# Assessing the impact of ABFT & Checkpoint composite strategies

George Bosilca[1], Aurélien Bouteiller[1], Thomas Hérault[1], Yves Robert[1,2] and Jack Dongarra[1]

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

{bosilca,bouteill,herault,dongarra}@icl.utk.edu
yves.robert@ens-lyon.fr

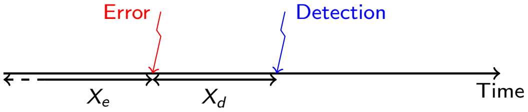
November 26, 2013 - JLPC Workshop

# Joint Lab

- Fault prediction: checkpointing vs. migration (PPL)
- Model to assess checkpoint protocols (CCPE, online)
- Checkpointing and prediction (JPDC, online)
- In-memory checkpointing (APDCM'13)
- Multi-criteria: time vs resource utilization (Europar'13)
- Multi-criteria: time vs energy (PMBS'13)
- Silent errors, checkpoints & verifications (PRDC'13)

# Detection latency

- Instantaneous error detection $\Rightarrow$ fail-stop failures
- Silent errors (data corruption) $\Rightarrow$ detection latency

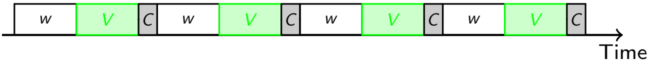



Error and detection latency

- Last checkpoint may have saved an already corrupted state
- Even when saving $k$ checkpoints: which one to roll back to?
- Critical failure: all checkpoints contain corrupted data

# Coupling checkpointing and verification

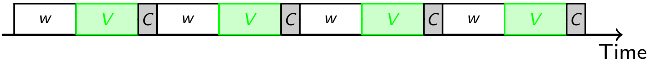
- Verification mechanism of cost $V$
- Simplest idea: verify work before each checkpoint




$V$ large compared to $w$ $\Rightarrow$ large $\text{WASTE}^{\text{ff}}$, can we improve that?

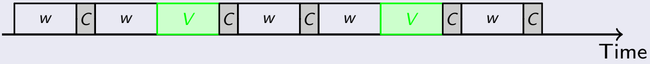
# Coupling checkpointing and verification

- Verification mechanism of cost $V$
- Simplest idea: verify work before each checkpoint



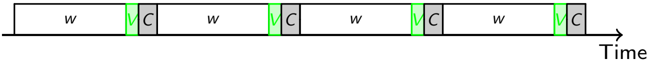

$V$ large compared to $w \Rightarrow$ large $\text{WASTE}^{\text{ff}}$, can we improve that?

**Is this better?**



# Coupling checkpointing and verification

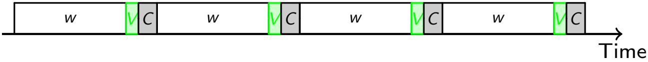
- Verification mechanism of cost $V$
- Simplest idea: verify work before each checkpoint




$V$ small in front of $w \Rightarrow$ large $\text{WASTE}^{\text{fail}}$, can we improve that?

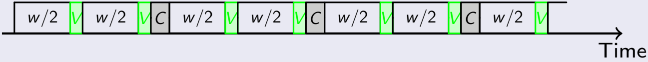
# Coupling checkpointing and verification

- Verification mechanism of cost $V$
- Simplest idea: verify work before each checkpoint

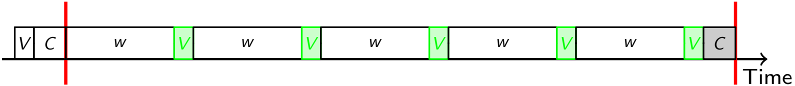



$V$ small in front of $w$ $\Rightarrow$ large $\text{Waste}^{\text{fail}}$, can we improve that?

