Scalable sparse matrix solvers on supercomputers

Hans De Sterck

Department of Applied Mathematics
University of Waterloo
hdesterck@uwaterloo.ca
Sparse matrix solvers

- solve $A \ x = b$

- $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$

- A large: $n = \text{millions, billions, ...}$

- A sparse: PDE discretization on a grid

- we want efficient, accurate and robust solvers
example 1: space weather prediction

- solar wind plasma (ionized gas) flows from sun to earth
- simulation goal: predict when solar eruptions reach earth (~ 4 days)
example 1: space weather prediction

- discretize Magnetohydrodynamics (MHD) equations
- implicit time discretization

- $A \mathbf{x} = \mathbf{b}$
- 8 million unknowns
example 1: space weather prediction

- problem with existing methods: **scalability**
  - 3D simulation
  - suppose: increase resolution by 2
  - \( n \Rightarrow 8\ n \)
  - if the methods scale well, \( t_{\text{exec}} \Rightarrow 8\ t_{\text{exec}} \)
    (linear or \( O(n) \) scaling)
  - existing methods often scale as \( O(n^2) \) or worse:
    \( t_{\text{exec}} \Rightarrow 64\ t_{\text{exec}} \)
  - high problem resolutions quickly get out of reach
- need scalable methods!
example 2: biomedical flows

- blood flow in compliant vessels
- incompressible Navier-Stokes + elasticity equations
- 3D: large systems, need scalable solvers on parallel computers
  (simulation results courtesy Jeff Heys)

\[-\nabla p + \Delta \vec{v} = 0\]  \hspace{1cm} (1)
\[\nabla \cdot \vec{v} = 0\]  \hspace{1cm} (2)
Overview of presentation

(A) introduction - scalable solvers for PDEs

(B) multigrid iterative solvers

(C) scalable parallel algebraic multigrid solvers

(D) future work
(A) introduction - scalable solvers for PDEs

Two types of scalability

1. Algorithmic scalability
   - On a single processor, using serial algorithm
   - Optimal algorithmic scalability:
     - Number of operations \( \text{op}_n = O(n) \)
     - Execution time \( t_{\text{exec}} = O(n) \)
     (both increase linearly with \( n \))
(1) algorithmic scalability
two types of scalability

(2) parallel scalability

- on a parallel computer, using parallelized algorithm
- communication overhead, may not scale well
- parallelization often attempts to minimize communication
- parallelized algorithm may be less scalable than serial algorithm (even with zero communication cost)
(2) parallel scalability

- consider constant problem size $n_{\text{proc}}$ per processor
- $p$ processors
- total problem size $n = p n_{\text{proc}}$
- double problem size, $2n = 2p n_{\text{proc}}$
- double number of processors, $2p$
- optimal scalability: $t_{\text{exec}} = \text{constant}$ as $n$ increases
(B) multigrid iterative solvers

• solve $A x = b$
• goal: $O(n)$ complexity

• 1D model problem:

\[-u''(x) = f(x)\]

\[u(0) = 0\]

\[u(1) = 0\]

• discretization:

\[-u_{i-1} + 2u_i - u_{i+1} \over h^2 = f_i\]
1D model problem

\[-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i\]

• A x = b

\[A = \frac{1}{h^2} \begin{bmatrix}
2 & -1 & & & \\
-1 & 2 & -1 & & \\
& -1 & 2 & -1 & \\
& & \ddots & \ddots & \ddots \\
& & & -1 & 2 & -1 \\
& & & & -1 & 2
\end{bmatrix}\]

• first idea: use an iterative method (exploiting sparsity)

• for example: Gauss method (relaxation)

\[u_{i}^{new} = \frac{1}{2} (u_{i-1}^{new} + u_{i+1}^{old} + h^2 f_i)\]
Gauss relaxation method: \(-u''(x) = 0\)

- need more iterations as \(n\) increases: not scalable
- but: for large \(n\), high frequency errors disappear fast!
- note: low frequency for large \(n\) (fine grid), is high frequency for small \(n\) (coarse grid)

- second idea: use hierarchy of grid levels
  - correct high-frequency errors on fine grid
  - correct low-frequency errors on coarse grid
multigrid hierarchy: V-cycle

- multigrid V-cycle:
  - relax (=smooth) on various grids
  - transfer error using restriction ($P^T$) and interpolation ($P$)
2D model problem: \[-u_{xx} - u_{yy} = f(x, y)\]

- high-frequency error is removed by relaxation
- low-frequency-error needs to be removed by coarse-grid correction
multigrid hierarchy: V-cycle

