Introduction to Parallel Computation

Lecture 25

Lennart Johnsson
Dept of Computer Science

---

Data Partitioning

Partitioning methods

- Coordinate bi-section
- Inertial bi-section
- Geometric bi-section
- Nodal bi-section
- Spectral bi-section
- Multi-level graph partitioning
- Space filling curves
- Hypergraph partitioning
Coordinate-Free: Spectral Bisection

• Based on theory of Fiedler (1970s), popularized by Pothen, Simon, Liou (1990)
• Motivation, by analogy to a vibrating string
• Basic definitions
• Vibrating string, revisited
• Implementation via the Lanczos Algorithm
  – To optimize sparse-matrix-vector multiply, we graph partition
  – To graph partition, we find an eigenvector of a matrix associated with the graph
  – To find an eigenvector, we do sparse-matrix vector multiply
  – No free lunch ...

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt

Motivation for Spectral Bisection

• Vibrating string
• Think of G = 1D mesh as masses (nodes) connected by springs (edges), i.e. a string that can vibrate
• Vibrating string has modes of vibration, or harmonics
• Label nodes by whether mode - or + to partition into N- and N+
• Same idea for other graphs (eg planar graph ~ trampoline)

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt
Basic Definitions

- **Definition**: The incidence matrix $\text{In}(G)$ of a graph $G(N,E)$ is an $|N|$ by $|E|$ matrix, with one row for each node and one column for each edge. If edge $e=(i,j)$ then column $e$ of $\text{In}(G)$ is zero except for the $i$-th and $j$-th entries, which are +1 and -1, respectively.

- Slightly ambiguous definition because multiplying column $e$ of $\text{In}(G)$ by -1 still satisfies the definition, but this won’t matter...

- **Definition**: The Laplacian matrix $\text{L}(G)$ of a graph $G(N,E)$ is an $|N|$ by $|N|$ symmetric matrix, with one row and column for each node. It is defined by
  - $\text{L}(G)_{i,i} =$ degree of node $i$ (number of incident edges)
  - $\text{L}(G)_{i,j} = -1$ if $i \neq j$ and there is an edge $(i,j)$
  - $\text{L}(G)_{i,j} = 0$ otherwise

Example of $\text{In}(G)$ and $\text{L}(G)$ for Simple Meshes

Nodes numbered in black
Edges numbered in blue

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt
Properties of Laplacian Matrix

- **Theorem 1:** Given G, L(G) has the following properties (proof on UC Berkeley 1996 CS267 web page)
  - L(G) is symmetric.
    - This means the eigenvalues of L(G) are real and its eigenvectors are real and orthogonal.
  - \( \text{In}(G) \ast (\text{In}(G))^T = L(G) \)
  - The eigenvalues of L(G) are nonnegative:
    - \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \)
  - The number of connected components of G is equal to the number of \( \lambda_i \) equal to 0.
  - **Definition:** \( \lambda_2(L(G)) \) is the algebraic connectivity of G
    - The magnitude of \( \lambda_2 \) measures connectivity
    - In particular, \( \lambda_2 \neq 0 \) if and only if G is connected.

Spectral Bisection Algorithm

- **Spectral Bisection Algorithm:**
  - Compute eigenvector \( v_2 \) corresponding to \( \lambda_2(L(G)) \)
  - For each node \( n \) of G
    - if \( v_2(n) < 0 \) put node \( n \) in partition N-
    - else put node \( n \) in partition N+
  - Why does this make sense? First reasons...
    - **Theorem 2 (Fiedler, 1975):** Let G be connected, and N- and N+ defined as above. Then N- is connected. If no \( v_2(n) = 0 \), then N+ is also connected. (proof on 1996 CS267 web page)
    - Recall \( \lambda_2(L(G)) \) is the algebraic connectivity of G
    - **Theorem 3 (Fiedler):** Let \( G_1(N,E_1) \) be a subgraph of \( G(N,E) \), so that \( G_1 \) is “less connected” than G. Then \( \lambda_2(L(G_1)) \leq \lambda_2(L(G)) \), i.e. the algebraic connectivity of \( G_1 \) is less than or equal to the algebraic connectivity of G. (proof on UCB 1996 CS267 web page)
Spectral Bisection Algorithm

- **Spectral Bisection Algorithm:**
  - Compute eigenvector $v_2$ corresponding to $\lambda_2(L(G))$
  - For each node $n$ of $G$
    - if $v_2(n) < 0$ put node $n$ in partition $N^-$
    - else put node $n$ in partition $N^+$

- Why does this make sense? More reasons...
  - **Theorem 4 (Fiedler, 1975):** Let $G$ be connected, and $N_1$ and $N_2$ be any partition into part of equal size $|N|/2$. Then the number of edges connecting $N_1$ and $N_2$ is at least $0.25 * |N| * \lambda_2(L(G))$.
    (proof on 1996 CS267 web page)

Motivation for Spectral Bisection (recap)

- Vibrating string has **modes of vibration**, or **harmonics**
- Modes computable as follows
  - Model string as masses connected by springs (a 1D mesh)
  - Write down $F=ma$ for coupled system, get matrix $A$
  - Eigenvalues and eigenvectors of $A$ are frequencies and shapes of modes
- Label nodes by whether mode - or + to get $N^-$ and $N^+$
- Same idea for other graphs (eg planar graph ~ trampoline)
Details for Vibrating String Analogy

- Force on mass $j = k^*[x(j-1) - x(j)] + k^*[x(j+1) - x(j)]$
  
  $$= -k^*[-x(j-1) + 2x(j) - x(j+1)]$$

- $F=ma$ yields $m^*x''(j) = -k^*[-x(j-1) + 2x(j) - x(j+1)]$ (*)

