Parallel analysis and scaling

A. Buttari and B. Uçar, CNRS-IRIT Toulouse, CNRS-LIP Lyon
MUMPS Users Group Meeting, April 2010
The solution of a sparse system with the MUMPS solver is achieved in three phases:

1. The **Analysis** phase
   - Fill-reducing pivot order
   - Symbolic factorization
   - Scaling
   - Amalgamation
   - Mapping
   - ...

2. The **Factorization** phase
   - \( LU = PA \)

3. The **Solve** phase
   - Forward/backward substitutions
Sparse direct solvers: the three phases

The solution of a sparse system with the MUMPS solver is achieved in three phases:

1. The **Analysis** phase
   - Fill-reducing pivot order
   - Symbolic factorization
   - Scaling
   - Amalgamation
   - Mapping
   - ...

2. The **Factorization** phase
   - $LU = PA$

3. The **Solve** phase
   - Forward/backward substitutions
The solution of a sparse system with the MUMPS solver is achieved in three phases:

1. The **Analysis** phase
   - Fill-reducing pivot order
   - Symbolic factorization
   - Scaling

2. The **Factorization** phase

3. The **Solve** phase
   - Forward/backward substitutions
Towards a Parallel Analysis
An approach to parallelization of the analysis

Briefly:

**Problem:** the sequential analysis of very large scale problems can be expensive
- memory consumption
- time to completion

**Solution:** parallelization of the analysis
Briefly:

**Problem:** the sequential analysis of very large scale problems can be expensive

- memory consumption
- time to completion

**Solution:** parallelization of the analysis

1. Parallel ordering of the problem using an external tool such as PT-SCOTCH or ParMETIS on 
   \[ \text{struct}(A + A^T) \]
An approach to parallelization of the analysis

Briefly:

**Problem:** the sequential analysis of very large scale problems can be expensive
- memory consumption
- time to completion

**Solution:** parallelization of the analysis

1. Parallel ordering of the problem using an external tool such as PT-SCOTCH or ParMETIS on
   \[\text{struct}(A + A^T)\]
2. Parallel symbolic factorization based on quotient graphs and restarting techniques
ParMETIS
- nested dissection stops at NP subdomains
- works only on $2^k$ processors
- quality of ordering degrades NP
- fast
PT-SCOTCH
- nested dissection does not stop at NP subdomains
- works on any number of processors
- quality of ordering is independent from NP
- a bit slower
Both
- get a distributed graph in input
- return an ordering and a separators tree on output
1 – First pass adjacency graph of the matrix to a parallel ordering tool (PT-SCOTCH or ParMetis). As a result, a pivotal order and a binary separators tree are returned.
2 – Then each processor separately performs the symbolic elimination of the variables contained in a subtree. This symbolic factorization is based on the usage of quotient graphs with a restarting technique that mixes left and right looking factorization methods.
3 – The host processor eliminates the variables in the top part of the tree using the same technique.
Parallel ordering and symbolic factoring

The distributed data are merged into a centralized data structure that is used in subsequent steps of the analysis phase like amalgamation, mapping etc.

4 – The distributed data are merged into a centralized data structure that is used in subsequent steps of the analysis phase like amalgamation, mapping etc.
• Quotient graphs
  ◦ keep the cost limited to $O(nnz)$ thanks to techniques like nodes absorption and redundant edges elimination
  ◦ ease the coupling between bottom and top part since the result of the symbolic facto on the subdomains can be represented as a clique in the quotient graph of the top-tree

• Restarting
Parallel ordering and symbolic factoring

- **Quotient graphs**
  - keep the cost limited to $O(nnz)$ thanks to techniques like nodes absorption and redundant edges elimination
  - ease the coupling between bottom and top part since the result of the symbolic factoring on the subdomains can be represented as a clique in the quotient graph of the top-tree

- **Restarting**
  1. in pivotal steps $1, ..., \tau$ are processed and only the adjacency information for variables $1 - \tau$ is updated in a right-looking way

• Quotient graphs
  ◦ keep the cost limited to $O(nnz)$ thanks to techniques like nodes absorption and redundant edges elimination
  ◦ ease the coupling between bottom and top part since the result of the symbolic facto on the subdomains can be represented as a clique in the quotient graph of the top-tree

