# Combinatorial Problems in High-Performance Computing

Rob H. Bisseling

Mathematical Institute, Utrecht University

March 24, 2010



Universiteit Utrecht

## Partitioning problems

- Parallel sparse matrix–vector multiplication
- Movie: chess matrix
- Hypergraphs
- 2D matrix partitioning
- Vector partitioning

## Matching problems

- Parallel edge-weighted matching
- Example graph

## Ordering problems

- Separated Block Diagonal structure
- Movie: Navier–Stokes
- Parallel computing revolution

## Conclusions and future work

# Joint work

My PhD Students:


Albert-Jan Yzelman

Bas Fagginger Auer

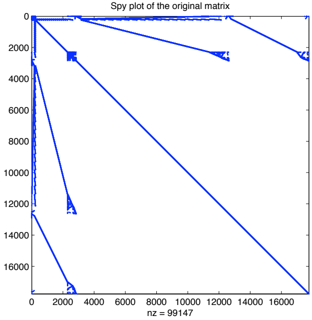
Other collaborators: Brendan Vastenhouw, Wouter Meesen, Tristan van Leeuwen, Fredrik Manne (Bergen, Norway), Ümit Çatalyürek (Ohio, USA)

Outline



Universiteit Utrecht

# Motivation: sparse matrix `memplus`




17758 × 17758 matrix with 126150 nonzeros.
Contributed to MatrixMarket in 1995 by Steve Hamm (Motorola). Represents the design of a memory circuit.
Iterative solver multiplies matrix repeatedly with a vector.

# Motivation: high-performance computer

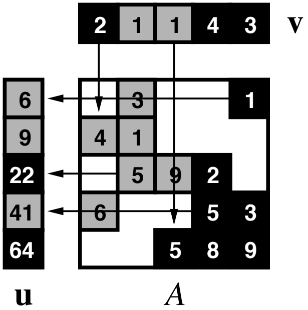
- National supercomputer Huygens named after Christiaan Huygens, Dutch astronomer who in 1655 proposed the form of the rings around Saturn
- Huygens, the machine, has 104 nodes
- Each node has 16 processors
- Each processor has 2 cores and an L3 cache
- Each core has an L1 and L2 cache

Now you go out and program this machine so that it works efficiently at all levels of its architecture!

# 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

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

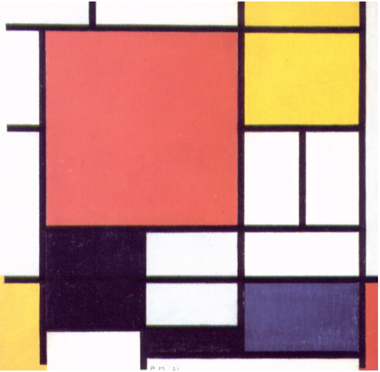



$p = 2$

4 phases: communicate, compute, communicate, compute

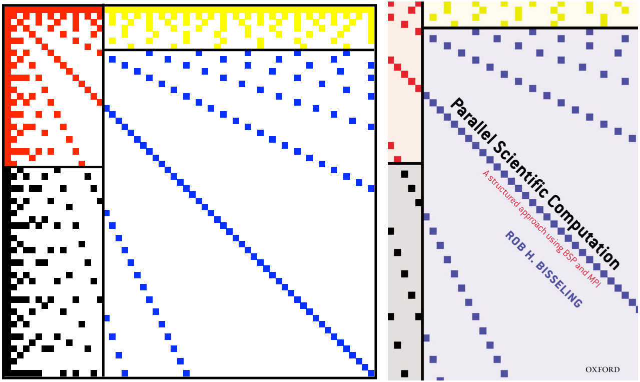
# Divide evenly over 4 processors

# Composition with Red, Yellow, Blue and Black



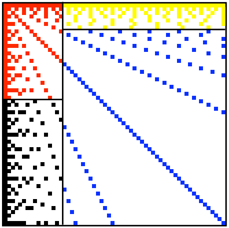
Piet Mondriaan 1921

# Matrix prime60




- Mondriaan block partitioning of $60 \times 60$ matrix `prime60` with 462 nonzeros, for $p = 4$
- $a_{ij} \neq 0 \Longleftrightarrow i|j \text{ or } j|i \qquad (1 \leq i,j \leq 60)$