- Writing (*) for $j=1,2,\ldots,n$ yields

$$\begin{bmatrix}
  x(1) \\
x(2) \\
\vdots \\
x(j) \\
x(n)
\end{bmatrix}
\begin{bmatrix}
  2x(1) - x(2) \\
-2x(1) + 2x(2) - x(3) \\
\vdots \\
-x(j-1) + 2x(j) - x(j+1) \\
2x(n-1) - x(n)
\end{bmatrix}
=-k^*
\begin{bmatrix}
  2 & -1 & & & \\
-1 & 2 & -1 & & \\
& \ddots & \ddots & \ddots & \\
& & -1 & 2 & -1 \\
& & & \ddots & \ddots & -1
\end{bmatrix}
\begin{bmatrix}
x(1) \\
x(2) \\
\vdots \\
x(j) \\
x(n)
\end{bmatrix}$$

$$(-m/k) x'' = L^*x$$

Vibrating Mass Spring System

Details for Vibrating String (continued)

- $-(m/k) x'' = L^*x$, where $x = [x_1,x_2,\ldots,x_n]^T$

- Seek solution of form $x(t) = \sin(\alpha^*t) \cdot x_0$

- $L^*x_0 = (m/k)\alpha^2 \cdot x_0 = \lambda \cdot x_0$

  - For each integer $i$, get $\lambda = 2^*(1-\cos(i\pi/(n+1)))$, $x_0 = \begin{bmatrix}
  \sin(1^i\pi/(n+1)) \\
  \sin(2^i\pi/(n+1)) \\
  \vdots \\
  \sin(n^i\pi/(n+1))
\end{bmatrix}$

  - Thus $x_0$ is a sine curve with frequency proportional to $i$

  - Thus $\alpha^2 = 2^k/m \cdot (1-\cos(i\pi/(n+1)))$ or $\alpha \sim \sqrt{m/k} \cdot \pi i/(n+1)$

- $L = \begin{bmatrix}
  2 & -1 & & & \\
-1 & 2 & -1 & & \\
& \ddots & \ddots & \ddots & \\
& & -1 & 2 & -1 \\
& & & \ddots & \ddots & -1
\end{bmatrix}$

  not quite Laplacian of 1D mesh, but we can fix that ...
Motivation for Spectral Bisection

- Vibrating string has modes of vibration, or harmonics
- Modes computable as follows
  - Model string as masses connected by springs (a 1D mesh)
  - Write down $F=ma$ for coupled system, get matrix $A$
  - Eigenvalues and eigenvectors of $A$ are frequencies and shapes of modes
- Label nodes by whether mode - or + to get $N$- and $N$+
- Same idea for other graphs (e.g., planar graph ~ trampoline)

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt

Eigenvectors of $L$ (1D mesh)

- Eigenvector 1 (all ones)
- Eigenvector 2
- Eigenvector 3

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt
2nd eigenvector of $L$(planar mesh)

4th eigenvector of $L$(planar mesh)
Computing $v_2$ and $\lambda_2$ of $L(G)$ using Lanczos

- Given any $n$-by-$n$ symmetric matrix $A$ (such as $L(G)$), Lanczos computes a $k$-by-$k$ "approximation" $T$ by doing $k$ matrix-vector products, $k \ll n$

Choose an arbitrary starting vector $r$

$b(0) = ||r||$

$j = 0$

repeat

$j = j+1$

$q(j) = r/b(j-1)$  
... scale a vector (BLAS1)

$r = Aq(j)$  
... matrix vector multiplication, the most expensive step

$a(j) = v(j)^T \cdot r$  
... dot product (BLAS1)

$r = r - a(j) \cdot v(j)$  
... "axpy" (BLAS1)

$b(j) = ||r||$  
... compute vector norm (BLAS1)

until convergence  
... details omitted

$$T = \begin{bmatrix}
  a(1) & b(1) & 0 & \cdots & 0 \\
  b(1) & a(2) & b(2) & \cdots & 0 \\
  b(2) & a(3) & b(3) & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  b(k-2) & a(k-1) & b(k-1) & \cdots & a(k) \\
  b(k-1) & a(k) & b(k) & \cdots & 0 \\
\end{bmatrix}$$

- Approximate $A$'s eigenvalues/vectors using $T$'s

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwskay11.ppt

---

A Few Finite Volume/Finite Element Examples

- Edge Based finite volume techniques (Mavriplis, Löhner)
  - → data structures: nodes and edges
- Mixed Finite volume finite element techniques (INRIA)
  - → data structure: cells (control volume)
- Communication is needed to transfer data between data structures
  - → gather operation: get/fetch
  - → scatter operation: send with add
- Proper mappings are needed to minimize communication time

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwskay11.ppt
A Few Finite Volume/Finite Element Examples

- **Objective**: Minimize off-processor data movement by
  1. achieving locality between nodes and elements
  2. avoiding redundant data traffic
- **Data mapping performed in two steps**:  
  1. unstructured mesh is partitioned into element blocks that are mapped to processors  
  2. nodes are mapped onto the processors to achieve maximum locality with the elements  
  3. simple heuristic algorithm to map the nodes

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

Parallel Partitioning

- **Example of mesh partitioning into 8 subdomains**:  
  1. Subdivide in parallel into 2 partitions
  2. Subdivide *in parallel* the 2 partition → leads to 4 partitions
  3. Subdivide *in parallel* the 4 partitions → leads to 8 partitions
  4. ……
- **Two-level parallelization**:  
  - over partitions  
  - over elements in each partition → maximum efficiency since algorithm processes in parallel the same number of data throughout the whole procedure.
Parallel Partitioning – Spectral Bisection

- Computational subtasks for Lanczos eigensolver
  - SAXPY operations
    → executed in parallel using Fortran 90
  - Dot products on each partition
  - Matrix-vector products
  - Eigenvalue analysis
    → performed on the front-end computer

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

A Few Finite Volume/Finite Element Examples

Iterative Solver Steps

1) Gather current solution from nodes to elements
2) Compute element residuals
3) Scatter (assemble) element residuals to nodes
4) Advance solution using an explicit or implicit solver
5) Go to step 1)

Only two data structures: nodes and elements

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Example: Partitioning of Bracket

Mesh with 98,058 tetrahedra
Mesh courtesy of Mark Shepard, RPI

- Partitioning quality for 128 partitions
  - Graph edges: 187,958
  - Edges cut: 14,627
  - Percent cut: 7.8%