• Restarting
  1. in pivotal steps $1, \ldots, \tau$ are processed and only the adjacency information for variables $1 - \tau$ is updated in a right-looking way
  2. **restart**: the adjacency information of variables $\tau - n$ is updated with respect to elements $1 - \tau$ in a left-looking way
Parallel ordering and symbolic facto

• Quotient graphs
  ◦ keep the cost limited to $O(nnz)$ thanks to techniques like nodes absorption and redundant edges elimination
  ◦ ease the coupling between bottom and top part since the result of the symbolic facto on the subdomains can be represented as a clique in the quotient graph of the top-tree

• Restarting
  1. in pivotal steps $1, \ldots, \tau$ are processed and only the adjacency information for variables $1 - \tau$ is updated in a right-looking way
  2. restart: the adjacency information of variables $\tau - n$ is updated with respect to elements $1 - \tau$ in a left-looking way
  3. apply steps 1 and 2 recursively on variables $\tau - n$
BRGM: N=3699643, NNZ=307580395

BRGM -- Max front size

BRGM -- Factor size

BRGM -- Flops

BRGM -- Scalability

9/28 A. Buttari and B. Ucañar, MUMPS Users Group Meeting, April 2010
CONESHL: \( N=126,2212, \text{NNZ}=8,475,3352 \)

CONESHL -- Max front size

CONESHL -- Factor size

CONESHL -- Flops

CONESHL -- Scalability

10/28 A. Buttari and B. Uçar, MUMPS Users Group Meeting, April 2010
10millions: N=10423737, NNZ=167722005

10millions -- Max front size

10millions -- Factor size

10millions -- Flops

10millions -- Scalability
• Better ordering with PT-SCOTCH
• ParMETIS faster than PT-SCOTCH
• ParMETIS ordering degrades with increasing parallelism
The behavior of the parallel analysis is defined by two parameters:

**ICNTL(28)** Analysis type:
- 0: Automatic decision (always =1 for the moment)
- 1: Sequential analysis
- 2: Parallel analysis

**ICNTL(29)** Ordering method for the parallel analysis
- 0: Automatic decision (always =1 for the moment)
- 1: PT-SCOTCH
- 2: ParMetis
Parallel analysis: interface

ICNTL(28)
    1
        ICNTL(7)
            0: AMD
            1: Given
            2: AMF
            3: SCOTCH
            4: PORD
            5: Metis
            6: QAMD
            7: Auto

    2
        ICNTL(29)
            0: Auto
            1: PT-SCOTCH
            2: ParMetis
Parallel Scaling
Matrix scaling

Definition

Given an $m \times n$ sparse matrix $A$, find diagonal matrices $D_1 > 0$ and $D_2 > 0$ such that all rows and columns of the scaled matrix

$$\hat{A} = D_1 AD_2$$

have equal norm.

Motivations

- Good pivoting strategy, numerical/optimal properties.
- Scaling combined with permutations can avoid many numerical difficulties [Duff and Pralet, SIMAX(2005)] during LU factorization:
  - Provides (weak) diagonal dominance,
  - Increases robustness of the factorization algorithms,
  - May improve the condition number.
The sequential algorithm (Ruiz 2001)

1: $D_1(0) \leftarrow I_{m \times m}$ \quad $D_2(0) \leftarrow I_{n \times n}$
2: for $k = 1, 2, \ldots$ until convergence do
3: \quad $D_R \leftarrow \text{diag} \left( \sqrt{\|r_i(k)\|_\ell} \right)$ \quad $i = 1, \ldots, m$
4: \quad $D_C \leftarrow \text{diag} \left( \sqrt{\|c_j(k)\|_\ell} \right)$ \quad $j = 1, \ldots, n$
5: \quad $D_1(k+1) \leftarrow D_1(k) D_R^{-1}$
6: \quad $D_2(k+1) \leftarrow D_2(k) D_C^{-1}$
7: \quad $A(k+1) \leftarrow D_1(k+1) A D_2(k+1)$
8: end for

Reminder

$\|x\|_\infty = \max\{|x_i|\}$

$\|x\|_1 = \sum |x_i|$

Notes

$\ell$: any vector norm (usually $\infty$- and 1-norms)
Convergence is achieved when

$$\max_{1 \leq i \leq m} \left\{ 1 - \|r_i(k)\|_\ell \right\} \leq \varepsilon \quad \text{and} \quad \max_{1 \leq j \leq n} \left\{ 1 - \|c_j(k)\|_\ell \right\} \leq \varepsilon$$
Features

Some properties (Ruiz 2001)

- Preserves symmetry; permutation independent; amenable to parallelization,
- With $\infty$-norm, linear convergence with asymptotic rate of $1/2$,
- With 1-norm, results are similar to those of the other well-known algorithms; convergence under certain conditions.

