Fault-Tolerant Multigrid Algorithms Based on Error Confinement and Full Approximation

Mirco Altenbernd and Dominik Göddeke

SIAM PP ’16
MS49: Resilience Towards Exascale Computing
April 2016

Institute for Applied Analysis and Numerical Simulation
Chair Computational Mathematics for Complex Simulations
The resilience challenge

Some important observations

- More components at exascale $\Rightarrow$ higher probability of failure
- Active debates to sacrifice reliability for energy efficiency
- MTBF $< 1\,\text{h} \Rightarrow$ all simulation software must be prepared

Classical techniques

- Reliability in hardware (ECC protection etc.) too power-hungry
- Checkpoint-restart too memory-intensive (and too slow)
- Triple modular redundancy too power-hungry, but: can be more energy-efficient and faster for large fault rates
Focus of this talk: ABFT for multigrid

General concept: algorithm-based fault tolerance
- Exploit algorithmic properties to detect and correct faults
- Can be more efficient than middleware

Previous work
- Scenario: Node loss
- Self-stabilisation properties

In this talk: Silent Data Corruption
- Scenario: Bitflips
- Black-box smoother protection
Resilient multigrid
Silent data corruption

- Soft transient faults that lead to wrong solutions
- Sometimes noticeable a posteriori (divergence), mostly not
- Causes: radiation, leaking voltage, silicon ageing, ... 

Core idea of our approach

- Use FAS multigrid to increase robustness
- Based on nonlinear MG: not just a correction on each level but a true approximation of the solution
- Linear case: numerically equivalent, less than one fine SPMV overhead per cycle
- One additional vector necessary
Focus on smoother stage

Relative cost of the smoother stage as a function of the number of smoothing steps

- Detailed model: in the paper
Theoretical justification for the down-cycle
- Obvious: residual $r$ converges to zero on finest grid
- Easy to prove: residual (monotonously) converges to zero on all grids

Theoretical justification for the up-cycle
- Slightly non-trivial proof: correction vector $c$ converges (monotonously) to zero on all grids

Consequence: good fault indicators
- Both readily available without additional computation
Black-box smoother protection

Practical realisation after smoothing on level $k$

- Compute index set of possibly faulty components of $c$ or $r$ by comparing against level-specific threshold from earlier iteration
- Extend by one layer of indices coupled by $A$
- Replace faulty components by unsmoothed values (down-cycle), or by recomputed correction from (non-faulty) coarser grid
- Adaptively update threshold
\begin{Verbatim}
1 \mathcal{L} = \text{detect\_and\_localise}(c, t_k^c) \quad \text{// detect faulty components}
2 \text{if } \mathcal{L} \neq \emptyset \text{ then}
3 \quad \tilde{v} = I_k^{k-1} v^{(0)} \quad \text{// restrict (non-faulty) initial approximation}
4 \quad \tilde{c} = u_{k-1} - \tilde{v} \quad \text{// calculate new correction vector}
5 \quad \tilde{c} = P_{k-1}^k \tilde{c} \quad \text{// prolongate new correction vector}
6 \quad \text{for } i \in \mathcal{L} \text{ do}
7 \quad \quad c(i) = \tilde{c}(i) \quad \text{// replace faulty components}
8 \quad \text{end}
9 \quad t = \text{calc\_threshold}(c) \quad \text{// calculate new threshold value}
10 \quad q = t / t_k^c \quad \text{// relative reduction of threshold value}
11 \quad \text{if } q > 1 \text{ then}
12 \quad \quad \text{for } i \in \{0, \ldots, L\} \setminus \{k\} \text{ do}
13 \quad \quad \quad t_i^c = t_i^c \cdot 10 \cdot q \quad \text{// store threshold value}
14 \quad \quad \text{end}
15 \quad \text{end}
16 \quad t_k^c = t \quad \text{// calculate and store threshold value}
17 \text{else}
18 \quad t_k^c = \text{calc\_threshold}(c)
19 \text{end}
\end{Verbatim}
Checksum protection for transfer stage

Checksums

- Use identity $1^T(Ax + y) = (1^TA)x + 1^Ty$
- Precompute $1^TA$ (column sums)
- Fault detection in $Ax + y$ by three dot products
- More elaborate schemes: detect and correct errors

Combined approach

- Black-box smoother protection, checksums for the rest
- Runtime comparison, fault-free case

<table>
<thead>
<tr>
<th></th>
<th>unprotected</th>
<th>transfer stage (checksums)</th>
<th>smoothing stage (new algorithm)</th>
<th>FTMG (both)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean time</td>
<td>26.1562</td>
<td>27.4128</td>
<td>26.7905</td>
<td>27.9090</td>
</tr>
<tr>
<td>factor</td>
<td>1</td>
<td>1.0480</td>
<td>1.0243</td>
<td>1.0670</td>
</tr>
</tbody>
</table>
Numerical experiments