# Communication volume for partitioned matrix



$$V(A_0, A_1, A_2, A_3) = V(A_0, A_1, A_2 \cup A_3) + V(A_2, A_3)$$

Here, $V(A_0, A_1, A_2, A_3)$ is the global matrix–vector communication volume corresponding to the partitioning $A_0, A_1, A_2, A_3$

# Avoid communication completely, if you can

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

# Permute the matrix by row and column permutations



First the black rows, then the red ones.
First the black columns, then the red ones.

# Combinatorial problem: sparse matrix partitioning

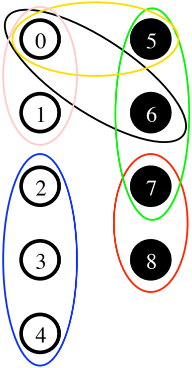
Problem: Split the set of nonzeros $A$ of the matrix into $p$ subsets, $A_0, A_1, \ldots, A_{p-1}$, minimising the communication volume $V(A_0, A_1, \ldots, A_{p-1})$ under the load imbalance constraint

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

The maximum amount of work should not exceed the average amount by more than a fraction $\epsilon$.

- $p = 2$ problem is already NP-complete (Lengauer 1990, circuit layout)
- Generalisation: heterogeneous processors with different speeds

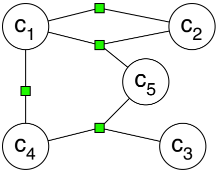
# The hypergraph connection




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

# Another view of hypergraphs

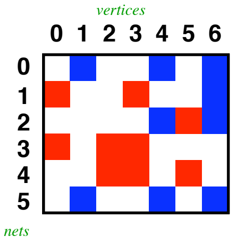
$$\begin{vmatrix} a_{11} & a_{12} & 0 & 0 & a_{15} \\ a_{21} & a_{22} & 0 & 0 & 0 \\ a_{31} & 0 & 0 & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \end{vmatrix}$$




(from Zoltan paper by Devine, Boman, et al. 2006)

- Hypergraph corresponding to a sparse matrix
- Columns are vertices. Rows (in green) are hyperedges.

# 1D matrix partitioning using hypergraphs

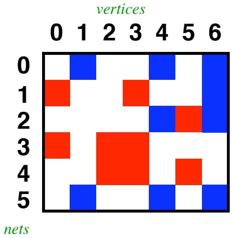



Column bipartitioning of $m \times n$ matrix

- 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 $\mathcal{V}$):

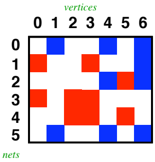
$$n_0 = \{1,4,6\}, \quad n_1 = \{0,3,6\}, \quad n_2 = \{4,5,6\},$$
$$n_3 = \{0,2,3\}, \quad n_4 = \{2,3,5\}, \quad n_5 = \{1,4,6\}.$$

# Minimising communication volume



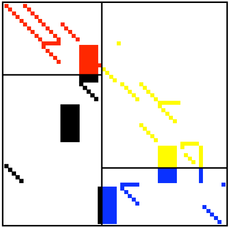

- Cut nets: $n_1$, $n_2$ cause one horizontal communication
- Use Kernighan–Lin/Fiduccia–Mattheyses for hypergraph bipartitioning
- Multilevel scheme: merge similar columns first, refine bipartitioning afterwards
- Used in PaToH (Çatalyürek and Aykanat 1999) for 1D matrix partitioning.

