

## Efficient Parallel Sparse Matrix–Vector Multiplication Using Graph and Hypergraph Partitioning

William Knottenbelt

Imperial College London

[wjk@doc.ic.ac.uk](mailto:wjk@doc.ic.ac.uk)

February 2015

- U.V. Çatalyürek and C. Aykanat: “Hypergraph-Partitioning-Based Decomposition for Parallel Sparse Matrix–Vector Multiplication”. *IEEE Trans. on Parallel and Distributed Systems*, 10(7), July 1999, pp. 673–693.
- A. Trifunovic: “Parallel Algorithms for Hypergraph Partitioning”. PhD thesis, Imperial College London, November 2005.
- J.T. Bradley, D.V. de Jager, W.J. Knottenbelt, A. Trifunovic: “Hypergraph Partitioning for Faster PageRank Computation”. *Proc. EPEW 2005*, pp. 155–171.
- A. Trifunovic and W.J. Knottenbelt: “A General Graph Model for Representing Exact Communication Volume in Parallel Sparse Matrix–Vector Multiplication”. *Proc. ISCIS 2006*, pp. 813–824.

## Recommended Software Tools

## Outline

- CHACO graph partitioning software:  
<http://www.cs.sandia.gov/~bahendr/chaco.html>
- PaToH hypergraph partitioning software:  
<http://bmi.osu.edu/~umit/software.html>
- METIS/ParMETIS graph partitioners and hMETIS hypergraph partitioner:  
<http://glaros.dtc.umn.edu/gkhome/views/metis>
- Parkway parallel hypergraph partitioner:  
<http://www.doc.ic.ac.uk/~at701/parkway/>

- Parallel Sparse Matrix–Vector Products
- Partitioning Objectives and Strategies
- Naïve Row-Striping
- 1D Graph Partitioning
- 1D Hypergraph Partitioning
- 2D Hypergraph Partitioning
- Comparison of Graph and Hypergraph Partitioning Techniques

- Parallel sparse matrix–vector product (and similar) operations form the kernel of many parallel numerical algorithms.
- Particularly widely used in iterative algorithms for solving very large sparse systems of linear equations (e.g. Jacobi and Conjugate-Gradient Squared methods).
- The data partitioning strategy adopted (i.e. the assignment of matrix and vector elements to processors) has a major impact on performance, especially in distributed memory environments.

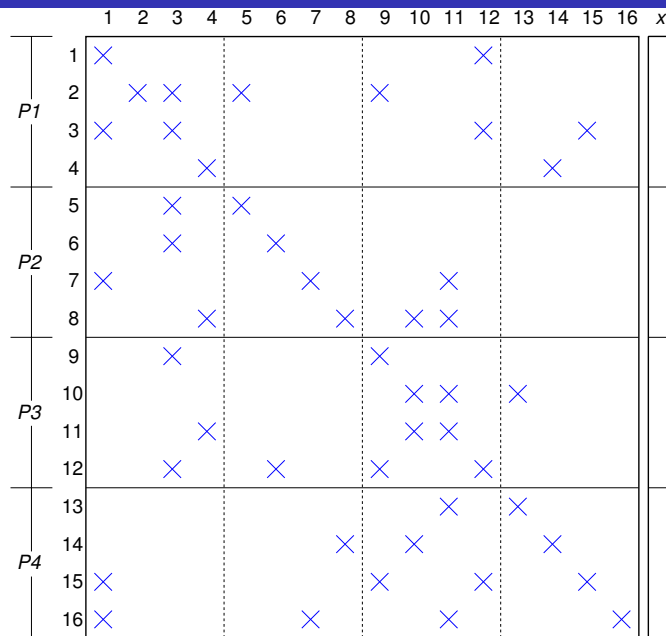
- Aim is to allocate matrix and vector elements across processors such that:
  - computational load is balanced
  - communication is minimised
- Candidate partitioning strategies:
  - random permutation applied to rows and columns with 2D checkerboard processor layout
  - naïve row (or column) striping
  - coarse-grained mapping of rows (or columns) and corresponding vector elements to processors using 1D graph or hypergraph-based data partitioning
  - fine-grained mapping of individual non-zero matrix elements and vector elements to processors using 2D hypergraph-based partitioning

