SAND2007-0165C

A Bipartite Graph Model and Algorithm for Fine-Grain Parallel Sparse Matrix Distribution

(Brief Announcement, SPAA 2007)

Erik G. Boman
Discrete Algorithms and Math
Sandia National Labs[*]
Albuquerque, NM 87185-1318
egboman@sandia.gov

Abstract

We consider how to distribute sparse matrices among processors to reduce communication cost in sparse matrix computations, in particular, sparse matrix-vector multiplication. We consider 2d distributions, where the distribution (partitioning) is not constrained to just rows or columns. The fine-grain model by Catalyurek and Aykanat [2] is a 2d distribution where nonzeros can be assigned to processors in an arbitrary, completely general way. The authors use a hypergraph model and suggest an algorithm based on hypergraph partitioning.

We propose a simpler bipartite graph model that also accurately describes communication volume, and show that it is the dual of the fine-grain hypergraph. Our model naturally leads to a new algorithm for fine-grain matrix distribution based on vertex separators, which could potentially be more efficient in practice than the hypergraph approach. Preliminary numerical results on matrices from different application areas indicate our new approach is competitive, and significantly better for some matrices from information retrieval.

Keywords: Load balancing, parallel sparse matrix computations, graph partitioning,hypergraph partitioning, information retrieval.

[*]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the DOE's National Nuclear Security Administration under contract number DE-AC-94AL85000.

1 Introduction

Sparse matrix-vector multiplication is a common kernel in many computations, e.g., iterative solvers for linear systems of equations and PageRank computation (power method). An important combinatorial problem is how to distribute the matrix and the vectors among processors such as to minimize the communication cost. We assume distributed-memory computers, where communication is expensive. Our work also applies to the more general computation $y = F(x)$, where x is the input vector, y is the output vector, and F is a reduction operator.

Sparse matrix-vector multiplication $y = Ax$ is usually parallelized such that the processor that owns element a_{ij} computes the contribution $a_{ij}x_j$. This is a local operation if x_j, y_i and a_{ij} all reside on the same processor; otherwise communication is required. In general, the following four steps are performed [1, 4]:

1. **Expand:** Send entries x_j to processors with a nonzero a_{ij} for some i.
2. **Local multiply:** $y_i += a_{ij}x_j$
3. **Fold:** Send partial y values to relevant processors.
4. **Sum:** Sum up the partial y values.

The partitioning problem we address is: Given a sparse matrix A and an integer $k > 1$, compute a parallel distribution over k processors of the nonzeros of A and also for the input and output vectors such that all processors have approximately equally many nonzeros (load balance), and the communication volume in sparse matrix-vector multiply is minimized.

Typically, matrices are partitioned in a 1d fashion, either by rows or by columns. This partitioning problem has been modeled both as graph partitioning in the symmetric case, and as bipartite graph partitioning [3] in the nonsymmetric and rectangular case. Hypergraph partitioning [1] has been shown to accurately model communication volume, and is therefore often preferred today. A hypergraph is a generalization of a graph. An hyperedge is a subset of vertices; thus hyperedges connect one or more vertices, while graph edges always connect two vertices.

Recently, several 2d decompositions have been proposed [2, 4]. The idea is to reduce the communication volume further by giving up the simplicity of the 1d structure. We focus on the fine-grain distribution [2], because it is the most general. We show that a bipartite graph model also accurately describes communication in fine-grain distribution. This leads to a new graph-based algorithm, which may potentially work better in practice.

2 Fine-Grain Models

2.1 The Fine-Grain Hypergraph Model

In fine-grain matrix distribution, each nonzero can be assigned independently. The hypergraph model [2] works as follows: Let each nonzero correspond to a vertex, and let each row and each column represent a hyperedge. A matrix A with m rows, n colums, and z nonzeros gives a hypergraph with z vertices and $m + n$ hyperedges. This hypergraph can then be partitioned using standard partitioning algorithms and software, but this does not take into account the special structure.

2.2 The New Fine-Grain Bipartite Graph Model

We start with the bipartite graph $G = (R, C, E)$ of the matrix, where R and C correspond to rows and columns, respectively. In 1d distribution, we partition either the rows (R) or the columns (C). For fine-grain distribution, we partition both R, C, and E into k sets. Note that we explicitly partition the edges E, which distinguishes our approach from previous work. To balance computation and memory, our primary objective is to balance the edges (matrix nonzeros). Vertex balance is a secondary objective.

We wish to analyze the communication requirements, so suppose that the vertices and edges have already been partitioned. Cut edges, that is, edges with endpoints in different partitions, may incur communication but not necessarily. The edge cut approximates but does not represent communication volume [1, 3]. Instead we assign communication cost to vertices. Clearly, a vertex where all incident edges belong to the same partition incurs no communication because all operations are local. Conversely, a vertex with at least one incident edge in a different partition does incur communication because that is a vector element x_i or y_j residing on a different processor than a_{ij}. We therefore have:

Fact 2.1 *The communication volume in matrix-vector multiply is equal to the number of vertices that have at least one incident edge in a different partition (boundary vertices).*

A crucial point is that by assigning edges to processors independently of the vertices, we can reduce the number of boundary vertices compared to the traditional 1d distribution, where only vertices are partitioned and the edge assignments are induced from the vertices.

Given a matrix with z nonzeros, the bipartite graph has z edges while the fine-grain hypergraph has $2z$ pins. Thus, algorithm based on the bipartite graph model may use less memory.

2.3 Equivalence between the Two Models

Since our bipartite graph model is exact, it must be equivalent to the fine-grain hypergraph model in some way. In fact, there is a simple and elegant relation between the two models.

