A Hybrid Solver Based on Block-Cimmino Method

Distributed implementation of the Conjugate Gradient accelerated Block Cimmino Method

I.S. Duff    R. Guivarch    D. Ruiz    Mohamed Zenadi

MUMPS User Days - May 29-30th, 2013
The Hybrid Block Cimmino
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

Partitioning the system $Ax = b$

\[
\begin{pmatrix}
A_1 \\
A_2 \\
\vdots \\
A_p
\end{pmatrix} x =
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_p
\end{pmatrix}
\]
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]

The Block Cimmino Iteration

\[ \delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)} \]

\[ x^{(k+1)} = x^{(k)} + \nu \sum_{i=1}^{p} \delta_i^{(k)} \]

where:

\[ A_i^+ = A_i^T (A_i A_i^T)^{-1} \]

and

\[ P_{\mathcal{R}(A_i^T)} = A_i^+ A_i \]
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is $\sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)}$

Acceleration

Apply CG to solve the SPD system

$$\sum_{i=1}^{p} A_i^+ A_i x = \sum_{i=1}^{p} A_i^+ b_i$$
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is $\sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)}$

Projections: $\delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)}$
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is \( \sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)} \)

Projections: \( \delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)} \)

Solve independently using MUMPS the systems for each partition

\[
\begin{bmatrix}
I & A_i^T \\
A_i & 0
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\quad b_i - A_i x
\end{bmatrix}
\]

where: \( u_i = A_i^+ (b_i - A_i x) = \delta_i \)
Block Cimmino: The basic facts

- Block row projection method [Elfving (Numer. Math. 1980)]
- CG acceleration: iteration matrix is \( \sum_{i=1}^{p} A_i^+ A_i = \sum_{i=1}^{p} P_{\mathcal{R}(A_i^T)} \)
- Can exploit up to 3 levels of parallelism (independency, sparsity structure, BLAS3 Kernels)

Projections: \( \delta_i^{(k)} = A_i^+ b_i - P_{\mathcal{R}(A_i^T)} x^{(k)} \)

Solve independently using MUMPS the systems for each partition

\[
\begin{bmatrix}
I & A_i^T \\
A_i & 0
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i
\end{bmatrix} =
\begin{bmatrix}0 \\
b_i - A_i x
\end{bmatrix}
\]

where: \( u_i = A_i^+ (b_i - A_i x) = \delta_i \)
In sequential:

- Computes all the projections at once
- Builds a block diagonal system of augmented system
- Analyzes and factorizes then solves using MUMPS
- Exploits the forest structure

\[
\begin{align*}
\begin{bmatrix}
\mathbf{u}_1 & \mathbf{v}_1 \\
\mathbf{u}_2 & \mathbf{v}_2 \\
\vdots & \vdots \\
\mathbf{u}_p & \mathbf{v}_p \\
\end{bmatrix} = \\
\begin{bmatrix}
\mathbf{r}_1 & \mathbf{0} \\
\mathbf{r}_2 & \mathbf{0} \\
\vdots & \vdots \\
\mathbf{r}_p & \mathbf{0} \\
\end{bmatrix}
\end{align*}
\]
Sequential Case: Computing projections

In sequential:

- Computes all the projections at once
### Sequential Case: Computing projections

In sequential:

- Computes all the projections at once
- Build a block diagonal system of augmented system

\[
\begin{bmatrix}
u_1 \\
v_1 \\
u_2 \\
v_2 \\
\vdots \\
u_p \\
v_p
\end{bmatrix} =
\begin{bmatrix}
0 \\
r_1 \\
0 \\
r_2 \\
\vdots \\
0 \\
r_p
\end{bmatrix}
\]
Sequential Case: Computing projections

In sequential:

