Sparse Matrix Partitioning

Rob H. Bisseling

Mathematical Institute, Utrecht University

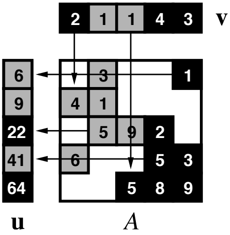
Course Introduction Scientific Computing
February 19, 2018

Universiteit Utrecht

Parallel sparse matrix–vector multiplication $\mathbf{u} := A\mathbf{v}$

A sparse $m \times n$ matrix, $\mathbf{u}$ dense m-vector, $\mathbf{v}$ dense n-vector

$$u_i := \sum_{j=0}^{n-1} a_{ij} v_j$$



$p = 2$

4 supersteps: communicate, compute, communicate, compute

Divide evenly over 4 processors

Universiteit Utrecht

Avoid communication completely, if you can

All nonzeros in a row or column have the same colour.

Permute the matrix rows/columns

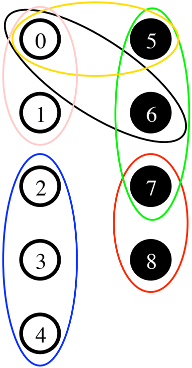
First the green rows/columns, then the blue ones.

Combinatorial problem: sparse matrix partitioning

Problem: Split the set of nonzeros A of the matrix into p subsets, $A_0, A_1, \ldots, A_{p-1}$, minimising the communication volume $V(A_0, A_1, \ldots, A_{p-1})$ under the load imbalance constraint

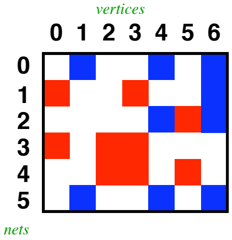
$$nz(A_i) \leq \frac{nz(A)}{p}(1+\epsilon), \quad 0 \leq i < p.$$

The hypergraph connection



Hypergraph with 9 vertices and 6 hyperedges (nets), partitioned over 2 processors, black and white

1D matrix partitioning using hypergraphs



- Hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N}) \Rightarrow$ exact communication volume in sparse matrix–vector multiplication.
- Columns $\equiv$ Vertices: $0, 1, 2, 3, 4, 5, 6$.
  Rows $\equiv$ Hyperedges (nets, subsets of $\mathcal{V}$):

$$n_0 = \{1, 4, 6\}, \quad n_1 = \{0, 3, 6\}, \quad n_2 = \{4, 5, 6\},$$
$$n_3 = \{0, 2, 3\}, \quad n_4 = \{2, 3, 5\}, \quad n_5 = \{1, 4, 6\}.$$

$(\lambda - 1)$-metric for hypergraph partitioning

- 138×138 symmetric matrix `bcsstk22`, $nz = 696$, $p = 8$
- Reordered to Bordered Block Diagonal (BBD) form
- Split of row i over λ_i processors causes a communication volume of $\lambda_i - 1$ data words

Cut-net metric for hypergraph partitioning

- Row split has unit cost, irrespective of λ_i

Mondriaan 2D matrix partitioning

- $p = 4$, $\epsilon = 0.2$, global non-permuted view

Fine-grain 2D matrix partitioning

- Each individual nonzero is a vertex in the hypergraph
  Çatalyürek and Aykanat, 2001.

Matrix lns3937 (Navier–Stokes, fluid flow)

Splitting the sparse matrix lns3937 into 5 parts.

Recursive, adaptive bipartitioning algorithm

MatrixPartition(A, p, ϵ)

input: p = number of processors, $p = 2^q$

ϵ = allowed load imbalance, $\epsilon > 0$.

output: p-way partitioning of A with imbalance $\leq \epsilon$.

if $p > 1$ **then**

$q := \log_2 p$;

$(A_0^{\mathrm{r}}, A_1^{\mathrm{r}}) := h(A, \mathrm{row}, \epsilon/q)$; hypergraph splitting

$(A_0^{\mathrm{c}}, A_1^{\mathrm{c}}) := h(A, \mathrm{col}, \epsilon/q)$;

(A_0, A_1) := best of $(A_0^{\mathrm{r}}, A_1^{\mathrm{r}})$, $(A_0^{\mathrm{c}}, A_1^{\mathrm{c}})$;

$maxnz := \frac{nz(A)}{p}(1+\epsilon)$;

