	0.292	0.012	0.484	0.700
		1024	239	184	226	15	20	22	73	70	67	0.005	0.196	0.194	0.009	0.330	0.455
		256	169	130	130	11	17	17	54	55	61	0.009	0.353	0.357	0.017	0.760	0.898
AAT11	pds-90pds-90-T	512	255	187	195	13	18	18	62	58	55	0.006	0.258	0.257	0.012	0.454	0.560
		1024	301	207	271	14	22	23	66	85	65	0.004	0.168	0.167	0.008	0.310	0.376
		256	159	105	115	8	22	24	122	84	101	0.002	0.244	0.251	0.007	0.770	1.850
AAT12	sgpf5y6sgpf5y6-T	512	196	113	150	11	36	32	181	90	109	0.002	0.222	0.220	0.005	0.485	1.349
		1024	215	103	170	18	47	46	167	110	133	0.002	0.162	0.159	0.005	0.285	0.738
		256	188	165	158	4	12	11	94	91	92	0.002	0.146	0.146	0.005	0.428	0.721
AAT13	watson_1watson_1-T	512	291	236	224	7	21	20	101	99	98	0.002	0.140	0.139	0.004	0.377	0.699
		1024	358	312	298	14	34	40	127	71	72	0.001	0.103	0.102	0.003	0.243	0.508
		256	217	177	170	7	14	12	111	98	104	0.001	0.087	0.094	0.003	0.281	0.564
AAT14	watson_2watson_2-T	512	363	282	253	10	20	22	104	107	124	0.001	0.087	0.088	0.003	0.275	0.484

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		1024	502	397	375	13	32	30	135	86	95	0.001	0.083	0.081	0.002	0.211	0.394
		256	140	110	113	12	19	18	75	64	66	0.007	0.267	0.273	0.014	0.549	0.812
avg.		512	194	148	161	15	24	24	81	69	72	0.005	0.209	0.209	0.012	0.409	0.607
		1024	228	172	218	20	29	29	89	80	70	0.004	0.151	0.149	0.009	0.341	0.467
							$C = AA$										
		256	145	163	163	40	33	33	99	79	76	0.025	0.170	0.172	0.043	0.317	0.373
AA1	144144	512	267	299	307	43	36	36	99	76	76	0.018	0.119	0.119	0.030	0.209	0.285
		1024	422	520	543	50	37	36	107	70	81	0.013	0.083	0.083	0.021	0.139	0.227
		256	195	196	197	9	9	10	52	36	39	0.032	0.241	0.243	0.047	0.322	0.447
AA2	2cubes_sphere2cubes_sphere	512	290	336	355	12	11	11	104	53	43	0.021	0.161	0.162	0.044	0.215	0.331
		1024	376	536	588	14	15	14	128	48	39	0.014	0.108	0.109	0.026	0.148	0.250
		256	236	215	222	8	14	11	75	51	51	0.005	0.040	0.040	0.009	0.060	0.074
AA3	Chevron4Chevron4	512	421	467	468	4	3	3	72	53	53	0.004	0.028	0.028	0.007	0.044	0.055

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		1024	671	863	874	10	3	4	217	52	55	0.003	0.020	0.020	0.005	0.032	0.043
		256	192	194	193	6	5	6	54	42	36	0.010	0.129	0.125	0.014	0.162	0.188
AA4	cp2k-h2o-.5e7cp2k-h2o-.5e7	512	351	355	362	6	7	6	53	42	46	0.006	0.081	0.079	0.010	0.114	0.135
		1024	273	622	648	6	7	7	125	47	40	0.004	0.052	0.051	0.007	0.072	0.106
		256	199	196	199	5	6	5	48	30	34	0.005	0.100	0.101	0.008	0.127	0.168
AA5	cp2k-h2o-e6cp2k-h2o-e6	512	348	357	360	6	6	6	66	38	39	0.004	0.063	0.063	0.005	0.087	0.116
		1024	476	588	606	6	6	7	88	41	43	0.002	0.041	0.041	0.004	0.059	0.084
		256	124	184	172	88	11	17	166	34	32	0.002	0.342	0.208	0.012	0.559	0.462
AA6	mac_econ_fwd500mac_econ_fwd500	512	199	297	289	109	14	21	227	34	41	0.002	0.224	0.135	0.030	0.431	0.387
		1024	317	398	401	138	17	24	241	38	60	0.001	0.138	0.088	0.020	0.307	0.320
		256	170	194	199	6	5	3	76	47	35	0.015	0.086	0.065	0.024	0.125	0.105
AA7	majorbasismajorbasis	512	286	343	348	8	6	6	69	51	46	0.011	0.064	0.050	0.018	0.096	0.102
		1024	430	536	567	11	9	10	70	54	45	0.008	0.047	0.038	0.014	0.071	0.091
		256	205	210	211	5	4	5	65	49	49	0.005	0.057	0.056	0.009	0.093	0.117
AA8	mario002mario002																

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		512	378	396	391	7	6	5	74	42	40	0.003	0.041	0.041	0.006	0.064	0.084
		1024	588	680	672	10	8	8	101	43	45	0.003	0.029	0.029	0.005	0.050	0.063
		256	246	239	235	11	21	20	58	38	44	0.007	0.077	0.076	0.012	0.111	0.134
AA9	mc2depimc2depi	512	444	425	419	17	27	30	64	40	45	0.005	0.056	0.054	0.009	0.083	0.098
		1024	581	723	710	38	30	31	114	41	37	0.004	0.040	0.039	0.007	0.061	0.079
		256	134	125	144	15	14	13	51	56	63	0.068	0.427	0.430	0.105	0.661	1.331
AA10	poisson3Dapoisson3Da	512	177	160	230	43	16	17	69	65	55	0.049	0.315	0.309	0.097	0.937	1.157
		1024	230	180	292	38	24	23	72	66	72	0.034	0.230	0.223	0.083	1.228	1.461
		256	33	141	140	205	17	17	1302	163	205	0.025	0.118	0.120	1.170	0.286	2.820
AA11	scircuitscircuit	512	40	178	174	515	28	27	975	218	266	0.017	0.090	0.092	1.198	0.268	3.088
		1024	39	204	190	1132	37	42	779	218	356	0.012	0.069	0.069	1.237	0.314	4.821
		256	214	222	220	8	6	5	70	43	43	0.005	0.046	0.047	0.008	0.071	0.078
AA12	t2emt2em	512	401	419	411	10	8	8	71	41	44	0.004	0.033	0.033	0.007	0.050	0.056
		1024	704	748	742	14	13	12	102	40	48	0.003	0.023	0.024	0.005	0.035	0.042

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		256	210	214	212	6	4	5	73	47	44	0.006	0.055	0.055	0.010	0.081	0.097
AA13	thermomech_dKthermomech_dK	512	395	406	415	5	6	5	79	52	48	0.005	0.041	0.041	0.009	0.061	0.074
		1024	692	748	758	8	5	6	89	54	46	0.004	0.030	0.030	0.007	0.048	0.059
		256	226	231	231	4	4	4	82	43	47	0.005	0.043	0.043	0.009	0.063	0.090
AA14	tmt_symtmt_sym	512	425	442	443	5	5	5	68	53	49	0.004	0.031	0.031	0.006	0.047	0.068
		1024	718	809	803	6	6	5	97	47	48	0.003	0.022	0.022	0.005	0.034	0.052
		256	172	190	192	4	4	4	52	45	47	0.013	0.089	0.093	0.022	0.145	0.199
AA15	torso2torso2	512	267	313	327	6	6	6	68	52	45	0.010	0.066	0.069	0.018	0.105	0.142
		1024	378	479	482	10	9	8	72	46	50	0.007	0.050	0.052	0.013	0.080	0.114
		256	190	186	189	5	3	3	56	50	41	0.008	0.102	0.102	0.013	0.141	0.170
AA16	xenon2xenon2	512	355	351	359	5	3	4	62	52	47	0.006	0.068	0.068	0.010	0.098	0.139
		1024	592	618	653	5	4	4	96	56	51	0.004	0.046	0.046	0.006	0.069	0.118
		256	168	191	193	11	8	8	82	48	49	0.010	0.102	0.097	0.021	0.156	0.218
avg.		512	283	334	344	14	9	9	91	53	53	0.007	0.072	0.068	0.018	0.119	0.171

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		1024	402	535	560	18	11	11	121	54	56	0.005	0.050	0.048	0.013	0.091	0.145
		$C = AB$															
		256	166	9	119	11	70	14	185	88	71	0.001	1.151	0.294	0.013	148.728	0.987
AB1	amazon0302amazon0302-user	512	239	8	158	15	99	14	282	90	84	0.001	0.599	0.170	0.033	153.874	0.884
		1024	309	9	226	26	109	16	271	88	111	0.001	0.312	0.098	0.024	97.508	0.524
		256	79	9	110	58	78	25	167	104	66	0.032	0.856	0.340	0.154	206.787	1.228
AB2	amazon0312amazon0312-user	512	101	9	152	78	98	22	276	101	56	0.016	0.458	0.202	0.114	185.904	0.916
		1024	63	8	159	60	117	22	214	96	61	0.014	0.246	0.119	0.073	152.334	0.830
		256	167	192	195	6	6	4	84	36	57	0.014	0.085	0.065	0.027	0.120	0.110
AB3	crashbasismajorbasis	512	274	342	349	8	6	5	87	47	52	0.011	0.063	0.050	0.021	0.095	0.103
		1024	428	544	575	11	8	9	72	56	48	0.008	0.047	0.038	0.014	0.073	0.084
		256	208	210	211	6	4	3	82	39	43	0.005	0.056	0.057	0.008	0.084	0.112
AB4	darcy003mario002	512	382	393	388	8	6	7	61	40	43	0.003	0.041	0.041	0.006	0.067	0.077
		1024	584	682	683	11	9	9	95	38	47	0.003	0.029	0.029	0.005	0.048	0.064