Practical considerations

- Numerical tests toward investigating the effects on LU decomposition, preconditioning [Duff and Pralet, SIMAX(2005)],
- Sequential codes also available in HSL library as MC77 [Ruiz (2001)],
- Parallel codes have been plugged into MUMPS.
Parallelization: Data distribution

Data: $\hat{A}^{(k)}$, $A$, $D_1^{(k)}$, $D_2^{(k)}$, $D_R$ and $D_C$.

**The scaled matrix $\hat{A}^{(k)}$**

Do not store $\hat{A}^{(k)} = D_1^{(k)}AD_2^{(k)}$ explicitly; access $a_{ij}^{(k)}$ by

$$d_1^{(k)}(i) \times |a_{ij}| \times d_2^{(k)}(j)$$

- Distribute $A$, $D_1$, and $D_2$. At every iteration $D_R$ and $D_C$ are computed afresh.
  - Matrix $A$ is already distributed (in another context). Each processor holds a set of entries $a_{ij}$ and their indices $(i,j)$.
  - Partition the diagonal elements of $D_1$ and $D_2$ among processors.

**Problem definition**

Given a partition on $A$, find the best partitions for $D_1$ and $D_2$. 

Parallelization: Dependencies

**Local computations**

Each processor $p$ should use each $(i, j, a_{ij})$ triplet to compute partial results on $d_R(i)$ and $d_C(j)$, e.g., in $\infty$-norm, sets

$$d_{R}^p(i) = \max \left\{ d_1^{(k)}(i) \times |a_{ij}| \times d_2^{(k)}(j) : a_{ij} \in p \right\}$$

**Communication operations**

The partial results should be combined/reduced for each $d_1^{(k+1)}(i)$. The owner of $d_1(i)$ should set, in $\infty$-norm,

$$d_1^{(k+1)}(i) = d_1^{(k)}(i) \times \frac{1}{\sqrt{\max\{d_{R}^p(i) : 1 \leq p \leq P\}}}.$$

The owner should send $d_1^{(k+1)}(i)$ back to the contributing processors.

• Similar discussion for $d_2(j)$. 

20/28 A. Buttari and B. Uçar, MUMPS Users Group Meeting, April 2010
Parallelization: $\infty$-norm algorithm for step $k$

**Row $r_i$**

Processors 2 and 4 contribute to $d_1^{(k+1)}(i)$. Whichever owns $d_1(i)$, receives one unit of data and sends one unit of data after computing the final $d_1^{(k+1)}(i)$.

**Column $c_j$**

Processors 1, 2, and 3 contribute to $d_2^{(k+1)}(j)$. Whichever owns $d_2(j)$, receives two units of data and sends two units of data after computing the final $d_2^{(k+1)}(j)$. 
Parallelization: Communication requirements

Common communication cost metric: the total volume.

**Communication for D$_1$**

- The volume of data a processor receives while reducing a $d_{1}^{(k+1)}(i)$ is equal to the volume of data it sends after computing $d_{1}^{(k+1)}(i)$.
- Nonzeros in row $r_i$ are split among $s_r(i)$ processors
  - All contribute to $d_{1}^{(k+1)}(i)$.
  - Reduction on $s_r(i)$ partial results.
  - If one of those $s_r(i)$ processors owns $d_{1}(i)$, $s_r(i) - 1$ partial results will be send to the owner.
  - If owned by somebody else, then $s_r(i)$ partial results will be send to the owner.

**Communication for D$_2$**

Similar observations.
**Communication requirements**

Nonzeros in row $r_i$ are split among $s_r(i)$ processors: total volume of communication is equal to

$$2 \times \sum (s_r(i) - 1)$$

(half for receiving contributions, half for sending back the results).

- The total volume of communication is the same for any $d_1(i)$ to processor assignment as long as that processor has at least one nonzero from row $r_i$.

Similar observation for the column $c_j$.

