Assessing the impact of ABFT & Checkpoint composite strategies

George Bosilca\textsuperscript{1}, Aurelien Bouteiller\textsuperscript{1}, Thomas Herault\textsuperscript{1}, Yves Robert\textsuperscript{1,2} and Jack Dongarra\textsuperscript{1}
1. University of Tennessee Knoxville, USA
2. École Normale Supérieure de Lyon, France
\{bosilca,bouteill,herault,dongarra\}@icl.utk.edu;
yves.robert@ens-lyon.fr
Outline

1 Motivation

2 ABFT & Periodic Ckpt

3 Performance Modeling

4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   • As function of $\alpha$ and $\mu$
   • Weak Scaling

6 Conclusion
Faults

- Assume independent failures
- Let $N$ be the number of components ("System Size")
- Let $r$ be the probability of a component to operate for 1h
- Let $R$ be the probability of the system to operate for 1h

\[
R = r^N \\
R \approx \frac{1}{e^{\lambda N}}, \quad \frac{1}{\lambda} = 1 - r
\]
Assume independent failures

Let $N$ be the number of components ("System Size")

Let $r$ be the probability of a component to operate for 1h

Let $R$ be the probability of the system to operate for 1h

$$R = r^N$$

$$R \approx \frac{1}{e^{\lambda N}}, \quad \frac{1}{\lambda} = 1 - r$$
Assume independent failures
Let $N$ be the number of components ("System Size")
Let $r$ be the probability of a component to operate for 1h
Let $R$ be the probability of the system to operate for 1h

\[ R = r^N \]

\[ R \approx \frac{1}{e^{\lambda N}}, \frac{1}{\lambda} = 1 - r \]
Fault Tolerance Techniques

General Techniques

- Replication
- Rollback Recovery
  - Coordinated Checkpointing
  - Uncoordinated Checkpointing & Message Logging
  - Hierarchical Checkpointing

Application-Specific Techniques

- Algorithm Based Fault Tolerance (ABFT)
- Iterative Convergence
- Approximated Computation
Reminder: Coordinated Checkpointing and Rollback Recovery

- Coordinated checkpoints over all processes
- Global restart after a failure

- General technique (we assume preemptive checkpointing capability)
- All processors need to roll back
- All memory needs to be saved
(Reminder?) Algorithm-Based Fault Tolerance

\[
\begin{pmatrix}
A \\
A & C
\end{pmatrix}
\xrightarrow{\text{Operation}}
\begin{pmatrix}
B \\
B & C'
\end{pmatrix}
\]

\[
C = \text{Cksum}(A)
\]
\[
C' = \text{Cksum}(B)
\]

**Principle of ABFT**

- Input Data \((A)\) and Result \((B)\) are distributed
- *Operation* preserves *Checksum* properties
- Apply the operation on Data + Checksum \((AC)\)
- In case of failure, recover the missing data by inversion of the checksum
for( aninsanenumber ) {
  /* Extract data from simulation, fill up matrix */
  sim2mat();

  /* Factorize matrix, Solve */
  dgeqrf();
  dsolve();

  /* Update simulation with result vector */
  vec2sim();
}
**Application**

**Typical Application**

```
for ( aninsanenumner ) {
  /* Extract data from simulation, */
  /* matrix */
  sim2mat();

  /* Factorize matrix, */
  /* Solve */
  dgeqrf();
  dsolve();

  /* Update simulation */
  /* with result vector */
  vec2sim();
}
```

Goodbye ABFT?!

(c) Yves, 2010 – 2013

Characteristics, Large part of (total) computation spent in factorization/solve

- Between LA operations:
  - use resulting vector / matrix with operations that do not preserve the checksums on the data
  - modify data not covered by ABFT algorithms
Problem Statement

Typical Application

```c
for (a number)
  /* 
   * Extract data from simulation, fill up matrix sim2mat ();
   */
  /* 
   * Factorize matrix, Solve dgeqrf (); dsolve ();
   */
  /* 
   * Update simulation with result vector vec2sim ();
   */

/* Update simulation with result vector */ vec2sim ();
```

How to use fault tolerant operations(*) within a non-fault tolerant(**) application?(***)

(*) ABFT, or other application-specific FT

(**) Or within an application that does not have the same kind of FT

(***) And keep the application globally fault tolerant...