Table 7.3: Results of $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$ (continued)

						Measured percent imbalance						Communication volume per proc.					
			Speedup			Mult. phase			Comm. phase			Average			Maximum		
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rr}}$
		256	170	218	216	8	5	5	75	53	44	0.009	0.055	0.055	0.016	0.083	0.105
AB5	thermomech_dKthermomech_dM	512	321	420	415	8	7	6	71	43	54	0.007	0.041	0.040	0.011	0.061	0.080
		1024	565	756	750	11	7	6	80	45	48	0.005	0.030	0.030	0.009	0.048	0.060
		256	151	58	163	11	15	7	110	58	55	0.007	0.192	0.115	0.023	1.915	0.275
avg.		512	241	83	267	14	18	9	124	59	56	0.005	0.124	0.078	0.022	1.618	0.220
		1024	307	116	403	19	23	11	126	61	59	0.004	0.080	0.053	0.016	1.196	0.170
		256	154	129	152	11	12	11	82	56	56	0.008	0.164	0.151	0.018	0.370	0.381
Overall		512	238	198	245	14	14	13	91	60	60	0.006	0.119	0.109	0.016	0.283	0.294
		1024	308	273	366	19	18	16	108	64	62	0.004	0.083	0.077	0.012	0.223	0.237

7.3 Performance Evaluation

For evaluating performances of the proposed models, the speedup values measured on JUQUEEN are reported. The metrics related with communication overhead and load balancing are also reported in order to give insights into the quality of partitioning. Table 7.3 contains these results for the three categories of the test dataset. Geometric averages of the results in each category are reported at the end of the category.

In the performance evaluation, parallel SpGEMM libraries such as CombBLAS and Trilinos are not used as baseline. This is because these libraries are very slow compared to our SpGEMM library. The reason behind the slowness of CombBLAS and Trilinos is that they do not calculate the output matrix C in a symbolic multiplication step so they have to perform dynamic memory allocation.

7.3.1 Effect of Balancing Constraint

In order to evaluate the quality of load balancing of the proposed models, the measured load imbalance values of the two phases of the proposed SpGEMM algorithms are reported in Table 7.3. While reporting these imbalance measures, barrier synchronization is used between the multiplication and communication phases.

As seen in Table 7.3, the load imbalance values for the multiplication phase is rather low and close to 10%, which the maximum allowable imbalance ratio given to PaToH. The load imbalance values for the multiplication phase are considerably smaller than that for the communication phase. This shows the benefit of the constraint enforced during the partitioning.

There are some instances that the imbalance values for the multiplication phase are higher than the maximum allowable imbalance ratio of 10%. This may be attributed to the “difficultness” of the problem which depends on the sparsity structure of the matrices. If tighter constraints are enforced (e.g. $\epsilon = 0.05$), the solution space will be more restricted. Hence, the cutsize will be larger and the partitioning will take more time to satisfy the tight constraint.

The high imbalances of $\mathcal{H}_{\text{cr}}$ for the communication phase can be attributed to the implementation issue discussed in Section 5.1.1. The correspondence between the second weights of vertices and the amount of summation performed in summation phase, in fact, depend on partitioning of vertices. In other words, the weights of vertices must be updated according to change in part of the vertex. Nevertheless, the second constraint can still be used to balance computational loads of the processors in the summation phase.

7.3.2 Effect of Reducing Communication Volume

The quality metrics on the communication overhead are given in Table 7.3. The average and maximum volume of messages sent by processors is given in 'Average' and "Maximum" columns, respectively. These communication volume values are normalized with respect to the number of multiply-and-add operations performed in the respective SpGEMM instance. Hence, these values represent the amount of communicated data words (i.e., floating point numbers) transferred per 1000 multiply-and-add operations.

As seen in the Table 7.3, the average and maximum volume of communication per multiply-and-add operation is small, i.e., below 0.4 words per 1000 multiply-and-add operations on the overall average. In other words, most of the multiplications do not require communication so that the communication overhead is reduced.

As seen in the Table 7.3, on the average, the maximum volume of communication is twice the average volume of communication. This is because the proposed models and methods mainly encode the balancing constraint on computational loads of processors. These models and partitioning methods can be enhanced to encode the balancing constraint on communication loads of processors.

7.3.3 Comparison of Performances of the Hypergraph Models $\mathcal{H}_{\text{cr}}$, $\mathcal{H}_{\text{rc}}$, and $\mathcal{H}_{\text{rr}}$

The relative performance of the proposed three hypergraph models will be discussed here. According to the averages given in Table 7.3, the superiority of the proposed models depends on the computation pattern of the SpGEMM instances.

For LP matrices in the $C = AA^T$ category, $\mathcal{H}_{\text{cr}}$ performs the best; and $\mathcal{H}_{\text{rr}}$ performs better than $\mathcal{H}_{\text{cr}}$. The reason behind the superior performance of $\mathcal{H}_{\text{cr}}$ can be observed in the average volume of communication column. On the average, $\mathcal{H}_{\text{cr}}$ achieves significantly smaller average volume of communication with respect to the other models. Hence, these results show the important impact of volume of communication on the parallel performance of SpGEMM operations on large-scale distributed memory architectures.

The inferior performance of $\mathcal{H}_{\text{rr}}$ can be attributed to the large messages since whole row of B matrix is communicated. This can be observed from the relatively greater difference in the average and maximum volume of communication.

For the category of $C = AA$, $\mathcal{H}_{\text{rr}}$ performs the best; and $\mathcal{H}_{\text{rc}}$ performs better than $\mathcal{H}_{\text{cr}}$. Although the average communication volume of $\mathcal{H}_{\text{cr}}$ is smaller than those of other models, $\mathcal{H}_{\text{cr}}$ suffers from the high imbalance in multiplication and summation phases. $\mathcal{H}_{\text{rr}}$ performs considerably better than $\mathcal{H}_{\text{rc}}$.

For the category of $C = AB$, $\mathcal{H}_{\text{rr}}$ performs the best; and $\mathcal{H}_{\text{cr}}$ performs better than $\mathcal{H}_{\text{rc}}$. A situation similar to that in $C = AA$ category can be seen here. Although the average communication volume of $\mathcal{H}_{\text{cr}}$ is smaller than those of other models, $\mathcal{H}_{\text{cr}}$ suffers from the high imbalance in multiplication and summation phases. However, $\mathcal{H}_{\text{cr}}$ still performs better than $\mathcal{H}_{\text{rc}}$. $\mathcal{H}_{\text{rr}}$ performs considerably better than the other models.

7.3.4 Performance Effects of Using the Communication Hypergraph Models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$

The results for using the communication hypergraph models for improving the communication induced by the three hypergraph models are presented in Table 7.4. Since the communication hypergraph models aim at reducing total amount of messages and balancing volumes of messages sent by processors, the average and maximum number of sent messages are reported in the table.

As seen in Table 7.4, in general, use of the communication hypergraph models yield decrease in the average number of messages. It is clear that the decrease in the average number of messages also corresponds to decrease in the total number of messages. The use of the communication hypergraph models can also maintain balance on the communication volume handled by processors. This effect can indirectly be seen in the "maximum number of messages" column. As seen in this column, there are significant differences between the maximum number of message values of the proposed hypergraphs models and those for when the communication hypergraph models are used.

In general, the use of the communication hypergraph models successfully decreases the total number of messages sent by processors. For some SpGEMM instances in the $C = AB$ category, the average number of messages is increased when the communication hypergraph models are used. This anomaly can be attributed to the restriction of the solution space depending on the communication patterns induced by those SpGEMM instances and the heuristics used in the partitioning.

Table 7.4: Results of communication hypergraph models $\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$, $\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$, and $\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$
									$C = AA^T$											
		256	215	215	215	207	215	212	17	13	13	13	10	12	17	13	13	13	10	12
AAT1	cont11_lcont11_l-T	512	409	409	409	399	410	407	19	15	14	11	12	10	19	15	14	11	12	10
		1024	621	692	735	709	749	741	18	15	14	12	11	10	18	15	14	12	11	10
		256	131	156	114	83	106	109	48	21	31	31	31	23	48	21	31	31	31	23
AAT2	fome13fome13-T	512	172	208	135	87	140	162	59	26	61	63	54	30	59	26	61	63	54	30
		1024	190	207	140	79	175	218	61	26	98	127	68	32	61	26	98	127	68	32
		256	131	153	99	81	104	105	45	19	61	48	41	26	45	19	61	48	41	26
AAT3	fome21fome21-T	512	165	216	123	115	136	155	50	21	75	61	55	29	50	21	75	61	55	29
		1024	180	246	148	161	191	216	52	23	84	74	56	28	52	23	84	74	56	28
		256	117	125	80	86	88	93	22	17	71	25	28	16	22	17	71	25	28	16
AAT4	fxm3_16fxm3_16-T	512	157	175	100	132	139	137	28	21	90	29	31	21	28	21	90	29	31	21
		1024	187	222	116	177	177	208	34	17	110	31	30	19	34	17	110	31	30	19

