Minimizing Communication Cost in Fine-Grain Partitioning of Sparse Matrices*

Bora Uçar and Cevdet Aykanat

Department of Computer Engineering, Bilkent University, 06800, Ankara, Turkey
{ubora,aykanat}@cs.bilkent.edu.tr

Abstract. We show a two-phase approach for minimizing various communication-cost metrics in fine-grain partitioning of sparse matrices for parallel processing. In the first phase, we obtain a partitioning with the existing tools on the matrix to determine computational loads of the processor. In the second phase, we try to minimize the communication-cost metrics. For this purpose, we develop communication-hypergraph and partitioning models. We experimentally evaluate the contributions on a PC cluster.

1 Introduction

Repeated matrix-vector multiplications (SpMxV) $y = Ax$ that involve the same large, sparse, structurally symmetric or nonsymmetric square or rectangular matrix are kernel operations in various iterative solvers. Efficient parallelization of these solvers requires matrix A to be partitioned among the processors in such a way that communication overhead is kept low while maintaining computational load balance. Because of possible dense vector operations, some of these methods require symmetric partitioning on the input and output vectors, i.e, conformal partitioning on x and y. However, quite a few of these methods allow unsymmetric partitionings, i.e., x and y may have different partitionings. The standard graph partitioning model has been widely used for one-dimensional (1D) partitioning of sparse matrices. Recently, Çatalyürek and Aykanat [3,4] and others [9,10,11] demonstrated some flaws and limitations of this model and developed alternatives. As noted in [10], most of the existing models consider minimizing the total communication volume. Depending on the machine architecture and the problem characteristics, communication overhead due to message latency may be a bottleneck as well [8]. Furthermore, maximum communication volume and latency handled by a single processor may also have crucial impacts on the parallel performance. In our previous work [17], we addressed these four communication-cost metrics in 1D partitioning of sparse matrices.

The literature on 2D matrix partitioning is rare. The 2D checkerboard partitioning approaches proposed in [12,15,16] are suitable for dense matrices or sparse matrices with structured nonzero patterns that are difficult to exploit. In

* This work was partially supported by The Scientific and Technical Research Council of Turkey under project EEEAG-199E013

A. Yazici and C. Şener (Eds.): ISCIS 2003, LNCS 2869, pp. 926–933, 2003.

particular, these approaches do not exploit sparsity to reduce the communication volume. Çatalyürek and Aykanat [3,6,7] proposed hypergraph models for 2D *coarse-grain* and *fine-grain* sparse matrix partitionings. In the *coarse-grain* model, a matrix is partitioned in a checkerboard-like style. In the *fine-grain* model, a matrix is partitioned on nonzero basis. The *fine-grain* model is reported to achieve better partitionings than the other models in terms of the total communication volume metric [6]. However, it also generates worse partitionings than the other models in terms of the total number of messages metric [6]. In this work, we adopt our two phase approach [17] to minimize the four communication-cost metrics in the fine-grain partitioning of sparse matrices. We show how to apply our *communication hypergraph model* to obtain unsymmetric partitioning and develop novel models to obtain symmetric partitioning.

2 Preliminaries

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as a set of vertices $\mathcal{V}$ and a set of nets. Every net n_i is a subset of vertices. Let d_j denote the number of nets containing a vertex v_j. Weights can be associated with vertices. $\Pi = \{\mathcal{V}_1, \cdots, \mathcal{V}_K\}$ is a K-way vertex partition of $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ if each part $\mathcal{V}_k$ is non-empty, parts are pairwise disjoint, and the union of parts gives $\mathcal{V}$. In Π, a net is said to *connect* a part if it has at least one vertex in that part. *Connectivity* λ_i of a net n_i denotes the number of parts connected by n_i. A net n_j is said to be cut if $\lambda_j > 1$ and uncut otherwise. The set of cut and uncut nets are called external and internal nets, respectively. In Π, the weight of a part is the sum of the weights of the vertices in that part. In the hypergraph partitioning problem, the objective is to minimize the cutsize:

$$cutsize(\Pi) = \sum_{n_i \in \mathcal{N}} (\lambda_i - 1). \tag{1}$$

This objective is referred to as the *connectivity−1 cutsize* metric [14]. The partitioning constraint is to satisfy a balancing constraint on the part weights, i.e.,

$$W_{max} \leq (1 + \epsilon) W_{avg} \tag{2}$$

where W_{max} is the weight of the part with the maximum weight and W_{avg} is the average part weight, and ϵ is an imbalance ratio. This problem is NP-hard [14].