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY of HOUSTON
Example: Partitioning of Bracket

<table>
<thead>
<tr>
<th>Partitioning Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of connected blocks</td>
<td>8.4 sec</td>
<td>14.3</td>
</tr>
<tr>
<td>Eigenvector computation</td>
<td>46.4 sec</td>
<td>78.9</td>
</tr>
<tr>
<td>Data ranking/reordering</td>
<td>2.1 sec</td>
<td>3.6</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>1.9 sec</td>
<td>3.2</td>
</tr>
<tr>
<td>Total</td>
<td>58.8 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Algebra Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-vector multiplication (SpMV)</td>
<td>22.1 sec</td>
<td>47.6</td>
</tr>
<tr>
<td>Inner-products (DDOT)</td>
<td>15.8 sec</td>
<td>34.1</td>
</tr>
<tr>
<td>Eigenvalue analysis</td>
<td>3.2 sec</td>
<td>6.9</td>
</tr>
<tr>
<td>SAXPY and miscellaneous</td>
<td>5.3 sec</td>
<td>11.4</td>
</tr>
<tr>
<td>Total</td>
<td>46.4 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY of HOUSTON

Partitioning Example
81,649 tetrahedra.
Mesh courtesy of Mark Shephard, RPI

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY of HOUSTON
Partitioning Example (81,649 tetrahedra)
Mesh courtesy Mark Shepard, RPI

8-way partitioning  16-way partitioning

Example: Partitioning
Partitioning on a 32-node CM-5, double precision

- Partitioning quality for 128 partitions
  - Graph edges: 152,878
  - Edges cut: 10,648
  - Percent cut: 7.0%

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Example: Partitioning

<table>
<thead>
<tr>
<th>Partitioning Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of connected blocks</td>
<td>7.1 sec</td>
<td>22.1</td>
</tr>
<tr>
<td>Eigenvector computation</td>
<td>21.9 sec</td>
<td>68.0</td>
</tr>
<tr>
<td>Data ranking/reordering</td>
<td>1.7 sec</td>
<td>5.3</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>1.5 sec</td>
<td>4.6</td>
</tr>
<tr>
<td>Total</td>
<td>32.2 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Algebra Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-vector multiplication (SpMV)</td>
<td>10.7 sec</td>
<td>48.9</td>
</tr>
<tr>
<td>Inner-products (DDOT)</td>
<td>5.1 sec</td>
<td>23.3</td>
</tr>
<tr>
<td>Eigenvalue analysis</td>
<td>2.0 sec</td>
<td>9.1</td>
</tr>
<tr>
<td>SAXPY and miscellaneous</td>
<td>4.1 sec</td>
<td>18.7</td>
</tr>
<tr>
<td>Total</td>
<td>21.9 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

Example: Partitioning for Falcon Jet Flow

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Falcon Jet

- 19,417 nodes, 109,914 elements, 32-node CM-5
- 1-point integration rule
- Mach 0.85, α=1, inviscid flow, 50 steps, CFL = 50
- Strategy I
  - Random mapping of elements
  - Random mapping of nodes
- Strategy II
  - Recursive Spectral Bisection applied to elements
  - Nodes reordered to achieve locality with elements

<table>
<thead>
<tr>
<th>Operation</th>
<th>Strategy I: No random mapping Time (ms)</th>
<th>Strategy II: Random mapping Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>-</td>
<td>61 sec</td>
</tr>
<tr>
<td>Gather</td>
<td>277</td>
<td>22 sec</td>
</tr>
<tr>
<td>Computation</td>
<td>246</td>
<td>243 sec</td>
</tr>
<tr>
<td>Scatter</td>
<td>323</td>
<td>36 sec</td>
</tr>
<tr>
<td>Total time</td>
<td>14min 6 sec</td>
<td>6 min 02 sec</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

Example: Partitioning of domain around M6 wing

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Example: Partitioning of domain around M6 wing

Mesh courtesy of Rainald Lohner,
266,556 tetrahedra, 48,011 nodes

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY of HOUSTON
Example: Partitioning of domain around M6 wing

Partitioning on a 64-node CM-5, double precision

- Partitioning quality for 256 partitions
  - Graph edges: 527,966
  - Edges cut: 57,063
  - Percent cut: 10.8%

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY of HOUSTON
### Example: Partitioning of domain around M6 wing

<table>
<thead>
<tr>
<th>Partitioning Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification of connected blocks</td>
<td>10.2 sec</td>
<td>13.4</td>
</tr>
<tr>
<td>Eigenvector computation</td>
<td>59.8 sec</td>
<td>78.5</td>
</tr>
<tr>
<td>Data ranking/reordering</td>
<td>3.5 sec</td>
<td>4.6</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>2.7 sec</td>
<td>3.5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>76.2 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

### Linear Algebra Operation Timings Percentage

<table>
<thead>
<tr>
<th>Linear Algebra Operation</th>
<th>Timings</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-vector multiplication (SpMV)</td>
<td>43.9 sec</td>
<td>73.4</td>
</tr>
<tr>
<td>Inner-products (DDOT)</td>
<td>5.2 sec</td>
<td>8.7</td>
</tr>
<tr>
<td>Eigenvalue analysis</td>
<td>3.6 sec</td>
<td>6.0</td>
</tr>
<tr>
<td>SAXPY and miscellaneous</td>
<td>7.1 sec</td>
<td>11.9</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>59.8 sec</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

### Partitioning in Solver Perspective

- M6 wing
  - Mach 0.84, attack angle $\alpha=3.06^\circ$ – Inviscid flow
  - 48,011 nodes, 256,566 tetrahedra
  - 1-point integration rule
  - 100 time steps at CFL=10
  - 64-node CM-5, double precision

