SuperSolvers: Hybrid, Adaptive and Composite Solvers

Padma Raghavan
Sanjukta Bhowmick and Keita Teranishi
Scalable Computing Laboratory
Department of Computer Science and Engineering
The Pennsylvania State University
Supported by the National Science Foundation
ACI-0102537 and DMR-0205232.

Lois C. McInnes and Boyana Norris
MCS, Argonne National Lab
SuperSolvers = Multimethod Sparse Solvers

- Many methods for **sparse** Ax=b, Complex tradeoffs
- Application needs vary
  - Accuracy, scaling
  - Conditioning – easy to hard even in one simulation
- Supersolvers: Automatically combine multiple methods for best solver performance
- For semi-implicit PDE based simulations

- When A is symmetric, positive definite
  - Direct
  - Iterative: CG
  - Multilevel, multigrid
- When A is nonsymmetric
  - Direct
  - KSP variants: GMRES, QMR, BiCG, CGSTAB, ...
- Multilevel schemes
- Preconditioning
  - Incomplete (variant of direct)
  - Sparse Approximate inverses
  - Multilevel
  - KSP
  - Smoothers, coarse grid solvers

"The impossibility of uniformly ranking linear system solvers in order of effectiveness...is widely appreciated" Keyes ...
Why SuperSolvers? ... Horror Matrices!

Matrix 1: augustus7
- Rank: 1,060,864
- Nz: 9,313,87
  (Kershaw sq. mesh)

Matrix 2: ldoor
- Rank: 952,203
- Nz: 46,522,475

Matrix 3: af_shell3
- Rank: 504,855
- Nz: 17,588,875

- Ill conditioned
- Complex sparsity structure
- Most methods fail except direct
- Direct has too much fill
SuperSolvers

- **Hybrid solvers**
  - Flexible direct- to- iterative, through preconditioning
    - Focus on parallel scalability, limiting memory, faster convergence
  - Keita Teranishi: Session P-1

- **Adaptive solvers** to reduce total simulation time
  - A single method is dynamically selected to match linear system properties

- **Composite solvers** for increased reliability
  - A sequence of methods are applied to the same linear system
Related Work

- Ern, Giovangili, Keyes and Smooke --- motivates “polyalgorithmic” solvers through empirical study on nonlinear elliptic PDEs (*SIAM J. Sci. Comp*).


- Barrett, Berry, Dongarra, Eijkhout and Romine --- multiprocessor implementation, different Krylov methods are applied in parallel to the same system (A is replicated at least in part).

- Bhowmick, Keyes – machine learning approaches
Nonlinear PDE-Based Simulations

- Partial differential equations (nonlinear) representing models, discretized and solved numerically
- Nonlinear systems are solved using Newton’s method
- Sparse linear systems at each nonlinear iteration

\[ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \]

- Simulation time is dominated by the time for linear solves
- Numerical properties of linear systems change during nonlinear iterations
Example: Driven Cavity Flow

- Velocity-vorticity formulation
- Flow driven by lid and/or buoyancy
- Logically regular grid, parallelized with DAs
- Finite difference discretization
- Grashof number and lid velocity determine the degree of nonlinearity

Solution Components

- velocity: $u$
- velocity: $v$
- vorticity: $\zeta$
- temperature: $T$

Governing Equations

\[ - \Delta u_x + \frac{\partial w}{\partial y} = 0; \quad - \Delta u_y + \frac{\partial w}{\partial x} = 0; \]
\[ - \Delta w + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} - Gr \frac{\partial T}{\partial x} = 0; \]
\[ - \Delta T + Pr \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) = 0; \]

Boundary condition: $w(x, y) = -\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$

Application code author: David Keyes

Figures from PETSc 2.1.3 tutorial
• Adaptive Solvers
Adaptive Solvers

- Time for linear solve across nonlinear iterations, solver unchanged
- Time varies

**Goal:** Reduce the total execution time by dynamically selecting the right linear solver
Adaptive Solver Selection

- Window of consecutive nonlinear iterations
- Switch at each window / criterion

**Sequence based switching**
- Methods in increasing order of time per iteration $t_i$
- Assumption: smaller $t_i \Rightarrow$ weaker method
- Selection of one before/one after
- Suitable if systems get progressively easier (or difficult) to

**Non-sequence based switching**
- All methods considered during selection
- Prediction using polynomials, wave function fits
Switching: Sequence Based

- **Switching based on:**
  - Incremental convergence rate
  - Rate of change in the number of linear iterations

- **Historic/user specified range** \([-\lambda, \lambda]\) for the convergence rate