- in every V-cycle, error is reduced by convergence factor $\rho$
- repeat $m$ V-cycles, error is reduced by $\rho^m$
- for large classes of elliptic PDEs, $\rho$ is bounded away from 1 uniformly in $n$
- ‘deeper’ cycles have same $\rho$ as ‘shallower’ cycles
amount of work per V-cycle

- work for 1 relaxation on fine grid is called a Work Unit (WU)
- 1 relaxation (or WU) is $O(n)$ ! (number of rows in $A^h$)
- number of WUs in V-cycle:
  \[ 2 \cdot (1 + (1/2)^2 + (1/2)^4 + (1/2)^8 + \ldots + (1/2)^{2m}) \text{ WUs} \]
- geometric series: $\sum_{i=0}^{m} (1/4)^i < 1 / (1-(1/4)) = 4/3$
- total work per V-cycle < 8/3 WUs : $O(n)$!
scalable solver: $O(n)$

- scalable method: $O(n)$
  - work per V-cycle is $O(n)$
  - number of V-cycles required is independent of $n$
    - because $\rho$ independent of $n$
(C) scalable parallel algebraic multigrid solvers

- algebraic multigrid:
  - unstructured grid problems
  - matrix problems without grid

automatically determine, only from information in matrix:

- coarse ‘grids’
- coarse grid operators $A^h$
- interpolation operators $P^h$

[Brandt, McCormick, Ruge, Stueben]
AMG coarse and fine grids

- select C-pts
- others points are F-pts
- F-pts interpolate from C-pts
AMG coarsening and interpolation

- only large $a_{ij}$, ‘strong connections’ are important
- define strength matrix $S$:

$$A = \begin{bmatrix}
  x & x & x \\
  x & x & x \\
  x & x & x & x \\
  x & x & x \\
  x & x & x & x
\end{bmatrix}$$

$$S = \begin{bmatrix}
  1 & 1 & 0 \\
  1 & 0 & 1 \\
  0 & 0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0
\end{bmatrix}$$

- consider the undirected graph of $S$
- apply parallel maximal independent set algorithms to graph($S$) [Luby, 1986]
classical AMG coarsening and interpolation

- **(C1)** Maximal Independent Set:
  - Independent: no two C-points are connected
  - Maximal: if one more C-point is added, the independence is lost

- **(C2)** All F-F connections require connections to a common C-point
  (for good interpolation)

- F-points have to be changed into C-points, to ensure (C2); (C1) is violated

more C-points, higher complexity

(Ruge, Stueben, Cleary)
AMG building blocks

Setup Phase:

- Select coarse “grids”
- Define interpolation, \( P^{(m)} \), \( m = 1,2,... \)
- Define restriction and coarse-grid operators

\[
R^{(m)} = P^{(m)T} \\
A^{(m+1)} = P^{(m)T} A^{(m)} P^{(m)}
\]

Solve Phase

Relax \( A^{(m)} u^m = f^m \)

Restrict \( P^{(m)T} \)

Solve \( A^{(m+1)} e^{m+1} = r^{m+1} \)

Interpolate \( P^{(m)} \)

Relax \( A^{(m)} u^m = f^m \)
AMG complexity - scalability

• Operator complexity $C_{op} = \frac{\sum_i \text{nonzeros}(A_i)}{\text{nonzeros}(A_0)}$

  - e.g., 3D, ideally: $C_{op} = 1 + 1/8 + 1/64 + \ldots < 8/7$

  - measure of memory use, and work in solve phase

• scalable algorithm:

  $O(n)$ operations per V-cycle ($C_{op}$ bounded)
  and
  number of V-cycles independent of $n$
  ($\rho$ independent of $n$)
Classical coarsening: scalability results

- example: finite difference Laplacian, parallel CLJP coarsening algorithm
- 2D (5-point): near-optimal scalability (250^2 dof/proc)

<table>
<thead>
<tr>
<th>Procs</th>
<th>C_{op}</th>
<th>t_{tot}</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4.48</td>
<td>2.89</td>
<td>9</td>
</tr>
<tr>
<td>64</td>
<td>4.50</td>
<td>3.85</td>
<td>9</td>
</tr>
<tr>
<td>256</td>
<td>4.50</td>
<td>5.01</td>
<td>9</td>
</tr>
</tbody>
</table>
Classical coarsening: complexity growth in some cases

- 3D (7-point): complexity growth

<table>
<thead>
<tr>
<th>dof</th>
<th>$C_{op}$</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>$32^3$</td>
<td>16.17</td>
<td>8</td>
</tr>
<tr>
<td>$64^3$</td>
<td>22.51</td>
<td>11</td>
</tr>
</tbody>
</table>

- increased memory use, long solution times, long setup times, loss of scalability
Classical coarsening: complexity growth in some cases

- 4D (9-point), 5D (11-point): complexity growth!!

