SAND2015-0990C

Hypergraph Partitioning for Parallel Sparse Matrix-Matrix Multiplication

Grey Ballard*
Sandia National Laboratories
gmballa@sandia.gov

Alex Druinsky
Lawrence Berkeley National Laboratory
adruinsky@lbl.gov

Nicholas Knight
University of California, Berkeley
knight@cs.berkeley.edu

Oded Schwartz
Hebrew University, Jerusalem, Israel
odedsc@cs.huji.ac.il

ABSTRACT

The performance of parallel algorithms for sparse matrix-matrix multiplication is typically determined by the amount of interprocessor communication performed, which in turn depends on the nonzero structure of the input matrices. In this paper, we characterize the communication cost of a sparse matrix-matrix multiplication algorithm in terms of the size of a cut of an associated hypergraph that encodes the computation for a given input nonzero structure. Obtaining an optimal algorithm corresponds to solving a hypergraph partitioning problem. Our hypergraph model generalizes several existing models for sparse matrix-vector multiplication, and we can leverage hypergraph partitioners developed for that computation to improve application-specific algorithms for multiplying sparse matrices.

1. INTRODUCTION

Sparse matrix-matrix multiplication (SpGEMM) is a fundamental computation in applications ranging from linear solvers to graph algorithms and data analysis (see [3] and references therein). SpGEMM algorithms are typically communication bound, spending much more of their time moving data than performing additions and multiplications. Furthermore, the computation is usually irregular and depends on the nonzero structures of the input matrices.

Hypergraphs are used to model the related computation of parallel sparse matrix-(dense) vector multiplication (SpMV), and software packages have been developed to improve performance for this computation (see, e.g., [4]). Hypergraphs can also be used to model vertical communication on a sequential machine for a general set of computations, where vertices correspond to computations and nets correspond to data [6]. More recently, hypergraph models have been devised to optimize the outer-product algorithm for SpGEMM [1]. In this work, we present a hypergraph model that generalizes the "fine-grain" SpMV model [4] and the outer-product model for SpGEMM [1]; we differ from [6] in that we consider interprocessor rather than vertical communication.

The main contributions of this work are (1) showing that SpGEMM can be modeled by a hypergraph so that its communication cost corresponds to the size of a cut induced by a partition of the vertices and (2) demonstrating that hypergraph partitioners can be used to determine efficient SpGEMM algorithms for an algebraic multigrid application.

*This research was supported in part by an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering, sponsored by Sandia Corporation (a wholly owned subsidiary of Lockheed Martin Corporation) as Operator of Sandia National Laboratories under its U.S. Department of Energy Contract No. DE-AC04-94AL85000.

2. THEORETICAL MODEL

Let $\mathbf{A}$ and $\mathbf{B}$ be I-by-K and K-by-J matrices over X, a set closed under two binary operations denoted by addition and multiplication. SpGEMM is $(\mathbf{A}, \mathbf{B}) \mapsto \mathbf{C}$, where $\mathbf{C}$ is an I-by-J matrix over X defined entrywise by $c_{ij} = \sum_{k \in [K]} a_{ik} b_{kj}$ ($[K]$ denotes the set $\{1, \ldots, K\}$). We let $S_{\mathbf{A}} \subseteq [I] \times [K]$, $S_{\mathbf{B}} \subseteq [K] \times [J]$, and $S_{\mathbf{C}} \subseteq [I] \times [J]$ denote the nonzero structures of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. Here we consider algorithms that evaluate and sum all *nontrivial* multiplications $a_{ik} b_{kj}$, where a_{ik} and $b_{kj} \neq 0$, and thus depend only on $S_{\mathbf{A}}$ and $S_{\mathbf{B}}$. To simplify, we ignore numerical cancellation, so $S_{\mathbf{A}}$ and $S_{\mathbf{B}}$ induce $S_{\mathbf{C}}$.

Definition 1. *Given input matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, let the* SpGEMM hypergraph *be* $\mathcal{H}(\mathbf{A}, \mathbf{B}) = (\mathcal{V}, \mathcal{N})$*, with vertices*

$$\mathcal{V} = \{v_{ikj} : (i,k) \in S_{\mathbf{A}} \wedge (k,j) \in S_{\mathbf{B}}\}$$

and nets $\mathcal{N} = \mathcal{N}^{\mathbf{A}} \cup \mathcal{N}^{\mathbf{B}} \cup \mathcal{N}^{\mathbf{C}}$*, where*