- use resulting vector / matrix with operations that do not preserve the checksums on the data
- modify data not covered by ABFT algorithms
Outline

1 Motivation

2 ABFT & Periodic Ckpt

3 Performance Modeling

4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   - As function of $\alpha$ and $\mu$
   - Weak Scaling

6 Conclusion
ABFT & PeriodicCheckpoint

ABFT & PeriodicCheckpoint: no failure
ABFT & Periodic Ckpt: failure during Library phase

- Process 0
- Process 1
- Process 2

Failure (during Library)

Rollback (partial) Recovery

ABFT Recovery
ABFT&PeriodicCkpt: failure during General phase

Process 0

Process 1

Process 2

Failure (during GENERAL)

Rollback (full)

Recovery

Application

Library
ABFT&PeriodicCkpt: Optimizations

- If the duration of the **GENERAL** phase is too small: don’t add checkpoints.
- If the duration of the **LIBRARY** phase is too small: don’t do ABFT recovery, remain in **GENERAL** mode.
  - This assumes a performance model for the library call.
**ABFT&PeriodicCkpt: Optimizations**

- If the duration of the *General* phase is too small: don’t add checkpoints.
- If the duration of the *Library* phase is too small: don’t do ABFT recovery, remain in *General* mode.
  - This assumes a performance model for the library call.
Outline

1 Motivation
2 ABFT & Periodic Ckpt
3 Performance Modeling
4 Periodic Checkpointing Protocols (for comparison)
5 Evaluation
   • As function of $\alpha$ and $\mu$
   • Weak Scaling
6 Conclusion
A few notations

Times, Periods

\( T_0 \): Duration of an Epoch (without FT)
\( T_L = \alpha T_0 \): Time spent in the Library phase
\( T_G = (1 - \alpha) T_0 \): Time spent in the General phase
\( P_G \): Periodic Checkpointing Period
\( T_{ff}, T_{ff}^G, T_{ff}^L \): “Fault Free” times
\( t_{lost}^G, t_{lost}^L \): Lost time (recovery overhreads)
\( T_{final}^G, T_{final}^L \): Total times (with faults)
A few notations

Costs

\[ C_L = \rho C: \text{ time to take a checkpoint of the Library data set} \]
\[ C_{\bar{L}} = (1 - \rho)C: \text{ time to take a checkpoint of the General data set} \]

\[ R, R_{\bar{L}}: \text{ time to load a full / General data set checkpoint} \]
\[ D: \text{ down time (time to allocate a new machine / reboot)} \]
\[ \text{Recons}_{\text{ABFT}}: \text{ time to apply the ABFT recovery} \]
\[ \phi: \text{ Slowdown factor on the Library phase, when applying ABFT} \]
**GENERAL phase, fault free waste**

**GENERAL phase**

![Diagram](image)

**Without Failures**

\[
T^\text{ff}_G = \begin{cases} 
T_G + \frac{C_L}{T_G} & \text{if } T_G < P_G \\
\frac{T_G}{P_G - C} \times P_G & \text{if } T_G \geq P_G 
\end{cases}
\]
Without Failures

\[ T_{L}^{ff} = \phi \times T_{L} + C_{L} \]
**GENERAL phase, failure overhead**

**GENERAL phase**

- Process 0
- Process 1
- Process 2

Failure (during GENERAL)

Rollback (full) Recovery

**Failure Overhead**

$$t_G^{\text{lost}} = \begin{cases} 
D + R + \frac{T_{ff}}{2} & \text{if } T_G < P_G \\
D + R + \frac{P_G}{2} & \text{if } T_G \geq P_G 
\end{cases}$$
**Library phase, failure overhead**