<table>
<thead>
<tr>
<th></th>
<th>dof</th>
<th>$C_{op}$</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>4D</td>
<td>$20^4$</td>
<td>127.5</td>
<td>8</td>
</tr>
<tr>
<td>5D</td>
<td>$9^5$</td>
<td>256.9</td>
<td>5</td>
</tr>
</tbody>
</table>

- excessive memory use

(results by Jeff Butler)
our approach to reduce complexity: PMIS coarsening
(De Sterck, Yang, SIAM J. Matrix Analysis, 2004, submitted)

• Parallel Modified Independent Set (PMIS)
• do not enforce condition (C2)
• convergence acceleration using GMRES
• weighted independent set algorithm: points $i$ that influence many equations ($\lambda_i$ large), are good candidates for C-points
• add random number between 0 and 1 to $\lambda_i$ to break ties
• parallel algorithm!
PMIS select 1

- select C-pts with maximal measure locally
- make neighbor F-pts
- remove neighbor edges
PMIS:
remove and update 1

- select C-pts with maximal measure locally
- make neighbors F-pts
- remove neighbor edges
PMIS:
select 2

- select C-pts with maximal measure locally
- make neighbors F-pts
- remove neighbor edges
PMIS: remove and update 2

- Select C-pts with maximal measure locally
- Make neighbors F-pts
- Remove neighbor edges
PMIS: final grid

- select C-pts with maximal measure locally
- make neighbor F-pts
- remove neighbor edges
PMIS coarsening: reduce complexity

- finite difference Laplacian (CLJP - PMIS+GMRES)

<table>
<thead>
<tr>
<th>dof</th>
<th>$C_{op}$</th>
<th>Iter</th>
<th>$t_{tot}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D $120^2$</td>
<td>4.16</td>
<td>12</td>
<td>0.22</td>
</tr>
<tr>
<td>$120^2$</td>
<td>1.90</td>
<td>24</td>
<td>0.24</td>
</tr>
<tr>
<td>3D $100^3$</td>
<td>25.94</td>
<td>12</td>
<td>129.42</td>
</tr>
<tr>
<td>$100^3$</td>
<td>2.36</td>
<td>20</td>
<td>27.68</td>
</tr>
<tr>
<td>4D $20^4$</td>
<td>127.5</td>
<td>8</td>
<td>88.39</td>
</tr>
<tr>
<td>$20^4$</td>
<td>2.95</td>
<td>11</td>
<td>4.31</td>
</tr>
<tr>
<td>5D $9^5$</td>
<td>256.9</td>
<td>5</td>
<td>73.92</td>
</tr>
<tr>
<td>$8^5$</td>
<td>3.14</td>
<td>8</td>
<td>0.91</td>
</tr>
<tr>
<td>$20^5$</td>
<td>4.02</td>
<td>12</td>
<td>181.93</td>
</tr>
</tbody>
</table>
parallel scaling tests: MCR linux cluster

- study algorithmic and parallel scalability
- 2304 processors
  (2.4-GHz Xeon, 2 GB/proc)
- 11.2 TeraFlops
  ($11.2 \times 10^{12}$ floating point operations per second)
- Quadrics fast interconnection network
- at Lawrence Livermore National Laboratory, California, USA

SONAD, Waterloo, 29 April 2005
hdercker@uwaterloo.ca
**PMIS results:** 7-point finite difference Laplacian in 3D, $40^3$ dof per proc

**CLJP and PMIS-GMRES(10)**

<table>
<thead>
<tr>
<th>proc</th>
<th>$C_{op}$</th>
<th>Levels</th>
<th>$t_{setup}$</th>
<th>$t_{solve}$</th>
<th>Iter</th>
<th>$t_{total}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.39</td>
<td>15</td>
<td>1.88</td>
<td>1.47</td>
<td>6</td>
<td>3.35</td>
</tr>
<tr>
<td>512</td>
<td>17.02</td>
<td>22</td>
<td>22.33</td>
<td>13.50</td>
<td>10</td>
<td>35.83</td>
</tr>
<tr>
<td>1331</td>
<td>17.19</td>
<td>23</td>
<td>29.57</td>
<td>16.68</td>
<td>10</td>
<td>46.25</td>
</tr>
<tr>
<td>1</td>
<td>2.32</td>
<td>7</td>
<td>0.41</td>
<td>0.87</td>
<td>13</td>
<td>1.28</td>
</tr>
<tr>
<td>512</td>
<td>2.37</td>
<td>10</td>
<td>5.04</td>
<td>7.73</td>
<td>25</td>
<td>12.77</td>
</tr>
<tr>
<td>1331</td>
<td>2.37</td>
<td>10</td>
<td>8.28</td>
<td>9.71</td>
<td>28</td>
<td>17.99</td>
</tr>
</tbody>
</table>
(D) future work