## Partitioning Objectives and Strategies

- Aim is to allocate matrix and vector elements across processors such that:
  - computational load is balanced
  - communication is minimised
- Candidate partitioning strategies:
  - random permutation applied to rows and columns with 2D checkerboard processor layout
  - naïve row (or column) striping
  - coarse-grained mapping of rows (or columns) and corresponding vector elements to processors using 1D graph or hypergraph-based data partitioning
  - fine-grained mapping of individual non-zero matrix elements and vector elements to processors using 2D hypergraph-based partitioning

## Naïve Row-Striping: Definition

- Assume an  $n \times n$  sparse matrix  $\mathbf{A}$ , an  $n$ -vector  $\mathbf{x}$  and  $p$  processors.
- Simply allocate  $n/p$  matrix rows and  $n/p$  vector elements to each processor (assuming  $p$  divides  $n$  exactly).
- If  $p$  does not divide  $n$  exactly, allocate one extra row and one extra vector element to those processors with rank less than  $n \bmod p$ .
- What are the advantages and disadvantages of this scheme?



- Consider the layout of a  $16 \times 16$  non-symmetric sparse matrix  $\mathbf{A}$  and vector  $\mathbf{x}$  onto 4 processors under a naïve row-striping scheme on the previous slide.
- What is:
  - (a) the computational load per processor?
  - (b) the total comms volume per matrix-vector product?

## 1D Graph Partitioning: Definition

## 1D Graph Partitioning: Definition (cont.)

- An  $n \times n$  sparse matrix  $\mathbf{A}$  can be represented as an undirected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ .
- Each row  $i$  ( $1 \leq i \leq n$ ) in  $\mathbf{A}$  corresponds to vertex  $v_i \in \mathcal{V}$  in the graph.
- The (vertex) weight  $w_i$  of vertex  $v_i$  is the total number of non-zeros in row  $i$ .
- For the edge-set  $\mathcal{E}$ , edge  $e_{ij}$  connects vertices  $v_i$  and  $v_j$  with (edge) weight:
  - 1 if either one of  $|a_{ij}| > 0$  or  $|a_{ji}| > 0$ ,
  - 2 if both  $|a_{ij}| > 0$  and  $|a_{ji}| > 0$
- Aim to partition the vertices into  $p$  mutually exclusive subsets (parts)  $\{P_1, P_2, \dots, P_p\}$  such that *edge-cut* is minimised and load is *balanced*.

- An edge  $e_{ij}$  is cut if the vertices which it contains are assigned to two different processors, i.e. if  $v_i \in P_m$  and  $v_j \in P_n$  where  $m \neq n$ .
- The edge-cut is the sum of the edge weights of cut edges and is an approximation for the amount of interprocessor communication.
  - Why is it not exact?

- Let

$$W_k = \sum_{i \in P_k} w_i \quad (\text{for } 1 \leq k \leq p)$$

denote the weight of part  $P_k$ , and  $\overline{W}$  denote the average part weight.

- A partition is said to be balanced if:

$$(1 - \varepsilon)\overline{W} \leq W_k \leq (1 + \varepsilon)\overline{W}$$

for  $k = 1, 2, \dots, p$ .

- Problem of finding a balanced  $p$ -way partition that minimises edge cut is NP-complete.
- But heuristics can often be applied to obtain good sub-optimal solutions.
- Software tools:
  - CHACO
  - METIS
  - ParMETIS
- Once partition has been computed, assign matrix row  $i$  to processor  $k$  if  $v_i \in P_k$ .