A recent variant of the above problem is called *multi-constraint hypergraph partitioning* [3,7,13]. In this problem, a vertex has a vector of weights. The partitioning objective is the same as above, however, the partitioning constraint is to satisfy a set of balancing constraints, one for each type of the weights.

In the fine-grain hypergraph model of Çatalyürek and Aykanat [6], an $M \times N$ matrix A with Z nonzeros is represented as a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ with $|\mathcal{V}| = Z$ vertices and $|\mathcal{N}| = M + N$ nets for 2D partitioning. There exists one vertex v_{ij} for each nonzero a_{ij}. There exists one net m_i for each row i and one net n_j for each column j. Each row-net m_i and column-net n_j contain all vertices v_{i*} and v_{*j}, respectively. Each vertex v_{ij} corresponds to scalar multiplication $a_{ij}x_j$. Hence, the computational weight associated with a vertex is 1. Each row-net m_i represents the dependency of y_i on the scalar multiplications with a_{i*}'s. Each

column-net n_j represents the dependency of scalar multiplications with a_{*j}'s on x_j. With this model, the problem of 2D partitioning a matrix among K processors reduces to the K-way hypergraph partitioning problem. In this model, minimizing the cutsize while maintaining the balance on the part weights corresponds to minimizing the total communication volume and maintaining the balance on the computational loads of the processors. An external column-net represents the communication volume requirement on a x-vector entry. This communication occurs in *expand phase*, just before the scalar multiplications. An external row-net represents the communication volume requirement on a y-vector entry. This communication occurs in *fold phase*, just after the scalar multiplications. Çatalyürek and Aykanat [6] assign the responsibility of expanding x_i and folding y_i to the processor that holds a_{ii} to obtain symmetric partitioning. Note that for the unsymmetric partitioning case, one can assign x_i to any processor holding a nonzero in column i without any additional communication-volume overhead. A similar opportunity exists for y_i. In the symmetric partitioning case, however, x_i and y_i may be assigned to a processor holding nonzeros both in the row and column i. In this work, we try to exploit the freedom in assigning vector elements to address the four communication-cost metrics.

A 10×10 matrix with 37 nonzeros and its 4-way fine-grain partitioning is given in Fig. 1(a). In the figure, the partitioning is given by the processor numbers for each nonzero. The computational load balance is achieved by assigning 9, 10, 9, and 9 nonzeros to processors in order.

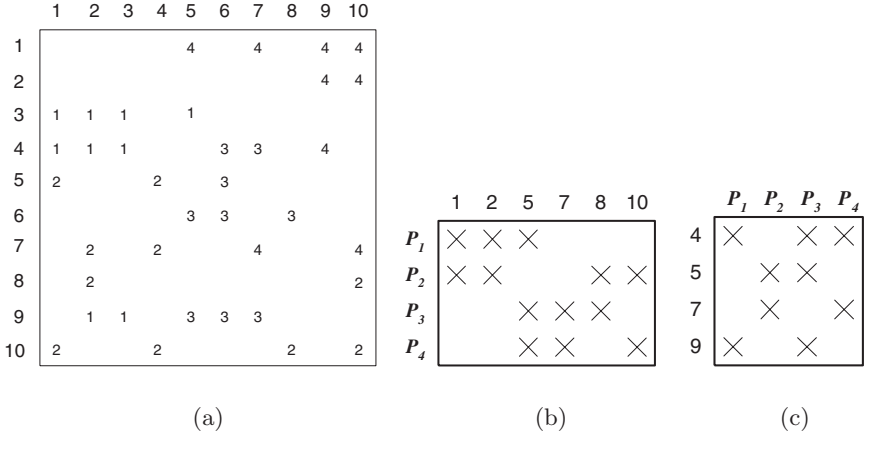


Fig. 1. (a) A 10×10 matrix and a 4-way partitioning, (b) communication matrix C_x, and (c) communication matrix C_y