<table>
<thead>
<tr>
<th>Operation</th>
<th>Timings</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>106 sec</td>
<td>18.5 MB/s/node</td>
</tr>
<tr>
<td>Gather</td>
<td>57 sec</td>
<td>30 MF/s/node</td>
</tr>
<tr>
<td>Computation</td>
<td>624 sec</td>
<td>15.7 MB/s/node</td>
</tr>
<tr>
<td>Scatter</td>
<td>83 sec</td>
<td>1.6 GF/s</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>12 min 44 sec</td>
<td></td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Partitioning in Solver Perspective

- M6 wing
  - Mach 0.84, attack angle $\alpha=3.06^\circ$ – Inviscid flow
  - 48,011 nodes, 256,566 tetrahedra
  - 1-point integration rule
  - 20 time steps at CFL=5, 80 time steps at CFL=10
  - 64-node CM-5, double precision

<table>
<thead>
<tr>
<th>Operation</th>
<th>CM-5</th>
<th>Cray C90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>123 sec</td>
<td>2532 sec</td>
</tr>
<tr>
<td>Gather</td>
<td>77 sec</td>
<td></td>
</tr>
<tr>
<td>Computation</td>
<td>528 sec</td>
<td>42 min 12 sec</td>
</tr>
<tr>
<td>Scatter</td>
<td>127 sec</td>
<td>0.44 GF/s</td>
</tr>
<tr>
<td>Total</td>
<td>14 min 15 sec</td>
<td>1.5 GF/s</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

Partitioning in Solver Perspective

- Airliner
  - Mach 0.768, attack angle $\alpha=1.116^\circ$ – Inviscid flow
  - 106,064 nodes, 575,986 tetrahedra
  - 1-point integration rule
  - 100 time steps at CFL=5
  - 128-node CM-5, double precision

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (sec)</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>157</td>
<td>17.2 MB/sec/node</td>
</tr>
<tr>
<td>Gather</td>
<td>47</td>
<td>32.6 MF/sec/node</td>
</tr>
<tr>
<td>Computation</td>
<td>439</td>
<td>16.0 MB/sec/node</td>
</tr>
<tr>
<td>Scatter</td>
<td>67</td>
<td>3.3 GF/s</td>
</tr>
<tr>
<td>Total</td>
<td>11 min 50 sec</td>
<td></td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Example: Partitioning of domain for F-18 Flow

- F-18 (Mesh courtesy of Rainald Lohner)
  - Mach 1.5 – Inviscid flow
  - 182,055 nodes, 1,010,174 tetrahedra
  - 1-point integration rule
  - 20 time steps at CFL=5, 80 time steps at CFL=10
  - 64-node CM-5, double precision

<table>
<thead>
<tr>
<th>Partitioning Example</th>
<th>256 processors</th>
<th>512 processors</th>
<th>1024 processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>230 sec</td>
<td>269 sec</td>
<td>150 sec</td>
</tr>
<tr>
<td>Gather</td>
<td>70 sec</td>
<td>46 sec</td>
<td>26 sec</td>
</tr>
<tr>
<td>Computation</td>
<td>623 sec</td>
<td>328 sec</td>
<td>181 sec</td>
</tr>
<tr>
<td>Scatter</td>
<td>107 sec</td>
<td>64 sec</td>
<td>38 sec</td>
</tr>
<tr>
<td>Total time</td>
<td>17 min 10 sec</td>
<td>11 min 47 sec</td>
<td>6 min 35 sec</td>
</tr>
<tr>
<td></td>
<td>5.8 GF/s</td>
<td>10.5 GF/s</td>
<td>18.7 GF/s</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY OF HOUSTON

Partitioning Examples

<table>
<thead>
<tr>
<th>Partitioning Example</th>
<th>No of nodes</th>
<th>No of elements</th>
<th>No of graph edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon Jet</td>
<td>19,417</td>
<td>109,914</td>
<td>217,669</td>
</tr>
<tr>
<td>M6 Wing</td>
<td>48,011</td>
<td>266,556</td>
<td>527,966</td>
</tr>
<tr>
<td>Airliner</td>
<td>106,064</td>
<td>575,986</td>
<td>1,136,029</td>
</tr>
<tr>
<td>F-18</td>
<td>182,055</td>
<td>1,010,174</td>
<td>1,999,646</td>
</tr>
<tr>
<td>M6 Wing - fine</td>
<td>367,723</td>
<td>2,132,448</td>
<td>4,244,312</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

UNIVERSITY OF HOUSTON
### Partitioning Examples - Scalability

<table>
<thead>
<tr>
<th>Partitioning Example</th>
<th>No of processors</th>
<th>No of partitions</th>
<th>No of Lanczos iter., tol. $10^{-3}$</th>
<th>Elapsed time</th>
<th>No of edge cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon Jet</td>
<td>128</td>
<td>128</td>
<td>1,156</td>
<td>44 sec</td>
<td>22,926</td>
</tr>
<tr>
<td>M6 Wing</td>
<td>256</td>
<td>256</td>
<td>1,413</td>
<td>76 sec</td>
<td>57,063</td>
</tr>
<tr>
<td>Airliner</td>
<td>512</td>
<td>512</td>
<td>1,606</td>
<td>124 sec</td>
<td>112,910</td>
</tr>
<tr>
<td>F-18</td>
<td>1,024</td>
<td>1,024</td>
<td>2,001</td>
<td>178 sec</td>
<td>220,413</td>
</tr>
<tr>
<td>M6 Wing - fine</td>
<td>2,048</td>
<td>2,047</td>
<td>2,410</td>
<td>201 sec</td>
<td>481,359</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

*University of Houston*
### Partitioning Examples – Comp. Analysis

Elapsed time for the gather-compute-scatter cycles

<table>
<thead>
<tr>
<th>Partitioning Example</th>
<th>Computation</th>
<th>Gather</th>
<th>Scatter</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon Jet</td>
<td>243 sec</td>
<td>22 sec</td>
<td>36 sec</td>
<td>5 min 01 sec</td>
</tr>
<tr>
<td>M6 Wing</td>
<td>624 sec</td>
<td>57 sec</td>
<td>83 sec</td>
<td>12 min 44 sec</td>
</tr>
<tr>
<td>Airliner</td>
<td>439 sec</td>
<td>47 sec</td>
<td>67 sec</td>
<td>9 min 13 sec</td>
</tr>
<tr>
<td>F-18</td>
<td>579 sec</td>
<td>59 sec</td>
<td>97 sec</td>
<td>12 min 25 sec</td>
</tr>
<tr>
<td>M6 Wing - fine</td>
<td>861 sec</td>
<td>99 sec</td>
<td>142 sec</td>
<td>18 min 22 sec</td>
</tr>
</tbody>
</table>