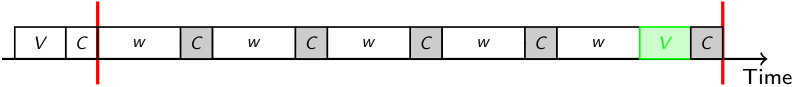
**Is this better?**



# Coupling checkpointing and verification




Small cost $V$: 5 verifications for 1 checkpoint

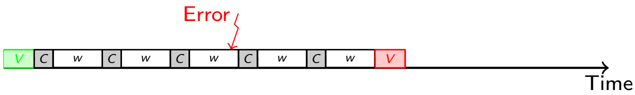



Large cost $V$: 5 checkpoints for 1 verification

More complicated periodic patterns? Different-size chunks?

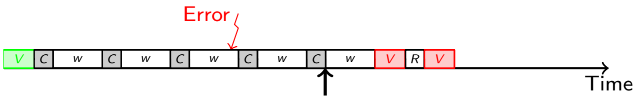
# $k$ checkpoints for 1 verification

Where did the error strike?



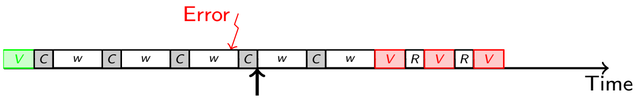
# $k$ checkpoints for 1 verification

Where did the error strike?



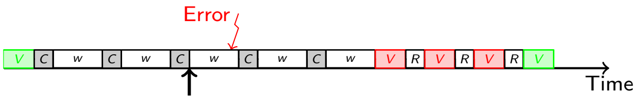
# $k$ checkpoints for 1 verification

Where did the error strike?



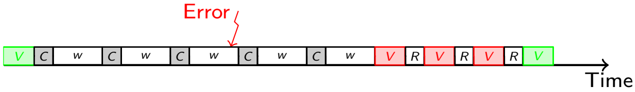
# $k$ checkpoints for 1 verification

Where did the error strike?



# $k$ checkpoints for 1 verification

Where did the error strike?




$$\text{Re-Exec} = 2(w + C) + (w + V)$$

# Outline

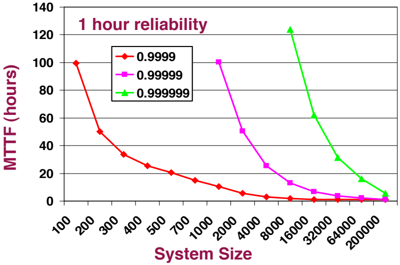
1. Motivation
2. ABFT&PeriodicCkpt
3. Performance Modeling
4. Periodic Checkpointing Protocols (for comparison)
5. Evaluation
   - As function of $\alpha$ and $\mu$
   - Weak Scaling
6. Conclusion

# 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

$$R = r^N$$

$$R \approx \frac{1}{e^{\lambda N}}, \frac{1}{\lambda} = 1 - r$$



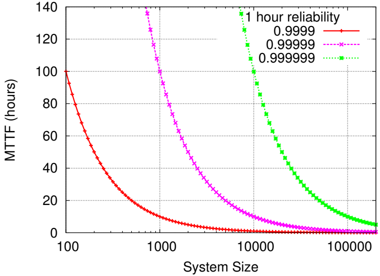

(Figure from Dan Reed "The Challenge of Complexity and Scale")

# 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

$$R = r^N$$

$$R \approx \frac{1}{e^{\lambda N}}, \frac{1}{\lambda} = 1 - r$$



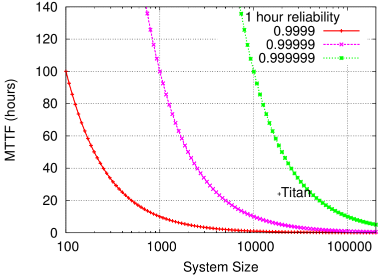

(Same figure with plotting software)

# 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

$$R = r^N$$

$$R \approx \frac{1}{e^{\lambda N}}, \frac{1}{\lambda} = 1 - r$$




(Same figure with plotting software)