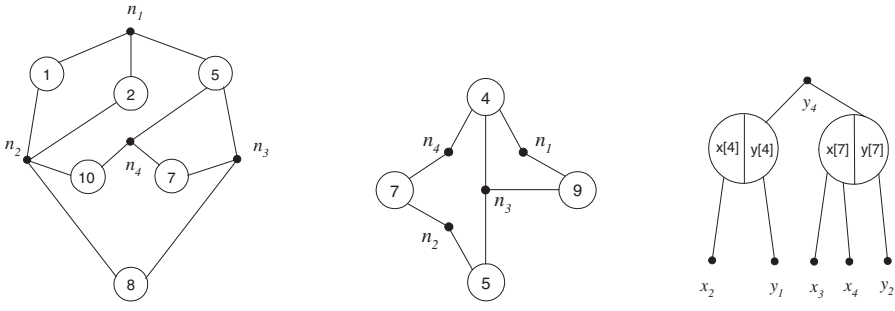
3 Minimizing the Communication Cost

Given a K-way fine-grain partitioning of a matrix, we identify two sets of rows and columns; *internal and coupling*. The internal rows or columns have nonzeros only in one part. The coupling rows or columns have nonzeros in more than one part. The set of x-vector entries that are associated with the coupling columns, referred to here as x_C, necessitate communication. Similarly, the set of y-vector entries that are associated with the coupling rows, referred to here as y_C, necessitate communication. Note that when symmetric partitioning requirement arises we add to x_C those x-vector entries whose corresponding entries are in y_C and vice versa. The proposed approach considers partitioning of these x_C and y_C vector entries to reduce the total message count and the maximum communication volume per processor. The other vector entries are needed by only one processor and should be assigned to the respective processors to avoid redundant communication. The approach may increase the total volume of communication of the given partitioning by at most $\max\{|x_C|, |y_C|\}$. We propose constructing two matrices C_x and C_y, referred to here as *communication matrices*, that summarize the communication on x- and y-vector entries, respectively. Matrix C_x has K rows and $|x_C|$ columns. For each row k we insert a nonzero in column j if processor P_k has nonzeros in column corresponding to $x_C[j]$ in the fine-grain partitioning. Hence, the rows of C_x correspond to processors in such a way that the nonzeros in the row k identify the subset of x_C-vector entries that are needed by processor P_k. Matrix C_y is constructed similarly. This time we put processors in columns and y_C entries in rows. Figure 1(b) and (c) show communication matrices C_x and C_y for the sample matrix given in (a).

3.1 Unsymmetric Partitioning Model

We use *row-net* and *column-net* hypergraph models for representing C_x and C_y, respectively. In the row-net hypergraph model, matrix C_x is represented as hypergraph $\mathcal{H}_x$ for columnwise partitioning. The vertex and net sets of $\mathcal{H}_x$ correspond to the columns and rows of matrix C_x, respectively. There exist one vertex v_j and one net n_i for each column j and row i, respectively. Net n_i contains the vertices corresponding to the columns which have a nonzero in row i. That is, $v_j \in n_i$ if $C_x[i, j] \neq 0$. In the column-net hypergraph model $\mathcal{H}_y$ of C_y, the vertex and net sets correspond to the rows and columns of the matrix C_y, respectively, with similar construction. Figure 2(a) and (b) show communication hypergraphs $\mathcal{H}_x$ and $\mathcal{H}_y$.

A K-way partition on the vertices of $\mathcal{H}_x$ induces a processor assignment for the expand operations. Similarly, a K-way partition on the vertices of $\mathcal{H}_y$ induces a processor assignment for the fold operations. In unsymmetric partitioning case, these two assignment can be found independently. In [17] we showed how to obtain such independent partitionings in order to minimize the four communication-cost metrics. The results of that work are immediately applicable to this case.