Performance of the gather-compute-scatter cycles

<table>
<thead>
<tr>
<th>Partitioning Example</th>
<th>Computation MF/s/node</th>
<th>Gather MB/s/node</th>
<th>Scatter MB/s/node</th>
<th>Overall MF/s/node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Falcon Jet</td>
<td>30.4</td>
<td>18.8</td>
<td>14.6</td>
<td>24.5</td>
</tr>
<tr>
<td>M6 Wing</td>
<td>30.0</td>
<td>18.5</td>
<td>15.7</td>
<td>24.5</td>
</tr>
<tr>
<td>Airliner</td>
<td>32.6</td>
<td>17.2</td>
<td>16.0</td>
<td>25.9</td>
</tr>
<tr>
<td>F-18</td>
<td>31.2</td>
<td>14.8</td>
<td>13.0</td>
<td>24.2</td>
</tr>
<tr>
<td>M6 Wing - fine</td>
<td>30.5</td>
<td>16.2</td>
<td>14.0</td>
<td>23.8</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes

---

### Partitioning Examples - Scalability

Source: J. Zdenek, K. Mathur, L. Johnsson, T.J. R. Hughes
Fracture Mechanics - Finite Element Method

Geometry for a Charpy V-notch test with meshes at time=0 and time 600=μsec. Each element is a 20-node brick

Source: A. Needleman, V. Tvergaard, K. Mathur

UNIVERSITY of HOUSTON
Fracture Mechanics - Finite Element Method

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Elements</th>
<th>Equations</th>
<th>Test Case</th>
<th>No of proc.</th>
<th>Overall GF/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36,864</td>
<td>475,875</td>
<td>Charpy</td>
<td>32</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>73,728</td>
<td>946,275</td>
<td>Charpy</td>
<td>64</td>
<td>1.7</td>
</tr>
<tr>
<td>3</td>
<td>147,456</td>
<td>1,846,467</td>
<td>Shear Band</td>
<td>128</td>
<td>3.0</td>
</tr>
<tr>
<td>4</td>
<td>294,912</td>
<td>3,671,619</td>
<td>Shear Band</td>
<td>256</td>
<td>6.3</td>
</tr>
<tr>
<td>5</td>
<td>589,824</td>
<td>7,272,963</td>
<td>Fracture</td>
<td>512</td>
<td>13.0</td>
</tr>
<tr>
<td>6</td>
<td>1,179,648</td>
<td>14,451,827</td>
<td>Fracture</td>
<td>1024</td>
<td>26.6</td>
</tr>
</tbody>
</table>

Source: A. Needleman, V. Tvergaard, K. Mathur

U N I V E R S I T Y o f H O U S T O N
Spectral Bisection: Summary

- Laplacian matrix represents graph connectivity
- Second eigenvector gives a graph bisection
  - Roughly equal "weights" in two parts
  - Weak connection in the graph will be separator
- Implementation via the Lanczos Algorithm
  - To optimize sparse-matrix-vector multiply, we graph partition
  - To graph partition, we find an eigenvector of a matrix associated with the graph
  - To find an eigenvector, we do sparse-matrix vector multiply

- Have we made progress?
  - The first matrix-vector multiplies are slow, but use them to learn how to make the rest faster

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdsary11.ppt

Geometric Partitioning – Test cases

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Spectral</th>
<th>Coordinate Bisection</th>
<th>Default Geometric</th>
<th>Best Geometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tapir</td>
<td>59</td>
<td>55</td>
<td>37</td>
<td>32</td>
</tr>
<tr>
<td>Airfoil2</td>
<td>117</td>
<td>172</td>
<td>100</td>
<td>93</td>
</tr>
<tr>
<td>Triangle</td>
<td>154</td>
<td>142</td>
<td>144</td>
<td>142</td>
</tr>
<tr>
<td>Airfoil3</td>
<td>174</td>
<td>230</td>
<td>152</td>
<td>148</td>
</tr>
<tr>
<td>PWT</td>
<td>362</td>
<td>562</td>
<td>529</td>
<td>499</td>
</tr>
<tr>
<td>Body</td>
<td>456</td>
<td>953</td>
<td>834</td>
<td>768</td>
</tr>
<tr>
<td>Wave</td>
<td>13,706</td>
<td>9,821</td>
<td>10,377</td>
<td>9,773</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Spectral</th>
<th>Coordinate Bisection</th>
<th>Default Geometric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tapir</td>
<td>1,278</td>
<td>1,387</td>
<td>1,239</td>
</tr>
<tr>
<td>Airfoil2</td>
<td>2,826</td>
<td>3,271</td>
<td>2,709</td>
</tr>
<tr>
<td>Triangle</td>
<td>2,989</td>
<td>2,907</td>
<td>2,912</td>
</tr>
<tr>
<td>Airfoil3</td>
<td>4,893</td>
<td>6,131</td>
<td>4,822</td>
</tr>
<tr>
<td>PWT</td>
<td>13,495</td>
<td>14,220</td>
<td>13,769</td>
</tr>
<tr>
<td>Body</td>
<td>12,077</td>
<td>22,497</td>
<td>19,905</td>
</tr>
<tr>
<td>Wave</td>
<td>143,015</td>
<td>162,833</td>
<td>145,155</td>
</tr>
</tbody>
</table>

http://www.cs.cmu.edu/~glmiller/www/Publications/GiMiTe98.pdf
Graph Partitioning

Geometric – Spheres, Random cuts

Source: J. Zdenek, K. Mathur, S.-H. Teng, L. Johnsson, T.J.R. Hughes

Graph Partitioning

<table>
<thead>
<tr>
<th>No of trials</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of cuts</td>
<td>23,136</td>
<td>20,819</td>
<td>20,841</td>
<td>19,909</td>
<td>19,335</td>
</tr>
<tr>
<td>Timings</td>
<td>7.7 sec</td>
<td>20.6 sec</td>
<td>32.1 sec</td>
<td>56.3 sec</td>
<td>133.8 sec</td>
</tr>
</tbody>
</table>