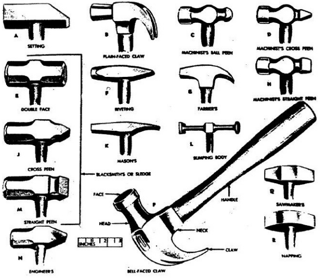
# Fault Tolerance Techniques

## General Techniques

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

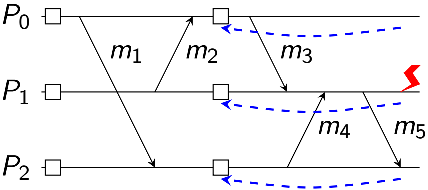
## Application-Specific Techniques

- Algorithm Based Fault Tolerance (ABFT)
- Iterative Convergence



# Coordinated Checkpointing and Rollback Recovery

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

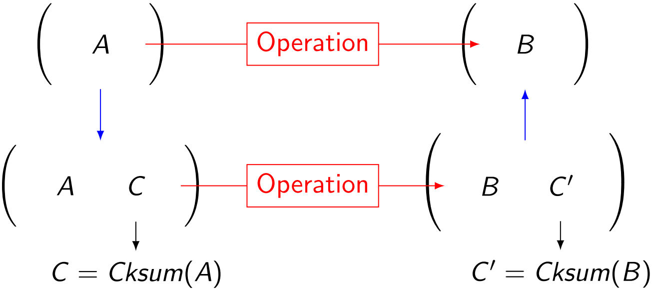



☺ General technique (we assume preemptive checkpointing capability)

☹ All processors need to roll back

☹ All memory needs to be saved

# Algorithm-Based Fault Tolerance




## 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

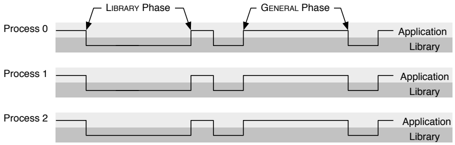
# Application

## Typical Application

```
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

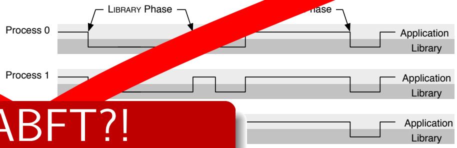
# Application

## Typical Application

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

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

  /* Upda   simulation
   * wi   result vector */
  vec2   n();
}
```




## Goodbye ABFT?!

## Characteristics

- ☺ Large t of (total) computat n spent in factorizatio olve
- Between LA op ations:
  - ☹ use resulting tor / matrix with operations at do not preserve data chec ums
  - ☹ modify data not cov d by ABFT algorithms