- Computes all the projections at once
- Build a block diagonal system of augmented system
- Analyse + Factorize then solve using MUMPS

\[
\begin{align*}
\begin{bmatrix}
    u_1 & 0 \\
    v_1 & 0 \\
    u_2 & 0 \\
    v_2 & 0 \\
    \vdots & \vdots \\
    u_p & 0 \\
    v_p & 0 \\
\end{bmatrix}
\begin{bmatrix}
    0 \\
    r_1 \\
    0 \\
    r_2 \\
    \vdots \\
    0 \\
    r_p \\
\end{bmatrix}
= 0
\end{align*}
\]
In sequential:
- Computes all the projections at once
- Build a block diagonal system of augmented system
- Analyse + Factorize then solve using MUMPS
- Exploits the Forest structure

\[
\begin{bmatrix}
  u_1 & \vdots & u_p \\
  v_1 & \vdots & v_p \\
\end{bmatrix}
= 
\begin{bmatrix}
  0 & \vdots & 0 \\
  r_1 & \vdots & r_p \\
\end{bmatrix}
\]
Sequential workflow

Augmented System Creation

MUMPS

A

Analyse → Factorize

\[ \sum_{i=1}^{p} A_i + A_i \]
Sequential workflow

A

Augmented System Creation

Factorize

Analyse

MUMPS

Conjugate Gradient

\[ \sum_{i=1}^{p} A_i^+ A_i p \]

Solve

5/22
Parallelism results: Running MUMPS in parallel

torso3 \(259,156 \times 259,156\) NZ(4,429,042)
16 partitions 33 iterations

<table>
<thead>
<tr>
<th>Number of processors</th>
<th>CPU</th>
<th>CG</th>
<th>Fact.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46.73s</td>
<td>21.28.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9.09s</td>
<td>11.66s</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5.16s</td>
<td>9.57s</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4.02s</td>
<td>8.03s</td>
<td></td>
</tr>
</tbody>
</table>
Distributed Scheme

- Distributed Conjugate Gradient Acceleration

---

Important Notice

There is only a single Block-CG distributed over these processes.
Distributed Scheme

- Distributed Conjugate Gradient Acceleration

Important Notice

There is only a single Block-CG distributed over these processes.
Distributed Scheme

- Distributed Conjugate Gradient Acceleration
Distributed Scheme

- Distributed Conjugate Gradient Acceleration
- Multiple levels of parallelism

Important Notice
There is only a single Block-CG distributed over these processes.
Distributed Conjugate Gradient Acceleration
Multiple levels of parallelism
Forest simulation

Important Notice
There is only a single Block-CG distributed over these processes.
Distributed Scheme

Create Aug. Syst.
Analyze Aug. Syst.
Find slaves
Create Intra-comm.
Feed MUMPS
Launch CG

8/22 MUMPS User Days - May 29-30th, 2013
Distributed Scheme

Create Aug. Syst.

Analyse Aug. Syst.

Find slaves

Create Intra-comm.

Feed MUMPS

Launch CG

CG0
Parallelism results: Distributed B-CG

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size</th>
<th>Nonzeros</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$: torso3</td>
<td>259,156</td>
<td>4,429,042</td>
<td>3D model of torso</td>
</tr>
<tr>
<td>$N_2$: CoupCons3D</td>
<td>416,800</td>
<td>17,277,420</td>
<td>structural problem</td>
</tr>
<tr>
<td>$N_3$: cage13</td>
<td>445,315</td>
<td>7,479,343</td>
<td>DNA electrophoresis</td>
</tr>
<tr>
<td>$N_4$: Hamrle3</td>
<td>1,447,360</td>
<td>5,514,242</td>
<td>Circuit Simulation</td>
</tr>
</tbody>
</table>

- Runs made on Hyperion - CICT
- 3.2Ghz Intel Xeon Quad-Core CPUs (2 per node)
- 36GB per node
Parallelism results: Distributed B-CG

(a) Factorization speedup

<table>
<thead>
<tr>
<th>Problem</th>
<th>Partitions</th>
<th>Factorization at 8 Cores</th>
<th>B-CG at 8 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>16</td>
<td>12.11 s.</td>
<td>6.34 s.</td>
</tr>
<tr>
<td>$N_2$</td>
<td>32</td>
<td>14.41 s.</td>
<td>63.49 s.</td>
</tr>
<tr>
<td>$N_3$</td>
<td>256</td>
<td>28.15 s.</td>
<td>13.64 s.</td>
</tr>
<tr>
<td>$N_4$</td>
<td>64</td>
<td>6.67 s.</td>
<td>773 s.</td>
</tr>
</tbody>
</table>
Block Cimmino vs MUMPS

32 cores machine with 500GB of memory [Conan - ENSEEIHT]

The reality:
On most test cases, MUMPS does better!