# General combinatorial problem



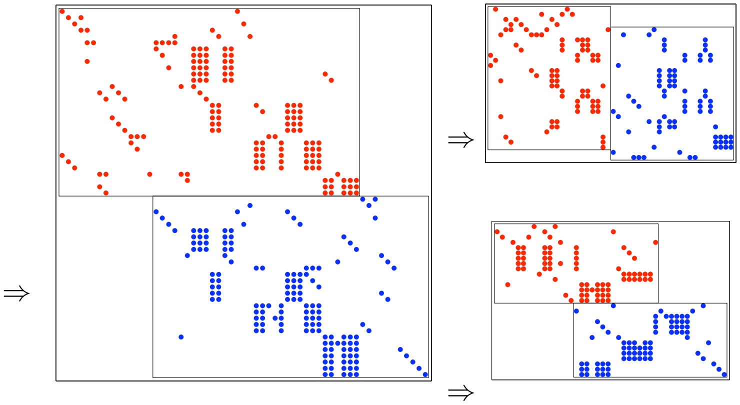

- Assign new pupils of a high school to 5 classes, while maintaining friendships (also in groups) and balancing class sizes.
- Well-known problem in VLSI circuit design.
- Can be solved by using MLpart, hMetis, PaToH, Zoltan, Par*k*way, or Mondriaan.

# Mondriaan 2D matrix partitioning



- Block partitioning (without row/column permutations) of $59 \times 59$ matrix `impcol_b` with 312 nonzeros, for $p = 4$
- Mondriaan package v1.0 (May 2002). Originally developed by Vastenhouw and Bisseling for partitioning term-by-document matrices for a parallel web search machine.

# Mondriaan 2D partitioning



- Recursively split the matrix into 2 parts.
- Try splits in row and column directions, allowing permutations. Each time, choose the best direction.

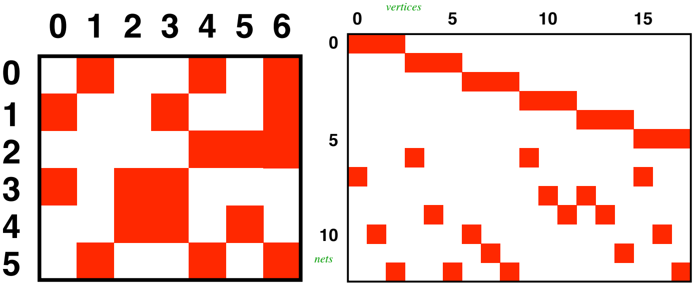
# Mondriaan 2.0, Released July 14, 2008

- New algorithms for vector partitioning. Often best achievable communication load balance (but not perfect).
- Much faster partitioning, by a factor of 10 compared to version 1.0.
- 10% better quality of the matrix partitioning.
- Inclusion of fine-grain partitioning method by Çatalyürek and Aykanat, 2001.
- Inclusion of hybrid between original Mondriaan and fine-grain methods.
- Can also handle non-powers of two for the number of processors.

# Fine-grain matrix partitioning

- Assign each nonzero of $A$ individually to a part.
- Each nonzero becomes a vertex in the hypergraph.
- Each matrix row and column becomes a hyperedge.
- Hence $nz(A)$ vertices and $m + n$ hyperedges.
- Proposed by Çatalyürek and Aykanat, 2001.

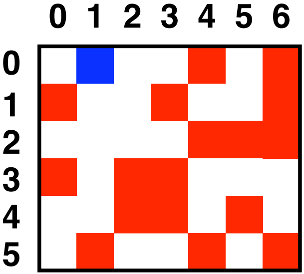
# Matrix view of fine-grain 2D partitioning

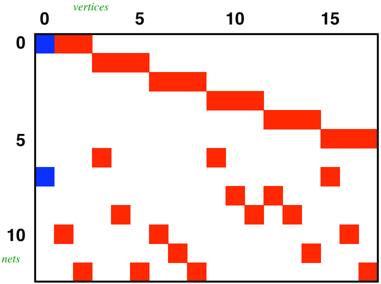



$A$ $\qquad$ $F = F_A$

- View the fine-grain hypergraph as an incidence matrix.
- $m \times n$ matrix $A$ with $nz(A)$ nonzeros
- $(m+n) \times nz(A)$ matrix $F = F_A$ with $2 \cdot nz(A)$ nonzeros
- $a_{ij}$ is $k$th nonzero of $A$ $\Leftrightarrow$ $f_{ik}$, $f_{m+j,k}$ are nonzero in $F$