# Application

## 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...

Typica

```
for( a
  /* I
   * s
   * r
  sim2

  /* I
   * S
  dge
  dsolv ();

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

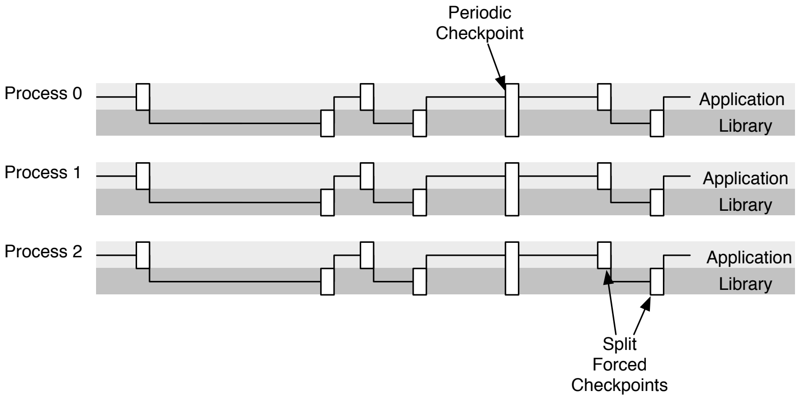
- Application
  Library
- Application
  Library
- Application
  Library

☹ use resulting vector / matrix with operations that do not preserve data checksums
☹ modify data not covered by ABFT algorithms

# Outline

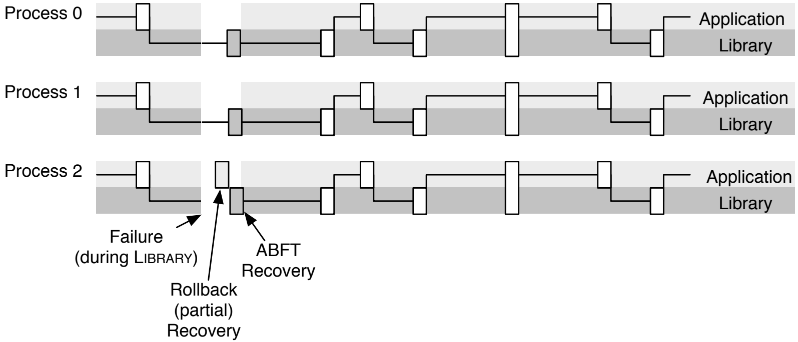
# ABFT&PeriodicCkpt

## ABFT&PeriodicCkpt: no failure



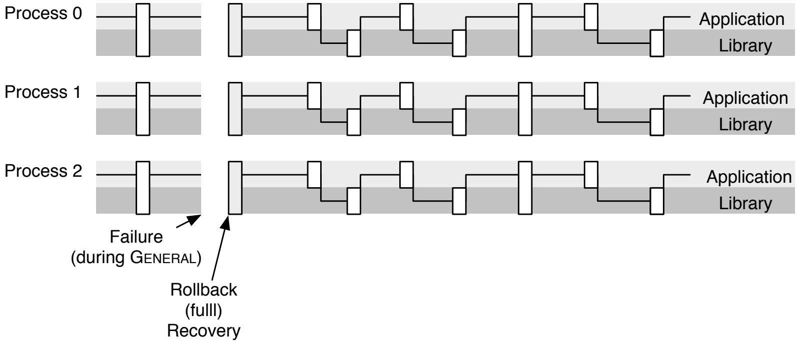
# ABFT&PeriodicCkpt

## ABFT&PeriodicCkpt: failure during Library phase

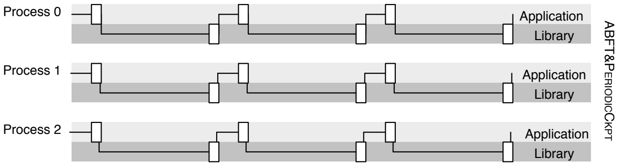


# ABFT&PeriodicCkpt

## ABFT&PeriodicCkpt: failure during General phase



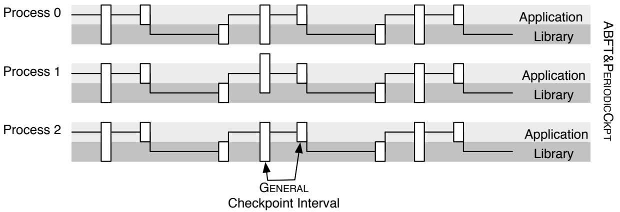
# ABFT&PeriodicCkpt: Optimizations




## 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

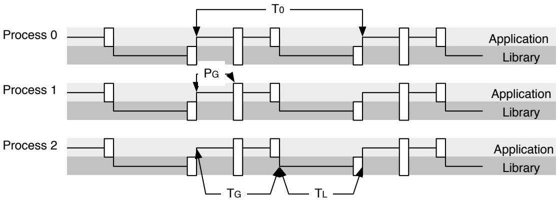



## ABFT&PeriodicCkpt: Optimizations

- 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

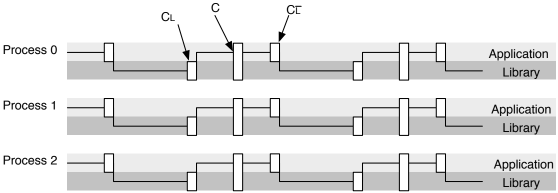
# A few notations




## 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
$T^{ff}$, $T_G^{ff}$, $T_L^{ff}$: "Fault Free" times
