Scalable Two-Level Preconditioners for CFD Computations on Many-Core Systems

Dr.-Ing. Achim Basermann, Melven Zöllner*
Survey

- Background DLR
- CFD Computations at DLR
- Storage Schemes for Sparse Matrices
- Distributed Schur Complement (DSC) Preconditioning
- Experiments with TRACE and TAU Matrices
- Conclusions and Future Work
DLR
German Aerospace Center

- Research Institution
- Space Agency
- Project Management Agency
Locations and Employees

7000 employees across 32 institutes and facilities at 16 sites.

Research Areas

- Aeronautics
- Space Research and Technology
- Transport
- Energy
Parallel Simulation System TRACE

- TRACE: Turbo-machinery Research Aerodynamic Computational Environment

- Developed by the Institute for Propulsion Technology of DLR

- Calculates internal turbo-machinery flows

- Finite volume method with block-structured grids

- The linearized TRACE modules require the parallel, iterative solution with preconditioning of large, sparse, non-symmetric real or complex systems of linear equations
Parallel Simulation System TAU

- TAU: developed for the aerodynamic design of aircrafts by the DLR Institute of Aerodynamics and Flow Technology

- Unstructured RANS solver (Reynolds-averaged Navier-Stokes), exploits finite volumes

- Requires the parallel, iterative solution with preconditioning of large, sparse, real, non-symmetric systems of linear equations
### Storage Schemes for Sparse Matrices

Compressed Row Storage (CSR) and Block Compressed Row Storage (BCSR)

#### Matrix:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>2</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

#### Non-zero values, row-wise:

1 2 3 4 5 6 7 8 9

#### Column indices, row-wise:

1 4 2 3 4 5 6 5 6

#### Row pointers:

1 3 6 8 10

- TRACE and TAU apply BCSR with 5x5 blocks.
- **Avantage**: less indirect addressing
- **Disadvantage**: A few zeros are stored.
Preconditioners: Incomplete *LU* Decomposition

**Incomplete** Gauss elimination

```
1: for i = 1, \ldots, n do
2:   for k = 1, \ldots, i - 1 \land (i, k) \in M do
3:     a_{i,k} \leftarrow a_{i,k} (a_{k,k})^{-1}
4:   for j = k + 1, \ldots, n \land (i, j) \in M do
5:     a_{i,j} \leftarrow a_{i,j} - a_{i,k} a_{k,j}
6:   end for
7: end for
8: end for
```

- **Block methods** apply sub-matrices, e.g.
  
  \[ a_{i,j} \in \mathbb{C}^{5 \times 5} \]

- The set *M* determines which entries are not dropped.

LU construction and forward and backward substitution are hard to parallelize!
Block-Jacobi-ILU Preconditioning for $K^{-1}Az = K^{-1}b$

- Idea: independent incomplete $LU$ decompositions of the diagonal blocks
  $\rightarrow$ highly parallel

- Disadvantage: Entries outside the diagonal blocks are neglected.
  $\rightarrow$ Preconditioner quality decreases with increasing #diagonal blocks.
Schur Complement Transformation for $Az = b$

$$\begin{pmatrix} D & E \\ F & G \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} D & E \\ F & G \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g - FD^{-1}f \end{pmatrix}$$

Solution method

1. Transformation: $g' := g - FD^{-1}f$
2. Solve Schur complement system: $Sy = g'$
3. Back transformation: $x = D^{-1}(f - Ey)$
Distributed Equation System

\[
\begin{pmatrix}
D_1 E_1 \\
F_1 G_1
\end{pmatrix}
\begin{pmatrix}
G_{1,2} \\
G_{1,3}
\end{pmatrix}
\begin{pmatrix}
D_2 E_2 \\
F_2 G_2
\end{pmatrix}
\begin{pmatrix}
G_{2,3}
\end{pmatrix}
\begin{pmatrix}
D_3 E_3 \\
F_3 G_3
\end{pmatrix}
\begin{pmatrix}
x_1 \\
y_1 \\
x_2 \\
y_2 \\
x_3 \\
y_3
\end{pmatrix}
=
\begin{pmatrix}
f_1 \\
g_1 \\
f_2 \\
g_2 \\
f_3 \\
g_3
\end{pmatrix}
\]
DSC Preconditioning (1)

- Use Schur complement transformation for distributed equation systems (DSC, *distributed Schur complement*)

- Iteratively improve block-Jacobi-ILU preconditioning by the consideration of the matrix entries outside the diagonal blocks.
Algorithm

1. ILU of the diagonal blocks:

\[
\begin{pmatrix}
\tilde{L}_{D_i} & (D_i^{-1}E_i) \\
(F_i U_{D_i}^{-1}) & \tilde{L}_{S_i}
\end{pmatrix}
\begin{pmatrix}
\tilde{U}_{D_i} \\
\tilde{U}_{S_i}
\end{pmatrix}
= \begin{pmatrix}
D_i & E_i \\
F_i & G_i
\end{pmatrix}
\]

