A nice little scheduling problem

Yves Robert  
Ecole Normale Supérieure de Lyon  
& Institut Universitaire de France  

CCGSC’2010 Asheville
A few nice little scheduling problems

- I made it to the 10 CCGSC workshops!
- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I talked about a nice little scheduling problem in 2006
- I talked about a nice little scheduling problem in 2008
A few nice little scheduling problems

- I made it to the 10 CCGSC workshops!
- I talked about a nice little scheduling problem in 1992
- I talked about a nice little scheduling problem in 1994
- I talked about a nice little scheduling problem in 1996
- I talked about a nice little scheduling problem in 1998
- I talked about a nice little scheduling problem in 2000
- I talked about a nice little scheduling problem in 2002
- I talked about a nice little scheduling problem in 2004
- I talked about a nice little scheduling problem in 2006
- I talked about a nice little scheduling problem in 2008

At last
a fundamental problem
in exascale computing!!
Checkpointing versus Migration for Post-Petascale Machines

Franck Cappello
INRIA-Illinois Joint Laboratory for Petascale Computing

Henri Casanova
University of Hawai‘i

Yves Robert
Ecole Normale Supérieure de Lyon
& Institut Universitaire de France

CCGSC’2010 Asheville
Dealing with failures

- Fault tolerant computing becomes **unavoidable**
  Caveat: same story told for a very long time! 😞

- Coming for real on future machines, e.g. Blue Waters
  INRIA-Illinois Joint Laboratory for Petascale Computing

- Techniques:
  - **failure avoidance** (as opposed to failure tolerance)
  - **checkpointing, migration**
Dealing with failures

- Fault tolerant computing becomes **unavoidable**
  Caveat: same story told for a very long time! 😞

- Coming for real on future machines, e.g. **Blue Waters**
  INRIA-Illinois Joint Laboratory for Petascale Computing

- Techniques:
  - **failure avoidance** (as opposed to failure tolerance)
  - **checkpointing, migration**
Dealing with failures

- Fault tolerant computing becomes **unavoidable**
  Caveat: same story told for a very long time! 😞

- Coming for real on future machines, e.g. **Blue Waters**
  INRIA-Illinois Joint Laboratory for Petascale Computing

- Techniques:
  - **failure avoidance** (as opposed to failure tolerance)
  - **checkpointing, migration**
Outline

1. Framework
2. Sequential jobs
3. Parallel jobs
4. Numerical results
5. To predict or not to predict
Outline

1. Framework
2. Sequential jobs
3. Parallel jobs
4. Numerical results
5. To predict or not to predict
Relying on failure prediction

- Applications **will** face resource faults during execution
- **Failure prediction** available
  (e.g. alarm when a disk or CPU becomes unusually hot)
- Application must dynamically prepare for, and recover from, expected failures

- Compare two well-known strategies:
  - **Checkpointing**: purely local, but can be very costly
  - **Migration**: requires availability of a spare resource

- **Remember, we assume accurate failure prediction**
Relying on failure prediction

- Applications **will** face resource faults during execution
- **Failure prediction** available
  (e.g. alarm when a disk or CPU becomes unusually hot)
- Application must dynamically prepare for, and recover from, expected failures

- Compare two well-known strategies:
  - **Preventive Checkpointing**: purely local, but can be very costly
  - **Preventive Migration**: requires availability of a spare resource

- **Remember**, we **assume accurate failure prediction**
Preventive checkpointing

- $D$: length of downtime intervals
- $\mu$: (average) length of execution intervals, a.k.a. MTTF
  - $R$: recovery time (beginning of interval)
  - $C$: checkpoint time (end of interval, just before failure)
Preventive migration

- $D$: length of downtime intervals
- $\mu$: (average) length of execution intervals
  - $M$: migration time (end of interval, just before failure)
  - Need spare node 😞
Notations

- $C$: checkpoint save time (in minutes)
- $R$: checkpoint recovery time (in minutes)
- $D$: down/reboot time (in minutes)
- $M$: migration time (in minutes)
- $\mu$: mean time to failure
  (e.g., $1/\lambda$ if failures are exponentially distributed)
- $N$: total number of cluster nodes
- $n$: number of spares (migration)
Checkpointing/migration comparison makes sense only if

\[ M < C + D + R \]

otherwise better use faulty machine as own spare

*Live migration* without any disk access,
thereby dramatically reducing migration time
Outline