However:
• BC breaks the complexity down so that the factorization goes faster
• BC consumes less memory:
 ◦ MUMPS + Cage13: MAX : 6.7GB / AVG : 4.0GB
**Block Cimmino vs MUMPS**

32 cores machine with 500GB of memory [Conan - ENSEEIHT]

<table>
<thead>
<tr>
<th>Problem</th>
<th>[MUMPS] Factorization</th>
<th>[BC] Factorization</th>
<th>B-CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>torso3</td>
<td>5.03 s.</td>
<td>3.21 s.</td>
<td>5.18 s.</td>
</tr>
<tr>
<td>Cage13</td>
<td>1452.44 s.</td>
<td>9.31 s.</td>
<td>3.18 s.</td>
</tr>
<tr>
<td>Hamrle3</td>
<td>413.21 s.</td>
<td>2.13 s.</td>
<td>282.90 s.</td>
</tr>
</tbody>
</table>

**The reality:** On most test cases, MUMPS does better!

**However:**

- BC breaks the complexity down so that the factorization goes faster
- BC consumes less memory:
  - MUMPS + Cage13: MAX: 6.7GB / AVG: 4.0GB
A path to orthogonality

### Issues

- Slow convergence
- Unpredictable convergence behaviour (usually plateaux based)
- Multiple solves requires a re-run of B-CG (too expensive)
A path to orthogonality

Issues

• Slow convergence
• Unpredictable convergence behaviour (usually plateaux based)
• Multiple solves requires a re-run of B-CG (too expensive)

Solution

• Enforce numerical orthogonality between partitions by adding extra variables and constraints
• Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves
A path to orthogonality

**Issues**

- Slow convergence
- Unpredictable convergence behaviour (usually plateaux based)
- Multiple solves requires a re-run of B-CG (too expensive)

**Solution**

- Enforce numerical orthogonality between partitions by adding extra variables and constraints
- Extract a condensed smaller subsystem (similar to Schur complement techniques) that can be reused for efficient further solves

⇒ **Augmented Block Cimmino Distributed solver (ABCD solver)**
Augmented Block Cimmino Distributed solver
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\),
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\), and enforce numerical orthogonality

Illustrative example
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_iA_j^T\), and enforce numerical orthogonality to obtain \(\bar{A} = [A \ C]\)

Illustrative example
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\), and enforce numerical orthogonality to obtain \(\tilde{A} = [A \ C]\)
- Add extra constraints to build an equivalent linear system:

\[
\begin{bmatrix}
A & C \\
0 & I
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
=
\begin{bmatrix}
b \\
0
\end{bmatrix},
\]

where \(y = 0\) ensures the same solution \(x\).
The augmentation process

- Partition the matrix with respect to its structure (not necessarily in block-tridiagonal form)
- For each pair of partitions \((j > i)\), expand with \(C_{i,j} = A_i A_j^T\), and enforce numerical orthogonality to obtain \(\bar{A} = [A \ C]\)
- Add extra constraints to build an equivalent linear system:

\[
\begin{bmatrix}
A & C \\
0 & I
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
b \\
0
\end{bmatrix},
\]

where \(y = 0\) ensures the same solution \(x\).

**Problem**

the extra partition \(Y = [0 \ I]\), linked to the constraints equations, is not orthogonal to the previous partitions in \(\bar{A} = [A \ C]\).
The augmentation process

To enforce this orthogonality, we project the column vectors $Y^T$ onto the null space of $\bar{A} = [A \ C]$ (orthogonal complement of $\mathcal{R}(\bar{A}^T)$) :

$$W^T = (I - P) Y^T,$$

where (as a result of the enforced orthogonality) :

$$P = P_{\mathcal{R}(\bar{A}^T)} = P_{\bigoplus_{i=1}^p \mathcal{R}(\bar{A}_i^T)} = \sum_{i=1}^p P_{\mathcal{R}(\bar{A}_i^T)}$$
The augmentation process

