Assessing the impact of ABFT & Checkpoint composite strategies

George Bosilca\textsuperscript{1}, Aurélien Bouteiller\textsuperscript{1}, Thomas Héralult\textsuperscript{1}, Yves Robert\textsuperscript{1,2} and Jack Dongarra\textsuperscript{1}

1. University of Tennessee Knoxville, USA
2. École Normale Supérieure de Lyon & INRIA, France

April 25, 2014 - ICL Lunch talk
1. Motivation

2. ABFT & Periodic Ckpt

3. Performance Modeling

4. Periodic Checkpointing Protocols (for comparison)

5. Evaluation
   - As function of $\alpha$ (% in library) and $\mu$ (MTBF)
   - Weak Scaling

6. Conclusion
Faults: a reality?

Optimists

- New technologies (graphene) solves the heat and voltage issues, making faults extremely unlikely
- Progress in the fabrication process will make the hardware much more reliable
- Simple methods to fix the problem directly at the hardware level: units duplication, checksum (ECC)
- How many “faults” you see on your daily working tool (laptop)? What would it be different with a Exascale tool?

Skeptics

- It is not because they could do it before but nobody did it, that they will not do it in the future (!)
- Do you have life insurance?
Faults: with big numbers come big responsibility

- 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

\[
R = r^N
\]

\[
R \approx \frac{1}{e^{\lambda N}}\cdot\frac{1}{\lambda} = 1 - r
\]
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
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
Algorithm-Based Fault Tolerance

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

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

**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
Typical Application

```c
for( aninsanenumber ) {
    /* Extract data from simulation, fill up matrix */
    sim2mat();

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

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

Characteristics

- Large part of (total) computation spent in factorization/solve
- Between LA operations:
  - use resulting vector / matrix with operations that do not preserve data checksums
  - modify data not covered by ABFT algorithms
Typical Application

```c
for ( aninsanenum )
    /* Extract data, simulation, matrix */
    sim2mat();

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

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

Characteristics

- Large part of (total) computation spent in factorization, solve
- Between LA operations:
  - use resulting vector / matrix with operations that do not preserve data checksums
  - modify data not covered by ABFT algorithms

Goodbye ABFT?!
Problem Statement

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 data checksums
😄 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 $\alpha$ (% in library) and $\mu$ (MTBF)
   - Weak Scaling

6. Conclusion
ABFT & Periodic Checkpoint: no failure
ABFT\&PeriodicCkpt: failure during Library phase

Process 0

Process 1

Process 2

Application

Library

Failure
(during LIBRARY)

Rollback
(partial)

Recovery
ABFT

Recovery
ABFT & Periodic Checkpoint: failure during General phase

Process 0

Process 1

Process 2

Application
Library

Failure (during General)

Rollback (full)
Recovery
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 $\alpha$ (% in library) and $\mu$ (MTBF)
   - 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

\( T_{\text{ff}}, T_{\text{ff}}, T_{\text{ff}} \): “Fault Free” times

\( t_{\text{lost}}^G, t_{\text{lost}}^L \): Lost time (recovery overheads)

\( T_{\text{final}}^G, T_{\text{final}}^L \): 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**

Without Failures

\[
T_{G}^{ff} = \begin{cases} 
T_{G} + C_{L} & \text{if } T_{G} < P_{G} \\
\frac{T_{G}}{P_{G} - C} \times P_{G} & \text{if } T_{G} \geq P_{G}
\end{cases}
\]
**Library phase, fault free waste**

**Library phase**

Process 0

Process 1

Process 2

Application

Library

Periodic Checkpoint

Split

Forced Checkpoints

Without Failures

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

General phase

Process 0
Process 1
Process 2
Application
Library
Application
Library
Application
Library
Failure (during GENERAL)
Rollback (full)
Recovery

Failure Overhead

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

Library phase

Process 0

Process 1

Process 2

Application

Library

Application

Library

Application

Library

Failure (during LIBRARY)