## 1D Graph Partitioning: Example

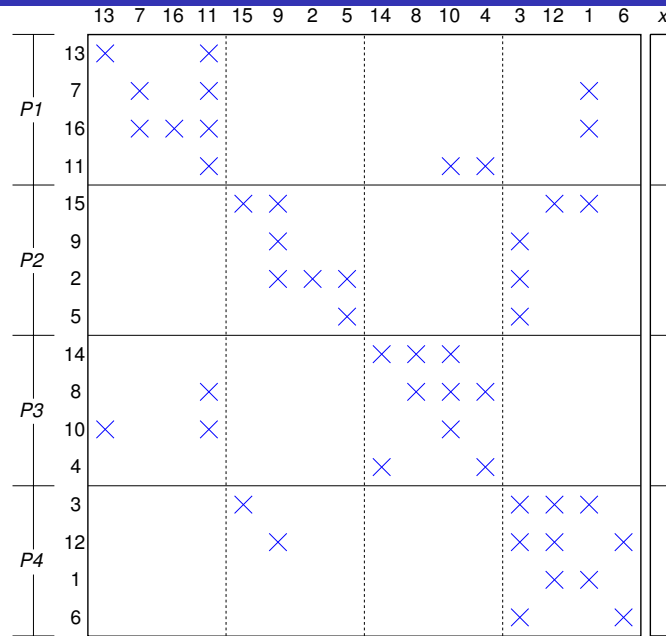
## 1D Graph Partitioning: Example (cont.)

- Consider the graph corresponding to the sparse matrix  $\mathbf{A}$  of the previous example.
- Assume the graph is partitioned into four parts as follows:

$$P_1 = \{v_{13}, v_7, v_{16}, v_{11}\} \quad P_2 = \{v_{15}, v_9, v_2, v_5\}$$

$$P_3 = \{v_{14}, v_8, v_{10}, v_4\} \quad P_4 = \{v_3, v_{12}, v_1, v_6\}$$

- Draw the graph representation and compute the edge cut.



- The row-striped layout of the sparse matrix **A** and vector **x** onto 4 processors under this graph-partitioning scheme is given on the previous slide.
- What is:
  - (a) the computational load per processor?
  - (b) the total comms vol. per matrix–vector product? How does the comms vol. compare to the edge cut?

## 1D Hypergraph Partitioning: Definition

## 1D Hypergraph Partitioning: Definition (cont.)

- An  $n \times n$  sparse matrix **A** can be represented as a hypergraph  $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ .
- $\mathcal{V}$  is a set of vertices and  $\mathcal{N}$  is a set of nets or hyperedges. Each  $n \in \mathcal{N}$  is a subset of the vertex set  $\mathcal{V}$ .
- Each row  $i$  ( $1 \leq i \leq n$ ) in **A** corresponds to vertex  $v_i \in \mathcal{V}$ .
- Each column  $j$  ( $1 \leq j \leq n$ ) in **A** corresponds to net  $N_j \in \mathcal{N}$ . In particular  $v_i \in N_j$  iff  $a_{ij} \neq 0$ .
- The (vertex) weight  $w_i$  of vertex  $v_i$  is the total number of non-zeros in row  $i$ .
- Given a partition  $\{P_1, P_2, \dots, P_p\}$ , the connectivity  $\lambda_j$  of net  $N_j$  denotes the number of different parts spanned by  $N_j$ . Net  $N_j$  is cut iff  $\lambda_j > 1$ .

- The cutsize or hyperedge cut of a partition is defined as:

$$\sum_{N_j \in \mathcal{N}} (\lambda_j - 1)$$

- Aim is to minimise the hyperedge cut while maintaining the balance criterion (which is same as for graphs).
- Again, problem of finding a balanced  $p$ -way partition that minimises the hyper-edge cut is NP-complete, but heuristics can be used to find sub-optimal solutions.
- Software tools:
  - hMETIS
  - PaToH
  - Parkway

- Consider the hypergraph corresponding to the sparse matrix **A** of the previous example.