1. Framework
2. Sequential jobs
3. Parallel jobs
4. Numerical results
5. To predict or not to predict
Checkpointing

Probability of node being active

\[ u_c = \max \left( 0, \frac{\mu - R - C}{\mu + D} \right) \]

Global throughput

\[ \rho_c = u_c \times N = \max \left( 0, \frac{\mu - R - C}{\mu + D} \right) \times N \]
Migration (1/2)

Probability of node being active

\[ u_m = \max \left( 0, \frac{\mu - M}{\mu + D} \right) \]

Global throughput

\[ \rho_m = u_m \times (N - n) = \max \left( 0, \frac{\mu - M}{\mu + D} \right) \times (N - n) \]
No shortage of spare nodes?

\[ \text{success}(n) = \sum_{k=0}^{n} \binom{N}{k} u_m^{N-k} (1 - u_m)^k \]

- Find \( n = \alpha(\varepsilon, N) \) that "guarantees" a successful execution with probability at least \( 1 - \varepsilon \)
- Solve numerically
Outline

1 Framework
2 Sequential jobs
3 Parallel jobs
4 Numerical results
5 To predict or not to predict
Number of processors required by typical jobs: two-stage log-uniform distribution biased to powers of two

- Let $N = 2^Z$ for simplicity
- Probability that a job is sequential: $\alpha_0 = p_1 \approx 0.25$
- Otherwise, the job is parallel, and uses $2^j$ processors with identical probability

$$\alpha_j = \alpha = (1 - p_1) \times \frac{1}{Z}$$

for $1 \leq j \leq Z = \log_2 N$
Number of processors required by typical jobs: \textit{two-stage log-uniform distribution biased to powers of two (says Dr. Feitelson)}

- Let $N = 2^Z$ for simplicity
- Probability that a job is sequential: $\alpha_0 = p_1 \approx 0.25$
- Otherwise, the job is parallel, and uses $2^j$ processors with identical probability

\[
\alpha_j = \alpha = (1 - p_1) \times \frac{1}{Z}
\]

for $1 \leq j \leq Z = \log_2 N$
<table>
<thead>
<tr>
<th>Framework</th>
<th>Sequential jobs</th>
<th>Parallel jobs</th>
<th>Results</th>
<th>No prediction</th>
</tr>
</thead>
</table>

**Distribution (2/3)**

- **Steady-state** utilization of whole platform:
  - all processors always active
  - constant proportion of jobs using any processor number

- Expectation of the number of jobs:
  - $K$ total number of jobs running
  - $\beta_j$ jobs that use $2^j$ processors exactly

Yves.Robert@ens-lyon.fr
Distribution (2/3)

- Steady-state utilization of whole platform:
  - all processors always active
  - constant proportion of jobs using any processor number

- Expectation of the number of jobs:
  - $K$ total number of jobs running
  - $\beta_j$ jobs that use $2^j$ processors exactly
Equations:

- \( K = \sum_{j=0}^{Z} \beta_j \)
- \( \beta_j = \alpha_j K \) for \( 0 \leq j \leq Z \)
- \( \sum_{j=0}^{Z} 2^j \beta_j = N \)

\[
\frac{N}{K} = \sum_{j=0}^{Z} 2^j \alpha_j = p_1 + \frac{1 - p_1}{Z} \sum_{j=1}^{Z} 2^j = p_1 + \frac{1 - p_1}{Z} (2N - 2)
\]

hence the value of \( K \) and the \( \beta_j \)
Distribution (3/3)

Equations:

- \( K = \sum_{j=0}^{Z} \beta_j \)
- \( \beta_j = \alpha_j K \) for \( 0 \leq j \leq Z \)
- \( \sum_{j=0}^{Z} 2^j \beta_j = N \)

\[
\frac{N}{K} = \sum_{j=0}^{Z} 2^j \alpha_j = p_1 + \frac{1 - p_1}{Z} \sum_{j=1}^{Z} 2^j = p_1 + \frac{1 - p_1}{Z} (2N - 2)
\]

hence the value of \( K \) and the \( \beta_j \)
If a job uses two processors, what is the expected interval time between failures?