- **Switching heuristic:**
  - Convergence rates are within the boundary: do not switch
  - Convergence rates exceed the upper bound: switch to preceding solver
  - Convergence rates are lower than the lower bound Or
  - Rate of increase of linear iterations is greater than a threshold \(\beta\)
    - Switch to succeeding solver
Switching: Interpolation Based

• Switching based on
  – Predicted time to solve remaining linear systems

• Predictions based on polynomial interpolation
  – Calculate number of nonlinear steps, $N$ required, from polynomial interpolation of rate of convergence
  – Time for linear solution: method $M_j$ at nonlinear iteration $t$ is given by an $n$-ordered polynomial

  $$ P_n^j (t) = \sum_{i=0}^{i=n} a_i^j t^i $$

  n - phases

• Switch based on predicted least cumulative time
  – At iteration $x$ switch to method $M_k$ such that

  $$ \sum_{t=x}^{t=m} P_n^k (t) = \min \left( \sum_{t=x}^{t=m} P_n^j (t) : \forall M_j \right) $$
Adaptive Solvers:
Driven Cavity Flow

Time per Nonlinear Iteration
Adaptive Solvers: Cumulative Time

Cumulative Time Over Nonlinear Iterations
Adaptive Solvers: Driven Cavity, Summary

- 39%-32% faster on average (30 simulations)
Adaptive Solvers: FUN3D

Time per Nonlinear Iteration

- Adaptive Switching is much better
- Within 10% of hand-optimized switching
• Composite Solvers
Composite Solvers

- Composite solver --- a specific ordered sequence of distinct methods

- A method fails if it does not convergence within \textit{max iterations}

- Upon failure of one method, the next method in the sequence is executed on the \textit{same} system

- The composite fails \textit{only if} all methods fail

- In the worst case, all methods have to be executed
A Combinatorial Model

- There are $n$ methods $M_1, M_2, \ldots, M_n$.
  - Method $M_i$ is associated with metrics:
    - normalized execution time $t_i$
    - probability of not converging (failure rate) $f_i$ or $f(i)$; ($r_i = 1 - f_i$ (reliability))
  - Cumulative failure rate of methods $M_1, M_2, \ldots, M_n$ is $f(1 \cap 2 \cap \ldots \cap n)$

The cumulative failure rate is lower than the individual failure rates of the methods

- The set $P$ contains all permutations of $\{1, 2, \ldots, n\}$
- $M_i$ denotes the $i$-th method in $P \in P$
- The composite $C$ executes methods in the order specified by $P$

Goal: Find ordering for the composite with the least time in the worst case (on average)
**Composite Solvers: Model**

- Assume that failures are mutually independent

- Reliability of $\hat{C}$
  \[
  \hat{F} = \prod_{i=1}^{n} \hat{f}_i ; \quad \hat{R} = 1 - \hat{F}
  \]
  
  - The reliability is independent of the ordering
  
  - The reliability is higher than any of the base method

- Execution time (worst case) of $\hat{C}$
  \[
  \hat{T} = \hat{t}_1 + \hat{f}_1 \hat{t}_2 + \ldots + \hat{f}_1 \hat{f}_2 \ldots \hat{f}_{n-1} \hat{t}_n
  \]
  
  - The execution time depends on the ordering

- Goal: To determine an optimal permutation
  \[
  \hat{T} = \min \{ \hat{T} : \hat{P} \in P \} \]
The Optimal Composite is ...

- Define $u_i = t_i / r_i$ as the utility ratio of $M_i$

- Let $P \in \mathcal{P}$ and let $C$ be the associated composite

- **Theorem:** $C$ is the optimal composite with $T = \min \{T : P \in \mathcal{P} \}$ if and only if the sequence of methods $M_1, M_2, \ldots, M_n$ is such that
  
  $$u_1 \leq u_2 \leq \ldots \leq u_{n-1} \leq u_n$$

- **Optimal** composite = methods in *increasing* order of utility ratios
Proof Sketch

• **Part I:** *If the utility ratios are in increasing order then the composite is optimal*

  • **By induction**
    
    Utility ratio of any composite ≤ utility ratio of any base method

  • **By contradiction**
    
    Any composite with utility ratios not in increasing order is not optimal

• **Part II:** *If the composite is optimal then the utility ratios are in increasing order*

  • This is proved by considering composites as paths in a layered graph
Composites as Paths in a Graph

Method M1; Time = 1.0; Failure = 0.9
Method M2; Time = 1.5; Failure = 0.3
Method M3; Time = 3.0; Failure = 0.2

The vertices on the shortest path gives the optimal composite.
Composites: Example

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Reliability</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1.0</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>M2</td>
<td>1.5</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>M3</td>
<td>3.0</td>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Least Time: (1,2,3)