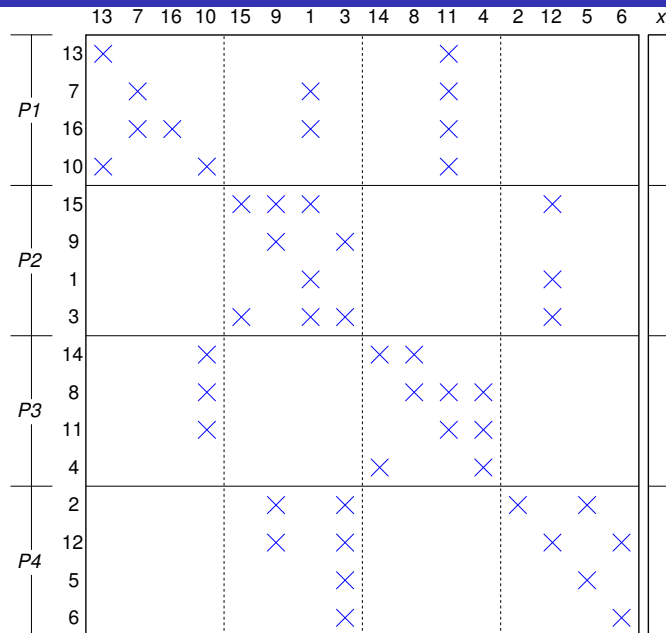
- Assume the hypergraph is partitioned into four parts as follows:

$$P_1 = \{v_{13}, v_7, v_{16}, v_{10}\} \quad P_2 = \{v_{15}, v_9, v_1, v_3\}$$

$$P_3 = \{v_{14}, v_8, v_{11}, v_4\} \quad P_4 = \{v_2, v_{12}, v_5, v_6\}$$

- Draw the hypergraph representation and compute the hyperedge cut.

## 1D Hypergraph Partitioning: Example (cont.)



## 1D Hypergraph Partitioning: Example (cont.)

- The row-striped layout of the sparse matrix **A** and vector **x** onto 4 processors under this hypergraph partitioning scheme is given on the previous slide.
- What is:
  - the computational load per processor?
  - the total comms vol. per matrix-vector product? How does the comms vol. compare to the edge cut?

- The most general mapping possible is to allocate individual non-zero matrix elements and vector elements to processors.
- General form of parallel sparse matrix–vector multiplication follows four stages, where each processor:
  - ① sends its  $x_j$  values to processors that possess a non-zero  $a_{ij}$  in column  $j$ ,
  - ② computes the products  $a_{ij}x_j$  for its non-zeros  $a_{ij}$  yielding a set of contributions  $b_{is}$  where  $s$  is a processor identifier.
  - ③ sends  $b_{is}$  values to the processor that has  $b_i$ .
  - ④ adds up received contributions for assigned vector elements, so 
$$b_i = \sum_{s=0}^{p-1} b_{is}$$
- Each non-zero is modelled by a vertex (weight 1) in the hypergraph; if  $a_{ij}$  is zero then add “dummy” vertex (weight 0).
- Model Stage 1 comms volume by net whose constituent vertices are the non-zeros of column  $j$ . Model Stage 3 comms volume by net whose constituent vertices are the non-zeros of row  $i$ .
- Now partition hypergraph into  $p$  parts such that the  $k - 1$  metric is minimised, subject to balance constraint.
- Assign non-zeros to processors according to partition.
- Assign  $b_i$ 's to processors appropriately according to whether row  $i$  and/or column  $i$  hyperedge is cut (if any).

## Comparison of Techniques

- A graph partition aims to minimise the number of non-zero entries in off-diagonal matrix blocks.
- A hypergraph partition aims to minimise actual communication; the partition may have more off-diagonal non-zero entries than a graph partition but these will tend to be column aligned.
- Either sort of partitioning is preferable to a naïve or random partition.
- Parallel partitioning tools are necessary for very large matrices, e.g. ParMETIS for graph partitioning, or Parkway, Zoltan, ... for hypergraph partitioning.