Table 7.4: Results of communication hypergraph models $\mathcal{H}_{\text{cr}}^{\text{C}}$, $\mathcal{H}_{\text{rc}}^{\text{C}}$, and $\mathcal{H}_{\text{rr}}^{\text{C}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{cr}}^{\text{C}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rc}}^{\text{C}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{rr}}^{\text{C}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{cr}}^{\text{C}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rc}}^{\text{C}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{rr}}^{\text{C}}$	$\mathcal{H}_{\text{cr}}$	$\mathcal{H}_{\text{cr}}^{\text{C}}$	$\mathcal{H}_{\text{rc}}$	$\mathcal{H}_{\text{rc}}^{\text{C}}$	$\mathcal{H}_{\text{rr}}$	$\mathcal{H}_{\text{rr}}^{\text{C}}$
		256	97	114	74	82	80	88	25	10	45	24	31	13	25	10	45	24	31	13
AAT5	fxm4_6fxm4_6-T	512	130	164	100	119	111	134	25	13	50	22	32	13	25	13	50	22	32	13
		1024	157	178	106	147	141	172	29	17	65	23	44	16	29	17	65	23	44	16
		256	88	118	67	64	75	55	46	20	64	53	44	58	46	20	64	53	44	58
AAT6	lp_pds_20lp_pds_20-T	512	96	141	82	85	101	117	52	23	79	65	52	27	52	23	79	65	52	27
		1024	110	147	85	97	128	146	48	24	80	89	45	25	48	24	80	89	45	25
		256	104	133	84	76	91	94	50	21	69	55	48	29	50	21	69	55	48	29
AAT7	pds-30pds-30-T	512	131	185	107	104	118	141	52	22	76	63	52	29	52	22	76	63	52	29
		1024	143	160	107	133	154	187	53	23	86	90	56	26	53	23	86	90	56	26
		256	129	155	99	88	103	101	48	19	63	47	43	29	48	19	63	47	43	29
AAT8	pds-40pds-40-T	512	169	213	132	117	134	151	51	22	74	66	55	32	51	22	74	66	55	32
		1024	184	202	143	162	181	214	53	23	90	94	56	30	53	23	90	94	56	30
		256	141	166	104	91	110	109	47	20	65	50	43	30	47	20	65	50	43	30
AAT9	pds-50pds-50-T	512	189	214	144	134	154	169	54	22	83	64	52	30	54	22	83	64	52	30

Table 7.4: Results of communication hypergraph models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$
		1024	209	186	171	188	209	235	55	25	86	100	57	30	55	25	86	100	57	30
		256	149	175	118	96	112	119	50	20	61	48	52	29	50	20	61	48	52	29
AAT10	pds-60pds-60-T	512	211	247	155	141	164	175	53	23	75	67	58	33	53	23	75	67	58	33
		1024	239	227	184	202	226	257	63	24	86	100	60	32	63	24	86	100	60	32
		256	169	187	130	105	130	131	48	22	59	59	45	30	48	22	59	59	45	30
AAT11	pds-90pds-90-T	512	255	282	187	160	195	202	55	22	86	84	66	30	55	22	86	84	66	30
		1024	301	233	207	224	271	297	61	25	103	122	59	32	61	25	103	122	59	32
		256	159	180	105	128	115	136	39	20	79	56	61	28	39	20	79	56	61	28
AAT12	sgpf5y6sgpf5y6-T	512	196	269	113	157	150	188	52	28	127	93	104	35	52	28	127	93	104	35
		1024	215	348	103	194	170	233	56	31	153	125	125	55	56	31	153	125	125	55
		256	188	178	165	161	158	158	13	10	12	12	12	9	13	10	12	12	12	9
AAT13	watson_1watson_1-T	512	291	287	236	224	224	236	19	11	15	11	16	11	19	11	15	11	16	11
		1024	358	437	312	308	298	319	26	14	21	14	19	11	26	14	21	14	19	11

Table 7.4: Results of communication hypergraph models $\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$, $\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$, and $\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$
		256	217	220	177	172	170	175	18	11	13	13	13	9	18	11	13	13	13	9
AAT14	watson_2watson_2-T	512	363	373	282	275	253	269	19	11	18	13	17	10	19	11	18	13	17	10
		1024	502	583	397	406	375	405	24	13	27	15	21	14	24	13	27	15	21	14
		256	140	159	110	102	113	114	33	17	42	33	32	22	33	17	42	33	32	22
avg.		512	194	231	148	145	161	178	39	19	55	41	40	22	39	19	55	41	40	22
		1024	228	258	172	193	218	251	42	21	67	54	43	23	42	21	67	54	43	23
$C = AA$																				
		256	145	140	163	153	163	163	49	30	43	25	36	17	49	30	43	25	36	17
AA1	144144	512	267	259	299	299	307	314	55	31	43	31	39	18	55	31	43	31	39	18
		1024	422	418	520	530	543	586	62	32	49	39	39	18	62	32	49	39	39	18
		256	195	185	196	177	197	191	43	28	29	23	24	17	43	28	29	23	24	17
AA2	2cubes_sphere2cubes_sphere	512	290	312	336	320	355	356	49	31	36	28	29	19	49	31	36	28	29	19
		1024	376	307	536	528	588	614	60	33	39	33	30	18	60	33	39	33	30	18

Table 7.4: Results of communication hypergraph models $\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$, $\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$, and $\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$
		256	236	230	215	210	222	212	17	14	9	9	9	9	17	14	9	9	9	9
AA3	Chevron4Chevron4	512	421	412	467	447	468	458	18	15	12	12	11	10	18	15	12	12	11	10
		1024	671	699	863	832	874	859	21	16	12	10	11	9	21	16	12	10	11	9
		256	192	184	194	169	193	185	40	27	27	21	22	18	40	27	27	21	22	18
AA4	cp2k-h2o-.5e7cp2k-h2o-.5e7	512	351	337	355	317	362	353	37	28	28	22	25	20	37	28	28	22	25	20
		1024	273	306	622	585	648	641	39	28	31	23	23	18	39	28	31	23	23	18
		256	199	198	196	176	199	195	35	22	23	20	23	15	35	22	23	20	23	15
AA5	cp2k-h2o-e6cp2k-h2o-e6	512	348	353	357	322	360	358	36	23	24	18	21	16	36	23	24	18	21	16
		1024	476	538	588	542	606	631	34	22	24	19	22	14	34	22	24	19	22	14
		256	124	123	184	164	172	167	39	25	14	16	19	18	39	25	14	16	19	18
AA6	mac_econ_fwd500mac_econ_fwd500	512	199	214	297	268	289	279	65	22	27	25	31	23	65	22	27	25	31	23
		1024	317	362	398	382	401	419	89	24	36	32	43	29	89	24	36	32	43	29
		256	170	167	194	179	199	198	22	14	10	11	9	7	22	14	10	11	9	7
AA7	majorbasismajorbasis	512	286	280	343	317	348	352	21	15	12	10	10	9	21	15	12	10	10	9

Table 7.4: Results of communication hypergraph models $\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$, $\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$, and $\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}^{\mathrm{C}}_{\mathrm{rr}}$
		1024	430	430	536	531	567	580	25	16	13	12	11	8	25	16	13	12	11	8
		256	205	205	210	205	211	208	15	14	10	9	9	8	15	14	10	9	9	8
AA8	mario002mario002	512	378	381	396	379	391	390	16	15	10	9	9	9	16	15	10	9	9	9
		1024	588	603	680	652	672	691	17	14	12	10	11	8	17	14	12	10	11	8
		256	246	245	239	198	235	177	17	15	9	16	10	16	17	15	9	16	10	16
AA9	mc2depimc2depi	512	444	457	425	357	419	323	18	14	9	16	9	15	18	14	9	16	9	15
		1024	581	622	723	599	710	566	19	15	11	17	10	16	19	15	11	17	10	16
		256	134	130	125	129	144	153	58	30	69	70	45	26	58	30	69	70	45	26
AA10	poisson3Dapoisson3Da	512	177	179	160	180	230	260	86	36	110	112	51	31	86	36	110	112	51	31
		1024	230	259	180	223	292	376	99	37	180	152	78	38	99	37	180	152	78	38
		256	33	75	141	143	140	158	40	24	39	48	66	22	40	24	39	48	66	22
AA11	scircuitscircuit	512	40	75	178	202	174	234	45	24	65	75	83	27	45	24	65	75	83	27
		1024	39	73	204	227	190	324	72	28	102	127	129	47	72	28	102	127	129	47

Table 7.4: Results of communication hypergraph models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$
		256	214	214	222	217	220	216	19	15	9	10	11	9	19	15	9	10	11	9
AA12	t2emt2em	512	401	396	419	406	411	408	19	15	11	10	10	10	19	15	11	10	10	10
		1024	704	697	748	728	742	725	20	16	11	10	11	9	20	16	11	10	11	9
		256	210	205	214	203	212	209	16	16	10	10	9	9	16	16	10	10	9	9
AA13	thermomech_dKthermomech_dK	512	395	391	406	386	415	408	19	14	10	9	10	8	19	14	10	9	10	8
		1024	692	678	748	705	758	755	21	16	12	11	11	9	21	16	12	11	11	9
		256	226	223	231	224	231	228	17	15	9	9	11	9	17	15	9	9	11	9
AA14	tmt_symtmt_sym	512	425	417	442	425	443	436	18	16	12	11	10	10	18	16	12	11	10	10
		1024	718	727	809	779	803	792	20	16	12	10	11	10	20	16	12	10	11	10
		256	172	168	190	180	192	187	16	13	9	9	8	7	16	13	9	9	8	7
AA15	torso2torso2	512	267	266	313	303	327	325	18	15	10	9	10	10	18	15	10	9	10	10
		1024	378	396	479	456	482	498	21	15	11	12	10	8	21	15	11	12	10	8
		256	190	186	186	169	189	182	26	21	22	18	18	15	26	21	22	18	18	15
AA16	xenon2xenon2	512	355	346	351	321	359	347	33	24	23	23	19	15	33	24	23	23	19	15

Table 7.4: Results of communication hypergraph models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$
		1024	592	608	618	576	653	640	39	25	26	25	22	15	39	25	26	25	22	15
		256	168	173	191	179	193	188	26	19	17	16	16	13	26	19	17	16	16	13
avg.		512	283	295	334	320	344	345	30	20	20	19	18	14	30	20	20	19	18	14
		1024	402	430	535	523	560	587	34	21	24	22	21	15	34	21	24	22	21	15
$C = AB$																				
		256	166	171	9	52	119	85	54	43	255	255	209	249	54	43	255	255	209	249
AB1	amazon0302amazon0302-user	512	239	323	8	31	158	114	104	42	511	511	313	435	104	42	511	511	313	435
		1024	309	558	9	24	226	176	191	39	1023	1023	329	541	191	39	1023	1023	329	541
		256	79	114	9	56	110	83	231	81	255	255	253	255	231	81	255	255	253	255
AB2	amazon0312amazon0312-user	512	101	222	9	69	152	114	386	102	511	511	499	511	386	102	511	511	499	511
		1024	63	412	8	36	159	144	704	116	1023	1023	854	991	704	116	1023	1023	854	991
		256	167	165	192	162	195	168	19	14	10	18	11	14	19	14	10	18	11	14
AB3	crashbasismajorbasis	512	274	274	342	294	349	302	23	16	11	19	10	16	23	16	11	19	10	16