Rollback (partial) Recovery

Failure Overhead

$$t_L^{\text{lost}} = D + R_L + \text{Recons}_{\text{ABFT}}$$
Overall

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

\[
T_{L_{\text{final}}} = \frac{1}{1 - \frac{D + R_L + \text{Recons}_{\text{ABFT}}}{\mu}} \times (\alpha \times T_L + C_L)
\]

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

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

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

\[
WASTE = 1 - \frac{T_0}{T_{G}^{\text{final}} + T_{L}^{\text{final}}}
\]
Outline

1. Motivation
2. ABFT & Periodic Ckpt
3. Performance Modeling
4. Periodic Checkpointing Protocols (for comparison)
5. Evaluation
   - As function of $\alpha$ (% in library) and $\mu$ (MTBF)
   - Weak Scaling
6. Conclusion
### PurePeriodicCkpt

**Optimization**

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

Optimization

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

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

1 Motivation

2 ABFT & Periodic Checkpointing

3 Performance Modeling

4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   - As function of $\alpha$ (% in library) and $\mu$ (MTBF)
   - 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 $\alpha$ (% in library) and $\mu$ (MTBF)
   - Weak Scaling

6 Conclusion
Model & Simulations: **PurePeriodicCkpt**

Parameter Values:
- \( T_0 = 1 \) week, \( C = R = 10 \) min, \( D = 1 \) min, \( \rho = 0.8 \), \( \phi = 1.03 \), \( \text{RECONS}_{\text{ABFT}} = 2 \)
Model & Simulations: BiPERIODICCKPT

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

$T_0=1w, C=R=10\text{min}, D=1\text{min}, \rho=0.8, \phi=1.03, R_{\text{CONS}}_{\text{ABFT}}=2$

Ratio of time spent in Library Phase ($\alpha$)

MTBF system (minutes)

SIMULATION | MODEL

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

**PurePeriodicCkpt**

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

**PurePeriodicCkpt**

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

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

### BiPeriodicCkpt

![BiPeriodicCkpt](image)

### ABFT&PeriodicCkpt

![ABFT&PeriodicCkpt](image)
Outline

1 Motivation
2 ABFT\&PeriodicCkpt
3 Performance Modeling
4 Periodic Checkpointing Protocols (for comparison)
5 Evaluation
   • As function of $\alpha$ (% in library) and $\mu$ (MTBF)
   • Weak Scaling
6 Conclusion
Let’s think at scale

- Number of components ↗⇒ MTBF ↘
- Number of components ↗⇒ Problem Size ↗
- Problem Size ↗⇒ Computation Time spent in Library phase ↗

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

Weak Scale Scenario #1

- Number of components, $x$, increases
- Memory per component $M_{ind}$ remains constant
- PbSize $n$ increases in $O(\sqrt{x})$ (e.g. matrix, $n^2 = xM_{ind}$)

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

- Number of components, \( x \), increases
- Memory per component \( M_{ind} \) remains constant
- PbSize \( n \) increases in \( O(\sqrt{x}) \) (e.g. matrix, \( n^2 = xM_{ind} \))

- \( \mu \) at \( x = 10^5 \): 1 day, is \( O(\frac{1}{x}) \)
- \( C \) (=\( R \)) at \( x = 10^5 \), is 1 minute, is in \( O(x) \)
- \( \rho \) 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 \( x = 10^5 \) nodes).
Weak Scale #2
Number of components, $x$, increases
Memory per component $M_{ind}$ remains constant
PbSize increases in $O(\sqrt{x})$ (e.g. matrix, $n^2 = xM_{ind}$)

$\mu$ at $x = 10^5$: 1 day, is $O(\frac{1}{x})$
$C$ (=$R$) at $x = 10^5$, is 1 minute, stays independent of $x$ ($O(1)$)
$\rho$ 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 $x = 10^5$ nodes).
Weak Scale #3

![Graph showing fault rates and waste nodes across different scales](image-url)
1 Motivation

2 ABFT & Periodic Ckpt

3 Performance Modeling

4 Periodic Checkpointing Protocols (for comparison)

5 Evaluation
   - As function of $\alpha$ (% in library) and $\mu$ (MTBF)
   - 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