# Communication for fine-grain 2D partitioning

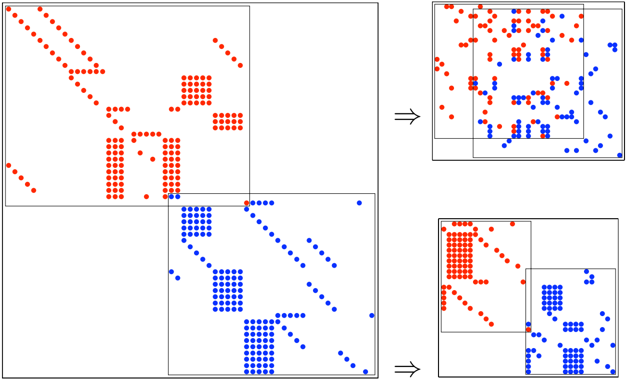



$A$ $F = F_A$

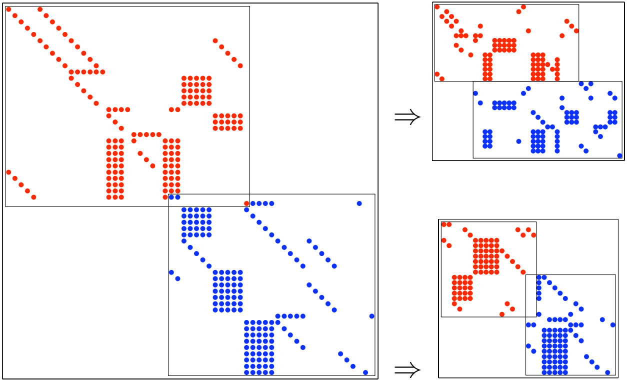
- Cut net in first $m$ nets (row nets) of hypergraph of $F$: nonzeros from row $a_{i*}$ are in different parts, hence horizontal communication in $A$.
- Cut net in last $n$ nets (col nets) of hypergraph of $F$: vertical communication in $A$.

# Fine-grain 2D partitioning



- Recursively split the matrix into 2 parts
- Assign individual nonzeros to parts
- For visualisation: move mixed rows to middle, red up, blue down. Same for columns.

# Hybrid 2D partitioning



- Recursively split the matrix into 2 parts
- Try splits in row and column directions, and fine-grain. Each time, choose the best of 3.
- Joint work with Tristan van Leeuwen and Ümit Çatalyürek, to be published

# Recursive, adaptive bipartitioning algorithm

MatrixPartition$(A, p, \epsilon)$

*input:* $\epsilon$ = allowed load imbalance, $\epsilon > 0$.

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

**if** $p > 1$ **then**

$q := \log_2 p$;

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

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

$(A_0^{\rm f}, A_1^{\rm f}) := h(A, {\rm fine}, \epsilon/q)$;

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

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

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

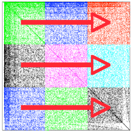
$\epsilon_1 := \frac{maxnz}{nz(A_1)} \cdot \frac{p}{2} - 1$; MatrixPartition$(A_1, p/2, \epsilon_1)$;

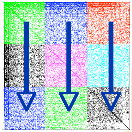
**else** output $A$;