Table 7.4: Results of communication hypergraph models $\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$, $\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$, and $\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$ (continued)

									Number of sent messages											
			Speedup						Average						Maximum					
Instance	Matrix	K	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{cr}}$	$\mathcal{H}_{\mathrm{cr}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rc}}$	$\mathcal{H}_{\mathrm{rc}}^{\mathrm{C}}$	$\mathcal{H}_{\mathrm{rr}}$	$\mathcal{H}_{\mathrm{rr}}^{\mathrm{C}}$
		1024	428	441	544	503	575	520	24	17	14	23	11	16	24	17	14	23	11	16
		256	208	206	210	203	211	209	16	14	10	9	9	9	16	14	10	9	9	9
AB4	darcy003mario002	512	382	382	393	378	388	392	16	14	10	9	11	10	16	14	10	9	11	10
		1024	584	607	682	663	683	681	17	14	12	10	11	10	17	14	12	10	11	10
		256	170	166	218	194	216	187	18	14	10	18	9	15	18	14	10	18	9	15
AB5	thermomech_dKthermomech_dM	512	321	313	420	372	415	363	18	14	11	16	10	15	18	14	11	16	10	15
		1024	565	569	756	679	750	665	20	15	12	20	11	18	20	15	12	20	11	18
		256	151	162	58	113	163	136	37	25	36	45	34	41	37	25	36	45	34	41
avg.		512	241	298	83	155	267	224	48	27	50	59	44	56	48	27	50	59	44	56
		1024	307	511	116	181	403	359	64	28	73	86	51	69	64	28	73	86	51	69
		256	154	166	129	134	152	147	30	19	27	25	24	19	30	19	27	25	24	19
Overall		512	238	268	198	210	245	249	35	21	34	30	28	21	35	21	34	30	28	21
		1024	308	359	273	302	366	390	41	22	42	38	32	22	41	22	42	38	32	22

7.3.5 Speedup Curves

The speedup curves are presented in Figures 7.1- 7.7. The names of input matrices are given on the top of each plot. The sequential running time T_s of a single SpGEMM operation is also given for each SpGEMM instance in the bottom right corner of each plot. T_s is given in terms of seconds. In the figures, the x-axis show the number of processors, i.e., 256, 512, and 1024. This axis is in logarithm scale. The y-axis shows the speedup.

As seen in the figures, at least one of the proposed models achieve significant scalability. In general, the use of the communication hypergraph improves the proposed hypergraph models, which consider only minimization of the communication volume. Especially for the $C = AA^T$ category, the best performance is obtained through using the communication hypergraph. This can be attributed to the very irregular sparsity pattern of the input LP constraint matrices. As a result, in most of the SpGEMM instances, the proposed models scale up to 1024 processors, thus these results verify the empirical validity of the proposed models and methods.

The superiority of the proposed algorithms and models depends on the amount of imbalance in computation and communication loads of processors. For the computation phase, the only performance issue computational load imbalance, whereas where are multiple quality criteria for the communication phase. Four of these quality criteria for evaluating the performance of the communication phase are presented in Tables 7.3 and 7.4.