$t_G^{lost}$, $t_L^{lost}$: Lost time (recovery overheads)
$T_G^{final}$, $T_L^{final}$: Total times (with faults)

# A few notations




## Costs

$C_L = \rho C$: time to take a checkpoint of the LIBRARY data set

$C_{\bar{L}} = (1 - \rho)C$: time to take a checkpoint of the GENERAL data set

$R, R_{\bar{L}}$: time to load a full / GENERAL data set checkpoint

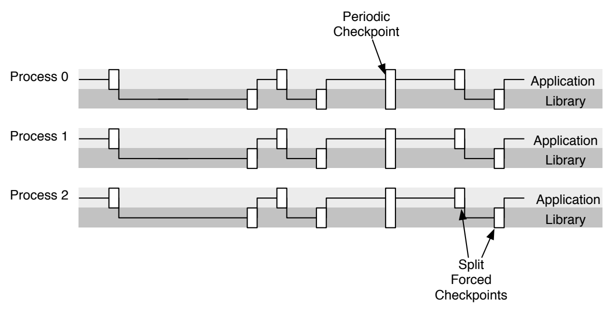
$D$: down time (time to allocate a new machine / reboot)

$\text{Recons}_{\text{ABFT}}$: time to apply the ABFT recovery

$\phi$: Slowdown factor on the LIBRARY phase, when applying ABFT

# General phase, fault free waste

## General phase

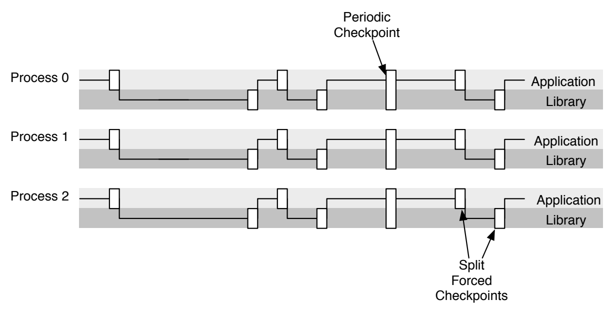



## Without Failures

$$T_G^{\text{ff}} = \begin{cases} T_G + C_{\bar{L}} & \text{if } T_G < P_G \\ \frac{T_G}{P_G - C} \times P_G & \text{if } T_G \geq P_G \end{cases}$$

# LIBRARY phase, fault free waste

## LIBRARY phase

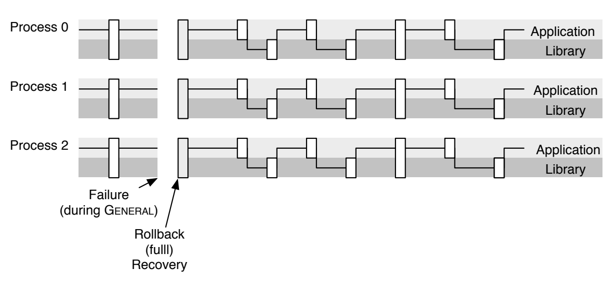



## Without Failures

$$T_L^{ff} = \phi \times T_L + C_L$$

# GENERAL phase, failure overhead

## GENERAL phase

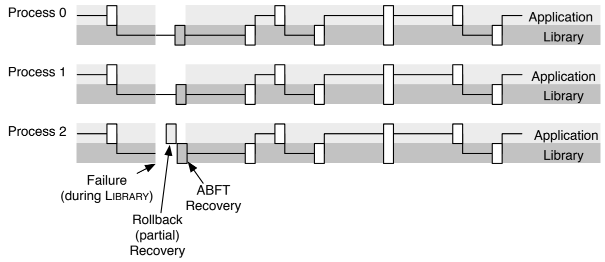