To enforce this orthogonality, we project the column vectors $Y^T$ onto the null space of $\tilde{A} = [A \ C]$ (orthogonal complement of $\mathcal{R}(\tilde{A}^T)$) :

$$W^T = (I - P) Y^T,$$

where (as a result of the enforced orthogonality) :

$$P = P_{\mathcal{R}(\tilde{A}^T)} = P \bigoplus_{i=1}^{p} \mathcal{R}(\tilde{A}_i^T) = \sum_{i=1}^{p} P_{\mathcal{R}(\tilde{A}_i^T)}$$

We finally obtain $[A \ C]$, where $[B \ S] = W$, an augmented matrix with mutually numerically orthogonal partitions.
The augmentation process

Illustrative example
The augmentation process

To keep the consistency within the solution of the new system:

\[
\begin{bmatrix}
A & C \\
B & S
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
b \\
f
\end{bmatrix}
\]

we compute the right hand side \( f \) as:

\[
f = [B \ S] \begin{bmatrix} x \\ 0 \end{bmatrix} = Y (I - P) \begin{bmatrix} x \\ 0 \end{bmatrix}
\]
The augmentation process

To keep the consistency within the solution of the new system:

\[
\begin{bmatrix}
A & C \\
B & S
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
=
\begin{bmatrix}
b \\
f
\end{bmatrix}
\]

we compute the right hand side \( f \) as:

\[
f = \begin{bmatrix} B & S \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = Y(I-P) \begin{bmatrix} x \\ 0 \end{bmatrix}
\]

\[
= -YP \begin{bmatrix} x \\ 0 \end{bmatrix} \quad \text{(since } Y = \begin{bmatrix} 0 & I \end{bmatrix} )
\]

\[
= -Y\bar{A}^+\bar{A} \begin{bmatrix} x \\ 0 \end{bmatrix}
\]

\[
f = -Y\bar{A}^+b
\]
Implicit Direct Solver

Since all the partitions in the new equivalent linear system

\[
\begin{bmatrix}
A & C \\
B & S
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
b \\
f
\end{bmatrix}
\]

are mutually numerically orthogonal, the Cimmino iteration matrix becomes the Identity matrix, and the solution can be directly obtained as:

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - W^+ Y \bar{A}^+ b
\]

\[
= \sum_{i=1}^{p} \bar{A}_i^+ b_i - W^+ Y \sum_{i=1}^{p} \bar{A}_i^+ b_i
\]
Knowing that $W = [B \quad S] = Y (I - P)$, with $Y = [0 \quad I]$, we have:

\[
WW^T = Y (I - P) (I - P)^T Y^T \\
= Y (I - P)^2 Y^T \\
= Y (I - P) Y^T \\
= [B \quad S] Y^T \\
= S
\]

Therefore $S = Y (I - P) Y^T$ and is SPD.

And the pseudo inverse $W^+ = W^T (WW^T)^{-1}$ is given by

\[
W^+ = W^T S^{-1} \\
W^+ = (I - P) Y^T S^{-1}
\]
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= \tilde{A}^+ b + W^+ f
= \tilde{A}^+ b - (I - P) Y^T S^{-1} Y \tilde{A}^+ b
\]
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

1. Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
2. Solve \( Sz = f \) (\( S \) should be small enough)
3. Expand \( z \) and then project it onto the null space of \( \bar{A} \) viz. \( u = (I - P) Y^T z \)
4. Then sum \( w + u \) to obtain the solution \( \begin{bmatrix} x \\ y \end{bmatrix} \) (where \( y = 0 \))

Note that we don't need to build \( B \), only \( S \) is used.
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \bar{A}^+ b + W^+ f \\
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- **Build** \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- **Solve** \( Sz = f \) (\( S \) should be small enough)
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix} = \bar{A}^+ b + W^+ f = \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- **Step 1**: Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- **Step 2**: Solve \( Sz = f \) (\( S \) should be small enough)
- **Step 3**: Expand \( z \) and then project it onto the null space of \( \bar{A} \) viz.
  \[
  u = (I - P) Y^T z
  \]
The solution is thus given by:

\[
\begin{bmatrix}
x \\
y
\end{bmatrix} = \bar{A}^+ b + W^+ f \\
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