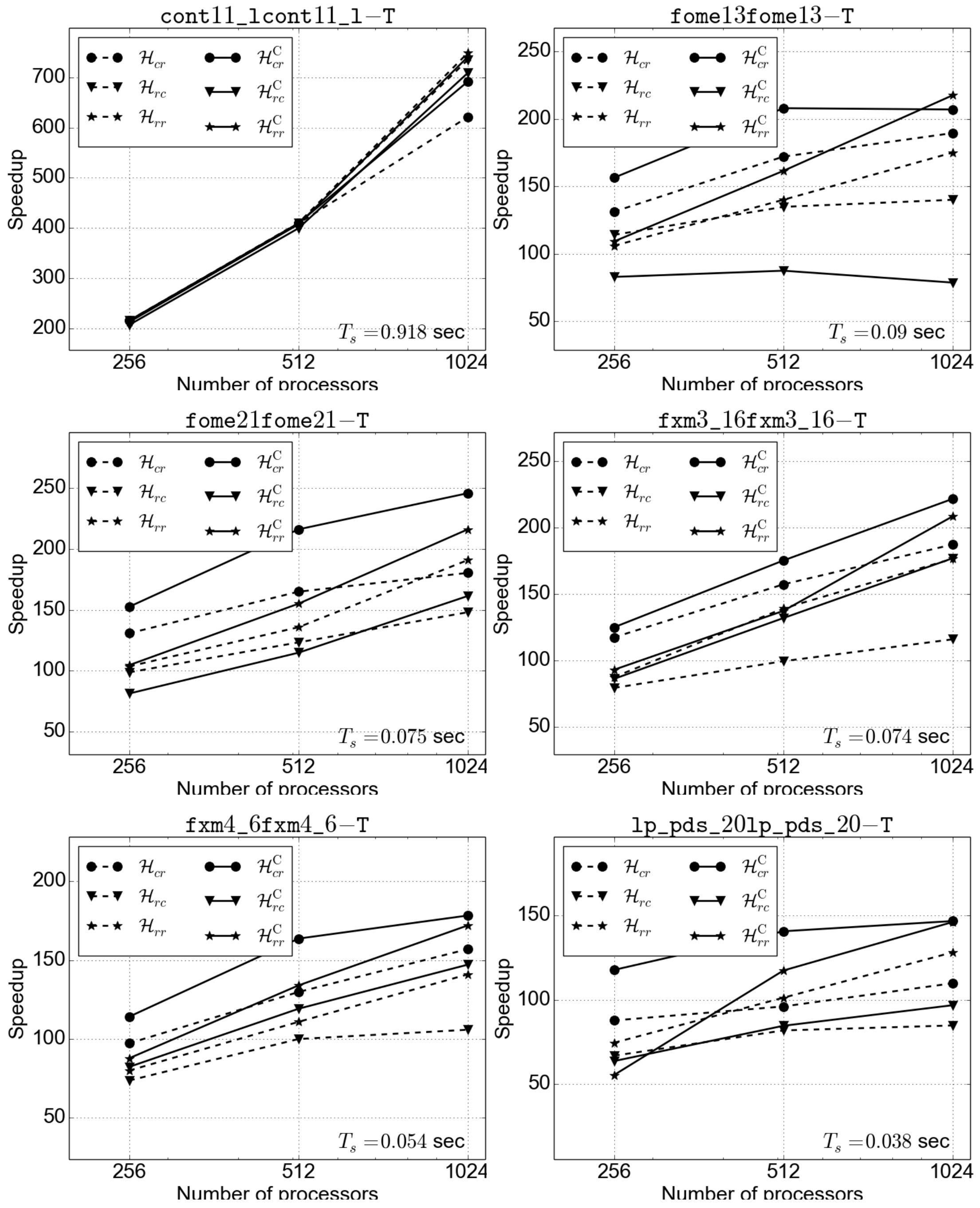


Figure 7.1: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA^T$ category

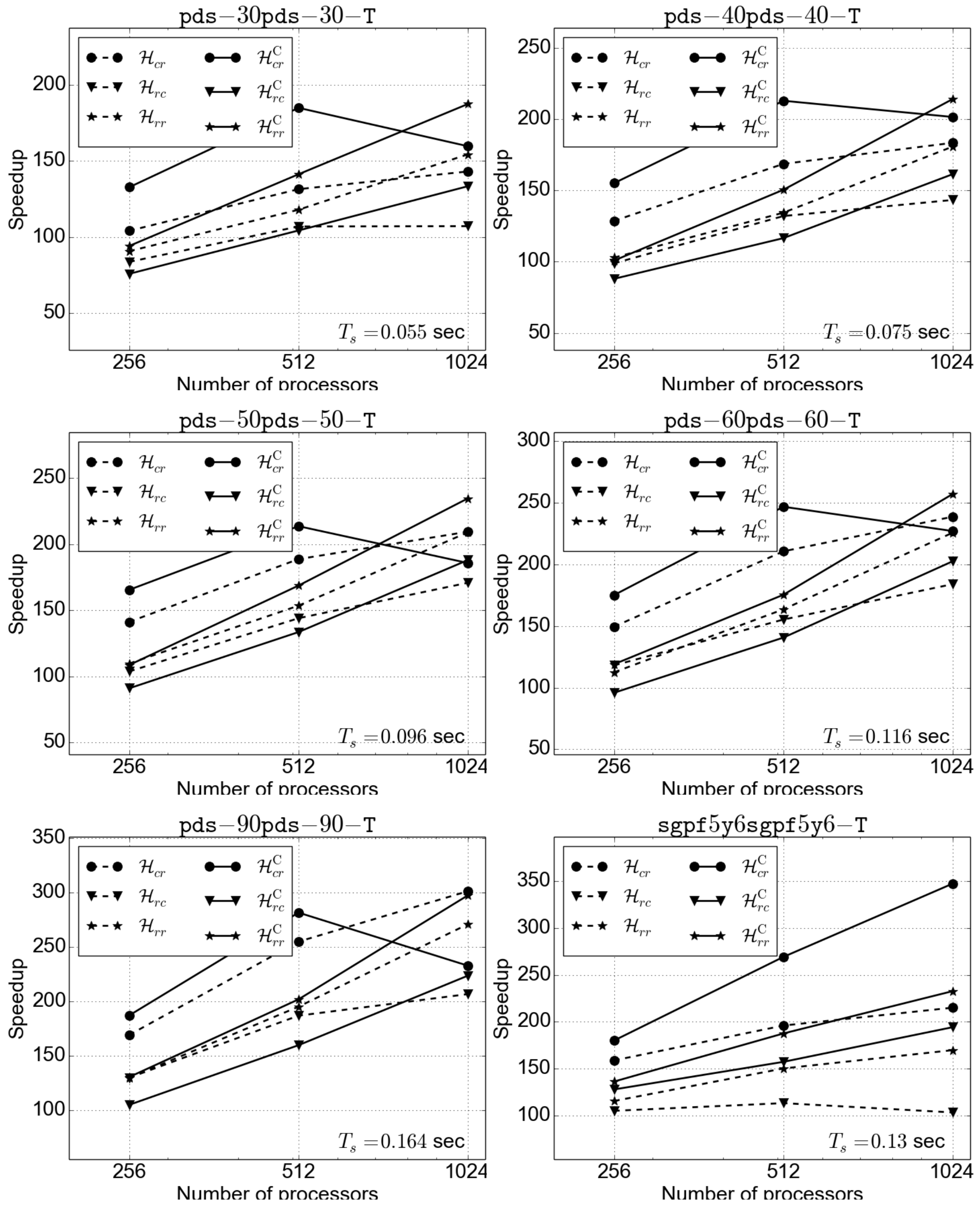


Figure 7.2: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA^T$ category

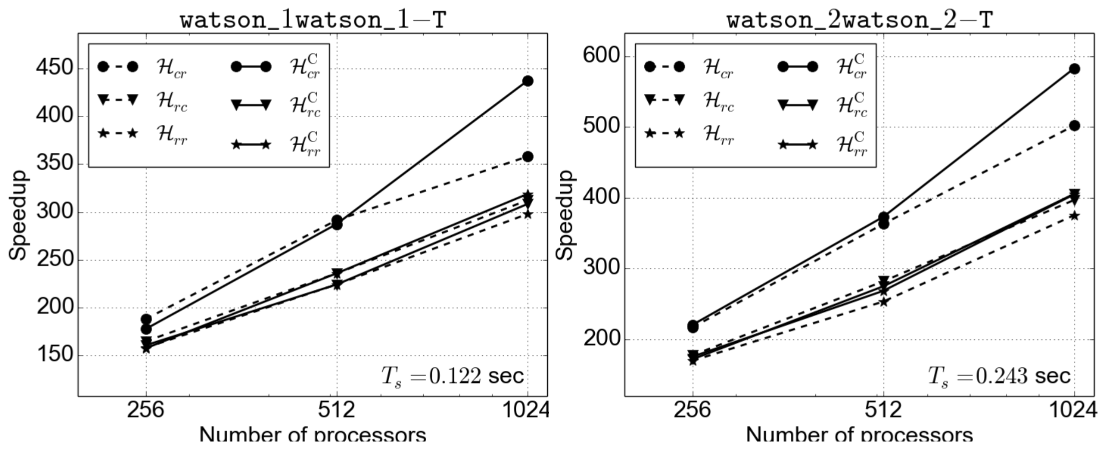


Figure 7.3: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA^T$ category

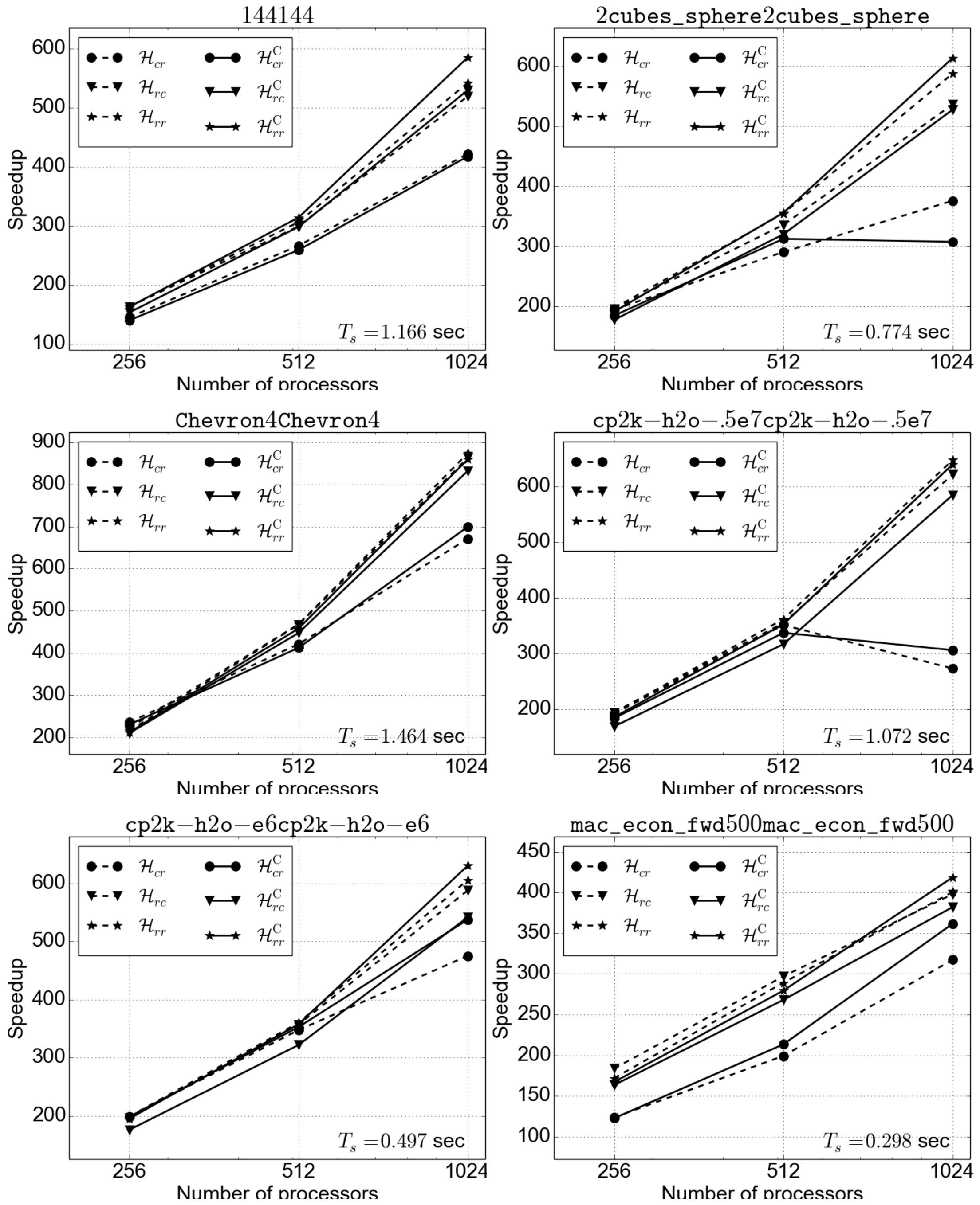
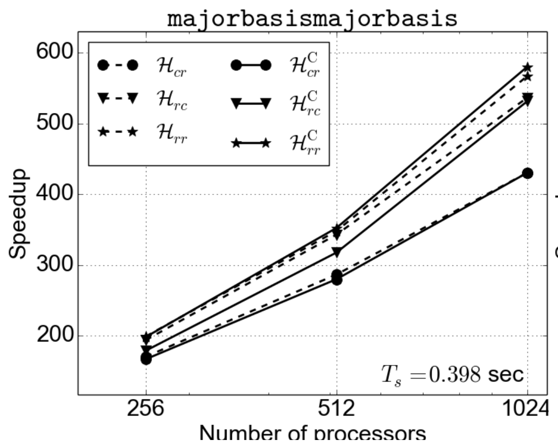
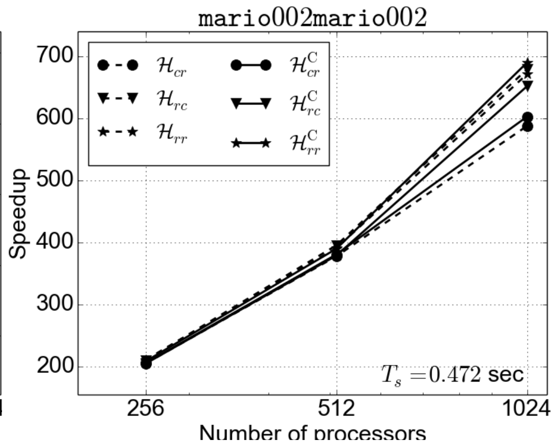
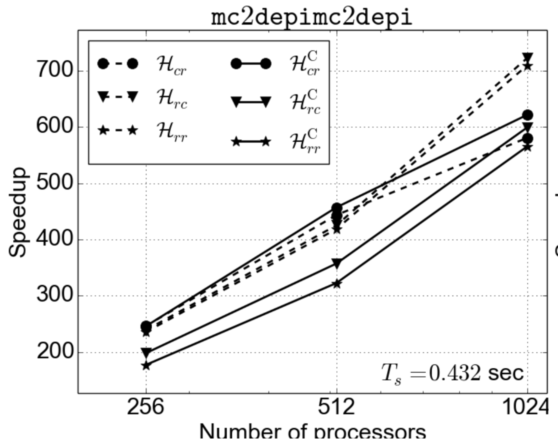
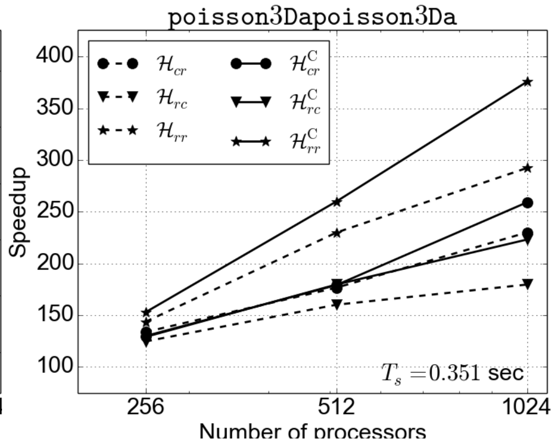