- $\mu_j$ mean of the minimum of $2^j$ i.i.d. variables
- If the variables are exponentially distributed, with scale parameter $\lambda$, then
  \[
  \mu_j = 1/(\lambda 2^j) = \mu/2^j
  \]
- If the variables are Weibull, with scale parameter $\lambda$ and shape parameter $a$, then
  \[
  \mu_j = \lambda \Gamma(1 + 1/(a2^j))
  \]
Failures

- If a job uses two processors, what is the expected interval time between failures?

- $\mu_j$ mean of the minimum of $2^j$ i.i.d. variables

- If the variables are exponentially distributed, with scale parameter $\lambda$, then

$$\mu_j = 1/(\lambda 2^j) = \mu/2^j$$

- If the variables are Weibull, with scale parameter $\lambda$ and shape parameter $a$, then

$$\mu_j = \lambda \Gamma(1 + 1/(a 2^j))$$
Checkpointing

Platform throughput

\[ \rho_{cp} = \sum_{j=0}^{Z} \beta_j \times 2^j \times \max \left( 0, \frac{\mu_j - R - C}{\mu_j + D} \right) \]

For the exponential distribution: \( \mu_j = \mu / 2^j \)
Migration

Platform throughput

\[
\rho_{mp} = \left( \sum_{j=0}^{Z} \beta_j \times 2^j \times \max \left( 0, \frac{\mu_j - M}{\mu_j + D} \right) \right) \times \frac{N - n}{N}
\]

Probability of success: same as for independent jobs
Scenarios

- Understand the impact of checkpointing vs. migration
- All results are in percentage improvement of migration over checkpointing (negative or positive values)
- All results use the following values:
  - $\mu = 1$ day, 1 week, 1 month, 1 year
  - $N = 2^{14}, 2^{17}, 2^{20}$
  - $\varepsilon = 10^{-4}, 10^{-6}$
- Number of required spares in parentheses
Scenario "today" – $C = R = 10$, $D = 1$, $M = 0.33$

<table>
<thead>
<tr>
<th></th>
<th>Sequential Jobs</th>
<th></th>
<th>Parallel Jobs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu$</td>
<td>$N$</td>
<td>$\varepsilon = 10^4$</td>
<td>$\varepsilon = 10^6$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1 day</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>$2^{14}$</td>
<td>1.19 (32)</td>
<td>1.16 (37)</td>
<td>3141.07 (32)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>1.26 (164)</td>
<td>1.25 (177)</td>
<td>3086.92 (164)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>1.28 (1086)</td>
<td>1.28 (1119)</td>
<td>3033.16 (1086)</td>
</tr>
<tr>
<td><strong>1 week</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>$2^{14}$</td>
<td>0.14 (9)</td>
<td>0.12 (12)</td>
<td>3521.14 (9)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>0.17 (35)</td>
<td>0.16 (40)</td>
<td>3511.74 (35)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>0.18 (184)</td>
<td>0.18 (198)</td>
<td>3501.72 (184)</td>
</tr>
<tr>
<td><strong>1 month</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>$2^{14}$</td>
<td>0.02 (5)</td>
<td>0.00 (7)</td>
<td>1541.89 (5)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>0.04 (13)</td>
<td>0.03 (17)</td>
<td>3354.95 (13)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>0.04 (55)</td>
<td>0.04 (63)</td>
<td>3352.86 (55)</td>
</tr>
<tr>
<td><strong>1 year</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>$2^{14}$</td>
<td>-0.01 (2)</td>
<td>-0.01 (3)</td>
<td>69.22 (2)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>0.00 (4)</td>
<td>-0.00 (6)</td>
<td>1037.00 (4)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>0.00 (11)</td>
<td>0.00 (13)</td>
<td>3381.52 (11)</td>
</tr>
</tbody>
</table>
## Scenario "2011" – \( C = R = 5, \, D = 1, \, M = 0.33 \)

