Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey

Workshop on Resiliency
Los Alamos Computer Science Symposium 2009

October 14, 2009
Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Motivations

Model Problem

FT Approach

Numerical Reconstructions

Performance Comparisons

Demo

Results

Conclusion

Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Motivations

Why fault tolerance?

1. The new generation of Exascale HPC Systems have increasingly complex architectures.

2. The Mean Time Between Failures (MTBF) of previous parallel computers used to be several weeks and now has dramatically decreased to just few minutes (Engelmann and Al Geist, 2002).

3. The Message Passing Interface (MPI) does not address the FT issue which is NOT an option for long-running and critical Applications.
1. Motivations

2. Model Problem

3. FT Approach

4. Numerical Reconstructions

5. Performance Comparisons

6. Demo

7. Results

8. Conclusion
The Parabolic 3D HEAT equation I

- The 3D heat equation writes

\[
\frac{\partial u}{\partial t} = \Delta u + F(x, y, z, t), \ (x, y, z, t) \in \Omega \times (0, T)
\]

\[
u|_{\partial \Omega} = g(x, y, z), \ u(x, y, z, 0) = u_0(x, y, z).
\]

We suppose that the time integration is done by a first order implicit Euler scheme,

\[
\frac{U^{n+1} - U^n}{dt} = \Delta U^{n+1} + F(x, y, z, t^{n+1}), \quad (1)
\]

and that \( \Omega \) is partitioned into \( N \) subdomains \( \Omega_j, j = 1..N \).
The Parabolic 3D HEAT equation II

- Example: Domain Decomposition with 5 subdomains allocated to 5 distributed computing nodes
The Parabolic 3D HEAT equation III

Why heat transfer problem?

1. Long time running numerical simulations
2. Reversibility in time not so easy (numerical instability)
3. ill-posed by nature
4. Reconstruction process of the lost data quite complex and challenging
5. NOT the case for
   - Elliptic operators
   - Linear Hyperbolic operators
Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Our Fault Tolerant Approach I

The design of Fault Tolerant applications is based on 2 complementary features:

1. Guaranty that if one or several machines get disconnected the code can still be executed

2. Provide an algorithm that can restart the time integration from the data that are available in the distributed memory or file system
   \[\implies (B)\] Non-Blocking, Periodic and Uncoordinated checkpointing
Our Fault Tolerant Approach II

(A) FT-MPI:
1. Ensures the Application will keep on running in presence of failures by respawning the failed processes. It does not provide any Checkpointing features.
2. Responsibility of the Application to recover the data-structures on the crashed processes.
3. The user has the ability to checkpoint only the minimum required data to restart the whole Application.

(B) Non-Blocking / Periodic / Uncoordinated Checkpointing:
1. Implementation at Application-Level
2. Two groups of processes:
   ⇒ the *solver* group which actually solves the problem and perform data checkpointing
   ⇒ the *spare* group which, so far, only saves the checkpoints coming from the *solver* processes
Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Now, we assume that we have detected a failure and need to restart the computation of the time dependent problem.

From the data stored in the spare processors main memory, we can rebuild in parallel the lost data on a subdomain using the following techniques:

1. The Forward Implicit Time Integration
2. The Backward Explicit Time Integration
Numerical Reconstruction Algorithms II

- The Forward Implicit Time Integration:

  $\Rightarrow$ Save $U$ at each $K$ time step interval and additionally, $BC$ at each time step

**Figure:** Available data on main memory processors before starting the reconstruction algorithms.

**Figure:** Reconstruction procedure in one dimension using forward time integration.
Numerical Reconstruction Algorithms III

- The Backward Explicit Time Integration:

  ➔ Save U only at each K time step interval

![Diagram ofbackward explicit time integration](image)

**Figure**: Available data on main memory processors before starting the reconstruction algorithms.

**Figure**: Reconstruction procedure in one dimension using explicit backward time stepping.

- Reconstruction Procedure in 3 successive computational steps
Explicit Backward Integration

\[ U^n = U^{n+1} - dt \Delta U^{n+1} - F(x, y, z, t^{n+1}) \]  \hspace{1cm} (2)

- The solution is granted by the Implicit Forward Integration in time (1)
- **This works only for few time steps!**
- Continue then with an hyperbolic regularization, the Telegraph equation (Eckhaus and Garbey, SIAM, 1990)

\[ \epsilon \frac{\partial^2 U}{\partial t^2} - \left( \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} \right) + \frac{\partial U}{\partial t} = F(x, y, z, t), \quad x\ y\ z \in (0, 1)^3, \quad t \in (0, T) \]  \hspace{1cm} (3)

- The smaller \( \epsilon \), the more unstable the scheme (3) and the flatter the cone of dependence
- The smaller \( \epsilon \), the better the asymptotic approximation

Space Marching Scheme (Beck and Murio, SIAM, 1994) on each direction: standard procedure in inverse heat problem

\[ \frac{U^n_{j+1} - 2U^n_j + U^n_{j-1}}{h^2} = \frac{U^{n+1}_j - U^{n-1}_j}{2dt} + F^n_j, \]  \hspace{1cm} (4)

Application of the Implicit Forward Integration reduced to the failed subdomain
Numerical Reconstruction Algorithms V

Figure: Numerical accuracy and order of the 3D overall reconstruction algorithm.
Performance Comparisons I

Performance of the 2 PFT Algorithm on a cluster of 56 dual Itanium2 processor (900MHz) nodes, all connected to a gigabit ethernet network.