• Build \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
• Solve \( Sz = f \) (\( S \) should be small enough)
• Expand \( z \) and then project it onto the null space of \( \bar{A} \) viz. 
  \[
  u = (I - P) Y^T z
  \]
• Then sum \( w + u \) to obtain the solution \( \begin{bmatrix} x \\ y \end{bmatrix} \) (where \( y = 0 \))
The solution is thus given by:

\[
\begin{bmatrix}
  x \\
  y \\
\end{bmatrix} = \bar{A}^+ b + W^+ f
\]

\[
= \bar{A}^+ b - (I - P) Y^T S^{-1} Y \bar{A}^+ b
\]

which can be computed through the 4 following steps:

- **Build** \( w = \bar{A}^+ b \) and then by simple restriction set \( f = -Yw \)
- **Solve** \( Sz = f \) (\( S \) should be small enough)
- **Expand** \( z \) and then project it onto the null space of \( \bar{A} \) viz. \( u = (I - P) Y^T z \)
- **Then sum** \( w + u \) to obtain the solution \( \begin{bmatrix} x \\ y \end{bmatrix} \) (where \( y = 0 \))

Note that we don’t need to build \( B \), only \( S \) is used.
Current and Future work

- The Augmented Block Cimmino Distributed Method
## Current and Future work

- The Augmented Block Cimmino Distributed Method

**Hamrle3 (1.447M) 64 partitions on 32 cores**

<table>
<thead>
<tr>
<th></th>
<th>BC (blk. size = 4)</th>
<th>ABCD (size $S = 54608$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>2.13 s.</td>
<td>3.4s.</td>
</tr>
<tr>
<td>CG</td>
<td>(615itr) 282.90 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>4.6s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>145.4s.</td>
</tr>
<tr>
<td>Ana. S</td>
<td>-</td>
<td>5.6s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>49.1s.</td>
</tr>
</tbody>
</table>
Current and Future work

- The Augmented Block Cimmino Distributed Method

<table>
<thead>
<tr>
<th>R6 (132k) 16 partitions on 32 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC (blk. size = 16)</td>
</tr>
<tr>
<td>Fact.</td>
</tr>
<tr>
<td>CG</td>
</tr>
<tr>
<td>Augmentation</td>
</tr>
<tr>
<td>Build S</td>
</tr>
<tr>
<td>Ana. S</td>
</tr>
<tr>
<td>Fact S</td>
</tr>
</tbody>
</table>

Thank you!
Current and Future work

- The Augmented Block Cimmino Distributed Method

<table>
<thead>
<tr>
<th></th>
<th>BC (blk. size = 1)</th>
<th>ABCD (size $S = 16695$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact.</td>
<td>1.7 s.</td>
<td>1.97s.</td>
</tr>
<tr>
<td>CG</td>
<td>(Failed) 176.5 s.</td>
<td>-</td>
</tr>
<tr>
<td>Augmentation</td>
<td>-</td>
<td>0.6s.</td>
</tr>
<tr>
<td>Build S</td>
<td>-</td>
<td>40.0s.</td>
</tr>
<tr>
<td>Ana. S</td>
<td>-</td>
<td>0.2s.</td>
</tr>
<tr>
<td>Fact S</td>
<td>-</td>
<td>18.0s.</td>
</tr>
</tbody>
</table>
Current and Future work

• The Augmented Block Cimmino Distributed Method
Current and Future work

- The Augmented Block Cimmino Distributed Method [on it!]
- Iteratively and implicitly solve $S^{-1}f$ [under investigation]
- More parallelism improvements [todo]
  - Distributed hypergraph partitioning (Zoltan)
  - Distributed input matrix
  - Improve parallel scaling
  - etc.
Current and Future work

- The Augmented Block Cimmino Distributed Method [on it!]
- Iteratively and implicitly solve $S^{-1}f$ [under investigation]
- More parallelism improvements [todo]
  - Distributed hypergraph partitioning (Zoltan)
  - Distributed input matrix
  - Improve parallel scaling
  - etc.

Thank you!