# Mondriaan matrix + PaToH hypergraph partitioner

| Name | Area | $p$ | Mon | fine | hybrid |
|---|---|---|---|---|---|
| c98a | Cryptology | 4 | 100128 | 125370 | 97188 |
| | | 16 | 227298 | 330724 | 225418 |
| | | 64 | 417670 | 588012 | 407192 |
| stanford | Web links | 4 | 886 | 935 | 845 |
| | | 16 | 3226 | 3398 | 3039 |
| | | 64 | 9668 | 9296 | 8307 |
| polyDFT | Polymers | 4 | 8772 | 8841 | 8582 |
| | | 16 | 34099 | 36480 | 34867 |
| | | 64 | 73337 | 82544 | 73292 |
| cage13 | DNA | 4 | 117124 | 89540 | 89337 |
| | | 16 | 250480 | 189084 | 189110 |
| | | 64 | 436944 | 333876 | 333562 |

# Zoltan parallel hypergraph partitioning



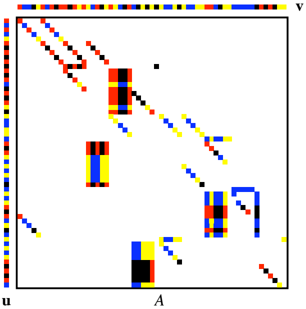


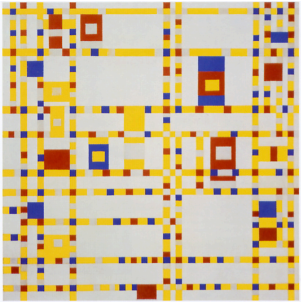
- Matrix to be split by columns into 2 parts.
- Matrix is stored by a two-dimensional Cartesian distribution
- This ensures scalability, while keeping the data distribution still relatively simple.
- Operations such as computing column inner products require horizontal and vertical communication.
- Version 3.1 September 2008 (Boman, Devine, Çatalyürek et al.)
- Zoltan includes row-based matrix partitioner Isorropia.



Universiteit Utrecht

# Vector partitioning



Broadway Boogie Woogie, 1942-43

- No extra communication if:
  $v_j \mapsto$ one of the owners of a nonzero in matrix column $j$
  $u_i \mapsto$ owner in matrix row $i$
- Joint work with Wouter Meesen, special issue of *ETNA* on combinatorial scientific computing (2005).

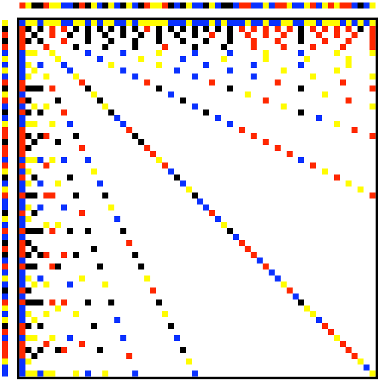
# Combinatorial problem: balance the communication

- Reduce the bulk synchronous parallel (BSP) cost

$$N_{\max} = \max_{0 \leq s < p} N(s),$$

  where $N(s) = \max(N_{\mathrm{send}}(s), N_{\mathrm{recv}}(s))$.
- Shown NP-complete (with help of Ali Pinar).
- In practice, optimal solution for a given matrix partitioning.
- But far from perfect communication balance: $N_{\max} \leq 4N_{\mathrm{avg}}$ observed ($\epsilon = 300\%$).
- Need to consider vector partitioning already during matrix partitioning (Uçar and Aykanat, *SIAM Review* 2007)

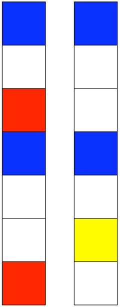
# Vector partitioning for prime60



# Similarity metric for column matching in coarsening

Column-scaled inner product:

$$W(u,v) = \frac{1}{\omega_{uv}} \sum_{i=0}^{m-1} u_i v_i = \text{weight of matching } u, v$$



- $\omega_{uv} = 1$ measures overlap
- $\omega_{uv} = \sqrt{d_u d_v}$ measures cosine of angle
- $\omega_{uv} = \min\{d_u, d_v\}$ measures relative overlap
- $\omega_{uv} = \max\{d_u, d_v\}$
- $\omega_{uv} = d_{u \cup v}$, Jaccard metric from information retrieval

Here, $d_u$ is the number of nonzeros of column $u$.