Source: J. Zdenek, K. Mathur, S.-H. Teng, L. Johnsson, T.J.R. Hughes

Graph Partitioning

Spectral bi-section

Source: J. Zdenek, K. Mathur, S.-H. Teng, L. Johnsson, T.J.R. Hughes
Data Partitioning

Partitioning methods

- Coordinate bi-section
- Geometric bi-section
- Inertial bi-section
- Nodal bi-section
- Spectral bi-section
- Multi-level graph partitioning
- Space filling curves
- Hypergraph partitioning

Multi-level Partitioning

http://bmi.osu.edu/resources/presentations/IPDP07-Catalyurek-DynamicLB.ppt
Multilevel Partitioning

- If we want to partition $G(N,E)$, but it is too big to do efficiently, what can we do?
  - 1) Replace $G(N,E)$ by a coarse approximation $G_c(N_c,E_c)$, and partition $G_c$ instead
  - 2) Use partition of $G_c$ to get a rough partitioning of $G$, and then iteratively improve it

- What if $G_c$ still too big?
  - Apply same idea recursively

---

Multilevel Partitioning - High Level Algorithm

```plaintext
(N+, N-) = Multilevel_Partition( N, E )

... recursive partitioning routine returns N+ and N- where N = N+ U N-

if |N| is small
  (1) Partition G = (N,E) directly to get N = N+ U N-
      Return (N+, N-)
else
  (2) Coarsen G to get an approximation G_c = (N_c, E_c)
  (3) (N_c+, N_c-) = Multilevel_Partition( N_c, E_c )
  (4) Expand (N_c+, N_c-) to a partition (N+, N-) of N
  (5) Improve the partition (N+, N-)
      Return (N+, N-)
endif
```

How do we
Coarsen?
Expand?
Improve?
Multilevel Kernighan-Lin

- Coarsen graph and expand partition using maximal matchings
- Improve partition using Kernighan-Lin

Maximal Matching

- **Definition**: A matching of a graph $G(N,E)$ is a subset $E_m$ of $E$ such that no two edges in $E_m$ share an endpoint
- **Definition**: A maximal matching of a graph $G(N,E)$ is a matching $E_m$ to which no more edges can be added and remain a matching
- A simple greedy algorithm computes a maximal matching:
  
  let $E_m$ be empty
  mark all nodes in $N$ as unmatched
  for $i = 1$ to $|N|$ ... visit the nodes in any order
    if $i$ has not been matched
      mark $i$ as matched
      if there is an edge $e=(i,j)$ where $j$ is also unmatched,
        add $e$ to $E_m$
      mark $j$ as matched
    endif
  endfor
Maximal Matching: Example

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt

Example of
How to coarsen a graph using a maximal matching

\[ G = (N, E) \]
\[ E_m \text{ is shown in red} \]
\[ \text{Edge weights shown in blue} \]
\[ \text{Node weights are all one} \]

\[ G_c = (N_c, E_c) \]
\[ N_c \text{ is shown in red} \]
\[ \text{Edge weights shown in blue} \]
\[ \text{Node weights shown in black} \]

http://www.cs.berkeley.edu/~demmel/cs267_Spr11/Lectures/lecture13_partition_jwdkay11.ppt
Expanding a partition of $G_c$ to a partition of $G$

Converting a coarse partition to a fine partition

Some Graphs for Partitioning Studies

<table>
<thead>
<tr>
<th>Matrix Name</th>
<th>Order</th>
<th>Nonzeros</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCSSTK28 (BC28)</td>
<td>4,410</td>
<td>107,307</td>
<td>Solid element model</td>
</tr>
<tr>
<td>BCSSTK29 (BC29)</td>
<td>13,992</td>
<td>302,748</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>BCSSTK30 (BC30)</td>
<td>28,204</td>
<td>1,007,284</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>BCSSTK31 (BC31)</td>
<td>35,588</td>
<td>572,914</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>BCSSTK32 (BC32)</td>
<td>44,609</td>
<td>985,046</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>BCSSTK33 (BC33)</td>
<td>8,736</td>
<td>201,383</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>BCSPPRT10 (BSP10)</td>
<td>5,303</td>
<td>8,271</td>
<td>Ethernet US power network</td>
</tr>
<tr>
<td>BRACK2 (BRCK)</td>
<td>62,631</td>
<td>366,559</td>
<td>3D Finite element mesh</td>
</tr>
<tr>
<td>CANT (CANT)</td>
<td>54,195</td>
<td>1,900,797</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>COPPER2 (COP2)</td>
<td>55,416</td>
<td>352,238</td>
<td>3D Finite element mesh</td>
</tr>
<tr>
<td>CYLINDER93 (CY93)</td>
<td>45,594</td>
<td>1,766,726</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>FINAN512 (FINC)</td>
<td>74,752</td>
<td>335,872</td>
<td>Linear programming</td>
</tr>
<tr>
<td>4ELT (4ELT)</td>
<td>15,606</td>
<td>45,878</td>
<td>2D Finite element mesh</td>
</tr>
<tr>
<td>INPROT (INPR)</td>
<td>46,949</td>
<td>1,177,809</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>LSPP (LHR)</td>
<td>70,304</td>
<td>1,208,902</td>
<td>3D Coefficient matrix</td>
</tr>
<tr>
<td>LSHP1466 (LS34)</td>
<td>3,466</td>
<td>10,215</td>
<td>Graded L-shape pattern</td>
</tr>
<tr>
<td>MAP (MAP)</td>
<td>267,241</td>
<td>937,103</td>
<td>Highway network</td>
</tr>
<tr>
<td>MEMPLUS (MEM)</td>
<td>17,758</td>
<td>126,150</td>
<td>Memory circuit</td>
</tr>
<tr>
<td>ROTOR (ROTR)</td>
<td>99,617</td>
<td>662,431</td>
<td>3D Finite element mesh</td>
</tr>
<tr>
<td>S3854 (S33)</td>
<td>22,143</td>
<td>9,339</td>
<td>Sequential circuit</td>
</tr>
<tr>
<td>SHELL93 (SHEL)</td>
<td>181,200</td>
<td>2,313,765</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>SHY161 (SHY1)</td>
<td>76,480</td>
<td>329,762</td>
<td>CFD/Navier-Stokes</td>
</tr>
<tr>
<td>YRCC (YROC)</td>
<td>212,453</td>
<td>5,885,929</td>
<td>3D Stiffness matrix</td>
</tr>
<tr>
<td>WAVE (WAVE)</td>
<td>156,317</td>
<td>1,959,331</td>
<td>3D Finite element mesh</td>
</tr>
</tbody>
</table>

