Augmenting Hypergraph Models with Message Nets to Reduce Bandwidth and Latency Costs Simultaneously

Oguz Selvitopi[1], Seher Acer[1], and Cevdet Aykanat[1]

[1] Department of Computer Engineering, Bilkent University, Turkey, 06800

Abstract

Efficient parallelization of the applications in scientific computing domain on distributed systems requires reducing the communication costs of the application in both terms: bandwidth and latency costs. Although there are many graph/hypergraph partitioning models that reduce the bandwidth cost in the literature, there exist only a few works that consider both of them. These works generally rely on a two-phase methodology in which different objectives of reducing bandwidth cost and latency cost are addressed in separate phases. Such approaches suffer from the fact that the addressed objective in a specific phase is oblivious to the objective addressed in the other phase.

In this work, we propose a single-phase hypergraph model for 1D partitioning that addresses both objectives simultaneously [1]. The widely-used standard hypergraph model in the literature [2] minimizes the bandwidth cost by reducing the total communication (message) volume as the nets of this model, which we refer to as *volume nets*, encode the amount of data transferred in case they become cut. We augment the standard hypergraph model by *message nets* that encode the messages communicated.

The addition of the message nets relies on recursive bipartitioning (RB). In the RB paradigm, the given hypergraph is recursively bipartitioned into two hypergraphs until the desired number of parts is reached. Let the RB process be currently at the ℓth level, prior to bipartitioning the ith hypergraph $\mathcal{H}_i^\ell = (\mathcal{V}_i^\ell, \mathcal{N}_i^\ell)$ in this level. Figure 1 displays the RB tree representing the corresponding RB process. As seen in the figure, the vertex sets of the hypergraphs in the current state of the RB tree induce a $(2^\ell + i)$-way vertex partition Π_{cur}. This vertex partition is also assumed to induce a $(2^\ell + i)$-way processor partition $\mathbb{P}_{\text{cur}}$ in which processor group $\mathcal{P}_i^\ell$ is held responsible for the items/tasks that are represented by the vertices in $\mathcal{V}_i^\ell$. After bipartitioning $\mathcal{H}_i^\ell$, the $(2^\ell + i + 1)$-way vertex partition $\Pi_{\text{new}} = (\Pi_{\text{cur}} - \mathcal{V}_i^\ell) \cup \mathcal{V}_{2i}^{\ell+1} \cup \mathcal{V}_{2i+1}^{\ell+1}$ is obtained. Bipartitioning $\mathcal{H}_i^\ell$ is assumed to also bipartition the processor group $\mathcal{P}_i^\ell$ into two processor groups $\mathcal{P}_L$ and $\mathcal{P}_R$, where $\mathbb{P}_{\text{new}} = \mathbb{P}_{\text{cur}} - \{\mathcal{P}_i^\ell\} \cup \{\mathcal{P}_L, \mathcal{P}_R\}$. We respectively denote the number of messages that processor group $\mathcal{P}_i^\ell$ and processor groups $\{\mathcal{P}_L, \mathcal{P}_R\}$ sends/receives to/from the other processor groups by M_{cur} and M_{new}. To encode the messages between processor group $\mathcal{P}_i^\ell$ and the other processor groups in $\mathbb{P}_{\text{cur}}$, we add message nets to $\mathcal{H}_i^\ell$ by utilizing Π_{cur}. A message net added to $\mathcal{H}_i^\ell$ represents a group of tasks/data items that necessitate send-

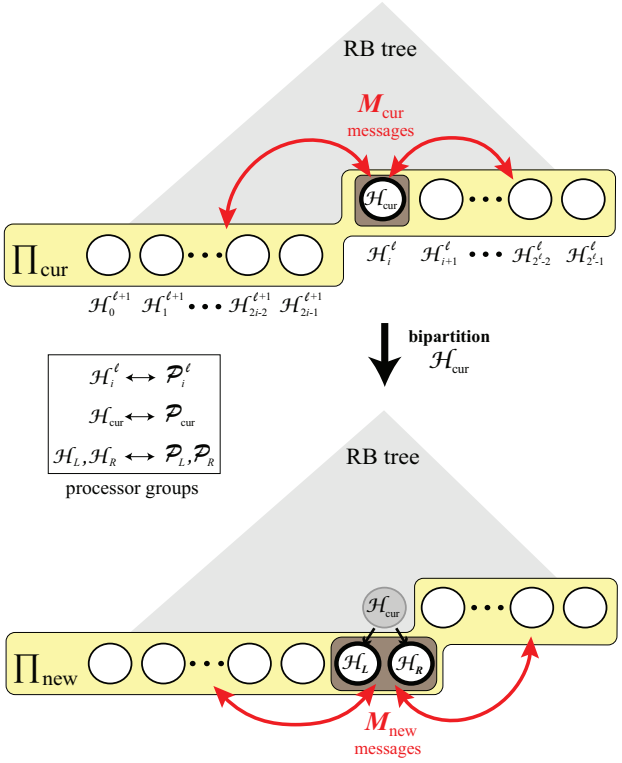

Figure 1: The state of the RB tree and the number of messages from/to $\mathcal{P}_{\text{cur}}$ and $\{\mathcal{P}_L, \mathcal{P}_R\}$ to/from the other processor groups before and after bipartitioning $\mathcal{H}_{\text{cur}}$. The processor groups corresponding to the vertex sets of the hypergraphs are shown in the box [1].

ing/receiving a message from/to $\mathcal{P}_i^\ell$ to/from another processor group.