Universiteit Utrecht

# Matching problem in partitioning

- Open problem: what are the correct weights?
- Another problem: given vertices (representing columns), and weights for adjacent columns (those with overlap $\geq 1$), compute the best matching. A vertex can only match with one other vertex. No polygamy.
- Compute the matching fast, perhaps in parallel.

# Parallel edge-weighted matching

- Approximation algorithm with $\geq \frac{1}{2}$ times the optimal total weight.
- Joint work with Fredrik Manne (2008).
- Basic idea: edge $(u, v)$ is dominating if it has the highest weight of all the edges incident to $u$ and $v$.
- Maintain a set of dominating edges and deplete it, each time updating the heaviest edge of each vertex, and removing the dominated edges.
- Parallel: deplete the local dominating set first; use ghost vertices.

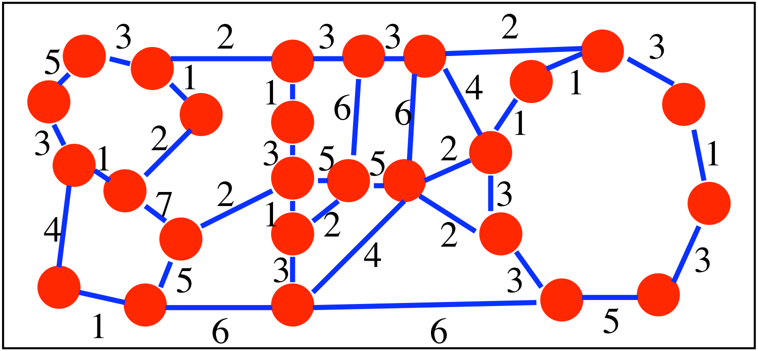
# Computation time optimal solution

- Computation time for the optimal algorithm by Harold Gabow (1990):

$$T = \mathcal{O}(mn + n^2 \log n),$$

  for $n$ vertices and $m$ edges.
- For $n = 10^6$ en $m = 10^7$, $T = 3 \times 10^{13}$.
- 4 hours 10 minutes on a dual-core PC of 1 Gflop/s per core. This takes too long!
- Even worse: actual speeds of graph computations are far from advertised peak flop rates.

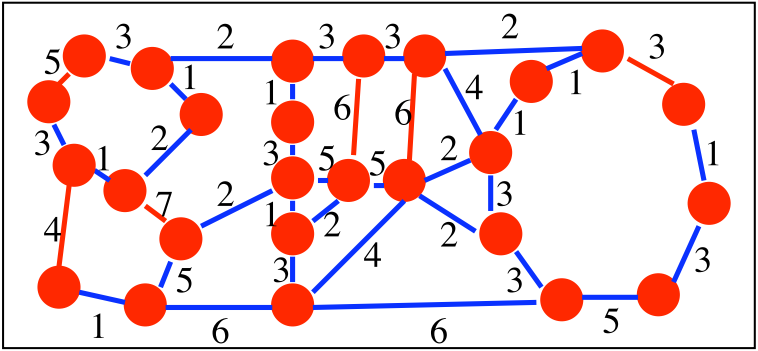
# Edge-weighted graph




$n = 26$ vertices, $m = 38$ edges

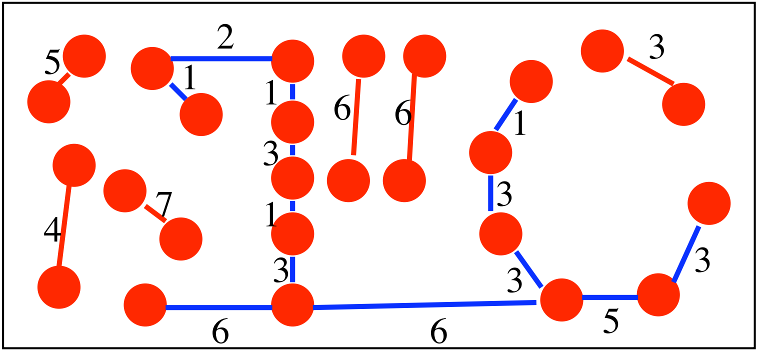
Total weight 120.