Source: G Karypis, V Kumar, Multi-level Graph Partitioning Schemes, Proc. 24th ICPP., 1995
Matching Strategies

Edge-Cut for 32-way partitioning, no refinement

<table>
<thead>
<tr>
<th></th>
<th>RM</th>
<th>HEM</th>
<th>LEM</th>
<th>HCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCSSTK31</td>
<td>14,489</td>
<td>84,024</td>
<td>412,361</td>
<td>115,471</td>
</tr>
<tr>
<td>BCSSTK32</td>
<td>184,236</td>
<td>148,637</td>
<td>680,637</td>
<td>153,945</td>
</tr>
<tr>
<td>BRACK2</td>
<td>75,832</td>
<td>53,115</td>
<td>187,688</td>
<td>69,370</td>
</tr>
<tr>
<td>CANT</td>
<td>817,500</td>
<td>487,543</td>
<td>1,633,878</td>
<td>521,417</td>
</tr>
<tr>
<td>COPTER2</td>
<td>69,184</td>
<td>59,135</td>
<td>208,318</td>
<td>59,631</td>
</tr>
<tr>
<td>CYLINDER93</td>
<td>522,619</td>
<td>286,901</td>
<td>1,473,731</td>
<td>354,154</td>
</tr>
<tr>
<td>4ELT</td>
<td>3,874</td>
<td>3,036</td>
<td>4,410</td>
<td>4,025</td>
</tr>
<tr>
<td>INPRO1</td>
<td>205,525</td>
<td>187,482</td>
<td>821,233</td>
<td>141,398</td>
</tr>
<tr>
<td>Rotor</td>
<td>147,971</td>
<td>110,988</td>
<td>424,359</td>
<td>98,530</td>
</tr>
<tr>
<td>SHELL93</td>
<td>373,028</td>
<td>237,212</td>
<td>1,443,868</td>
<td>258,689</td>
</tr>
<tr>
<td>TROLL</td>
<td>1,095,607</td>
<td>806,810</td>
<td>4,941,507</td>
<td>883,002</td>
</tr>
<tr>
<td>WAVE</td>
<td>239,090</td>
<td>212,742</td>
<td>745,495</td>
<td>192,729</td>
</tr>
</tbody>
</table>

RM = Random Matching
HEM = Heavy-Edge Matching (visit nodes randomly, select neighbor with max edge weight)
LEM = Length-Edge Matching (visit nodes randomly, select neighbor with least edge weight)
HCM = Heavy Clique Matching (match based on edge density forming multi-nodes)

Source: G. Karypis, V. Kumar, Multi-level Graph Partitioning Schemes, Proc. 24th ICPP, 1995

---

Performance of Refinement Strategies

<table>
<thead>
<tr>
<th></th>
<th>GR</th>
<th>KLR</th>
<th>BGR</th>
<th>BKLR</th>
<th>BKLGR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32EC RTtime</td>
<td>32EC RTtime</td>
<td>32EC RTtime</td>
<td>32EC RTtime</td>
<td>32EC RTtime</td>
</tr>
<tr>
<td>BCSSTK31</td>
<td>45,267</td>
<td>1.09</td>
<td>46,852</td>
<td>2.33</td>
<td>46,281</td>
</tr>
<tr>
<td>BCSSTK32</td>
<td>66,336</td>
<td>1.39</td>
<td>71,091</td>
<td>2.89</td>
<td>72,048</td>
</tr>
<tr>
<td>BRACK2</td>
<td>22,451</td>
<td>2.04</td>
<td>20,720</td>
<td>4.92</td>
<td>20,786</td>
</tr>
<tr>
<td>CANT</td>
<td>323.4K</td>
<td>3.30</td>
<td>320.5K</td>
<td>6.82</td>
<td>325.0K</td>
</tr>
<tr>
<td>COPTER2</td>
<td>31,338</td>
<td>2.24</td>
<td>31,215</td>
<td>5.42</td>
<td>32,064</td>
</tr>
<tr>
<td>CYLINDER93</td>
<td>201.0K</td>
<td>1.95</td>
<td>200.0K</td>
<td>4.32</td>
<td>199.0K</td>
</tr>
<tr>
<td>4ELT</td>
<td>1,834</td>
<td>0.44</td>
<td>1,833</td>
<td>0.96</td>
<td>2,028</td>
</tr>
<tr>
<td>INPRO1</td>
<td>75,676</td>
<td>1.28</td>
<td>75,911</td>
<td>3.41</td>
<td>76,315</td>
</tr>
<tr>
<td>Rotor</td>
<td>38,214</td>
<td>4.98</td>
<td>38,312</td>
<td>13.09</td>
<td>36,834</td>
</tr>
<tr>
<td>SHELL93</td>
<td>91,723</td>
<td>9.27</td>
<td>79,523</td>
<td>52.4</td>
<td>84,123</td>
</tr>
<tr>
<td>TROLL</td>
<td>317.5K</td>
<td>9.55</td>
<td>309.7K</td>
<td>27.4</td>
<td>314.2K</td>
</tr>
<tr>
<td>WAVE</td>
<td>74,486</td>
<td>8.72</td>
<td>72,343</td>
<td>19.36</td>
<td>71,941</td>
</tr>
</tbody>
</table>