**Library phase**

Failure Overhead:

\[ t^{\text{lost}}_L = D + R_L + \text{Recons}_{\text{ABFT}} \]
Overall

Time (with overheads) of **LIBRARY** phase is constant (in $P_G$):

$$T_L^{\text{final}} = \frac{1}{1 - \frac{D+R_L+\text{ReconsABFT}}{\mu}} \times (\alpha \times T_L + C_L)$$

Time (with overheads) of **GENERAL** phase accepts two cases:

$$T_G^{\text{final}} = \begin{cases} 
\frac{1}{1 - \frac{D+R+\mu T_G}{2}} \times (T_G + C_L) & \text{if } T_G < P_G \\
\frac{\mu T_G}{(1-C/P_G)(1-\frac{D+R+\frac{P_G}{\mu}}{2})} & \text{if } T_G \geq P_G 
\end{cases}$$

Which is minimal in the second case, if

$$P_G = \sqrt{2C(\mu - D - R)}$$
From the previous, we derive the waste, which is obtained by

\[ \text{WASTE} = 1 - \frac{T_0}{T_G^{\text{final}}} + \frac{T_f^{\text{final}}}{T_L^{\text{final}}} \]
Motivation

ABFT & Periodic Ckpt

Performance Modeling

Periodic Checkpointing Protocols (for comparison)

Evaluation
  - As function of $\alpha$ and $\mu$
  - Weak Scaling

Conclusion
Optimal Checkpoint Interval

\[ P_{PC}^{opt} = \sqrt{2C(\mu - D - R)} \]
**BiPeriodicCkpt**

**Optimization**

\[ P_{\text{BPC},G}^{\text{opt}} = \sqrt{2C(\mu - D - R)} \]

\[ P_{\text{BPC},L}^{\text{opt}} = \sqrt{2CL(\mu - D - R)} \]
Outline

1 Motivation
2 ABFT & Periodic Checkpointing (ABFT & Periodic Ckpt)
3 Performance Modeling
4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   - As function of $\alpha$ and $\mu$
   - Weak Scaling

6 Conclusion
Outline

1. Motivation
2. ABFT & Periodic Ckpt
3. Performance Modeling
4. Periodic Checkpointing Protocols (for comparison)
5. Evaluation
   - As function of $\alpha$ and $\mu$
   - Weak Scaling
6. Conclusion
Model & Simulations: **PurePeriodicCkpt**

- $T_0=1w$, $C=R=10\text{min}$, $D=1\text{min}$, $\rho=0.8$, $\phi=1.03$, $\text{RECONS}_{\text{ABFT}}=2$
Model & Simulations: **BiPeriodicCkpt**

\[
T_0 = 1 \text{w}, \ C = R = 10 \text{min}, \ D = 1 \text{min}, \ \rho = 0.8, \ \phi = 1.03, \ \text{Recons}_{ABFT} = 2
\]

- **MTBF system (minutes)**
- **Ratio of time spent in Library Phase (\( \alpha \))**

![Graph showing simulation model comparison for BiPeriodicCkpt](image-url)
Model & Simulations: ABFT&PeriodicCkpt

\[ T_0 = 1w, C = R = 10 \text{min}, D = 1 \text{min}, \rho = 0.8, \phi = 1.03, \text{Recons}_{\text{ABFT}} = 2 \]
Model: PurePeriodicCkpt vs. BiPeriodicCkpt

T₀=1w, C=R=10min, D=1min, ρ=0.8, φ=1.03, ReconsABFT=2

<table>
<thead>
<tr>
<th>Ratio of time spent in Library Phase (α)</th>
<th>MTBF system (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>10</td>
</tr>
<tr>
<td>0.4</td>
<td>20</td>
</tr>
<tr>
<td>0.6</td>
<td>30</td>
</tr>
<tr>
<td>0.8</td>
<td>40</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
</tr>
</tbody>
</table>

PurePeriodicCkpt

BiPeriodicCkpt
Model & Simulations: PurePeriodicCkpt vs. ABFT&PeriodicCkpt

**PurePeriodicCkpt**

**ABFT&PeriodicCkpt**
Model & Simulations: BiPeriodicCkpt vs. ABFT&PeriodicCkpt

BiPeriodicCkpt

ABFT&PeriodicCkpt
Outline

1 Motivation
2 ABFT & PERIODIC CKPT
3 Performance Modeling
4 Periodic Checkpointing Protocols (for comparison)
5 Evaluation
   • As function of $\alpha$ and $\mu$
   • Weak Scaling
6 Conclusion
Toward Exascale, and Beyond!