$$\begin{aligned}
\mathcal{N}^{\mathbf{A}} &= \{\, n^{\mathbf{A}}_{ik} : (i,k) \in S_{\mathbf{A}} \,\} \;\; \textit{with} \;\; n^{\mathbf{A}}_{ik} = \{\, v_{ikj} : j \in [J] \,\}, \\
\mathcal{N}^{\mathbf{B}} &= \{\, n^{\mathbf{B}}_{kj} : (k,j) \in S_{\mathbf{B}} \,\} \;\; \textit{with} \;\; n^{\mathbf{B}}_{kj} = \{\, v_{ikj} : i \in [I] \,\}, \\
\mathcal{N}^{\mathbf{C}} &= \{\, n^{\mathbf{C}}_{ij} : (i,j) \in S_{\mathbf{C}} \,\} \;\; \textit{with} \;\; n^{\mathbf{C}}_{ij} = \{\, v_{ikj} : k \in [K] \,\}.
\end{aligned}$$

In $\mathcal{H}(\mathbf{A}, \mathbf{B})$, each vertex corresponds to a nontrivial multiplication, and each net in $\mathcal{N}^{\mathbf{A}}$, $\mathcal{N}^{\mathbf{B}}$, and $\mathcal{N}^{\mathbf{C}}$ corresponds to a nonzero of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and contains all nontrivial multiplications in which that nonzero participates.

We now consider performing SpGEMM with input matrices $\mathbf{A}$ and $\mathbf{B}$ on a parallel machine with p processors

with disjoint memories. A *parallelization* is a p-way partition of $\mathcal{V}$ (assigning multiplications to processors) and a *data distribution* is a triple of p-way partitions of $S_{\mathbf{A}}$, $S_{\mathbf{B}}$, and $S_{\mathbf{C}}$ (assigning nonzeros to processors). The communication proceeds in two phases: the *expand* phase, where the processors exchange nonzero entries of $\mathbf{A}$ and $\mathbf{B}$ (initially distributed according to the partitions of $S_{\mathbf{A}}$ and $S_{\mathbf{B}}$) in order to perform their multiplications (assigned according to the partition of $\mathcal{V}$), and the *fold* phase, where the processors communicate to reduce partial sums for nonzero entries of $\mathbf{C}$ (finally distributed according to the partition of $S_{\mathbf{C}}$).

DEFINITION 2. *Given a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_p\}$ of $\mathcal{V}$, for each $i \in [p]$, we define Q_i, the* ith cut *of $\mathcal{H}$, to be the subset of $\mathcal{N}$ having nonempty intersections with both $\mathcal{V}_i$ and $\mathcal{V} \setminus \mathcal{V}_i$.*

LEMMA 1. *Given an SpGEMM computation with parallelization $\{\mathcal{V}_1, \ldots, \mathcal{V}_p\}$ and any data distribution, the number of words each processor i sends or receives is at least $|Q_i|$, and the critical-path cost is at least $\max_{i \in [p]} |Q_i|$.*

PROOF. For each processor i, for each net in Q_i, processor i must either receive or send the corresponding nonzero, since at most one processor owns each nonzero at the start and end of the computation. (While singleton nets may not uniquely correspond to a nonzero in $\mathbf{A}$, $\mathbf{B}$, or $\mathbf{C}$, they are never cut.) The bound on the critical-path communication cost is obtained by maximizing over processors. □

Lemma 1 yields a lower bound over all parallelizations, subject to a computational load balance constraint.

DEFINITION 3. *For any $\epsilon \in [0, p-1]$, let Π_ϵ be the set of all partitions $\{\mathcal{V}_1, \ldots, \mathcal{V}_p\}$ of $\mathcal{V}$ where $|\mathcal{V}_i| \leq (1+\epsilon)|\mathcal{V}|/p$ for each $i \in [p]$. We say an SpGEMM computation with parallelization $\{\mathcal{V}_1, \ldots, \mathcal{V}_p\} \in \Pi_\epsilon$ is* ϵ-load balanced.

THEOREM 1. *For any ϵ-load balanced SpGEMM computation, its critical-path communication cost is at least*

$$\min_{\{\mathcal{V}_1, \ldots, \mathcal{V}_p\} \in \Pi_\epsilon} \max_{i \in [p]} |Q_i|.$$

The next result shows that this critical-path lower bound is tight up to a logarithmic factor.

