Overview of MUMPS (A multifrontal Massively Parallel Solver)

E. Agullo, P. Amestoy, A. Buttari, P. Combes, A. Guermouche, J.-Y. L’Excellent, T. Slavova, B. Uçar

CERFACS, ENSEEIHT-IRIT, INRIA (LIP and LaBRI),
(http://mumps.enseeiht.fr or http://graal.ens-lyon.fr/MUMPS)

1. Introduction-History

2. Sparse Direct Methods and preprocessing

3. Parallel sparse solver - Memory issues

4. Conclusion-On going work
Objective: Solve $Ax = b$, where $A$ is large and sparse.

At the beginning: LTR (Long Term Research) European project, from 1996 to 1999

Led to first public domain version

Now:
- MUMPS is supported by CERFACS (Toulouse), ENSEEIHT-IRIT (Toulouse) and INRIA (Lyon, Bordeaux).
- Public Domain (available free of charge)
- 300,000 lines of F90/C/MPI code
- ~1000 users, 3 requests per day
Current development team:
- Patrick Amestoy, ENSEEIHT-IRIT (Toulouse)
- Alfredo Buttari, INRIA Lyon
- Philippe Combes, CNRS/ENS Lyon
- Abdou Guermouche, LaBRI-INRIA (Bordeaux)
- Jean-Yves L’Excellent, INRIA
- Bora Uçar, CERFACS

Phd Students
- Emmanuel Agullo, ENS-Lyon
- Tzvetomila Slavova, CERFACS (Toulouse).
Overview of MUMPS (A multifrontal Massively Parallel Solver)
Outline

1. Introduction-History

2. Sparse Direct Methods and preprocessing

3. Parallel sparse solver - Memory issues

4. Conclusion-On going work
MUMPS solves large systems of linear equations of the form $Ax=b$ by factorizing $A$ into $A=LU$ or $LDLT$. It uses a multifrontal technique which is a direct method.

- 3 main steps (plus initialization and termination):

![Diagram showing the steps: JOB = -1 → ANALYSIS JOB = 1 → FACTORIZATION JOB = 2 → SOLVE JOB = 3 → JOB = -2]
MUMPS solves large systems of linear equations of the form $Ax=b$ by factorizing $A$ into $A=LU$ or $LDLT$. It uses a multifrontal technique which is a direct method.

- 3 main steps (plus initialization and termination):

- $JOB=-1$: initialize solver type ($LU$, $LDLT$) and default parameters
MUMPS solves large systems of linear equations of the form $Ax=b$ by factorizing $A$ into $A=LU$ or $LDLT$. It uses a multifrontal technique which is a direct method.

- **3 main steps (plus initialization and termination):**

  - **JOB = −1**: analyse the matrix, build an ordering, prepare factorization
  - **JOB = 1**: analysis
  - **JOB = 2**: factorization
  - **JOB = 3**: solve
  - **JOB = −2**: initialization and termination
MUMPS solves large systems of linear equations of the form $Ax=b$ by factorizing $A$ into $A=LU$ or $LDLT$. It uses a multifrontal technique which is a direct method.

- 3 main steps (plus initialization and termination):

  - JOB = 2: (parallel) numerical factorization
  - $A = LU$
MUMPS solves large systems of linear equations of the form $Ax=b$ by factorizing $A$ into $A=LU$ or $LDLT$. It uses a multifrontal technique which is a direct method.

- **3 main steps (plus initialization and termination)**:

  \[ \text{JOB} = -1 \rightarrow \text{ANALYSIS JOB = 1} \rightarrow \text{FACTORIZATION JOB = 2} \rightarrow \text{SOLVE JOB = 3} \rightarrow \text{JOB = -2} \]

- **JOB = 3**: (parallel) solution step forward and backward substitutions \( (Ly = b, Ux = y) \)
MUMPS solves large systems of linear equations of the form \( Ax = b \) by factorizing \( A \) into \( A = LU \) or \( LDL^T \). It uses a multifrontal technique which is a direct method.