## Failure Overhead

$$t_G^{\text{lost}} = \begin{cases} D + R + \frac{T_G^{\text{ff}}}{2} & \text{if } T_G < P_G \\ D + R + \frac{P_G}{2} & \text{if } T_G \geq P_G \end{cases}$$

# LIBRARY phase, failure overhead

## LIBRARY phase




## Failure Overhead

$$t_L^{\text{lost}} = D + R_{\bar{L}} + \text{Recons}_{\text{ABFT}}$$

# Overall

## Overall

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

$$T_L^{\text{final}} = \frac{1}{1 - \frac{D+R_{\bar{L}}+\text{Recons}_{\text{ABFT}}}{\mu}} \times (\alpha \times T_L + C_L)$$

Time (with overehads) of GENERAL phase accepts two cases:

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

Which is minimal in the second case, if

$$P_G = \sqrt{2C(\mu - D - R)}$$

## Waste

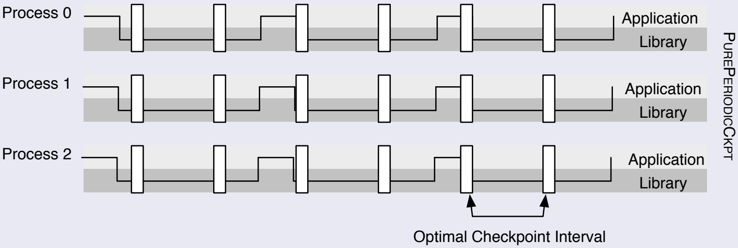
From the previous, we derive the waste, which is obtained by

$$\textsc{Waste} = 1 - \frac{T_0}{T_G^{\text{final}} + T_L^{\text{final}}}$$

# Outline

# PUREPERIODICCKPT

## PUREPERIODICCKPT

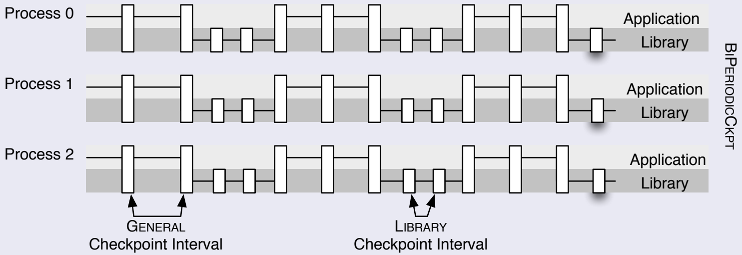



## Optimization

$$P_{PC}^{\text{opt}} = \sqrt{2C(\mu - D - R)}$$

# BiPeriodicCkpt

## BiPeriodicCkpt




## Optimization

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