Test problems
- Different flavours of anisotropic diffusion-convection-reaction
- 2D, $Q_1$ FEM, eight levels, 263169 DOF

Solver configuration
- FAS-MG, V-cycle, 4+4 Jacobi $\omega = 0.7$ smoothing steps
- Bilinear interpolation for restriction and prolongation
- Natural injection for approximations

Fault injection
- 1% bitflip probability in current approximation after each smoothing step on each level (deliberately unrealistically high)
- Actual component and bit fully random
- 100 repetitions per experiment
Numerical experiments

Histograms of iterations for poisson, anisotropic diffusion (andi), diffusion-convection (dico) and anisotropic diffusion-convection-reaction (andicore)
### Numerical experiments (Summary)

<table>
<thead>
<tr>
<th>problem</th>
<th>#faults</th>
<th>#detects</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>direct</td>
<td>delayed</td>
<td>fault-free</td>
<td>FTMG</td>
<td>unprotected</td>
</tr>
<tr>
<td>poisson</td>
<td>3.82</td>
<td>1.89</td>
<td>0.61</td>
<td>7</td>
<td>7.79</td>
<td>10.82 (3)</td>
</tr>
<tr>
<td>andi</td>
<td>10.24</td>
<td>4.94</td>
<td>0.49</td>
<td>18</td>
<td>18.25</td>
<td>29.09 (15)</td>
</tr>
<tr>
<td>dico</td>
<td>4.92</td>
<td>2.01</td>
<td>0.39</td>
<td>8</td>
<td>8.46</td>
<td>11.89 (4)</td>
</tr>
<tr>
<td>andicore</td>
<td>5.77</td>
<td>2.40</td>
<td>0.25</td>
<td>10</td>
<td>10.55</td>
<td>16.14 (9)</td>
</tr>
</tbody>
</table>

- Divergent runs in brackets
- Some faults not directly detected: delayed repair with ‘faultier’ correction
Detailed analysis of one andi test

- Blue bump: delayed repair with already faulty correction
- Two other significant faults: well repaired
Detailed analysis of one andi test

<table>
<thead>
<tr>
<th>iter</th>
<th>cycle</th>
<th>level</th>
<th>step</th>
<th>change (%)</th>
<th>#det</th>
<th>#corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>up</td>
<td>6</td>
<td>1</td>
<td>9.99e+1</td>
<td>9</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>up</td>
<td>7</td>
<td>3</td>
<td>9.38e+1</td>
<td>25</td>
<td>49</td>
</tr>
<tr>
<td>18</td>
<td>down</td>
<td>7</td>
<td>4</td>
<td>1e+2</td>
<td>9</td>
<td>25</td>
</tr>
</tbody>
</table>
### Detailed analysis of one andi test

<table>
<thead>
<tr>
<th>iter</th>
<th>cycle</th>
<th>level</th>
<th>step</th>
<th>change (%)</th>
<th>#det</th>
<th>#corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>down</td>
<td>5</td>
<td>3</td>
<td>2.14e-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>down</td>
<td>5</td>
<td>4</td>
<td>2.72e-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>up</td>
<td>6</td>
<td>1</td>
<td>9.99e+1</td>
<td>9</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>up</td>
<td>5</td>
<td>3</td>
<td>5.55e-10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>down</td>
<td>7</td>
<td>3</td>
<td>1.81e-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>up</td>
<td>7</td>
<td>3</td>
<td>9.38e+1</td>
<td>25</td>
<td>49</td>
</tr>
<tr>
<td>9</td>
<td>down</td>
<td>4</td>
<td>2</td>
<td>6.75e-3</td>
<td>49</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>up</td>
<td>1</td>
<td>2</td>
<td>4.47e-4</td>
<td>17</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>up</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>up</td>
<td>1</td>
<td>1</td>
<td>1.25e-4</td>
<td>28</td>
<td>54</td>
</tr>
<tr>
<td>18</td>
<td>down</td>
<td>7</td>
<td>4</td>
<td>1e+2</td>
<td>9</td>
<td>25</td>
</tr>
</tbody>
</table>

- Irrelevant faults (small changes) are tolerated
- Small additional corrections (red)
Performance results
Summary
Summary and acknowledgements

- Fault tolerance becomes increasingly important
- Black-box ‘computer science’ techniques exist and work well, but: substantial overhead
- ABFT techniques may do better
- In this talk: some (early) ideas for multigrid

Papers and more information
- Silent data corruption: almost submitted

DFG Priority Program 1648 ‘Software for Exascale Computing’, grants GO 1758-2/1 and -2/2