(a) (b) (c)

Fig. 2. Communication hypergraphs: (a) $\mathcal{H}_x$, (b) $\mathcal{H}_y$, and (c) a portion of $\mathcal{H}$

3.2 Symmetric Partitioning Model

When we require symmetric partitioning on vectors x and y, the partitionings on $\mathcal{H}_x$ and $\mathcal{H}_y$ can not be obtained independently. Therefore, we combine hypergraphs $\mathcal{H}_x$ and $\mathcal{H}_y$ into a single hypergraph $\mathcal{H}$ as follows. For each part P_k, we create two nets x_k and y_k. For each $x_C[i]$ and $y_C[i]$ pair, we create a single vertex v_i. For each net x_k, we insert v_i into its vertex list if processor P_k needs $x_C[i]$. For each y_k, we insert v_j into its vertex list if processor P_k contributes to $y_C[j]$. We show vertices v_4 and v_7 of $\mathcal{H}$ in Fig. 2(c). Since communication occurs in two distinct phases, vertices have two weights associated with them. The first weight of a vertex v_i is the communication volume requirement incurred by $x_C[i]$; hence we associate weight $d_i - 1$ with the vertex v_i. The second weight of a vertex v_i is the communication volume requirement incurred by $y_C[i]$; as in [17] we associate a unit weight of 1 with each v_i.

In a K-way partition of $\mathcal{H}$, an x_k-type net connecting λ_{xk} parts necessitates $\lambda_{xk} - 1$ messages to be sent to processor P_k during the expand phase. The sum of these values thus represents the total number of messages sent during the expand phase. Similarly, a y_k-type net connecting λ_{yk} parts necessitates $\lambda_{yk} - 1$ messages to be sent by P_k during the fold phase. The sum of these values represents the total number of messages sent during the fold phase. The sum of the *connectivity* $- 1$ values for all nets thus represents the total number of messages. Therefore, by minimizing the objective function in Eq. 1, partitioning $\mathcal{H}$ minimizes the total number of messages. The vertices in part $\mathcal{V}_k$ represent the x-vector entries to be expanded and the respective y-vector entries to be folded by processor P_k. The load of the expand operations are exactly represented by the first components of the vertex weights if for each $v_i \in \mathcal{V}_k$ we have $v_i \in x_k$. If, however, $v_i \notin x_k$, the weight of a vertex for the expand phase will be one less than the required. We hope these shortages to occur, in some extent, for every processor to cancel the diverse effects on the communication-volume load balance. The weighting scheme for the fold operations is adopted with the rationale that every $y_C[i]$ assigned to a processor P_k will relieve P_k from sending

a unit-volume message. If the net sizes are close to each other then this scheme will prove to be a reasonable one. As a result, balancing part sizes for the two set of weights, e.g., satisfying Eq. 2, will relate to balancing the communication-volume loads of processors in the expand and the fold phases, separately. For the minimization of the maximum number of messages per processor metric we do not spend explicit effort. We merely rely on its loose correlation with the total number of messages metric.

In the above discussion, each net is associated with a certain part and hence a processor. This association is not defined in the standard hypergraph partitioning problem. We can enforce this association by adding K special vertices, one for each processor P_k, and inserting those vertices to the nets x_k and y_k. Fixing those special vertices to the respective parts and using *partitioning with fixed vertices* feature of hypergraph partitioning tools [1,5] we can obtain the specified partitioning on $\mathcal{H}$. However, existing tools do not handle fixed vertices within multi-constraint framework. Therefore, instead of obtaining balance on the communication-volume loads of processors in the expand and the fold phases separately, we add up the weights of vertices and try to obtain balance on aggregate communication-volume loads of processors.