# Fast approximation algorithm



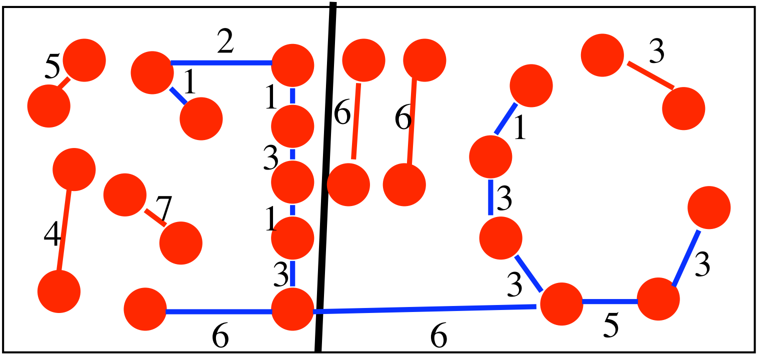

Red edges are dominant

# Fast approximation algorithm

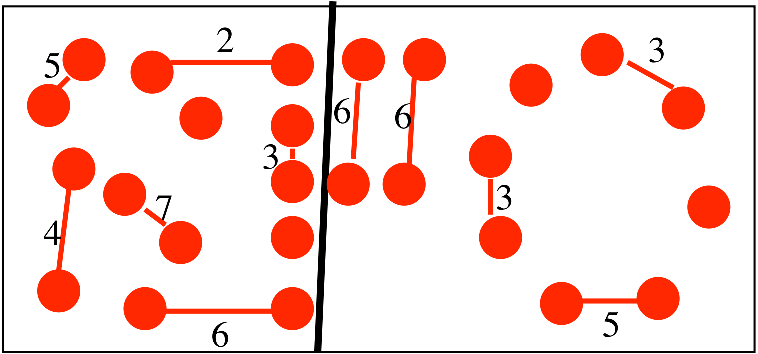



Dominated edges disappear

# Parallel and fast approximation algorithm



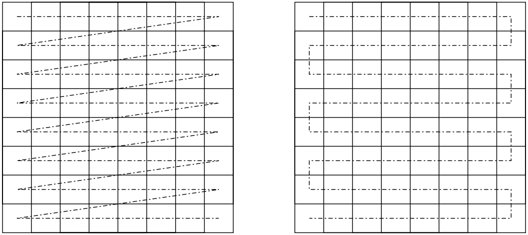
# The solution found





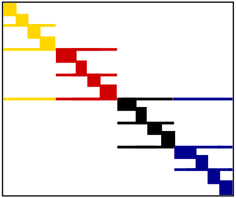

Universiteit Utrecht

# Ordering a sparse matrix to improve cache use



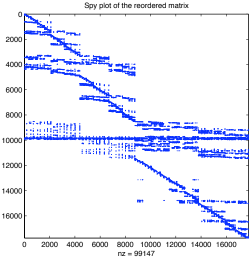
- Compressed Row Storage (CRS, left) and zig-zag CRS (right) orderings.
- Zig-zag CRS avoids unnecessary end-of-row jumps in cache, thus improving access to the input vector in a matrix–vector multiplication.
- Joint work with Albert-Jan Yzelman, *SIAM Journal on Scientific Computing* 2009.

# Separated block-diagonal (SBD) structure



- SBD structure is obtained by recursively partitioning the rows of a sparse matrix, each time moving the cut (mixed) rows to the middle. Columns are permuted accordingly.
- Mondriaan is used in one-dimensional mode, splitting only in the row direction.
- The cut rows are sparse and serve as a gentle transition between accesses to two different vector parts.