2. Transformation:

\[
\begin{pmatrix}
f_i' \\
y_i^0
\end{pmatrix} = \begin{pmatrix}
\tilde{L}_{D_i} & (D_i^{-1}E_i) \\
(F_i U_{D_i}^{-1}) & \tilde{L}_{S_i} \tilde{U}_{S_i}
\end{pmatrix}^{-1}
\begin{pmatrix}
f_i \\
g_i
\end{pmatrix}
\]

3. Approximate the solution of the Schur complement systems:

\[
y_i + (\tilde{L}_{S_i} \tilde{U}_{S_i})^{-1} \sum_{j=1}^{3} G_{i,j} y_j = y_i^0 \quad \text{für} \quad i = 1, 2, 3
\]

4. Back transformation:

\[
\begin{pmatrix}
x_i^k \\
y_i^k
\end{pmatrix} = \begin{pmatrix}
\tilde{U}_{D_i} & (D_i^{-1}E_i) \\
F_i & G_i
\end{pmatrix}^{-1}
\begin{pmatrix}
f_i' \\
y_i^k
\end{pmatrix}
\]

Formulation saves forward/back operations ↔ Saad/Sosonkina 1999
Whole Solver: ILU Construction

Preconditioning within the DSC algorithm

Processor $i$

Local rows

Block incomplete $LU$ for the local rows

Block incomplete $LU$ for the local interface rows

Local interface rows
Whole Solver: Outer and Inner Iteration

Schematic view on each processor

Also possible for outer iteration:
- Flexible QMR
- Flexible BiCG
- Flexible BiCGstab
(Szyld, Vogel 2001; Vogel 2007)
Hardware System

- RWTH Bull HPC cluster
  - Intel Westmere X5675 CPUs
  - 6 cores per CPU with 3.06 GHz
  - 12 cores (2 CPUs) per node

- Computations with 1 MPI process per core
Experiments: CSR versus BCSR Format

Block-Jacobi-ILU preconditioning with 12 processes

<table>
<thead>
<tr>
<th>Block size</th>
<th>Execution time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>77</td>
</tr>
<tr>
<td>5</td>
<td>59</td>
</tr>
</tbody>
</table>

ILU construction
Iterations

TAU matrix:
\( n=541,980; \)
\( nz=170,610,950; \)
ILU fill-in ratio \( \approx 0.8; \)
\(|\text{rel. res.}| < 10^{-5} \)
Experiments: Strong Scaling, Iterations

TRACE mat. UHBR: n=4,497,520; nz=552,324,700; threshold=5·10^{-4}; |rel. res.|<10^{-5}
Experiments: Strong Scaling, Time

TRACE mat. UHBR: \( n=4,497,520; \ nz=552,324,700; \) threshold=5\( \times 10^{-4}; \ |\text{rel. res.}|<10^{-5} \)
linearTRACE Performance: Internal versus DSC Solver
(2x Intel XEON E5520 with 4 cores each, 2.26 GHz)

dsc2011 solver for linearTRACE
(8 processes, test case "THD stator": dim = 0.8 Mio, nnz = 90 Mio)

<table>
<thead>
<tr>
<th>Time in seconds</th>
<th>Internal solver (gmres100, ssor(0.7,3))</th>
<th>DSC2011 (fgmres40, dsc gmres 5, ilut(0.01,1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>#140</td>
<td>#57</td>
<td></td>
</tr>
</tbody>
</table>

- Trace (setup matrix etc)
- Solver iteration
- Prec. preparation (ilut)
linearTRACE Memory Usage: Internal versus DSC Solver
(2x Intel XEON E5520 with 4 cores each, 2.26 GHz)

dsc2011 solver for linearTRACE
(8 processes, test case "THD stator": dim = 0.8 Mio, nnz = 90 Mio)

Peak memory [Gigabyte]

<table>
<thead>
<tr>
<th>Peak memory [Gigabyte]</th>
<th>internal solver ( gmres100, ssor(0.7,3) )</th>
<th>dsc2011 ( fgmres40, dsc gmres 5, ilut(0.01,1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- BCSR format application significantly outperforms CSR format application for real TRACE and TAU problems.

- DSC method achieves higher scalability and faster iteration than block-local methods.

- DSC method very suitable for TRACE and TAU problems

Future Work

- Hybrid parallelization is appropriate to further improve scalability.
Questions?
DSC Method: Effect of the Interface Iteration

(2x Intel XEON E5520 with 4 cores each, 2.26 GHz)

Results on 8 cores

TAU matrix:
n=541,980;
nz=170,610,950;
threshold = 10^{-3};
|rel. residual| < 10^{-7}