- 3 main steps (plus initialization and termination):

\[
\text{JOB} = -2 : \text{termination} \\
\text{deallocate all MUMPS data structures}
\]
Preprocessing and postprocessing:

- Symmetric permutations to reduce fill: \(Ax = b \rightarrow PAP^tPx = b\)
- Numerical pivoting, scaling to preserve numerical accuracy
- Maximum transversal (set large entries on the diagonal)
- Preprocessing for parallelism (influence of task mapping on parallelism)
- Iterative refinement, error analysis

Default (often automatic/adaptive) setting of the options is available. However, a better knowledge of the options can help the user to further improve:

- memory usage,
- time for solution,
- numerical accuracy.
Preprocessing - illustration

Original ($A = \text{LHR01}$)  Preprocessed matrix ($A'(\text{LHR01})$)

Modified Problem: $A'x' = b'$ with $A' = P_{n}D_{r}PAQP^{t}D_{c}$
### Impact of fill-reducing heuristics

<table>
<thead>
<tr>
<th></th>
<th>METIS</th>
<th>SCOTCH</th>
<th>PORD</th>
<th>AMF</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GUPTA2</strong></td>
<td>2757.8</td>
<td>4510.7</td>
<td>4993.3</td>
<td>2790.3</td>
<td>2663.9</td>
</tr>
<tr>
<td><strong>SHIP_003</strong></td>
<td><strong>83828.2</strong></td>
<td>92614.0</td>
<td>112519.6</td>
<td>96445.2</td>
<td>155725.5</td>
</tr>
<tr>
<td><strong>TWOTONE</strong></td>
<td>29120.3</td>
<td><strong>27764.7</strong></td>
<td>37167.4</td>
<td>29847.5</td>
<td>29552.9</td>
</tr>
<tr>
<td><strong>WANG3</strong></td>
<td>4313.1</td>
<td>5801.7</td>
<td>5009.9</td>
<td>6318.0</td>
<td>10492.2</td>
</tr>
<tr>
<td><strong>XENON2</strong></td>
<td><strong>99273.1</strong></td>
<td>112213.4</td>
<td>126349.7</td>
<td>237451.3</td>
<td>298363.5</td>
</tr>
</tbody>
</table>

E. Agullo, P. Amestoy, A. Buttari, P. Combes, A. Guermouche, J.-Y. L'Excellent, T. Slavova, B. Ucar

Overview of MUMPS (A multifrontal Massively Parallel Solver)
Memory is divided into two parts (that can overlap in time):

- the factors
- the active memory

Elimination tree represents tasks dependencies
### Impact of fill-reducing heuristics

**Peak of active memory for multifrontal approach \((\times 10^6 \text{ entries})\)**

<table>
<thead>
<tr>
<th></th>
<th>METIS</th>
<th>SCOTCH</th>
<th>PORD</th>
<th>AMF</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUPTA2</td>
<td>58.33</td>
<td>289.67</td>
<td>78.13</td>
<td><strong>33.61</strong></td>
<td>52.09</td>
</tr>
<tr>
<td>SHIP_003</td>
<td>25.09</td>
<td>23.06</td>
<td>20.86</td>
<td><strong>20.77</strong></td>
<td>32.02</td>
</tr>
<tr>
<td>TWOTONE</td>
<td>13.24</td>
<td>13.54</td>
<td>11.80</td>
<td><strong>11.63</strong></td>
<td>17.59</td>
</tr>
<tr>
<td>WANG3</td>
<td>3.28</td>
<td>3.84</td>
<td><strong>2.75</strong></td>
<td>3.62</td>
<td>6.14</td>
</tr>
<tr>
<td>XENON2</td>
<td>14.89</td>
<td>15.21</td>
<td><strong>13.14</strong></td>
<td>23.82</td>
<td>37.82</td>
</tr>
</tbody>
</table>

---

E. Agullo, P. Amestoy, A. Buttari, P. Combes, A. Guermouche, J.-Y. L'Excellent, T. Slavova, B. Ucar
## Time for factorization (seconds)

<table>
<thead>
<tr>
<th></th>
<th>1p</th>
<th>16p</th>
<th>32p</th>
<th>64p</th>
<th>128p</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONESHL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>METIS</td>
<td>970</td>
<td>60</td>
<td>41</td>
<td>27</td>
<td>14</td>
</tr>
<tr>
<td>PORD</td>
<td>1264</td>
<td>104</td>
<td>67</td>
<td>41</td>
<td>26</td>
</tr>
<tr>
<td><strong>AUDI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>METIS</td>
<td>2640</td>
<td>198</td>
<td>108</td>
<td>70</td>
<td>42</td>
</tr>
<tr>
<td>PORD</td>
<td>1599</td>
<td>146</td>
<td>83</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>
Sparse Direct Methods and preprocessing