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

# Outline

# Outline

# Model & Simulations: PurePeriodicCkpt

PurePeriodicCkpt

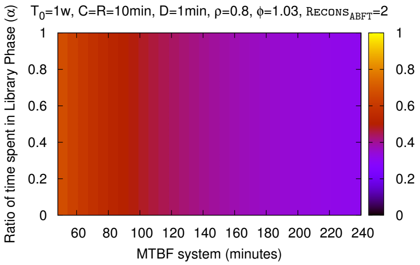
# Model & Simulations: BiPeriodicCkpt

BiPeriodicCkpt

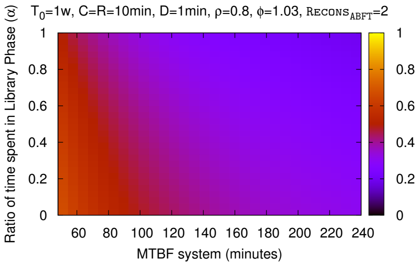
# Model & Simulations: ABFT&PeriodicCkpt

ABFT&PeriodicCkpt

# Model: PurePeriodicCkpt vs. BiPeriodicCkpt

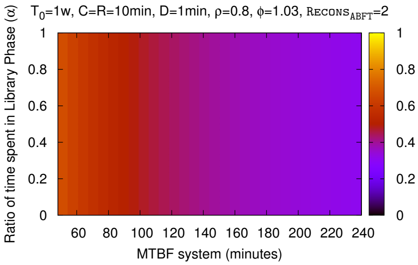



PurePeriodicCkpt

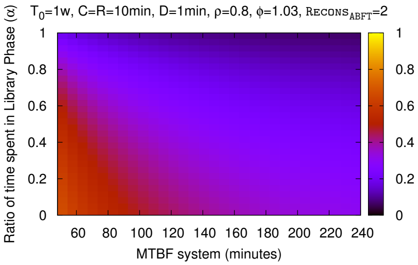



BiPeriodicCkpt

# Model & Simulations: PurePeriodicCkpt vs. ABFT&PeriodicCkpt

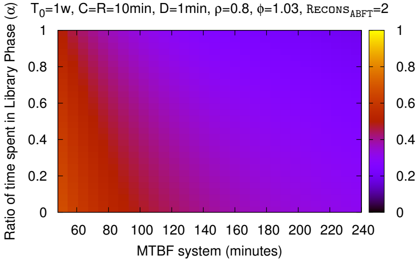



PurePeriodicCkpt

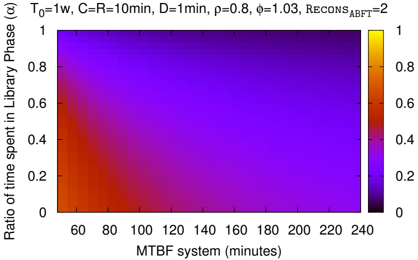



ABFT&PeriodicCkpt

# Model & Simulations: BiPeriodicCkpt vs. ABFT&PeriodicCkpt




BiPeriodicCkpt




ABFT&PeriodicCkpt

# Outline

# Toward Exascale, and Beyond!

## Let's think at scale

- Number of components $\nearrow \Rightarrow$ MTBF $\searrow$
- Number of components $\nearrow \Rightarrow$ Problem Size $\nearrow$
- Problem Size $\nearrow \Rightarrow$
  Computation Time spent in LIBRARY phase $\nearrow$

☺ ABFT&PERIODICCKPT should perform better with scale

☹ By how much?

# Weak Scale #1

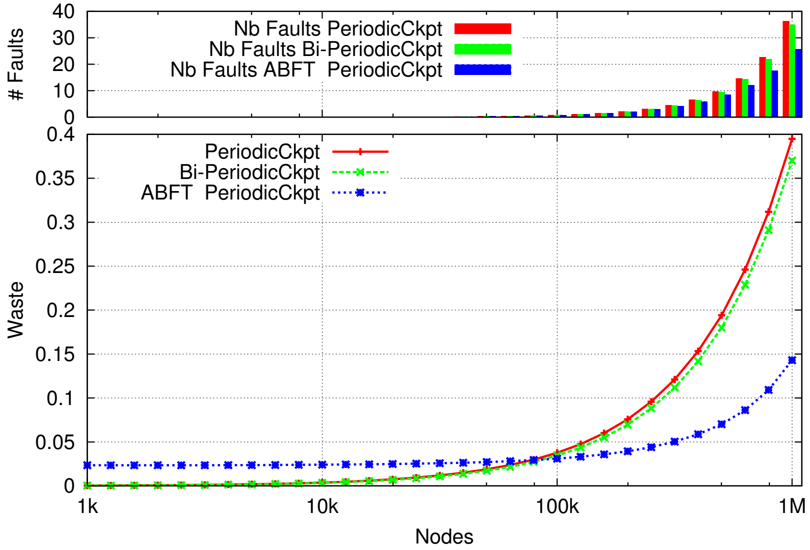
