Hypergraph Decomposition of Sparse Matrices for Vector-Matrix Multiplication

S. Dadizadeh

{*sara88*}*@cs.ubc.ca*
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada

Abstract

Keywords: parallel computing, hypergraph partitioning, sparse matrix multiplication

1 Introduction

Sparse matrix multiplication is one of the most heavily used operations in scientific computing, especially in applications which solve partial differential equations. It is imperative to have scalable matrix operations which perform these computations as efficiently as possible. To achieve the best speed-up possible, we must minimize communication overhead since there are few computations relative to the size of the matrix. Our approach is to model the matrix as a hypergraph and use the MLFM technique to split the hypergraph into partitions, where we assign a partition to each processor. The problem of finding an optimal partition is NP-hard, but because partitioning is critical in several applications, heursitic algorithms with near-linear runtime were developed [1].

2 Strategy

We first need to represent the matrix A as a hypergraph; the model is given in section 2.1. We then use the multilevel Fiduccia-Mattheyses (MLFM) framework to perform the hypergraph partitioning, the algorithm is described in 2.2. In section 3 we delve into implementation details. In 3.1 we give an overview of hMetis, a library that implements the MLFM partitioning algorithm. Finally in 3.2 we describe the implementation details of the computation and communication, done in MPI.

2.1 Hypergraph Partitioning Problem

Our goal is a sparse-matrix vector product of the form $\vec{y} = A\vec{x}$, where $\vec{y}$ and $\vec{x}$ are dense vectors, and A is a sparse matrix. Matrix A is represented as the hypergraph $H_R(V_R, N_C)$. The vertex and net sets V_R and N_C correspond to the rows and columns of A, respectively. The vertices in a net n_j are called its pins and denoted as $pins[n_j]$. There exist one vertex v_i and one net n_j for each row i and column j, respectively. Net n_j contains the vertices corresponding to the columns which have a nonzero entry on row j ($v_i \in n_j$ if and only if $a_{ij} \neq 0$). Given this representation, we build a hypergraph H, where the k-way partitioning of H assigns vertices of H to k disjoint nonempty partitions. In a partition Π of H, a net that has at least one pin is said to connect that part. The connectivity set Λ_j of a net n_j denotes the number of parts connected by n_j. A net n_j is cut if it connects more than one part ($\Lambda_j > 1$), and uncut otherwise ($\Lambda_j = 1$). The hypergraph partitioning problem is the task of dividing a hypergraph into two or more parts such that the cutsize is minimized, since cuts represent interprocessor communication.

This decomposition scheme, where we represent rows of A as vertices, and columns of A as hyperedges, is called the column-net model for rowwise decomposition. The nets of H_R represent the dependency relations of the atomic tasks on the $\vec{x}$-vector components. Each net n_j incurs the computation $y_i = y_i + a_{ij}x_j$ for each vertex $v_i \in n_j$. This means that each net n_j denotes the set of atomic tasks (vertices) that need x_j. Figure 1 illustrates this dependency relation [2].

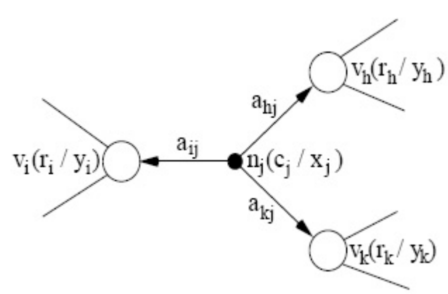

Figure 1: Dependency relation view of the column-net model

2.2 Hypergraph Partitioning Framework

The choice of partitioning algorithm depends on the number of movable objects (i.e. the number of vertices). The multilevel Fiduccia-Mattheyses (MLFM) framework scales well with large numbers of vertices, and creates the best known partitioning results. It consists of three components: clustering, top-level partitioning, and refinement or "uncoarsening". During the clustering stage, vertices are combined into clusters based on connectivity, leading to a smaller, clustered hypergraph. Then the smallest (i.e. top-level) hypergraph is partitioned with a fast initial solution generator, and iteratively improved, using the Fidduccia-Mattheyses (FM) partitioning framework (which is a single-level version of the algorithm we use here). During refinement, solutions are projected from one level to the next and iteratively improved, once again by the FM algorithm.