Least Failure: (3,2,1)

Optimal: (2,3,1)

The failure rate of any composite is 0.054
Execution Times

- CU requires only 56% of the linear solution time required by a base method
Parallel Implementation

A parallel composite = sequence of fully parallel linear solvers

- The matrix is distributed across $P$ processors
- Methods are assumed to use the same matrix distribution across $P$ processors
- The next method can be invoked on the same system without matrix redistribution
Execution Times

- CU is ~ 40%-48% of the best base method
Scalability of Parallel Composites

- Speedups of composite CU = best base method
  - From iteration scaling
Supersolver Codes

- Initial implementation using PETSc (www.mcs.anl.gov/petsc)
- Can collect performance metrics without high overhead

Diagram:

Application Driver

Nonlinear Solvers (SNES)

Linear Solvers (SLES)

PC

KSP

Solve

F(u) = 0

Multi-method Solver

Application Initialization

Function Evaluation

Jacobian Evaluation

Post Processing
**SuperSolver System**

- Multimethod Algorithms
  - Multimethod Solver 1
  - Multimethod Solver 2
  - Multimethod Solver 3

- **MainServer**
  - Application 1
  - Application 2
  - Application 3

- **External Libraries**
  - Hypre
  - Trilinos
  - PETSc

**Issues:** variations in languages, implementations, architectures, ...Bhowmick+ Curfman+Norris in ANL
Conclusions

• Serial and parallel hybrid, adaptive and composite solvers

• Hybrids: Direct-Iterative through preconditioning
  – Limiting memory, multiprocessor communication latency-tolerance for preconditioner application

• Adaptive Solvers: dynamic method selection, i.e., faster linear solution for faster simulations
  – Polynomial model of solver time for prediction
  – FUN3D: ~ close to ideal

• Composite Solvers: reliable linear solution for faster simulations
  – Optimal composite: combinatorial model
  – Optimal composites can improve nonlinear convergence rates
  – Application time – halved, improved scalability

• Parallel performance/scalability as good as base methods or better from iteration scaling
### Adaptive Solvers

**Driven Cavity Flow: Problem Parameters**

<table>
<thead>
<tr>
<th>Base Methods</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Krylov Method</strong></td>
<td>BiCG</td>
<td>GMRES(30)</td>
<td>TFQMR</td>
<td>BiCG</td>
</tr>
<tr>
<td><strong>Preconditioner</strong></td>
<td>ILU(2)</td>
<td>ILU(2)</td>
<td>ILU(2)</td>
<td>ILUT</td>
</tr>
</tbody>
</table>

- ILU(l): Incomplete LU with l level of fill
- ILUT: Incomplete LU with drop threshold .0001

- **Mesh Size:** 128 by 128
- **Prandtl Number:** 1
- **Grashof Numbers:** [400, 450, 500, 580, 600, 650]
- **Lid Velocity:** [10, 13, 15, 20, 25]
Failures and Reliability

Driven Cavity: 24 simulations
Driven Cavity Flow: Numerical Solution

- Uses inexact Newton Method with pseudo-transient continuation to solve systems with high nonlinearity

- Newton’s Method:
  Solve \( f(u) = 0; \quad f'(u^{'-1})\delta u' = -f(u^{'-1}); \quad u' = u^{'-1} + \alpha \delta u' \)

- Pseudo-transient Continuation, nonlinear equation is modified to:
  \( \frac{1}{\tau^l} + f(x) = 0; \quad \tau^l = \text{pseudo timestep at iteration} \ l \)

- Uses a Krylov iterative method to solve approximately the Newton correction equation
## Composite Solvers: Problem Parameters

<table>
<thead>
<tr>
<th>Base Methods</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krylov Method</td>
<td>GMRES(30)</td>
<td>TFQMR</td>
<td>GMRES(30)</td>
<td>TFQMR</td>
</tr>
<tr>
<td>Preconditioner</td>
<td>ILU(1)</td>
<td>ILUT</td>
<td>ILU(0)</td>
<td>ILU(0)</td>
</tr>
<tr>
<td>Ordering</td>
<td>QMD</td>
<td>RCM</td>
<td>RCM</td>
<td>RCM</td>
</tr>
</tbody>
</table>

ILU(l): Incomplete LU with l level of fill
ILUT: Incomplete LU with drop threshold .01

Mesh Size: 128 by 128
Prandtl Number: 1
Grashof Numbers: [580, 620, 660, 700, 740, 780]
Lid Velocity: [10, 13, 16, 20]

| Composites (values of metrics obtained through sampling) |
|--------|--------|--------|--------|--------|
| CU     | B2     | B3     | B1     | B4     |
| C1     | B3     | B1     | B2     | B4     |
| C2     | B4     | B3     | B2     | B1     |
| C3     | B2     | B1     | B3     | B4     |
## Composite Solvers: Problem Parameters