HEM for coarsening
32EC = Edges cut for 32 partitions, RTtime = Refinement Time
GR = Greedy Refinement (one iteration of KLR)
KLR = Kernighan-Lin Refinement
BGR = Boundary Greedy Refinement (only consider boundary nodes in greedy refinement)
BKLR = Boundary Kernighan-Lin Refinement (only consider boundary nodes in KLR refinement)
BKLGR = Boundary Kernighan-Lin Greedy Refinement (BKLGR for coarse graphs, BGR for fine)

Source: G Karypis, V Kumar, Multi-level Graph Partitioning Schemes, Proc. 24th ICPP, 1995
### Performance of Matching Strategies for BKLGR

<table>
<thead>
<tr>
<th></th>
<th>RM</th>
<th>HEM</th>
<th>LEM</th>
<th>HCM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32EC</td>
<td>CTime</td>
<td>UTime</td>
<td>32EC</td>
</tr>
<tr>
<td>BCSSTK31</td>
<td>44,810</td>
<td>5.93</td>
<td>2.46</td>
<td>45,991</td>
</tr>
<tr>
<td>BCSSTK32</td>
<td>71,416</td>
<td>9.21</td>
<td>2.91</td>
<td>69,361</td>
</tr>
<tr>
<td>BRACK2</td>
<td>20,693</td>
<td>6.86</td>
<td>3.41</td>
<td>21,152</td>
</tr>
<tr>
<td>CANT</td>
<td>323.0K</td>
<td>20.34</td>
<td>8.99</td>
<td>323.0K</td>
</tr>
<tr>
<td>CLOPTER2</td>
<td>32,330</td>
<td>5.18</td>
<td>2.95</td>
<td>30,938</td>
</tr>
<tr>
<td>CYLINDER93</td>
<td>198.0K</td>
<td>16.49</td>
<td>5.25</td>
<td>198.0K</td>
</tr>
<tr>
<td>4ELT</td>
<td>1.826</td>
<td>0.82</td>
<td>0.76</td>
<td>1.894</td>
</tr>
<tr>
<td>INPRO1</td>
<td>78,375</td>
<td>10.4</td>
<td>2.9</td>
<td>75,203</td>
</tr>
<tr>
<td>ROTOR</td>
<td>38,723</td>
<td>12.94</td>
<td>5.6</td>
<td>36,512</td>
</tr>
<tr>
<td>SHELL93</td>
<td>84,523</td>
<td>35.18</td>
<td>10.24</td>
<td>81,756</td>
</tr>
<tr>
<td>TROLL</td>
<td>317.4K</td>
<td>67.75</td>
<td>14.16</td>
<td>307.0K</td>
</tr>
</tbody>
</table>

32EC = edges cut for 32 partitions, CTime = Coarsening Time, UTime = time for partitioning coarse graph, refinement and projection to next level finer graph
RM = Random Matching
HEM = Heavy-Edge Matching (visit nodes randomly, select neighbor with max edge weight)
LEM = Light-Edge Matching (visit nodes randomly, select neighbor with least edge weight)
HCM = Heavy Clique Matching (match based on density)

Source: G Karypis, V Kumar, Multi-level Graph Partitioning Schemes, Proc. 24th ICPP., 1995
Partitioning Software

- JOSTLE, http://staffweb.cms.gre.ac.uk/~c.walshaw/jostle
- METIS, http://glaros.dtc.umn.edu/gkhome/views/metis
- ParMETIS, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview
- SCOTCH, http://www.labri.u-bordeaux.fr/perso/pelegrin/scotch

References

References (cont’d)

- **SPECTRAL BISECTION**
- A multilevel algorithm for partitioning graphs, B. Hendrickson, R. Leland, in Proc. of Supercomputing ’95, ACM, 1995 http://dl.acm.org/citation.cfm?id=921398
- A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm, G. Karypis, V. Kumar, 8th SIAM Conference on Parallel Processing for Scientific Computing, 1997 http://glaros.dtc.umn.edu/gkhome/node/88
- Multilevel graph partitioning schemes, G. Karypis, V. Kumar, ICPP 95, glaros.dtc.umn.edu/gkhome/node/papers/wlKPP95.pdf
- METIS
- Analysis of Multilevel Graph Partitioning, G. Karypis, V. Kumar, Supercomputing ’95, http://glaros.dtc.umn.edu/gkhome/node/86
- A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm, G. Karypis, V. Kumar, 8th SIAM Conference on Parallel Processing for Scientific Computing, 1997 http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970011904_1997011816.pdf
- Multilevel graph partitioning schemes, G. Karypis, V. Kumar, ICPP 95, glaros.dtc.umn.edu/gkhome/node/papers/wlKPP95.pdf
- SPACE FILLING CURVES
- Sur une courbe, qui remplit toute une aire plane, G. Poincaré, Mathematische Annalen, vol. 36, no. 1, pp 157 – 160, 1890
- DOI:10.1007/BF01199431, http://www.springerlink.com/content/w232301n53960133
- DOI:10.1007/BF01199431, http://www.springerlink.com/content/w232301n53960133
- Paralel Multilevel k-way Partitioning Scheme for Irregular Graphs, G. Karypis, V. Kumar, in Proc. of Supercomputing ‘95, ACM, 1995 http://dl.acm.org/citation.cfm?id=921398
References (cont’d)

- SPACE FILLING CURVES
  - Using Space-filling Curves for Computation Reordering, Guohua Jin, J. Mellor-Crummey, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.65.1757

- MISC

- MESHES
  - http://www2.cs.uni-paderborn.de/cs/eg-berlin/RESEARCH/PART/Graphs.html
  - http://staffweb.cms.qe.ac.uk/~wo06/partition