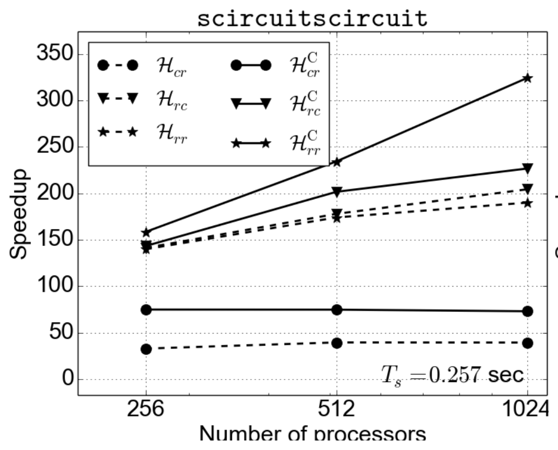
Figure 7.4: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA$ category











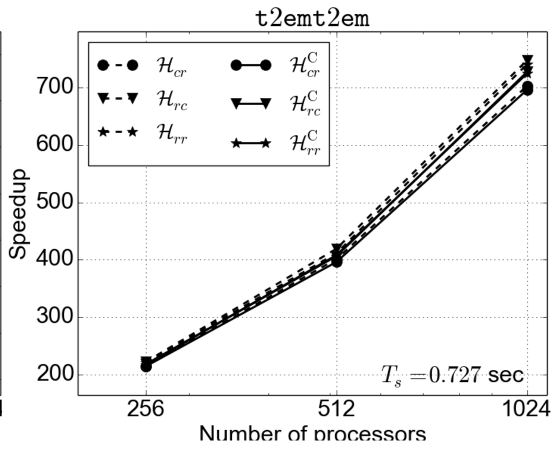


Figure 7.5: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA$ category

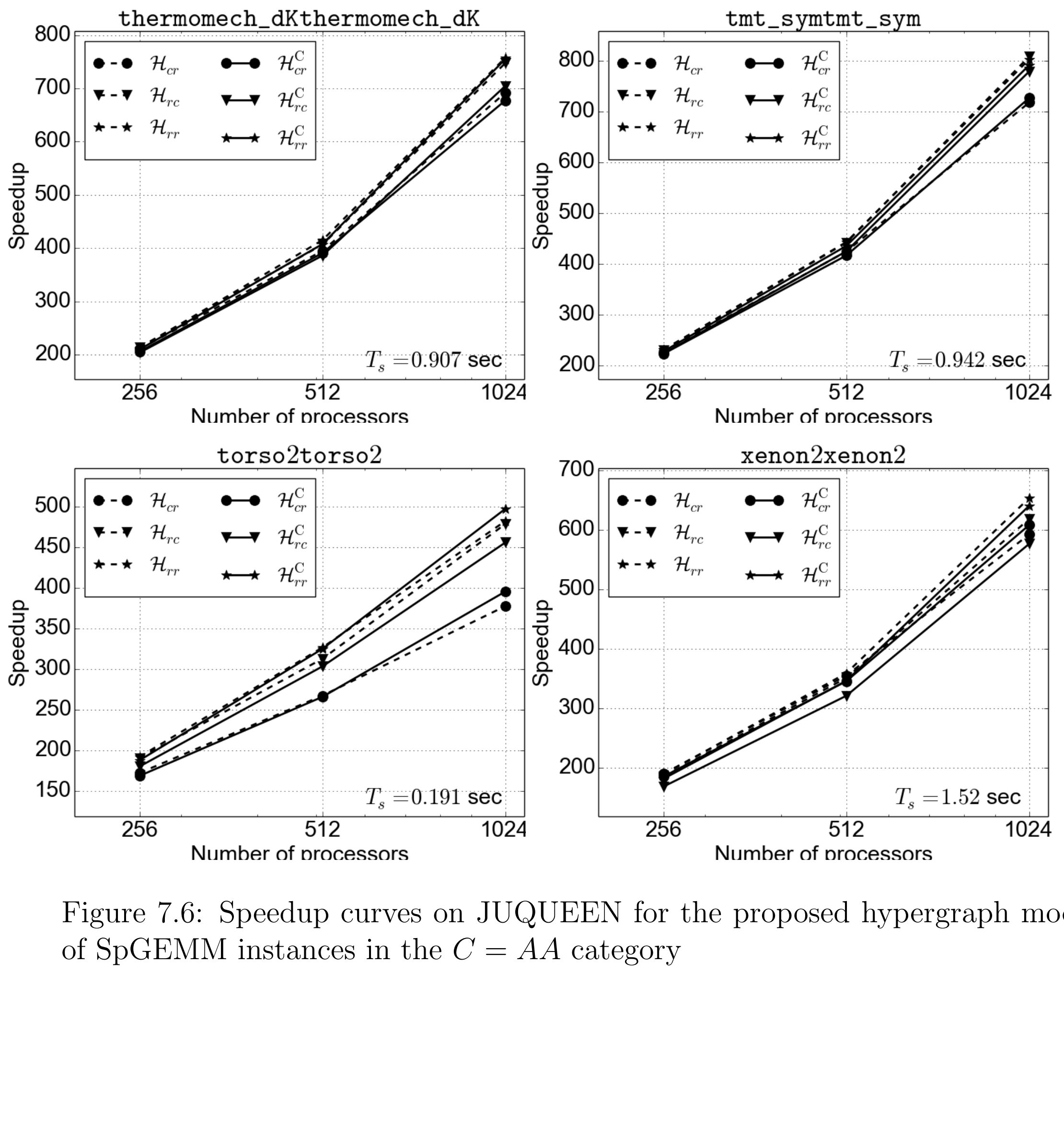


Figure 7.6: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AA$ category

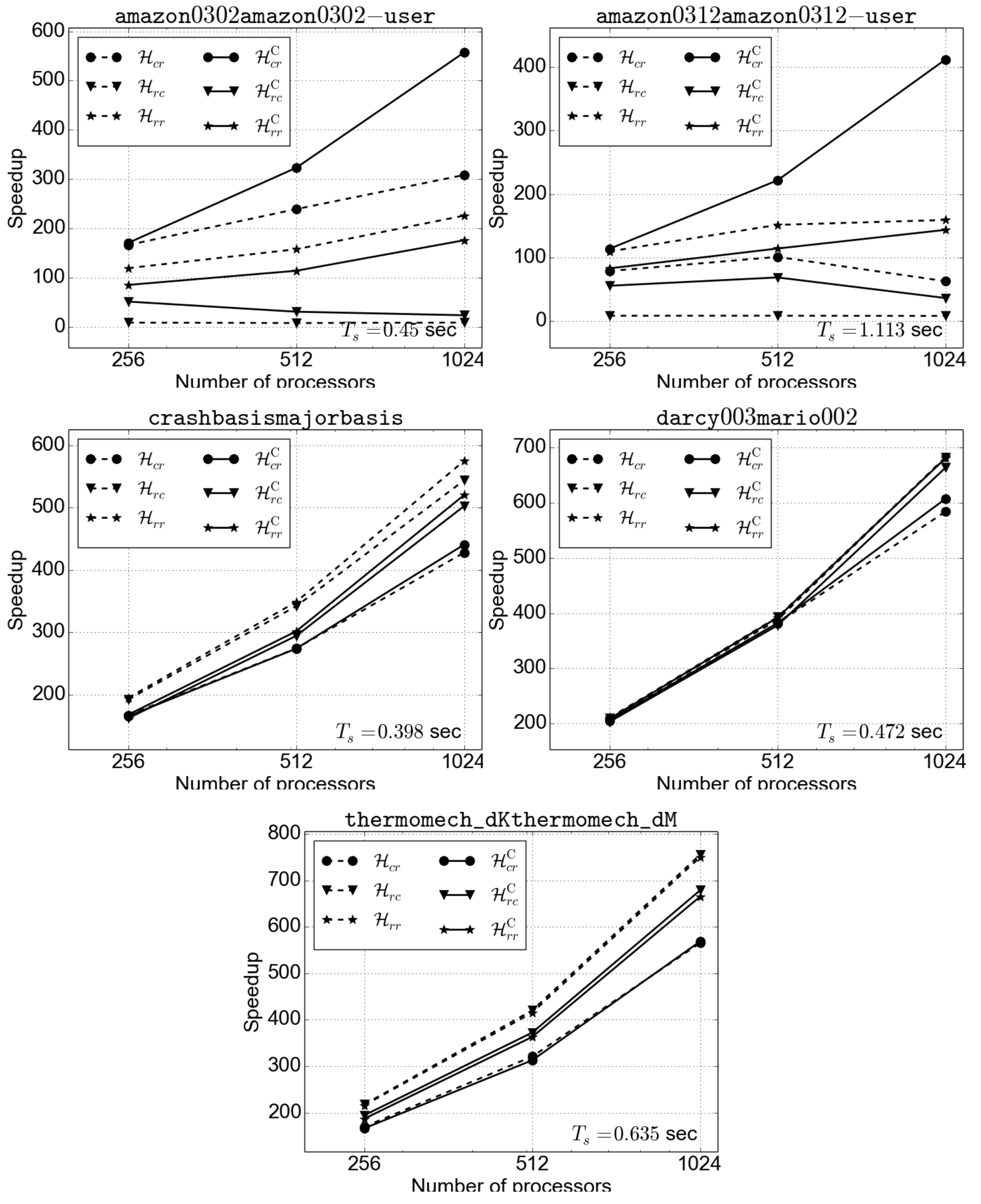


Figure 7.7: Speedup curves on JUQUEEN for the proposed hypergraph models of SpGEMM instances in the $C = AB$ category

Chapter 8

Conclusion

We proposed three parallel SpGEMM algorithms and three hypergraph models for these algorithms. These models achieve simultaneous partitioning of input and output matrices for sparse matrix-matrix multiplication (SpGEMM) of the form $C = AB$. The proposed algorithms contain two separate phases: multiplication and communication phases. In all of the three hypergraph models, there exists a vertex in order to represent an atomic task of computation in the multiplication phase of the two-phase SpGEMM algorithms.. In all models, there exists a hyperedge (net) for a communicated entity of matrices in order to encode the total volume of communication that will occur during the communication phase of the two-phase SpGEMM algorithms. The constraints used in partitioning the proposed hypergraph models correspond to balancing computational loads of processors. The partitioning objective of minimizing cutsize corresponds to minimizing the total volume of communication, which occur in the communication phase.