Preprocessing - Maximum weighted matching (I)

- **Objective**: Set large entries on the diagonal
  - Unsymmetric permutation and scaling
  - Preprocessed matrix $B = D_1 AQ D_2$
    is such that $|b_{ii}| = 1$ and $|b_{ij}| \leq 1$

Original ($A = lhr01$)  
Permutated ($A' = AQ$)
Influence of maximum weighted matching on the performance

| Matrix     | Symmetry | \(|LU| (10^6)| Flops (10^9) | Backwd Error |
|------------|----------|-------------|--------------|---------------|
| TWOTONE    | OFF      | 28          | 235          | 1221          |
|            | ON       | 43          | 22           | 29            |
| FIDAPM11   | OFF      | 100         | 16           | 10            |
|            | ON       | 46          | 28           | 29            |

On very unsymmetric matrices: reduce flops, factor size and memory used.

In general improve accuracy, and reduce number of iterative refinements.

Improve reliability of memory estimates.
**Influence of maximum weighted matching on the performance**

| Matrix          | Symmetry | $|LU|$ (10$^6$) | Flops (10$^9$) | Backwd Error |
|-----------------|----------|---------------|----------------|---------------|
| TWOTONE         | OFF      | 28            | 235            | 1221          | $10^{-6}$    |
|                 | ON       | 43            | 22             | 29            | $10^{-12}$   |
| FIDAPM11        | OFF      | 100           | 16             | 10            | $10^{-10}$   |
|                 | ON       | 46            | 28             | 29            | $10^{-11}$   |

- On very unsymmetric matrices: reduce flops, factor size and memory used.
- In general improve accuracy, and reduce number of iterative refinements.
- Improve reliability of memory estimates.
Perform an unsymmetric weighted matching
Preprocessing symmetric matrices - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries

Matched entry

Selected Matched entry

E. Agullo, P. Amestoy, A. Buttari, P. Combes, A. Guermouche, J.-Y. L'Excellent, T. Slavova, B. Uçar

Overview of MUMPS (A multifrontal Massively Parallel Solver)
Preprocessing symmetric matrices - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal

\[ B = Q^t A Q \]

Selected entries

E. Agullo, P. Amestoy, A. Buttari, P. Combes, A. Guermouche, J.-Y. L'Excellent, T. Slavova, B. Ucar
Preprocessing symmetric matrices - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal
- Compression: $2 \times 2$ diagonal blocks become supervariables.

![Diagram of matrix preprocessing and compression](image-url)
Preprocessing symmetric matrices - Compressed ordering

- Perform an unsymmetric weighted matching
- Select matched entries
- Symmetrically permute matrix to set large entries near diagonal
- Compression: $2 \times 2$ diagonal blocks become supervariabales.

Influence of using a compressed graph (with scaling)

<table>
<thead>
<tr>
<th>Compression :</th>
<th>Total time (seconds)</th>
<th>Nb of entries in factors in Millions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>CONT-300</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>CVXQP3</td>
<td>28</td>
<td>11</td>
</tr>
<tr>
<td>STOKES128</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Outline

1. Introduction-History

2. Sparse Direct Methods and preprocessing

3. Parallel sparse solver - Memory issues

4. Conclusion-On going work
Out-of-core

Solving sparse linear systems

\[ Ax = b : 10 \text{ M variables} \]
\[ \Rightarrow A = LU \quad (\text{Direct methods}) \]

Current limits: BRGM matrix

- \( 3.7 \times 10^6 \) variables
- \( 156 \times 10^6 \) non zeros in \( A \)
- \( 4.5 \times 10^9 \) non zeros in \( LU \)
- \( 26.5 \times 10^{12} \) flops

Physical constraint
Solving sparse linear systems

\[ Ax = b : 10 \text{ M variables} \]
\[ \Rightarrow A = LU \text{ (Direct methods)} \]

Current limits: BRGM matrix

- 3.7 \times 10^6 \text{ variables}
- 156 \times 10^6 \text{ non zeros in } A
- 4.5 \times 10^9 \text{ non zeros in } LU
- 26.5 \times 10^{12} \text{ flops}

Physical constraint

Core memory

Memory required

Memory crash
Solving sparse linear systems

\[ Ax = b : 10 \text{ M variables} \]

\[ \Rightarrow A = LU \text{ (Direct methods)} \]