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>( N )</th>
<th>Sequential Jobs</th>
<th>Parallel Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \varepsilon = 10^4 )</td>
<td>( \varepsilon = 10^6 )</td>
</tr>
<tr>
<td>1 day</td>
<td>( 2^{14} )</td>
<td>0.48 (32)</td>
<td>0.45 (37)</td>
</tr>
<tr>
<td></td>
<td>( 2^{17} )</td>
<td>0.55 (164)</td>
<td>0.54 (177)</td>
</tr>
<tr>
<td></td>
<td>( 2^{20} )</td>
<td>0.57 (1086)</td>
<td>0.57 (1119)</td>
</tr>
<tr>
<td>1 week</td>
<td>( 2^{14} )</td>
<td>0.04 (9)</td>
<td>0.02 (12)</td>
</tr>
<tr>
<td></td>
<td>( 2^{17} )</td>
<td>0.07 (35)</td>
<td>0.07 (40)</td>
</tr>
<tr>
<td></td>
<td>( 2^{20} )</td>
<td>0.08 (184)</td>
<td>0.08 (198)</td>
</tr>
<tr>
<td>1 month</td>
<td>( 2^{14} )</td>
<td>-0.01 (5)</td>
<td>-0.02 (7)</td>
</tr>
<tr>
<td></td>
<td>( 2^{17} )</td>
<td>0.01 (13)</td>
<td>0.01 (17)</td>
</tr>
<tr>
<td></td>
<td>( 2^{20} )</td>
<td>0.02 (55)</td>
<td>0.02 (63)</td>
</tr>
<tr>
<td>1 year</td>
<td>( 2^{14} )</td>
<td>-0.01 (2)</td>
<td>-0.02 (3)</td>
</tr>
<tr>
<td></td>
<td>( 2^{17} )</td>
<td>-0.00 (4)</td>
<td>-0.00 (6)</td>
</tr>
<tr>
<td></td>
<td>( 2^{20} )</td>
<td>0.00 (11)</td>
<td>0.00 (13)</td>
</tr>
</tbody>
</table>
### Scenario "2015" – $C = 10$$R = 0.21$, $D = 0.25$, $M = 0.33$

<table>
<thead>
<tr>
<th>Framework</th>
<th>Sequential jobs</th>
<th>Parallel jobs</th>
<th>Results</th>
<th>No prediction</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>µ</th>
<th>N</th>
<th>$\varepsilon = 10^4$</th>
<th>$\varepsilon = 10^6$</th>
<th>$\varepsilon = 10^4$</th>
<th>$\varepsilon = 10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 day</td>
<td>$2^{14}$</td>
<td>-0.12 (18)</td>
<td>-0.14 (22)</td>
<td>-27.96 (18)</td>
<td>-27.98 (22)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>-0.07 (82)</td>
<td>-0.08 (91)</td>
<td>-27.92 (82)</td>
<td>-27.92 (91)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>-0.05 (501)</td>
<td>-0.06 (523)</td>
<td>-27.90 (501)</td>
<td>-27.90 (523)</td>
</tr>
<tr>
<td>1 week</td>
<td>$2^{14}$</td>
<td>-0.04 (6)</td>
<td>-0.05 (8)</td>
<td>-13.14 (6)</td>
<td>-13.15 (8)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>-0.02 (20)</td>
<td>-0.02 (24)</td>
<td>-29.07 (20)</td>
<td>-29.08 (24)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>-0.01 (91)</td>
<td>-0.01 (101)</td>
<td>-29.07 (91)</td>
<td>-29.07 (101)</td>
</tr>
<tr>
<td>1 month</td>
<td>$2^{14}$</td>
<td>-0.02 (3)</td>
<td>-0.03 (5)</td>
<td>-2.63 (3)</td>
<td>-2.64 (5)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>-0.01 (8)</td>
<td>-0.01 (11)</td>
<td>-30.74 (8)</td>
<td>-30.74 (11)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>-0.00 (30)</td>
<td>-0.00 (35)</td>
<td>-30.74 (30)</td>
<td>-30.74 (35)</td>
</tr>
<tr>
<td>1 year</td>
<td>$2^{14}$</td>
<td>-0.01 (2)</td>
<td>-0.01 (2)</td>
<td>-0.22 (2)</td>
<td>-0.22 (2)</td>
</tr>
<tr>
<td></td>
<td>$2^{17}$</td>
<td>-0.00 (3)</td>
<td>-0.00 (4)</td>
<td>-1.69 (3)</td>
<td>-1.69 (4)</td>
</tr>
<tr>
<td></td>
<td>$2^{20}$</td>
<td>-0.00 (7)</td>
<td>-0.00 (9)</td>
<td>-17.00 (7)</td>
<td>-17.00 (9)</td>
</tr>
</tbody>
</table>

Yves.Robert@ens-lyon.fr

Checkpointing. Or not. 27/39
Summary

- Sequential jobs: comparable performance (within 2%)
- Parallel jobs, short term: prefer migration
- Parallel jobs, 2015: picture not so clear

Good news for migration:
- small number of spares
- insensitive to target value of success probability
Summary