We also proposed models for reducing the total number of messages while maintaining balance on communication volumes handled by processors during the communication phase of the SpGEMM algorithms. The performance improvement by the proposed hypergraph models for reducing communication volume was further enhanced by the use of the communication hypergraph models in a second stage. In this second stage, the partitioning information of the first stage was preprocessed. This preprocessing step consisted of construction of the respective communication hypergraph model and partitioning it. Minimizing the total number of messages transferred over network was shown to be encoded by the partitioning objective of minimizing cutsize. Maintaining balance on communication volumes handled by processors was shown to be encoded by the partitioning constraint of balancing part weights.

The validity of the proposed models and methods were empirically tested on a wide range of sparse matrices. We developed an SpGEMM library based on the MPI (Message Passing Interface) library. This library contains the proposed three parallel SpGEMM algorithms and matrix partitioning tools for these algorithms Parallel SpGEMM runs on large-scale distributed memory IBM BlueGene/Q system, named JUQUEEN, showed that the proposed models achieve high speedup values.

Chapter 9

Future Work

In this thesis, we only consider multiplication of two sparse matrices. There exist applications that involve triple matrix product. One of such applications is algebraic multigrid solver. The product $P^T AP$ is formed to construct the grid hierarchy of an algebraic multigrid partial differential equation (PDE) solver [51, 52]. Hypergraph partitioning based models and methods can be investigated in order to model the communication costs and computation loads; and this model can be used to reduce the communication cost during this triple matrix product.

Obtaining increasing speedup on an unbounded number of processors becomes very important to reach exascale computing power via combining compute nodes by using interconnection networks. For this type of parallel systems, the significant factor in the communication overhead becomes the number of messages when the number of processors increases. In other words, communication overhead due to volume is dominated by the overhead of message latency [53].

One of the solutions for decreasing message latency overhead is using collective communication primitives instead of using point-to-point communication when the number of messages is large enough. The collective communication achieves reducing the number of concurrently sent messages with respect to the point-to-point scheme. So, network congestion is reduced. Efficient collective communication schemes can be integrated in the communication phase of our parallel SpGEMM library.

Appendix A

The Parallel SpGEMM Library

A.1 Quick Start

1. Download the latest version of the library from `http://sites.google.com/site/kadircs/`.
2. Decompress the downloaded archive.
3. Set the variable named `ROOT` defined in `Makefile.inc` to the full path of the uncompressed library folder.
4. Compile the source code using the following command:

   ```
   > make
   ```

5. The following command runs the library for a small dataset. The name of the matrix is `smallA` and $C = AA$ operation is performed.

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

A.2 File Format for Sparse Matrices

The SpGEMM library uses binary format for input and output matrices. Storage size of the binary format is less than the storage size of the Matrix Market [54] format so read and write operations of large sparse matrices are faster. The following command can be used for conversion of a sparse matrix in Matrix Market format to the binary format utilized by the library:

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

For converting a sparse matrix in the binary format utilized by the library to Matrix Market format:

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

A.3 Preprocessing Step for Partitioning Input and Output Matrices

Prior to multiplication of matrices, partitioning information must be obtained via this preprocessing step.

The hypergraph models for outer-product and inner-product formulations depend on the sparsity pattern of the output C matrix and all of the multiplication routines require C matrix for symbolic multiplication. So the output C matrix must be generated before preprocessing and multiplication steps using the `./sspmxm/smult` program as follows:

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

The parameters of `./sspmxm/smult`:

1) `A.bin`	input matrix A in binary format
2) `B.bin`	input matrix B in binary format
3) `C.bin`	output matrix C in binary format
4) `resultFile.txt`	the statistics related with the program are appended to this file
5) `CSR_SCHEME`	the scheme used for multiplication. `CSR_SKIP_ZERO_ROWS` ensures that A-matrix rows that will not incur any computation are skipped using the sparsity pattern of the output C matrix.

Partition information for the parallel SpGEMM computation is obtained by using the `./preprocess/preprocess` program, which takes the following sequence of parameters:

1) `P` number of partitions, which is also equal to number of processors

2) `FORMULATION` matrix multiplication formulation used in the parallel algorithm:

- `FORM_OUTER`: Outer-product formulation
- `FORM_INNER`: Inner-product formulation
- `FORM_ROWROW`: Row-by-row formulation

2) `METRIC` the objective of partitioning:

- `CON`: the cutsize calculated according to the "connectivity-1 metric"
- `CUT`: the cutsize calculated according to the "cutnet metric"

3) `IMBAL`	maximum allowed value for the constraint of the partitioning. The given value must be in the range of [0.0 ... 0.5]
4) `RCNETCOST`	for experimentation only, use `RCNNZ`
5) `ZNETCOST`	scheme used for calculating costs of nets in the hypergraph • `ZABVERT`: cost of a net is the equal to the input vertices connected by that net • `ZUNIT`: all nets have unit cost
6) `MATRIX FORMAT`	• `MTX`: Matrix Market format • `TRIBIN`: Binary file consisting of row, column, value tuples
7) `MATRIX A`	path of the input matrix A stored in binary format
8) `MATRIX B`	path of the input matrix B stored in binary format
9) `MATRIX C`	path of the output matrix C stored in binary format. C matrix will be used in construction of the hypergraph model of the SpGEMM computation
10) `RESULTFILE`	experimental results related with partitioning will be either printed to standard output or appended to a the given text file • `stdout`: standard out • `FILENAME`: name of the file to which results will be appended

11) `PART`

- `DOPART`: perform partitioning and write partition information into file
- `LOADPARTVEC`: read partition information form file
- `CONSTRUCT_HYPERGRAPH_ONLY`: only constructs the hypergraph of the SpGEMM computation and exits

12) `PARTVECFILE` name of text file which will contain partition information

13) `PARTMTX`

- `PARTMTX`: input and output matrices will also be partitioned and written to files
- `PARTONLY`: matrices will not be partitioned

14) `HYPERGRAPH MODEL`

- `HPMODEL_C_NZ`: hypergraph model for outer-product formulation (nonzero-based partitioning of C matrix)
- `HPMODEL_C_ROW`: hypergraph model for outer-product formulation (row-based partitioning of C matrix)
- `HPMODEL_C_COL`: hypergraph model for outer-product formulation (column-based partitioning of C matrix)
- `HPMODEL_INNER_B_SUBCOL`: hypergraph model for inner-product formulation (A-resident. Only required nonzeros of B matrix is communicated.)
- `HPMODEL_ROWROW_AC_ROW`: hypergraph model for row-by-row formulation (Both A and C matrices are partitioned rowwise.)

15) `STRMTXC`	use the value of the above-mentioned parameter `MATRIX C`
16) `STRMTXCSS`	use `null` because this parameter is used for other models and methods that are implemented for experimental purposes

17) `OUTPUT VERTEX WEIGHTS`	weighting scheme of output vertices, which represent the communication operations • `ZERO`: zero weight • `UNIT`: unit weight • `ABVERTICES_OF_ZNET`: weight of an output vertex is equal to 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`	weighting scheme of output vertices, which represent the communication operations • `ABMULT`: number of nets that connect the vertex • `ABMULTIROW`: another weighting scheme for testing purposes
20) `PARTREPEAT`	number of partitionings. The given value must be greater than 0.

A.4 Parallel SpGEMM Computation

The parallel SpGEMM computation is performed using the `./pspmxm/pspmxm` program. This program needs an MPI library installed in the system. The following sequence of parameters is required by the program:

1) `P`	total number of MPI ranks
2) `STRMTXA`	use `null` because this parameter is used for other models and methods that are implemented for experimental purposes

3) `STROUTPUTMTX`	use `null` because this parameter is used for other models and methods that are implemented for experimental purposes
4) `MATRIX C`	path to the output matrix C that will be computed in parallel
5) `COMMUNICATION PATTERN`	communication pattern that will be used in the communication phase. Use `NOCOMM` because this parameter is used for other models and methods that are implemented for experimental purposes.
6) `MULTIPLICATION TYPE`	use `SYMBOLIC`
7) `PARTVEC`	path to the text file that contains partition information
8) `CSR SCHEME`	• `CSR_NORMAL`: Gustavson's sequential SpGEMM algorithm • `CSR_SKIP_ZERO_ROWS`: The outermost for-loop iterates over nonempty rows of matrix C
9) `SEND MODE`	use `NORMAL`
10) `WRITE C MATRIX`	• `WRITEMTX`: write the computed C matrix into file • `DONTWRITEMTX`: do not write the computed C matrix
2) `FORMULATION`	matrix multiplication formulation used in the parallel algorithm
11) `HYPERGRAPH MODEL`	the hypergraph model used in partitioning
12) `MATRIX A`	path to input matrix A stored in binary format
13) `MATRIX B`	path to input matrix B stored in binary format
14) `MATRIX C`	path to output matrix C stored in binary format to be loaded for the parallel symbolic multiplication

15) PERFORM NUMERICAL CHECK	• PERFORM_CHECK_IN_PREPROCESSING: check numerical equality of the loaded input matrix C and the matrix C computed by the parallel SpGEMM operation • NO_CHECK_IN_PREPROCESSING: no check
16) PARTREPEAT	number of partitionings. The given value must be greater than 0.