Current limits: BRGM matrix

- \( 3.7 \times 10^6 \) variables
- \( 156 \times 10^6 \) non zeros in A
- \( 4.5 \times 10^9 \) non zeros in LU
- \( 26.5 \times 10^{12} \) flops

Out-of-core

- Core memory
- Disks
- Memory required
- Use of disks
Out-of-core solution phase

For a symmetric case: \( Ax = b \) \( \Rightarrow \) \( A = LDL^T \)

matrix pattern

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1
\end{array}
\]

elimination tree

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1
\end{array}
\]

L factors

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1
\end{array}
\]

Location of factors on DISK

Out-Of-Core solution phase \((LDL^T x = b)\) is often as costly as the factorization
**Objective**: Reduce time for solution of $Ax = b$
in a parallel limited memory environment (OOC)

**Time performance** of the solution phase is related to:

- the bandwidth and number of disk accesses for reading data
- the number of processors and volume of factors per processor
- the regularity in the disk accesses

_Scheduling can significantly improve the parallel performance of the solution_

<table>
<thead>
<tr>
<th>Matrix name</th>
<th>Order</th>
<th>Nb entries (Millions)</th>
<th>Factor size (MBytes)</th>
<th>Nb Nodes in the tree</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>QIMONDA07</td>
<td>8 613 291</td>
<td>66.9</td>
<td>2 534</td>
<td>3 083 998</td>
<td>Qimonda AG</td>
</tr>
<tr>
<td>CAS4R-L15</td>
<td>2 423 135</td>
<td>195.8</td>
<td>4 832</td>
<td>864 447</td>
<td>EADS</td>
</tr>
</tbody>
</table>

Publicly available matrices can be obtained on [gridtlse.org](http://gridtlse.org) web site.
**Objective**: Reduce time for solution of $Ax = b$

in a parallel limited memory environment (OOC)

**Time performance** of the solution phase is related to:

- the bandwidth and number of disk accesses for reading data
- the number of processors and volume of factors per processor
- the regularity in the disk accesses

*Scheduling can significantly improve the parallel performance of the solution*

<table>
<thead>
<tr>
<th>Matrix name</th>
<th>Order</th>
<th>Nb entries (Millions)</th>
<th>Factor size (MBytes)</th>
<th>Nb Nodes in the tree</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>QIMONDA07</td>
<td>8 613 291</td>
<td>66.9</td>
<td>2 534</td>
<td>3 083 998</td>
<td>Qimonda AG</td>
</tr>
<tr>
<td>CAS4R-L15</td>
<td>2 423 135</td>
<td>195.8</td>
<td>4 832</td>
<td>864 447</td>
<td>EADS</td>
</tr>
</tbody>
</table>

Publicly available matrices can be obtained on gridtlse.org web site.
### QIMONDA07 (Qimonda AG company)

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Nb of Procs</th>
<th>Factor Size (MB)</th>
<th>Workspace (MB)</th>
<th>Fwd (sec)</th>
<th>Bwd (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STANDARD</strong></td>
<td>1</td>
<td>2 534</td>
<td>12</td>
<td>171.5</td>
<td>177.2</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>1</td>
<td>2 534</td>
<td>12</td>
<td>170.6</td>
<td>176.8</td>
</tr>
<tr>
<td><strong>STANDARD</strong></td>
<td>8</td>
<td>317</td>
<td>12</td>
<td>25.2</td>
<td>137.6</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>8</td>
<td>317</td>
<td>12</td>
<td>29.0</td>
<td>45.2</td>
</tr>
<tr>
<td><strong>STANDARD</strong></td>
<td>32</td>
<td>79</td>
<td>8.2</td>
<td>11.0</td>
<td>53.1</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>32</td>
<td>79</td>
<td>8.2</td>
<td>10.2</td>
<td>10.7</td>
</tr>
</tbody>
</table>

### CAS4R-L15 (EADS)

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Nb of Procs</th>
<th>Factor Size (MB)</th>
<th>Workspace (MB)</th>
<th>Fwd (sec)</th>
<th>Bwd (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STANDARD</strong></td>
<td>2</td>
<td>2547</td>
<td>559</td>
<td>336.3</td>
<td>270.1</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>2</td>
<td>2547</td>
<td>559</td>
<td>334.0</td>
<td>269.7</td>
</tr>
<tr>
<td><strong>STANDARD</strong></td>
<td>8</td>
<td>756</td>
<td>183</td>
<td>165.5</td>
<td>203.3</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>8</td>
<td>756</td>
<td>183</td>
<td>117.7</td>
<td>99.9</td>
</tr>
<tr>
<td><strong>STANDARD</strong></td>
<td>32</td>
<td>170</td>
<td>42</td>
<td>47.0</td>
<td>102.3</td>
</tr>
<tr>
<td><strong>NEW</strong></td>
<td>32</td>
<td>170</td>
<td>42</td>
<td>44.5</td>
<td>69.8</td>
</tr>
</tbody>
</table>
Memory related to communication buffers