4 Experiments

We have conducted experiments on the matrices given in Table 1. In the table, N and NNZ show, respectively, the dimension of the matrix and the number of nonzeros. The `Srl.Time` column lists the timings for serial SpMxV operations in milliseconds. We used PaToH [5] library to obtain 24-way fine-grain partitionings on the test matrices. In all partitioning instances, the computational-load imbalance were below 7 percent. `Part.Mthd` give the partitioning method applied: `PTH` refers to the fine-grain partitioning of Çatalyürek and Aykanat [6], `CHy` refers to partitioning the communication hypergraphs with fixed vertices and aggregate vertex weights. For these two methods, we give timings under column `Part.Time`, in seconds. For each partitioning method, we dissect the communication requirements into the expand and fold phases. For each phase, we give the total communication volume, the maximum communication volume handled by a single processor, the total number of messages, and the maximum number of messages per processor under columns `tV`, `xV`, `tM`, and `xM`, respectively. In order to see whether the improvements achieved by method `CHy` in the given performance metrics hold in practice, we also give timings, the best among 20 runs, for parallel SpMxV operations, in milliseconds, under column `Prll.Time`. All timings are obtained on machines equipped with 400 MHz Intel Pentium II processor and 64 MB RAM running Linux kernel 2.4.14 and Debian GNU/Linux 3.0 distribution. The parallel SpMxV routines are implemented using LAM/MPI 6.5.6 [2]. To compare our method against `PTH`, we opted for obtaining symmetric partitioning. For each matrix, we run `PTH` 20 times starting from different random seeds and selected the partition which gives the minimum in total communication volume metric. Then, we constructed the communication hypergraph with respect to `PTH`'s best partitioning and run `CHy` 20 times, again starting from

different random seeds, and selected the partition which gives the minimum in total number of messages metric. Timings for these partitioning methods are for a single run. In all cases, running CHy adds at most half of the time required by PTH to the framework of fine-grain partitioning.

In all of the partitioning instances, CHy reduces the total number of messages to somewhere between 0.47 (*fom12*) and 0.74 (*CO9*) of PTH. In all partitioning instances, CHy increases the total communication volume to somewhere between 1.32 (*creb*) and 1.86 (*pds20*) of PTH. This is expected, because a vertex v_i may be assigned to a part $\mathcal{V}_k$ while P_k does not need any of $x_C[i]$ or $y_C[i]$. However, reductions in parallel running times are seen for all matrices except *lpl1*. The highest speed-up achieved by PTH and CHy is 5.96 and 6.38, respectively, on *fxm3*.

Table 1. Communication patterns and running times for 24-way parallel SpMxV

Matrix	Size		Part.		Expand Phase				Fold Phase				Prll.	Srl.
	N	NNZ	Mthd	Time	tV	xV	tM	xM	tV	xV	tM	xM	Time	Time
CO9	10789	249205	PTH	11.43	2477	524	290	21	4889	473	358	22	4.55	12.54
			CHy	0.66	5121	318	223	22	7367	714	259	18	4.05	
creb	9648	398806	PTH	26.71	9344	750	490	23	12660	871	504	23	6.64	19.30
			CHy	2.51	13047	715	313	23	16157	1068	341	20	5.92	
ex3s1	17443	679857	PTH	48.88	7964	602	312	22	26434	1762	356	20	8.39	33.52
			CHy	13.58	19537	1128	195	23	36347	2252	270	16	7.91	
fom12	24284	329068	PTH	22.07	7409	559	228	23	21208	1143	228	13	5.03	19.86
			CHy	13.76	16713	983	96	10	28151	1541	119	8	4.03	
fxm3	41340	765526	PTH	37.73	1843	279	212	23	2662	282	236	17	6.39	38.13
			CHy	0.29	3299	215	142	16	4027	456	156	15	5.97	
lpl1	39951	541217	PTH	27.04	7646	1062	226	20	13752	961	253	17	5.73	29.81
			CHy	5.09	15079	892	166	22	20582	1507	186	12	5.83	
mod2	34774	604910	PTH	32.83	5015	845	267	23	9421	1135	278	22	6.92	30.81
			CHy	2.18	10523	656	181	23	14142	1517	198	17	5.92	
pds20	33874	320196	PTH	18.65	5373	557	299	23	13548	956	317	19	5.23	17.82
			CHy	6.09	14066	794	177	18	21302	1436	201	13	4.95	
pltex	26894	269736	PTH	14.29	1883	172	167	16	7065	508	273	20	4.27	14.64
			CHy	1.11	4533	311	89	16	8828	782	139	10	3.63	
world	34506	582064	PTH	30.84	4934	794	300	23	9710	1295	316	23	7.35	29.63
			CHy	2.50	10679	656	181	23	14854	1745	205	18	6.05	