# SBD structure for matrix memplus




- Matrix is shown after 100 bipartitionings.
- The recursive, fractal-like nature makes the ordering method work, irrespective of the actual cache characteristics (e.g. sizes of L1, L2, L3 cache).
- The ordering is cache-oblivious.

# Combinatorial problem: try to forget it all

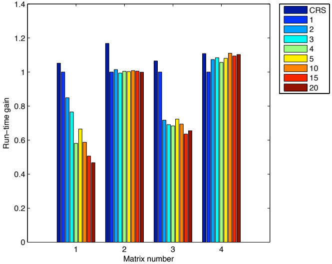
- Ordering the matrix in SBD format makes the matrix-vector multiplication cache-oblivious. Forget about the exact cache hierarchy. It will always work.
- We also like to forget about the cores: core-oblivious. And then processor-oblivious (Wise 2004 at Dagstuhl), node-oblivious, totally oblivious.
- All that is needed is a good ordering of the rows and columns of the matrix, and subsequently of its nonzeros.
- If you cut the nonzeros somewhere, there is hopefully little connection between the two parts.

# Matrix lns3937 (Navier–Stokes, fluid flow)

(Loading movie...take a breath)

Splitting the sparse matrix lns3937 into 5 parts. Film made using MondriaanMovie by Bas Fagginger Auer, part of Mondriaan v3.0, to be released Spring 2010.

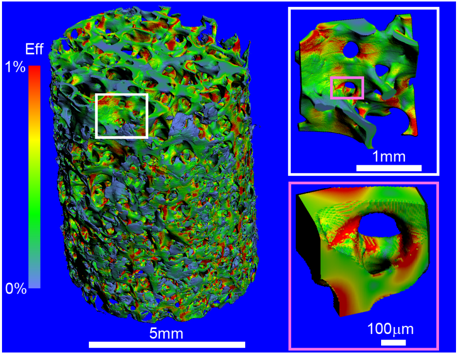
# Wall clock timings on supercomputer Huygens




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`

# Aim: huge computations

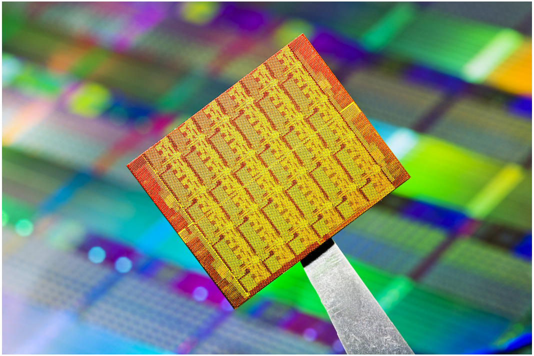



Costas Bekas (IBM Zürich), Peter Arbenz (ETH Zürich), 2008
20 minutes computation on 16384 cores, osteoporosis studies.
Matrix of $1.5 \times 10^9$ rows and columns.
Parallel partitioning is the bottleneck.

# Pictures of a revolution: the guillotine

King Louis XVI of France executed at the Place de la Concorde in Paris, January 23, 1793. Source: http://www.solarnavigator.net/history/french_revolution.htm

# The parallel computing revolution



Intel Single-Chip Cloud computer with 48 cores, announced December 2, 2009. Energy consumption from 25 to 125 Watt, depending on use. Each pair of cores has a variable clock frequency. Source: http://techresearch.intel.com

# Conclusions

- Flop counts become less and less important.
- It's all about restricting movement: moving less data, moving fewer electrons.
- We have presented 3 combinatorial problems: partitioning, matching, ordering. Solution of these is an enabling technology for high-performance computing.
- Reordering is a promising method for oblivious computing. We have shown its utility in enhancing cache performance.

# Future Mondriaan work

- Release 3.0, scheduled Spring 2010.
  - Ordering to SBD and BBD structure: cut rows in the middle, and at the end, respectively
  - Visualisation through Matlab interface and MondriaanMovie
  - Two metrics: $\lambda - 1$ for parallelism, and cut-net for other applications
  - Interface to PaToH hypergraph partitioner