- Communication buffer size becomes critical in Out-of-core.
- Idea: send message by packets that fit in a small buffer of fixed size.
- Cost: more synchronizations (receiver must be ready to receive before we can send the next packet).

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Communication scheme</th>
<th>Large buffers</th>
<th>Small buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIKW_1</td>
<td></td>
<td>264</td>
<td>4.2</td>
</tr>
<tr>
<td>CONESHL_MOD</td>
<td></td>
<td>66</td>
<td>3.7</td>
</tr>
<tr>
<td>CONV3D64</td>
<td></td>
<td>286</td>
<td>16.1</td>
</tr>
<tr>
<td>ULTRASOUND80</td>
<td></td>
<td>75</td>
<td>8.2</td>
</tr>
</tbody>
</table>

**Table:** Size of the communication buffers (MB) with 32 processors.

- Balance to be found to avoid too many synchronizations between senders and receivers.
Memory related to communication buffers

- Communication buffer size becomes critical in Out-of-core.
- Idea: send message by packets that fit in a small buffer of fixed size.
- Cost: more synchronizations (receiver must be ready to receive before we can send the next packet).

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Communication scheme</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Large buffers</td>
<td>Small buffers</td>
</tr>
<tr>
<td>AUDIKW_1</td>
<td>264</td>
<td>4.2</td>
</tr>
<tr>
<td>CONESHL_MOD</td>
<td>66</td>
<td>3.7</td>
</tr>
<tr>
<td>CONV3D64</td>
<td>286</td>
<td>16.1</td>
</tr>
<tr>
<td>ULTRASOUND80</td>
<td>75</td>
<td>8.2</td>
</tr>
</tbody>
</table>

**Total IC : 1260**

**Tab.**: Size of the communication buffers (MB) with 32 processors.

- Balance to be found to avoid too many synchronizations between senders and receivers.
Memory related to communication buffers

- Communication buffer size becomes critical in Out-of-core.
- Idea: send message by packets that fit in a small buffer of fixed size.
- Cost: more synchronizations (receiver must be ready to receive before we can send the next packet).

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Communication scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Large buffers</td>
</tr>
<tr>
<td>AUDIKW_1</td>
<td>264</td>
</tr>
<tr>
<td>CONESHL_MOD</td>
<td>66</td>
</tr>
<tr>
<td>CONV3D64</td>
<td>286</td>
</tr>
<tr>
<td>ULTRASOUND80</td>
<td>75</td>
</tr>
</tbody>
</table>

**Total OOC:** 800

[Table]: Size of the communication buffers (MB) with 32 processors.

- Balance to be found to avoid too many synchronizations between senders and receivers.
Preliminary work: scheduling influences peak memory needed

- Modify tree mapping to reduce the minimum memory requirement in parallel executions.

<table>
<thead>
<tr>
<th>Factors</th>
<th>Performance oriented</th>
<th>Memory oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Core</td>
<td>Max</td>
<td>4038</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>3345</td>
</tr>
<tr>
<td>Out-Of-Core</td>
<td>Max</td>
<td>3028</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>2251</td>
</tr>
</tbody>
</table>

**Table:** Estimated total memory (MB) - AUDIKW_1 matrix - 16 processors.
Outline

1. Introduction-History

2. Sparse Direct Methods and preprocessing

3. Parallel sparse solver - Memory issues

4. Conclusion-On going work
Future functionalities (on-going work)

- Out-of-core continuing
- Parallel analysis phase (orderings, scalings)
- Rank-detection and null space basis
- Scheduling for memory (OOC, Fault-Tolerance)
- Multi-core: (Flops not best metric to minimize)

On-going projects

- ANR CIS-SOLSTICE (ANR-06-CIS6-010)
- France-Berkeley (2008-)
- REDIMPS: CNRS/JST (Japan) cooperation project (2007-2009)
- SEISCOPE (http://www-geoazur.unice.fr/SEISCOPE)