Let’s think at scale

- Number of components $\uparrow \Rightarrow$ MTBF $\downarrow$
- Number of components $\uparrow \Rightarrow$ Problem Size $\uparrow$
- Problem Size $\uparrow \Rightarrow$
  Computation Time spent in Library phase $\uparrow$

😊 ABFT & PeriodicCkpt should perform better with scale
🤔 By how much?
Weak Scale #1

Weak Scale Scenario #1

- Number of components, $n$, increase
- Memory per component remains constant
- Problem Size increases in $O(\sqrt{n})$ (e.g. matrix operation)

- $\mu$ at $n = 10^5$: 1 day, is in $O\left(\frac{1}{n}\right)$
- $C (=R)$ at $n = 10^5$, is 1 minute, is in $O(n)$
- $\alpha$ is constant at 0.8, as is $\rho$.
  (both Library and General phase increase in time at the same speed)
Weak Scale #1
Weak Scale Scenario #2

- Number of components, \( n \), increase
- Memory per component remains constant
- Problem Size increases in \( O(\sqrt{n}) \) (e.g. matrix operation)

\( \mu \) at \( n = 10^5 \): 1 day, is \( O\left(\frac{1}{n}\right) \)
\( C (=R) \) at \( n = 10^5 \), is 1 minute, is in \( O(n) \)
\( \rho \) remains constant at 0.8, but Library phase is \( O(n^3) \) when General phases progresses in \( O(n^2) \) (\( \alpha \) is 0.8 at \( n = 10^5 \) nodes).
Weak Scale #2
Weak Scale #3

Weak Scale Scenario #3

- Number of components, \( n \), increase
- Memory per component remains constant
- Problem Size increases in \( O(\sqrt{n}) \) (e.g. matrix operation)

- \( \mu \) at \( n = 10^5 \): 1 day, is \( O\left(\frac{1}{n}\right) \)
- \( C \) (= \( R \)) at \( n = 10^5 \), is 1 minute, stays independent of \( n \) \((O(1))\)
- \( \rho \) remains constant at 0.8, but Library phase is \( O(n^3) \) when General phases progresses in \( O(n^2) \) (\( \alpha \) is 0.8 at \( n = 10^5 \) nodes).
Weak Scale #3

<table>
<thead>
<tr>
<th>Nb Faults PeriodicCkpt</th>
<th>Nb Faults Bi-PeriodicCkpt</th>
<th>Nb Faults ABFT PeriodicCkpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k</td>
<td>α = 0.55</td>
<td></td>
</tr>
<tr>
<td>10k</td>
<td>α = 0.8</td>
<td></td>
</tr>
<tr>
<td>100k</td>
<td>α = 0.92</td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td>α = 0.975</td>
<td></td>
</tr>
</tbody>
</table>

Waste

<table>
<thead>
<tr>
<th>PeriodicCkpt</th>
<th>Bi-PeriodicCkpt</th>
<th>ABFT PeriodicCkpt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>α = 0.55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>α = 0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>α = 0.92</td>
<td></td>
</tr>
<tr>
<td></td>
<td>α = 0.975</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1 Motivation

2 ABFT & Periodic Ckpt

3 Performance Modeling

4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   - As function of $\alpha$ and $\mu$
   - Weak Scaling

6 Conclusion
Conclusion

- method of composing fault tolerance approaches
  - applications that alternate between ABFT-aware and ABFT-unaware sections
  - each section is protected by its own mechanism
- performance model shows good opportunity for scaling
  - even when checkpointing hypothesis is optimistic
  - composite approach benefits from checkpointing improvements too
- Energy Efficiency? Checkpointing on Buddies?
  Checksumming? Better techniques to recover the ABFT-protected data in some cases.