- Sequential jobs: comparable performance (within 2%)
- Parallel jobs, short term: prefer migration
- Parallel jobs, 2015: picture not so clear

Good news for migration:
- small number of spares
- insensitive to target value of success probability
Outline

1. Framework
2. Sequential jobs
3. Parallel jobs
4. Numerical results
5. To predict or not to predict

Yves.Robert@ens-lyon.fr
Checkpointing versus … checkpointing

- No failure prediction available
- No more migration 😞
- Checkpoint periodically
- How to determine optimal period $T$?
- Impact on platform throughput?
Optimal period $T$ (1/3)

$W = \text{expected percentage of time lost, or “wasted”}$:

$$W = \frac{C}{T} + \frac{T}{2\mu} \quad (1)$$

- First term in (1) by definition:
  - $C$ time-steps devoted to checkpointing every $T$ time-steps
- Every $\mu$ time-steps, a failure occurs
  - $\Rightarrow$ loss of $T/2$ time-steps in average

$W$ minimized for $T_{opt} = \sqrt{2C\mu}$ (Young’s approximation)

$$W_{min} = \sqrt{\frac{2C}{\mu}}$$

Yves.Robert@ens-lyon.fr
Optimal period $T$ (1/3)

$W =$ expected percentage of time lost, or “wasted”:

$$W = \frac{C}{T} + \frac{T}{2\mu} \quad (1)$$

- First term in (1) by definition:
  - $C$ time-steps devoted to checkpointing every $T$ time-steps
- Every $\mu$ time-steps, a failure occurs
  $\Rightarrow$ loss of $T/2$ time-steps in average

$W$ minimized for $T_{opt} = \sqrt{2C\mu}$ (Young’s approximation)

$$W_{min} = \sqrt{\frac{2C}{\mu}}$$
Optimal period $T$ (2/3)

\[ W = \frac{C}{T} + \frac{T}{2} + \frac{R + D}{\mu} \]

\[ W_{\text{min}} = \frac{R + D}{\mu} + \sqrt{\frac{2C}{\mu}} \]

Different from Daly:

target = steady-state operation of platform

target \neq \text{minimizing expected duration of a given job}
Optimal period $T$ (3/3)

\[ W_{\text{min}} = \frac{R + D}{\mu} + \sqrt{\frac{2C}{\mu}} \]  

(2)

$W_{\text{min}}$ larger than 1 for very small $\mu$  
(likely to happen with jobs requiring many processors)

$W_{\text{min}} \leq 1$ iff $\mu \geq 1/\nu_b^2$, where

\[ \nu_b = \frac{-\sqrt{2C} + \sqrt{2C + 4(R + D)}}{2(R + D)} \]

\[ W^*_{\text{min}} = \min(W_{\text{min}}, 1) \]
Platform throughput

Sequential jobs

\[ \rho = (1 - W_{\text{min}}^*) N \]

Parallel jobs

\[ \rho = \sum_{j=0}^{Z} (1 - W_{\text{min}}^*(j)) 2^j \beta_j \]

use \( \mu_j \) instead of \( \mu \) in (2) to derive \( W_{\text{min}}^*(j) \)
Numerical results: yield $\rho/N$ for scenario “2015”

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\mu = 1$ month</th>
<th>$\mu = 1$ year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>per. chkpt.</td>
<td>prev. chkpt.</td>
</tr>
<tr>
<td>$2^8$</td>
<td>$96.04%$</td>
<td>$99.81%$</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>$88.23%$</td>
<td>$98.50%$</td>
</tr>
<tr>
<td>$2^{14}$</td>
<td>$62.28%$</td>
<td>$88.75%$</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>$10.66%$</td>
<td>$40.04%$</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>$1.33%$</td>
<td>$5.01%$</td>
</tr>
<tr>
<td>$2^8$</td>
<td>$98.89%$</td>
<td>$99.98%$</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>$96.80%$</td>
<td>$99.88%$</td>
</tr>
<tr>
<td>$2^{14}$</td>
<td>$90.59%$</td>
<td>$99.01%$</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>$70.46%$</td>
<td>$92.41%$</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>$15.96%$</td>
<td>$54.77%$</td>
</tr>
</tbody>
</table>
Limiting job size

- MTTF $\mu = 1$ year
- Exponentially distributed failures
- Scenario “2015”
- Tightly coupled parallel job with $2^{20}$ nodes (whole platform)