THEOREM 2. *For an SpGEMM computation with parallelization $\{\mathcal{V}_1, \ldots, \mathcal{V}_p\}$, there exists a data distribution such that the number of words processor i sends/receives is $O(|Q_i|)$ and the critical-path communication cost is $O(\log p \cdot \max |Q_i|)$.*

PROOF. We construct a data distribution by assigning the nonzero corresponding to each net in $\mathcal{N}^{\mathbf{A}}$, $\mathcal{N}^{\mathbf{B}}$, and $\mathcal{N}^{\mathbf{C}}$ to one of the processors owning a vertex in that net. The expand phase proceeds in at most $O(\log p)$ steps. Each nonzero of $\mathbf{A}$ and $\mathbf{B}$ is associated with a binary-tree broadcast among the processors whose parts $\mathcal{V}_i$ intersect the corresponding nets. Processor i receives each of its nonzeros at most once and sends each at most twice, for a total cost of $O(|Q_i|)$. Further, at each step j, processor i performs its assigned sends or receives from all the broadcast trees in which it is involved at level j (at most $|Q_i|$), and so each step involves at most $O(\max_{i \in [p]} |Q_i|)$ sends/receives along the critical path. The fold phase is similar, using binary-tree reductions. □

Table 1: Communication cost of forming $\mathbf{P}^T\mathbf{AP}$.

		$\mathbf{AP}$		$\mathbf{P^T(AP)}$	
N	p	row	fine	row	fine
19,683	27	5,528	4,649	10,712	964
91,125	125	5,528	5,823	10,712	1,324
250,047	343	5,528	6,160	10,712	1,444
531,441	729	5,528	6,914	10,712	1,491
970,299	1,331	5,528	6,679	10,712	1,548

3. EXPERIMENTAL RESULTS

Next, we experimentally study the communication costs of computing the product $\mathbf{P}^T\mathbf{AP}$, formed to produce the grid hierarchy of an algebraic multigrid PDE solver. Here, $\mathbf{A}$ is the adjacency matrix of a graph $G_{\mathbf{A}}$ of $N = n^3$ vertices arranged as a 3D lattice, where every non-boundary vertex is adjacent to itself and its 26 closest neighbors. The matrix $\mathbf{P}$ corresponds to a bipartite graph $G_{\mathbf{P}} = (U, V, E)$, where U is the vertex set of $G_{\mathbf{A}}$ and V is the set of $(n/3)^3$ disjoint 3-by-3-by-3 subcubes of U. Every subcube $v \in V$ is adjacent to each vertex $u \in U$ that belongs to v or is one of v's neighbors.

The standard *row-wise* approach for computing $\mathbf{P}^T\mathbf{AP}$ [5] works as follows. Partition the 3D lattice into p equal subcubes and assign the rows of $\mathbf{A}$ that belong to each subcube to the corresponding processor, which then uses its assigned rows to compute the corresponding rows of $\mathbf{AP}$. Next, partition the rows of $\mathbf{P}^T$ similarly and compute $\mathbf{P}^T(\mathbf{AP})$.

We compare the maximum per-processor communication costs of the row-wise approach to the *fine-grained* approach, where we construct a parallelization by partitioning the hypergraphs $\mathcal{H}(\mathbf{A}, \mathbf{P})$ and $\mathcal{H}(\mathbf{P}^T, \mathbf{AP})$ using the PaToH library. The results are shown in Table 1, where we follow a weak-scaling scheme that increases N and p proportionally. The table shows that in the $\mathbf{AP}$ step, the cost of the row-wise approach is comparable to or less than that of the fine-grained approach. We believe the slight advantage to the row-wise approach is due to the fact that PaToH minimizes communication volume rather than maximum per-processor cost. In the $\mathbf{P}^T(\mathbf{AP})$ step, the cost of the row-wise approach is 7 to 11 times greater than that of the fine-grained approach. Although this gap slightly narrows as we scale, likely because of the mismatch in cost functions, the fine-grained approach's advantage is dramatic, and demonstrates that better algorithms for $\mathbf{P}^T\mathbf{AP}$ should be developed. Improvements will be described in a future paper [2].

4. REFERENCES

[1] K. Akbudak and C. Aykanat. Simultaneous input and output matrix partitioning for outer-product–parallel sparse matrix-matrix multiplication. *SISC*, 36(5):C568–C590, 2014.

[2] G. Ballard, J. Hu, and C. Siefert. Reducing communication costs for sparse matrix multiplication within algebraic multigrid. In preparation, 2015.

[3] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SISC*, 34(4):C170–C191, 2012.

[4] Ü. Çatalyürek and C. Aykanat. A fine-grain hypergraph model for 2D decomposition of sparse matrices. In *IPDPS '01*, pages 118–123, 2001.

[5] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. ML 5.0 smoothed aggregation user's guide. Sandia National Laboratories, TR SAND2006-2649, 2006.

[6] S. Krishnamoorthy, Ü. Çatalyürek, J. Nieplocha, A. Rountev, and P. Sadayappan. Hypergraph partitioning for automatic memory hierarchy management. In *SC '06*, pages 34–46, 2006.