<table>
<thead>
<tr>
<th>Base Methods</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Krylov Method</strong></td>
<td>GMRES(30) RASM(1) Jacobi</td>
<td>GMRES(30) RASM(1) SOR</td>
<td>TFQMR RASM(3) ILU(0)</td>
<td>TFQMR RASM(4) ILU(0)</td>
</tr>
<tr>
<td><strong>Preconditioner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subdomain</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Solver</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mesh Size: 128 by 128
Prandtl Number: 1
Grashof Numbers: [700 750 800 850 900 950]
Lid Velocity: [73 80 83 85]

### Composites

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td></td>
<td></td>
<td>B3</td>
<td>B4</td>
</tr>
<tr>
<td>C1</td>
<td>B3</td>
<td></td>
<td>B2</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>B2</td>
<td>B1</td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>C3</td>
<td>B4</td>
<td>B1</td>
<td></td>
<td>B2</td>
</tr>
</tbody>
</table>
# Summary of Results

For 24 Simulations on Sequential Implementation

## Average Performance Data

<table>
<thead>
<tr>
<th>Average Metric</th>
<th>Base1</th>
<th>Base2</th>
<th>Base3</th>
<th>Base4</th>
<th>Mean</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Solution Time</td>
<td>1,083</td>
<td>1,000</td>
<td>1,041</td>
<td>1,291</td>
<td>1,104</td>
<td>1,166</td>
<td>1,458</td>
<td>583.3</td>
<td>541.6</td>
</tr>
<tr>
<td>Nonlinear Solution Time</td>
<td>1,125</td>
<td>1,041</td>
<td>1,083</td>
<td>1,333</td>
<td>1,145</td>
<td>1,208</td>
<td>1,500</td>
<td>625</td>
<td>604</td>
</tr>
<tr>
<td>Linear Iteration Count</td>
<td>1,125</td>
<td>625</td>
<td>1,416</td>
<td>1,166</td>
<td>1,083</td>
<td>1,205</td>
<td>1,291</td>
<td>391.6</td>
<td>391.6</td>
</tr>
<tr>
<td>Nonlinear Iteration Count</td>
<td>5.2</td>
<td>3.1</td>
<td>5.8</td>
<td>5</td>
<td>4.7</td>
<td>3.1</td>
<td>3.1</td>
<td>2.7</td>
<td>2.3</td>
</tr>
<tr>
<td>Number of Failures</td>
<td>4.0</td>
<td>.41</td>
<td>5</td>
<td>4.1</td>
<td>3.41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Summary of Results
For 24 Simulations over 8 processors

Average Performance Data

<table>
<thead>
<tr>
<th>Average Metric</th>
<th>Base 1</th>
<th>Base2</th>
<th>Base3</th>
<th>Base4</th>
<th>Mean</th>
<th>CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Solution Time</td>
<td>19.59</td>
<td>18.11</td>
<td>10.14</td>
<td>10.9</td>
<td>14.6</td>
<td>9.3</td>
</tr>
<tr>
<td>Nonlinear Solution Time</td>
<td>21.74</td>
<td>19.69</td>
<td>10.8</td>
<td>11.59</td>
<td>15.9</td>
<td>10.02</td>
</tr>
<tr>
<td>Linear Iteration Count</td>
<td>4133.3</td>
<td>3363.3</td>
<td>867.2</td>
<td>893.4</td>
<td>2464.3</td>
<td>807</td>
</tr>
<tr>
<td>Nonlinear Iteration Count</td>
<td>23.66</td>
<td>17.5</td>
<td>5.83</td>
<td>5.87</td>
<td>13.23</td>
<td>5.1</td>
</tr>
<tr>
<td>Number of Failures</td>
<td>23.6</td>
<td>16.58</td>
<td>1.5</td>
<td>1.41</td>
<td>10.8</td>
<td>0</td>
</tr>
</tbody>
</table>

Speedup and Efficiency

<table>
<thead>
<tr>
<th>Metric</th>
<th>Base 1</th>
<th>Base2</th>
<th>Base3</th>
<th>Base4</th>
<th>CU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>.47</td>
<td>.52</td>
<td>.95</td>
<td>.88</td>
<td>1.02</td>
</tr>
<tr>
<td>Speedup</td>
<td>3.11</td>
<td>4.16</td>
<td>7.59</td>
<td>7.07</td>
<td>8.18</td>
</tr>
</tbody>
</table>