- Experiences a failure every 0.5 minutes!
- Throughput close to 0 for both fault tolerance and fault avoidance 😞
Limiting job size

- MTTF $\mu = 1$ year
- Exponentially distributed failures
- Scenario “2015”
- Tightly coupled parallel job with $2^{20}$ nodes (whole platform)
- Experiences a failure every 0.5 minutes!
- Throughput close to 0 for both fault tolerance and fault avoidance 😞
Yield $\rho/N$ for scenario “2015” and capped job sizes

<table>
<thead>
<tr>
<th>max job size</th>
<th>per. chkpt.</th>
<th>prev. chkpt.</th>
<th>prev. mig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{20}$</td>
<td>1.33%</td>
<td>5.01%</td>
<td>3.47%</td>
</tr>
<tr>
<td>$2^{19}$</td>
<td>2.67%</td>
<td>10.01%</td>
<td>6.93%</td>
</tr>
<tr>
<td>$2^{18}$</td>
<td>5.33%</td>
<td>20.02%</td>
<td>13.87%</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>10.66%</td>
<td>40.04%</td>
<td>27.73%</td>
</tr>
<tr>
<td>$2^{16}$</td>
<td>21.32%</td>
<td>63.07%</td>
<td>55.46%</td>
</tr>
<tr>
<td>$2^{15}$</td>
<td>42.64%</td>
<td>79.04%</td>
<td>74.72%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>max job size</th>
<th>per. chkpt.</th>
<th>prev. chkpt.</th>
<th>prev. mig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{20}$</td>
<td>15.96%</td>
<td>54.77%</td>
<td>45.65%</td>
</tr>
<tr>
<td>$2^{19}$</td>
<td>31.92%</td>
<td>73.57%</td>
<td>68.13%</td>
</tr>
<tr>
<td>$2^{18}$</td>
<td>55.59%</td>
<td>85.54%</td>
<td>82.56%</td>
</tr>
<tr>
<td>$2^{17}$</td>
<td>70.46%</td>
<td>92.41%</td>
<td>90.84%</td>
</tr>
<tr>
<td>$2^{16}$</td>
<td>80.05%</td>
<td>96.11%</td>
<td>95.30%</td>
</tr>
<tr>
<td>$2^{15}$</td>
<td>86.36%</td>
<td>98.03%</td>
<td>97.62%</td>
</tr>
</tbody>
</table>

$N = 2^{20}$, $\mu = 1$ month

$\mu = 1$ year

Yves.Robert@ens-lyon.fr
Conclusion

- Short term: prefer preventive migration to preventive checkpointing
- Longer term: not so clear, but may prefer preventive checkpointing

Long-term scenarios and very large scale platforms:
- Poor scaling of non-prediction-based traditional fault tolerance
- Even with perfect prediction, fault avoidance not much better
- Necessary to cap job size to achieve reasonable throughput

- Simulator: http://navet.ics.hawaii.edu/~casanova/software/resilience.tgz
Perspectives

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

- "Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

- Ahum, don't you see it coming? ...
  ... a nice little scheduling problem! 😊
  multi-criteria throughput/energy/reliability
  add replication

- Need combine all three approaches!
Perspectives

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

- "Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

- Ahum, don’t you see it coming? ... ... a nice little scheduling problem! 😊
  multi-criteria throughput/energy/reliability
  add replication

- Need combine all three approaches!
Perspectives

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

- "Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

- Ahum, don’t you see it coming? ...
  
  ... a nice little scheduling problem! 😊

  multi-criteria throughput/energy/reliability
  add replication

- Need combine all three approaches!
Perspectives

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

- "Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

- Ahum, don’t you see it coming? ...
  ... a nice little scheduling problem! 😊
  multi-criteria throughput/energy/reliability
  add replication

- Need combine all three approaches!
Perspectives

- Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

- "Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

- Ahum, don’t you see it coming? ...
  ...
  ... a nice little scheduling problem! 😊
  multi-criteria throughput/energy/reliability
  add replication

- Need combine all three approaches!
Software/hardware techniques to reduce checkpoint, recovery, migration times and to improve failure prediction

"Self-fault-tolerant" algorithms (e.g. asynchronous iterative)

Ahum, don’t you see it coming? ...
... a nice little scheduling problem! 😊
multi-criteria throughput/energy/reliability
add replication

Need combine all three approaches!