There are two types of message nets: send nets and receive nets. A send net s_k added to hypergraph $\mathcal{H}_i^\ell$ to be bipartitioned signifies a message to be sent from processor group $\mathcal{P}_i^\ell$ to processor group $\mathcal{P}_k$. Net s_k connects the vertices corresponding to the data items to be sent from $\mathcal{P}_i^\ell$ to $\mathcal{P}_k$. We add a send net to $\mathcal{H}_i^\ell$ for each such processor group that $\mathcal{P}_i^\ell$ will send a message to. In a dual manner, a receive net r_k added to $\mathcal{H}_i^\ell$ signifies a message to be received by $\mathcal{P}_i^\ell$ from $\mathcal{P}_k$. Net r_k connects the vertices corresponding to the tasks that necessitate $\mathcal{P}_i^\ell$ to receive data items from $\mathcal{P}_k$. We add a receive net to $\mathcal{H}_i^\ell$ for each such processor group that $\mathcal{P}_i^\ell$ will receive a message from.

These formed message nets are added to the standard hypergraph model, which already contains the volume nets. The costs of volume nets are set to t_w, per word transfer time, and the costs of message nets are set to t_s, message startup time. Bipartitioning this augmented hypergraph with the objective of minimizing the cutsize corresponds to reducing both the total volume and the total message count as it contains both volume and message nets. The number of cut message nets in this bipartition amounts to the increase in the number of messages between processor group $\mathcal{P}_{\text{cur}}$ and the other processor groups, i.e., $M_{\text{new}} - M_{\text{cur}}$.

The cost addition of the message nets throughout the entire RB process is $O(p \log K)$ and relatively cheap compared to the partitioning overhead, where p is the number of pins in the standard hypergraph model and K is the number of parts/processors.

We evaluated the proposed model with 1D row-parallel sparse matrix vector multiplication (SpMV), and compared it against the standard hypergraph model on 30 matrices [3]. Table 1 displays the results obtained by the proposed model as normalized with respect to the results of the standard model. The number of processors in our experiments are 128, 256, 512, 1024 and 2048. The ratio t_s/t_w varies in practice for different message sizes and depends on the protocol used for transmitting messages as well as the characteristics of the target application. For this reason, we experimented with different t_s/t_w values: 10, 50, 100 and 200.

As seen in the table, we achieved up to 46%, 52%, 51%, 48% and 43% reduction in the total number of messages for 128, 256, 512, 1024 and 2048 processors, respectively. Using a higher value for t_s/t_w magnifies both the reduction in total message count and the increase in total volume. Compared to the standard hypergraph model, the proposed model results in a higher total volume since it considers both bandwidth and latency components simultaneously while the standard model solely optimizes the bandwidth component.

Table 1: Communication statistics, partitioning times and parallel SpMV runtimes of the proposed model normalized with respect to those of the standard model as the geometric average of 30 matrices [1].

		volume		#messages		part.	**parallel SpMV**
t_s/t_w	K	tot	max	tot	max	time	**time**
10	128	1.08	1.11	0.82	0.87	1.07	**0.956**
	256	1.10	1.16	0.78	0.83	1.13	**0.904**
	512	1.12	1.22	0.75	0.83	1.13	**0.838**
	1024	1.16	1.29	0.73	0.84	1.25	**0.792**
	2048	1.20	1.37	0.71	0.88	1.28	**0.774**
50	128	1.17	1.25	0.65	0.76	1.08	**0.924**
	256	1.25	1.44	0.59	0.70	1.14	**0.846**
	512	1.33	1.57	0.56	0.69	1.21	**0.760**
	1024	1.41	1.69	0.57	0.74	1.24	**0.715**
	2048	1.48	1.85	0.59	0.80	1.33	**0.708**
100	128	1.24	1.43	0.59	0.73	1.09	**0.954**
	256	1.35	1.66	0.53	0.68	1.17	**0.858**
	512	1.45	1.86	0.51	0.68	1.19	**0.768**
	1024	1.54	1.92	0.53	0.71	1.31	**0.706**
	2048	1.61	2.06	0.57	0.80	1.41	**0.707**
200	128	1.33	1.60	0.54	0.72	1.15	**1.031**
	256	1.46	1.87	0.48	0.67	1.19	**0.872**
	512	1.57	2.02	0.49	0.67	1.25	**0.778**
	1024	1.65	2.09	0.52	0.72	1.37	**0.722**
	2048	1.70	2.17	0.57	0.79	1.48	**0.712**

The more accurate representation of the communication costs together with the significant reductions in total message count led up to improvements of 8%, 15%, 24%, 29% and 29% in parallel SpMV runtime for 128, 256, 512, 1024 and 2048 processors, respectively.

References

[1] O. Selvitopi, S. Acer, and C. Aykanat, "A recursive hypergraph bipartitioning framework for reducing bandwidth and latency costs simultaneously," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2016, doi:10.1109/TPDS.2016.2577024.

[2] U. Catalyurek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, pp. 673–693, July 1999. [Online]. Available: http://portal.acm.org/citation.cfm?id=311796.311798

[3] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049663