Bibliography

[1] M. Challacombe, "A general parallel sparse-blocked matrix multiply for linear scaling SCF theory," *Computer Physics Communications*, vol. 128, no. 12, pp. 93 – 107, 2000.

[2] J. VandeVondele, U. Borstnik, and J. Hutter, "Linear scaling self-consistent field calculations with millions of atoms in the condensed phase," *Journal of Chemical Theory and Computation*, vol. 8, no. 10, pp. 3565–3573, 2012.

[3] M. Challacombe, "A simplified density matrix minimization for linear scaling self-consistent field theory," *The Journal of Chemical Physics*, vol. 110, no. 5, pp. 2332–2342, 1999.

[4] S. Itoh, P. Ordejn, and R. M. Martin, "Order-N tight-binding molecular dynamics on parallel computers," *Computer Physics Communications*, vol. 88, no. 2-3, pp. 173 – 185, 1995.

[5] H. B. Schlegel, J. M. Millam, S. S. Iyengar, G. A. Voth, A. D. Daniels, G. E. Scuseria, and M. J. Frisch, "Ab initio molecular dynamics: Propagating the density matrix with gaussian orbitals," *The Journal of Chemical Physics*, vol. 114, no. 22, pp. 9758–9763, 2001.

[6] X.-P. Li, R. W. Nunes, and D. Vanderbilt, "Density-matrix electronic-structure method with linear system-size scaling," *Physical Review B*, vol. 47, pp. 10891–10894, Apr 1993.

[7] J. M. Millam and G. E. Scuseria, "Linear scaling conjugate gradient density matrix search as an alternative to diagonalization for first principles electronic structure calculations," *The Journal of Chemical Physics*, vol. 106, no. 13, pp. 5569–5577, 1997.

[8] A. D. Daniels, J. M. Millam, and G. E. Scuseria, "Semiempirical methods with conjugate gradient density matrix search to replace diagonalization for molecular systems containing thousands of atoms," *The Journal of Chemical Physics*, vol. 107, no. 2, pp. 425–431, 1997.

[9] "CP2K home page." `http://www.cp2k.org/`.

[10] M. O. Rabin and V. V. Vazirani, "Maximum matchings in general graphs through randomization," *Journal of Algorithms*, vol. 10, no. 4, pp. 557–567, 1989.

[11] P. D'Alberto and A. Nicolau, "R-kleene: A high-performance divide-and-conquer algorithm for the all-pair shortest path for densely connected networks," *Algorithmica*, vol. 47, no. 2, pp. 203–213, 2007.

[12] R. Yuster and U. Zwick, "Detecting short directed cycles using rectangular matrix multiplication and dynamic programming," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, (Philadelphia, PA, USA), pp. 254–260, Society for Industrial and Applied Mathematics, 2004.

[13] J. R. Gilbert, V. B. Shah, and S. Reinhardt, "A unified framework for numerical and combinatorial computing," *Computing in Science & Engineering*, vol. 10, no. 2, pp. 20–25, 2008.

[14] V. B. Shah, *An interactive system for combinatorial scientific computing with an emphasis on programmer productivity.* PhD thesis, UNIVERSITY OF CALIFORNIA Santa Barbara, 2007.

[15] A. Buluç and J. R. Gilbert, "Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments," *SIAM Journal of Scientific Computing (SISC)*, vol. 34, no. 4, pp. 170 – 191, 2012.

[16] A. Buluc and J. Gilbert, “On the representation and multiplication of hypersparse matrices,” in *IEEE International Symposium on Parallel and Distributed Processing*, IPDPS’08, pp. 1–11, April 2008.

[17] A. Buluç and J. R. Gilbert, “Highly parallel sparse matrix-matrix multiplication,” Tech. Rep. UCSB-CS-2010-10, University of California, Santa Barbara, Computer Science Department, June 2010.

[18] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.

[19] G. Karypis, A. Gupta, and V. Kumar, “A parallel formulation of interior point algorithms,” in *Supercomputing 94*, 1994.

[20] R. H. Bisseling, T. M. Doup, and L. Loyens, “A parallel interior point algorithm for linear programming on a network of transputers,” *Annals of Operations Research*, vol. 43, pp. 51–86, 1993.

[21] E. Boman, O. Parekh, and C. Phillips, “LDRD final report on massively-parallel linear programming: the parPCx system,” tech. rep., SAND2004-6440, Sandia National Laboratories, 2005.

[22] J. Kepner and J. Gilbert, *Graph Algorithms in the Language of Linear Algebra*. Society for Industrial and Applied Mathematics, 2011.

[23] B. Uçar and C. Aykanat, “Minimizing communication cost in fine-grain partitioning of sparse matrices,” in *Computer and Information Sciences-ISCIS 2003*, pp. 926–933, Springer, 2003.

[24] B. Uçar and C. Aykanat, “Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies,” *SIAM Journal on Scientific Computing*, vol. 25, no. 6, pp. 1837–1859, 2004.

[25] K. Akbudak and C. Aykanat, “Parallel Sparse Matrix-Matrix Multiplication Library,” Technical report BU-CE-1402, Computer Engineering Department, Bilkent University, Ankara, Turkey, 2014.

[26] "Message passing interface forum." http://www.mpi-forum.org/.

[27] P. Sulatycke and K. Ghose, "Caching-efficient multithreaded fast multiplication of sparse matrices," in *Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing 1998*, pp. 117–123, Mar 1998.

[28] C. Berge and E. Minieka, *Graphs and hypergraphs*, vol. 7. North-Holland publishing company Amsterdam, 1973.

[29] Ü. V. Çatalyürek and C. Aykanat, "Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication," *IEEE Transactions on Parallel Distributed Systems*, vol. 10, no. 7, pp. 673–693, 1999.

[30] T. Lengauer, *Combinatorial algorithms for integrated circuit layout.* Chichester, U.K.: Willey–Teubner, 1990.

[31] Basic Linear Algebra Subprograms Technical (BLAST) Forum, University of Tennessee, Knoxville, Tennessee, *BLAST Forum Standard*, 2001. http://www.netlib.org/blas/blast-forum/.

[32] "Sparse Basic Linear Algebra Subprograms (BLAS) Library." http://math.nist.gov/spblas/.

[33] F. G. Gustavson, "Two fast algorithms for sparse matrices : Multiplication and permuted transposition," *ACM Transactions on Mathematical Software (TOMS)*, vol. 4, no. 3, pp. 250–269, 1978.

[34] J. R. Gilbert, C. B. Moler, and R. Schreiber, "Sparse matrices in MATLAB : Design and implementation," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 1, pp. 333–356, 1992.

[35] K. Nusbaum, "Optimizing Tpetra's sparse matrix-matrix multiplication routine," tech. rep., SAND2011-6036, Sandia National Laboratories, 2011.

[36] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, *et al.*,

"An overview of the trilinos project," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 397–423, 2005.

[37] A. Buluç and J. R. Gilbert, "The Combinatorial BLAS: design, implementation, and applications," *International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 496–509, 2011.

[38] R. A. van de Geijn and J. Watts, "SUMMA: scalable universal matrix multiplication algorithm," *Concurrency - Practice and Experience*, vol. 9, no. 4, pp. 255–274, 1997.

[39] G. Ballard, A. Buluc, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo, "Communication optimal parallel multiplication of sparse random matrices," in *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA'13, (New York, NY, USA), pp. 222–231, ACM, 2013.

[40] J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger, "Communication-optimal parallel recursive rectangular matrix multiplication," in *Proceedings of 27th International Parallel Distributed Processing Symposium*, pp. 261–272, IEEE, May 2013.

[41] E. Solomonik, A. Bhatele, and J. Demmel, "Improving communication performance in dense linear algebra via topology aware collectives," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, (New York, NY, USA), pp. 77:1–77:11, ACM, 2011.

[42] L. E. Cannon, *A Cellular Computer to Implement the Kalman Filter Algorithm.* PhD thesis, Bozeman, MT, USA, 1969. AAI7010025.

[43] U. Borštnik, J. VandeVondele, V. Weber, and J. Hutter, "Sparse matrix multiplication: The distributed block-compressed sparse row library," *Parallel Computing*, vol. 40, no. 5, pp. 47–58, 2014.

[44] B. Hendrickson and T. G. Kolda, "Partitioning rectangular and structurally nonsymmetric sparse matrices for parallel computation," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2048–2072, 2000.

[45] Ü. V. Çatalyürek, C. Aykanat, and B. Uçar, "On two-dimensional sparse matrix partitioning: Models, methods, and a recipe," *SIAM Journal on Scientific Computing*, vol. 32, no. 2, pp. 656–683, 2010.

[46] Ü. V. Çatalyürek and C. Aykanat, *PaToH: A Multilevel Hypergraph Partitioning Tool, Version 3.0.* Computer Engineering Department, Bilkent University, Ankara, Turkey., 1999.

[47] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[48] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, p. 1, 2011.

[49] "Sparse Matrix-Matrix Multiplication Library v1.0." `https://sites.google.com/site/kadircs/`.

[50] "MPICH, high-performance and widely portable implementation of the Message Passing Interface (MPI) standard." `https://www.mpich.org/`.

[51] M. Adams and J. W. Demmel, "Parallel multigrid solver for 3D unstructured finite element problems," in *Supercomputing, ACM/IEEE 1999 Conference*, pp. 27–27, IEEE, 1999.

[52] W. L. Briggs, S. F. McCormick, *et al.*, *A multigrid tutorial, Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[53] R. Selvitopi, M. Ozdal, and C. Aykanat, "A novel method for scaling iterative solvers: Avoiding latency overhead of parallel sparse-matrix vector multiplies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 632–645, March 2015.

[54] "Matrix Market Text File Formats." `http://math.nist.gov/MatrixMarket/formats.html`.