Let the dual of a hypergraph $H = (V, E)$ be another hypergraph $H' = (V', E')$, where a vertex in V corresponds to an edge in E', and an edge in E corresponds to a vertex in V'. Let H be the hypergraph for the fine-grain model.

Theorem 2.2 *The dual hypergraph H' is the bipartite graph G.*

Proof: H has a vertex for each nonzero, G has an edge for ecah nonzero. H contains one hyperedge for each row (r_i) and column (c_j), G has one vertex for each row and column. There is an edge $(i, j) \in E_G$ iff hyperedges r_i and c_j intersect, that is, $r_i \cap c_j \neq \emptyset$.

Hence partitioning the vertices of H corresponds to partitioning the edges in G. In our bipartite graph model, we also explicitly partition the vertices in G, while the hyperedges in H are only partitioned implicitly in the hypergraph algorithm.

3 A Vertex Separator Algorithm

An edge separator in the fine-grain hypergraph model corresponds to a vertex separator in the bipartite graph. Thus, we can derive a fine-grain decomposition from a vertex separator for the bipartite graph.

For simplicity, we consider only bisection. The general (k-way) problem can be solved using recursive bisection. First compute a small balanced vertex separator S for the bipartite graph using any vertex separator algorithm. This partitions the vertices into (V_0, V_1, S). Assign V_0 and incident edges to processor 0, and similarly for V_1. We have a choice for the vertices in S; these should be assigned to the same partition but we can choose either one. We can use this flexibility to achieve a secondary goal, e.g. load balance in the vectors (vertices). Since our primary goal is load balance in the matrix nonzeros, we seek to balance the edges in the bipartite graph. This can be done by weighting each vertex by its degree.

4 Experiments

We compare the communication volume for parallel sparse matrix-vector for a set of sparse matrices from different application areas (information retrieval, linear programming, circuit simulation, chemical engineering) that were used in [4]. The first five matrices are rectangular, and the next five are square but structurally nonsymmetric.

We compare our fine-grain bipartite graph separator algorithm versus the fine grain hypergraph algorithm, and include 1d row and column partitions using hypergraph partitioning as a baseline. Though NP-hard problems, several good codes for graph and hypergraph partitioning are available, all based on the multilevel method. We used PaToH 3.0 as our hypergraph partitioner, and Chaco 2.0 to find vertex separators. In both cases the imbalance tolerance was 0.05. Only bisection is used as Chaco is limited to that. (Metis computed poorly balanced separators and is not shown.)

We see from Table 4 that there is no consistent winner. The hypergraph (h.g.) fine-grain method (with PaToH) obtained significantfly lower volume in four out of ten cases (including one case where

Name	1d row	1d col	2d h.g.	2d b.g.
dfl001	1427	630	540	879
cre_b	6551	477	468	484
tbdmatlab	4591	6319	4307	3381
nug30	167504	27828	33198	–
tbdlinux	14731	15795	10198	7861
west0381	51	52	37	37
gemat1	1859	37	33	33
memplus	2183	2222	148	190
onetone2	202	280	212	216
lhr34	99	64	64	294

Table 1: Communication volume ($k = 2$

.

Name	rows	cols	2d h.g.	2d b.g.
dfl001	6071	12230	0.20	0.34
cre_b	9648	77137	3.46	6.45
tbdmatlab	19859	5979	9.89	3.62
nug30	52260	379350	12.8	–
tbdlinux	112757	20167	79.6	12.2
west0381	381	381	0.01	0.01
gemat1	4929	4929	0.14	0.24
memplus	17758	17758	0.67	0.27
onetone2	36057	36057	0.85	0.34
lhr34	35152	35152	3.71	0.30

Table 2: Matrix sizes and partitioning times (sec.).

Chaco failed), while the bipartite (b.p.) approach (with Chaco) was clearly better in two cases. Interestingly, these two matrices were both term-by-document matrices from information retrieval.

The partitioning times are also difficult to interpret, with each fine-grain method faster about half of the time. We remark that in some cases, Chaco stopped the coarsening early due to insufficient progress, causing a time-consuming coarse partition phase on a large graph.

5 Conclusions

We presented a new interpretation of the bipartite graph that is useful for sparse matrix partitioning. Our model is the dual of the hypergraph fine-grain model. Our vertex separator algorithm is similar in time and quality to the hypergraph algorithm, but the results can vary substantially among problems. An advantage of our bipartite approach is that we explicitly obtain distributions for both the matrix and the vectors, so there is no need for a subsequent "vector partitioning" phase. In future work we will study the performance for larger k using recursive bisection (nested dissection), and also use parallel partitioners like Zoltan. We do not know how sensitive the results are to different implementations of similar multilevel partitioning methods. An open question is to identify matrix classes that are consistently solved better by one approach.

Acknowledgments

We thank Bruce Hendrickson, John Gilbert, and Umit Catalyurek for related discussions, and Rob Bisseling for providing the test matrices.

References

[1] Ü. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Dist. Systems*, 10(7):673–693, 1999.

[2] Ü. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2d decomposition of sparse matrices. In *Proc. IPDPS 8th Int'l Workshop on Solving Irregularly Structured Problems in Parallel (Irregular 2001)*, April 2001.

[3] B. Hendrickson and T. G. Kolda. Partitioning rectangular and structurally nonsymmetric sparse matrices for parallel computation. *SIAM Journal on Scientific Computing*, 21(6):2048–2072, 2000.

[4] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Review*, 47(1):67–95, 2005.