Twice the requirements of parallel sparse matrix-vector multiply operation.
## Computations (per iteration)

<table>
<thead>
<tr>
<th>Op.</th>
<th>SpMxV</th>
<th>1-norm</th>
<th>∞-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>(\text{nnz}(A))</td>
<td>(2 \times \text{nnz}(A))</td>
<td>0</td>
</tr>
<tr>
<td>mult</td>
<td>(\text{nnz}(A))</td>
<td>(2 \times \text{nnz}(A) + m + n)</td>
<td>(2 \times \text{nnz}(A) + m + n)</td>
</tr>
<tr>
<td>comparison</td>
<td>0</td>
<td>0</td>
<td>(2 \times \text{nnz}(A))</td>
</tr>
</tbody>
</table>

## Communication (per iteration)

The communication operations both in the 1-norm and \(\infty\)-norm algorithms are the same as those in the computations:

\[
\begin{align*}
  \mathbf{y} & \leftarrow \mathbf{A} \mathbf{x} \\
  \mathbf{x} & \leftarrow \mathbf{A}^T \mathbf{y}
\end{align*}
\]

when the partitions on \(\mathbf{x}\) and \(\mathbf{y}\) are equal to those on \(D_2\) and \(D_1\).
Parallelization: Our partitioning approach

### What we did?

- Use simple strategies while ensuring that each scaling entry is assigned to a processor that contributes to that entry.

\[ d_1(i) \] assigned to the processor \( p \) that has an entry \( a_{ij} \) with \( j \) giving \( \min\{|i - j|\} \); in case of ties to the processor with the smallest rank.

\[ d_2(j) \] assigned to the processor \( p \) that has an entry \( a_{ij} \) with \( i \) giving \( \min\{|i - j|\} \); in case of ties to the processor with the smallest rank.

### What could be done?

The freedom can be used to optimize some other metrics [U. and Aykanat, SISC(2004); Bisseling and Meesen, ETNA(2005)].

- Communication cost: Minimize number of messages, maximum volume/message per processor.
- Balance the number of \( d_1(i) \) and/or \( d_2(j) \) per processor.
## Parallelization results: Speedup values

<table>
<thead>
<tr>
<th>matrix</th>
<th>Seq. Time (s.)</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>aug3dcqp</td>
<td>8.30</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>3.06</td>
<td>1.9</td>
</tr>
<tr>
<td>a2nnnsnsl</td>
<td>20.71</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>7.24</td>
<td>1.5</td>
</tr>
<tr>
<td>a0nsdsil</td>
<td>20.92</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>7.22</td>
<td>1.5</td>
</tr>
<tr>
<td>lhr71</td>
<td>78.25</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>18.10</td>
<td>2.0</td>
</tr>
<tr>
<td>G3_circuit</td>
<td>455.25</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>173.11</td>
<td>1.9</td>
</tr>
<tr>
<td>thermal2</td>
<td>573.24</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>208.20</td>
<td>1.6</td>
</tr>
</tbody>
</table>

- Averages of 10 different partitions (with PaToH [Çatalyürek and Aykanat, Tech.Rep (1999)]),
- PC cluster with a Gigabit Ethernet switch (Intel Pentium IV 2.6 GHz), PC cluster with an Infiniband interconnect (dual 150 Opteron AMD processors)

Best three and worst three speedup values are shown—speedup tends to be higher with larger number of nonzeros.
Settings in MUMPS and some numerical results

**Default behavior** \((\text{ICNTL}(8)=7)\)

One \(\infty\)-norm scaling, three \(1\)-norm scaling.

Behaved better than a number of alternatives (among about 700 matrices, without scaling for 46 matrices parameter \text{ICNTL}(14) had to be adjusted; with this setting for 15 matrices).

Typically reduces the number of delayed pivots and off-diagonal pivoting, and hence reduces the memory requirements

<table>
<thead>
<tr>
<th>scaling</th>
<th>Flops ((\times10^6))</th>
<th>#entries in factors ((\times10^6))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>estimated</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>C-54</td>
<td>281</td>
<td>1.42</td>
</tr>
<tr>
<td>a0nsdsil</td>
<td>7.7</td>
<td>0.42</td>
</tr>
</tbody>
</table>
Conclusions

Thank you.

Parallel analysis and scaling are good 😊