5 Conclusion and Future Work

We showed a two-phase approach that encapsulates various communication-cost metrics in 2D partitioning of sparse matrices. We developed models to obtain symmetric and unsymmetric partitioning on input and output vectors. We tested performance of the proposed models on practical implementations. In this work, a sophisticated hypergraph partitioning tool that can handle fixed vertices in the context of multi-constraint partitioning was needed. Since the existing tools do not handle this type of partitioning, we are considering to develop such a method.

References

1. Charles J. Alpert, Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Hypergraph partitioning with fixed vertices. *IEEE Transactions on Computer-Aided Design*, 19(2):267–272, 2000.
2. Greg Burns, Raja Daoud, and James Vaigl. LAM: an open cluster environment for MPI. In John W. Ross, editor, *Proceedings of Supercomputing Symposium '94*, pages 379–386. University of Toronto, 1994.
3. Ü. V. Çatalyürek. *Hypergraph Models for Sparse Matrix Partitioning and Reordering.* PhD thesis, Bilkent Univ., Computer Eng. and Information Sci., Nov 1999.
4. Ü. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, 1999.
5. Ü. V. Çatalyürek and C. Aykanat. PaToH: A multilevel hypergraph partitioning tool, ver. 3.0. Tech. Rep. BU-CE-9915, Computer Eng. Dept., Bilkent Univ., 1999.
6. Ü. V. Çatalyürek and Cevdet Aykanat. A fine-grain hypergraph model for 2d decomposition of sparse matrices. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS), 8th International Workshop on Solving Irregularly structured Problems in Parallel (Irregular 2001)*, April 2001.
7. Ü. V. Çatalyürek and Cevdet Aykanat. A hypergraph-partitioning approach for coarse-grain decomposition. In *Proceedings of Scientific Computing 2001 (SC2001)*, pages 10–16, Denver, Colorado, November 2001.
8. J. J. Dongarra and T. H. Dunigan. Message-passing performance of various computers. *Concurrency—Practice and Experience*, 9(10):915–926, 1997.
9. B. Hendrickson. Graph partitioning and parallel solvers: has the emperor no clothes? *Lecture Notes in Computer Science*, 1457:218–225, 1998.
10. B. Hendrickson and T. G. Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, 26:1519–1534, 2000.
11. B. Hendrickson and T. G. Kolda. Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing. *SIAM Journal on Scientific Computing*, 21(6):2048–2072, 2000.
12. B. Hendrickson, R. Leland, and S. Plimpton. An efficient parallel algorithm for matrix-vector multiplication. *Int. J. High Speed Comput.*, 7(1):73–88, 1995.
13. G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint hypergraph partitioning. Tech. Rep. 99-034, University of Minnesota, Dept. Computer Science/Army HPC Research Center, Minneapolis, MN 55455, November 1998.
14. T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley–Teubner, Chichester, U.K., 1990.
15. J. G. Lewis, D. G. Payne, and R. A. van de Geijn. Matrix-vector multiplication and conjugate gradient algorithms on distributed memory computers. In *Proceedings of the Scalable High Performance Computing Conference*, 1994.
16. A. T. Ogielski and W. Aiello. Sparse matrix computations on parallel processor arrays. *SIAM Journal on Numerical Analysis*, 1993.
17. Bora Uçar and Cevdet Aykanat. Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies. *Siam J. Sci. Comput.*, Submitted to, 2002.