$\epsilon_0 := \frac{maxnz}{nz(A_0)} \cdot \frac{p}{2} - 1$; MatrixPartition$(A_0, p/2, \epsilon_0)$;

$\epsilon_1 := \frac{maxnz}{nz(A_1)} \cdot \frac{p}{2} - 1$; MatrixPartition$(A_1, p/2, \epsilon_1)$;

else output A;

Mondriaan package

- Ordering to SBD and BBD structure: cut rows are placed in the middle, and at the end, respectively.
- Visualisation through Matlab interface, MondriaanPlot, and MondriaanMovie
- Metrics: $\lambda - 1$ for parallelism, and cut-net for other applications
- Library-callable, so you can link it to your own program

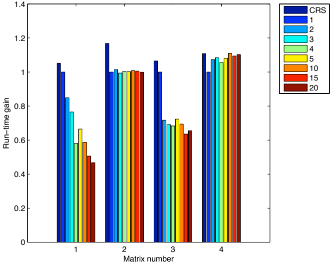
Partition the columns till the end, $p = n = 59$

- The recursive, fractal-like nature makes the SBD ordering method work, irrespective of the actual cache characteristics (e.g. sizes of L1, L2, L3 cache).
- The ordering is cache-oblivious.

Try to forget it all

- Ordering the matrix in SBD format makes the matrix-vector multiplication cache-oblivious. Forget about the exact cache hierarchy. It will always work.
- We also like to forget about the cores: core-oblivious. And then processor-oblivious, node-oblivious.
- All that is needed is a good ordering of the rows and columns of the matrix, and subsequently of its nonzeros.

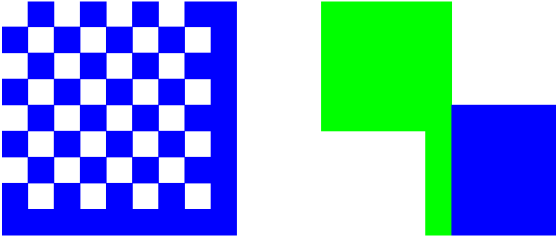
Wall clock timings of SpMV on Huygens



Splitting into 1–20 parts

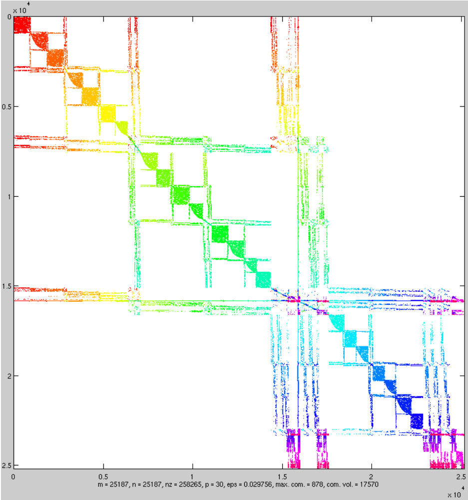
- Experiments on 1 core of the dual-core 4.7 GHz Power6+ processor of the Dutch national supercomputer Huygens.
- 64 kB L1 cache, 4 MB L2, 32 MB L3.
- Test matrices: 1. `stanford`; 2. `stanford_berkeley`; 3. `wikipedia-20051105`; 4. `cage14`

Doubly Separated Block-Diagonal structure



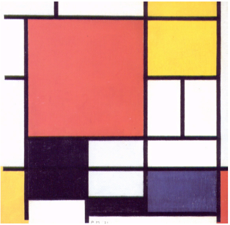
- 9×9 chess-arrowhead matrix, $nz = 49$, $p = 2$, $\epsilon = 0.2$.
- DSBD structure is obtained by recursively partitioning the sparse matrix, each time moving the cut rows and columns to the middle.
- The nonzeros must also be reordered by a Z-like ordering.
- Mondriaan is used in two-dimensional mode.

Screenshot of Matlab interface



- Matrix `rhpentium`, split over 30 processors

Summary



- We have presented two combinatorial problems: partitioning and ordering. Solution of these is an enabling technology for high-performance computing, in particular for linear system solving.
- Matrix reordering is a promising method for oblivious computing. We have shown its utility in enhancing cache performance.
- The Mondriaan package provides both partitioning and ordering methods. based on hypergraph partitioning.
- Visualisation can help in designing new algorithms!