1. small $\leftrightarrow (10 \times 10 \times 50)$
2. medium $\leftrightarrow (15 \times 15 \times 76)$
3. large $\leftrightarrow (18 \times 18 \times 98)$
Figure: Overhead of saving the local solutions with 36, 49, 64 and 81 processes: Small data set.
Figure: Overhead of saving the local solutions with 36, 49, 64 and 81 processes: **Medium** data set.
Figure: Overhead of saving the local solutions with 36, 49, 64 and 81 processes: Large data set.
Demo I

Time Step = 1 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo II

Time Step = 1 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo III

Time Step = 1 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo IV

Time Step = 2 ; Checkpoint Interval = 10

Solver processes

Spare processes

Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Demo V

Time Step = 2 ; Checkpoint Interval = 10
Demo VI

Time Step = 3 ; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 3 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo VIII

Time Step = 4 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo IX

Time Step = 4 ; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 5 ; Checkpoint Interval = 10
Time Step = 5 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XII

Time Step = 6 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XIII

Time Step = 7 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XIV

Time Step = 8 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XV

Time Step = 9 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XVI

Time Step = 10 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XVII

Time Step = 11 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XVIII

Time Step = 11 ; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 12; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 12 ; Checkpoint Interval = 10
Demo XXI

Time Step = 13 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXII

Time Step = 13 ; Checkpoint Interval = 10

Solver processes

Spare processes

Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Time Step = 14 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXIV

Time Step = 14 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXV

Time Step = 15 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXVI

Time Step = 15 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXVII

\[ \text{Time Step } = 16 \; ; \; \text{Checkpoint Interval } = 10 \]
Demo XXVIII

Time Step = 17 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXIX

Time Step = 18 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXX

Time Step = 18 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXXI

**Time Step = 18 ; Checkpoint Interval = 10**

**Solver processes**

**Spare processes**
Demo XXXII

Time Step = 18 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XXXIII

Time Step = 18 ; Checkpoint Interval = 10

(1) Explicit Backward Integration

Solver processes

Spare processes
Demo XXXIV

Time Step = 18 ; Checkpoint Interval = 10 (1) Explicit Backward Integration

Solver processes

Spare processes
Demo XXXV

Time Step = 18; Checkpoint Interval = 10

(1) Explicit Backward Integration

Solver processes

Spare processes
## Demo XXXVI

<table>
<thead>
<tr>
<th>Time Step $= 18$ ; Checkpoint Interval $= 10$</th>
<th>(2) Space Marching Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spare processes</td>
<td></td>
</tr>
<tr>
<td>Solver processes</td>
<td></td>
</tr>
</tbody>
</table>

### Solver processes

- Red squares indicate failed processes.
- Blue squares indicate spare processes available.

### Spare processes

- Blue squares indicate spare processes available.
Demo XXXVII

Time Step = 18 ; Checkpoint Interval = 10

(2) Space Marching Scheme

Solver processes

Spare processes
Demo XXXVIII

Time Step = 18; Checkpoint Interval = 10

(2) Space Marching Scheme

Solver processes

Spare processes
### Demo XXXIX

- **Time Step** = 18
- **Checkpoint Interval** = 10

#### (2) Space Marching Scheme

<table>
<thead>
<tr>
<th>Solver processes</th>
<th>Spare processes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Demo XL

Time Step = 18 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XLI

Time Step = 18 ; Checkpoint Interval = 10

(3) Forward Implicit Integration

Solver processes

Spare processes
Demo XLII

Time Step = 18 ; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 18; Checkpoint Interval = 10
Demo XLIV

Time Step = 19 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XLV

Time Step = 20 ; Checkpoint Interval = 10

Solver processes

Spare processes
Demo XLVI

Time Step = 21 ; Checkpoint Interval = 10

Solver processes

Spare processes
Time Step = 21 ; Checkpoint Interval = 10
Fault Tolerant Algorithms for Heat Transfer Problems

Hatem Ltaief, Edgar Gabriel and Marc Garbey
Results I

- Recovery Performance (A)

![Failure Overheads in Percentage: Small data set](image)

**Figure:** Failure Overheads in Percentage: **Small** data set.
Figure: Failure Overheads in Percentage: Medium data set.
Figure: Failure Overheads in Percentage: Large data set.
Results IV

- Recovery Performance (B)

Figure: Recovery Time in Seconds: Small data set.
Results V

Figure: Recovery Time in Seconds: Medium data set.
Results VI

Figure: Recovery Time in Seconds: Large data set.
Motivations
Model Problem
FT Approach
Numerical Reconstructions
Performance Comparisons
Demo
Results
Conclusion
Conclusion

- Our Fault Tolerant approach is based on efficient numerical algorithms

- Solution: combine explicit reconstruction techniques that are an explicit backward integration with some stabilization terms and space marching

- Challenging problem because it is very ill posed

- One more job to do for spare processors could be solution verification on the fly (the least square extrapolation method)
Thank You!