3 Details of the Implementation

First we build a hypergraph representation of the matrix A in the format hMetis specifies. From hMetis we obtain a partitioning, which we use to communicate data to the processors. Once the processors have obtained the necessary information, they perform the computation, and at the end we aggregate the results to obtain $\vec{y}$.

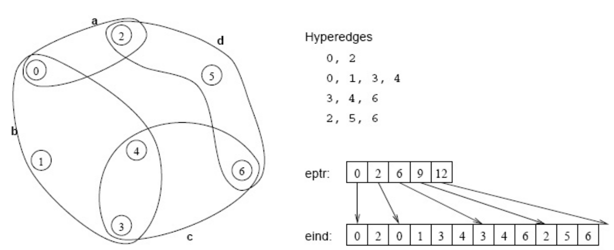

Figure 2: The `eptr` and `eind` arrays which describe the hyperedges of the hypergraph.

3.1 Overview of hMetis

hMetis is a hypergraph partitioning package that produces high quality partitions in reasonable time. The main function we are concerned with is `HMETIS_PartKway (int nvtxs, int nhedges, int *vwgts, int *eptr, int *eind, int *hewgts, int nparts, int ubfactor, int *options, int *part, int *edgecut)`. `nvtxs` and `nhedges` are the number of vertices and hyperedges, respectively. The parameters `eptr` and `eind` are two arrays which describe the hyperedges in the graph. `eptr` is of size `nhedges`+1, and is used to index into the second array `eind` that stores the actual hyperedges. Each hyperedge is stored as a sequence of the vertices that it spans, in consecutive locations in `eind`. `nparts` is the number of desired partitions. Figure 2 illustrates a simple hypergraph with its corresponding `eptr` and `eind` arrays. `part` returns the computed partition as an array of `nvtxs` size. Specifically, `part[i]` contains the partition number in which vertex i belongs to [3].

3.2 Communication and Computation

In the column-net model, a partition Λ of H_R with $v_i \in P_k$ assigns row i, y_i and x_i to processor P_k for rowwise decomposition. A cut net n_j indicates that the processor who contains the corresponding vertex v_j should send its local x_j to all the processors in the connectivity set Λ_j of net n_j except itself. For example, in Figure 3, P_1 should send x_5 to both P_2 and P_3. Each net n_j incurs the computation $y_i = y_i + a_{ij}x_j$ for each vertex $v_i \in n_j$. Each processor P_k is responsible for computing y_i for the vertices (rows) that it has been assigned.

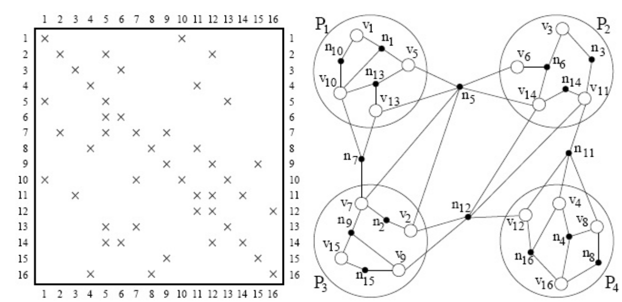

Figure 3: A 16x16 sparse matrix A and its column-net representation in a 4-way partitioning

4 Performance

5 Conclusions

References

[1] D.A. Papa and I.L. Markov. *Hypergraph Partitioning and Clustering*. CRC Press, 2007.

[2] Umit V. Catalyurek and Cevdet Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. on Parallel and Distributed Computing*, 10:673–693.

[3] Kumar Karypis. hmetis: A hypergraph partitioning package. 1998.