- scalable multigrid solvers for PDEs of **hyperbolic** type
  - O(n) scaling for hyperbolic PDE systems is difficult
  - AMG is a challenge
- scalable solvers for **nonlinear** PDE systems: use nested iteration

- improve **scalability** for very large machines...
### Top 500 Supercomputer list (November 2004)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer</th>
<th>Country</th>
<th>TeraFlops</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawrence Livermore National Laboratory</td>
<td>IBM BlueGene/L</td>
<td>US</td>
<td>135</td>
<td>65,536</td>
</tr>
<tr>
<td>2</td>
<td>NASA/Ames Research Center/NAS</td>
<td>SGI Altix</td>
<td>US</td>
<td>51</td>
<td>10,160</td>
</tr>
<tr>
<td>3</td>
<td>The Earth Simulator Center</td>
<td>NEC Earth-Simulator</td>
<td>Japan</td>
<td>35</td>
<td>5,120</td>
</tr>
<tr>
<td>4</td>
<td>Barcelona Supercomputer Center</td>
<td>IBM eServer</td>
<td>Spain</td>
<td>20</td>
<td>3,564</td>
</tr>
<tr>
<td>5</td>
<td>Lawrence Livermore National Laboratory</td>
<td>Intel Itanium2</td>
<td>US</td>
<td>19</td>
<td>4,096</td>
</tr>
<tr>
<td>6</td>
<td>Los Alamos National Laboratory</td>
<td>ASCI Q - HP AlphaServer</td>
<td>US</td>
<td>13</td>
<td>8,192</td>
</tr>
<tr>
<td>7</td>
<td>Virginia Tech</td>
<td>1100 Dual 2.3 GHz Apple XServe</td>
<td>US</td>
<td>12</td>
<td>2,200</td>
</tr>
<tr>
<td>8</td>
<td>IBM - Rochester</td>
<td>IBM BlueGene/L</td>
<td>US</td>
<td>11</td>
<td>8,192</td>
</tr>
<tr>
<td>9</td>
<td>Naval Oceanographic Office</td>
<td>IBM eServer</td>
<td>US</td>
<td>10</td>
<td>2,944</td>
</tr>
<tr>
<td>10</td>
<td>NCSA</td>
<td>Dell P4 Xeon</td>
<td>US</td>
<td>10</td>
<td>2,500</td>
</tr>
<tr>
<td>11</td>
<td>ECMWF</td>
<td>IBM eServer</td>
<td>UK</td>
<td>10</td>
<td>2,176</td>
</tr>
<tr>
<td>12</td>
<td>ECMWF</td>
<td>IBM eServer</td>
<td>UK</td>
<td>10</td>
<td>2,176</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Shanghai Supercomputer Center</td>
<td>Dawning 4000A, Opteron</td>
<td>China</td>
<td>8</td>
<td>2,560</td>
</tr>
<tr>
<td>18</td>
<td>Los Alamos National Laboratory</td>
<td>LNX Opteron</td>
<td>US</td>
<td>8</td>
<td>2,816</td>
</tr>
<tr>
<td>19</td>
<td>Lawrence Livermore National Laboratory</td>
<td>MCR Linux Cluster Xeon</td>
<td>US</td>
<td>8</td>
<td>2,304</td>
</tr>
<tr>
<td>20</td>
<td>Lawrence Livermore National Laboratory</td>
<td>ASCI White, IBM SP Power3</td>
<td>US</td>
<td>7</td>
<td>8,192</td>
</tr>
</tbody>
</table>

- scalable results were presented for MCR (#19), 2,000 procs
- next target: Blue Gene/L (#1), 65,000 procs
LLNL Blue Gene/L

- dual-processor nodes optimized for data access
- each node: one processor for simulation, one for communication
- only 256MB ram per processor
- lightweight, single-process linux kernel
- Blue Gene/L will be fully operational in July 2005, with 130,000 procs
LLNL Blue Gene/L

- our code currently runs on LLNL Blue Gene/L

- one preliminary result, on 8,000 processors: PMIS works, CLJP runs out of memory...

- more tests to follow...
- scalability up to 130,000 processors?
collaborators

• Ulrike Yang, Rob Falgout
  Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, California
• Tom Manteuffel, Steve McCormick, John Ruge
  University of Colorado at Boulder
• Jeff Heys
  Arizona State University
• Jeff Butler
  University of Waterloo