## Weak Scale Scenario #1

- 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 #1



40
30
20
10
0
# Faults
Nb Faults PeriodicCkpt
Nb Faults Bi-PeriodicCkpt
Nb Faults ABFT PeriodicCkpt
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Waste
PeriodicCkpt
Bi-PeriodicCkpt
ABFT PeriodicCkpt
1k
10k
100k
1M
Nodes

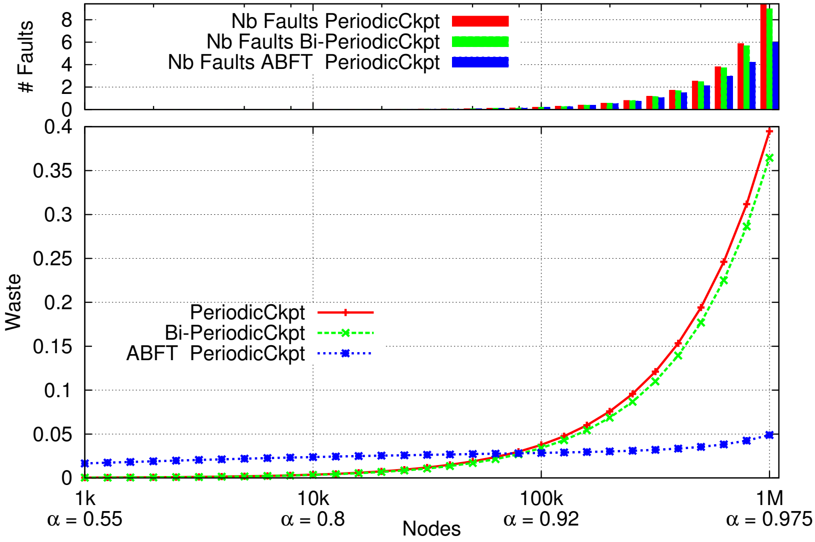
# Weak Scale #2

## 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



# Faults
8
6
4
2
0
Nb Faults PeriodicCkpt
Nb Faults Bi-PeriodicCkpt
Nb Faults ABFT PeriodicCkpt
Waste
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
PeriodicCkpt
Bi-PeriodicCkpt
ABFT PeriodicCkpt
1k
α = 0.55
10k
α = 0.8
100k
α = 0.92
1M
α = 0.975
Nodes

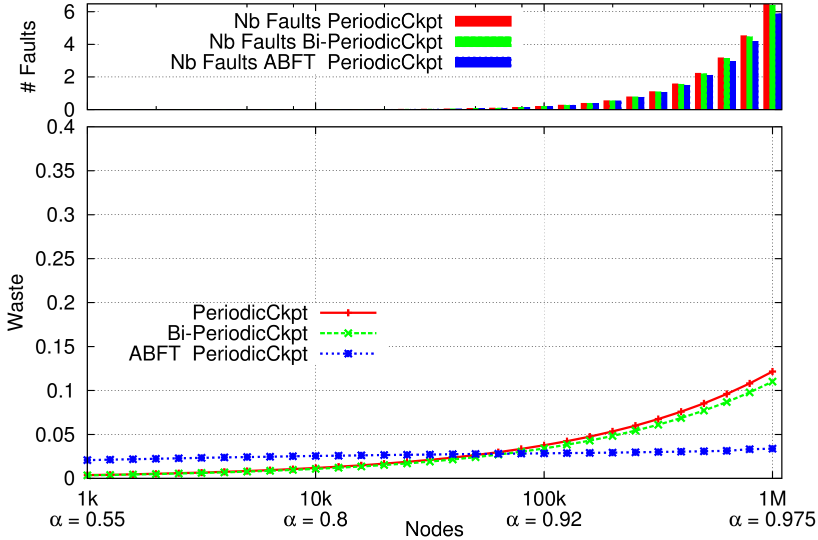
# Weak Scale #3

## Weak Scale Scenario #3

- 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



Nb Faults PeriodicCkpt
Nb Faults Bi-PeriodicCkpt
Nb Faults ABFT PeriodicCkpt
# Faults
0
2
4
6
Waste
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
PeriodicCkpt
Bi-PeriodicCkpt
ABFT PeriodicCkpt
1k
α = 0.55
10k
α = 0.8
100k
α = 0.92
1M
α = 0.975
Nodes

# Outline

# 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.