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[illegible]SC worksh[illegible]
- 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 [illegible] 2002
- I tal[illegible] [illegible]4
- I talked [illegible] [illegible]00[illegible]
- I talked [illegible] 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

Outline

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

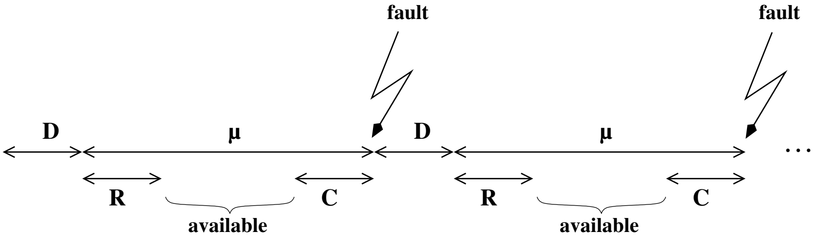
- Compare two well-known strategies:
  - Checkpointing: purely local, but can be very costly
  - Migration: requires availability of a spare resource

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

Relying on failure prediction

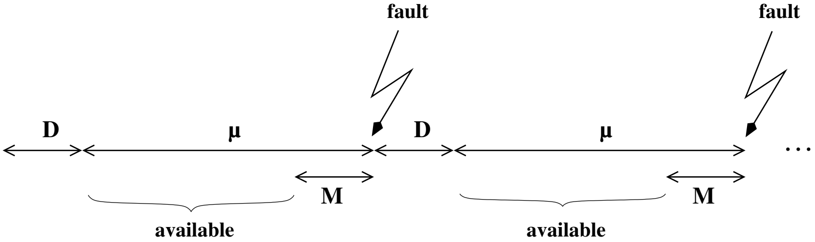
- Applications **will** face resource faults during execution
- Failure prediction available
  (e.g. alarm when a disk or CPU becomes unusually hot)
- Application must dynamically prepare for, and recover from, expected failures
- Compare two well-known strategies:
  - Preventive Checkpointing: purely 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
- μ: (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
- μ: (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)
- μ: mean time to failure
  (e.g., $1/\lambda$ if failures are exponentially distributed)
- N: total number of cluster nodes
- n: number of spares (migration)

Caveat

- Checkpointing/migration comparison makes sense only if

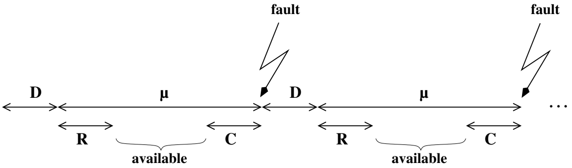
$$M < C + D + R$$

otherwise better use faulty machine as own spare

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

Outline

Checkpointing



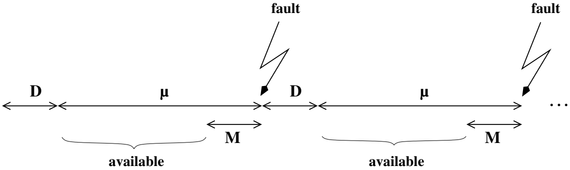
Probability of node being active

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

Global throughput

$$\rho_c = u_c \times N = \max\left(0, \frac{\mu - R - C}{\mu + D}\right) \times N$$

Migration (1/2)



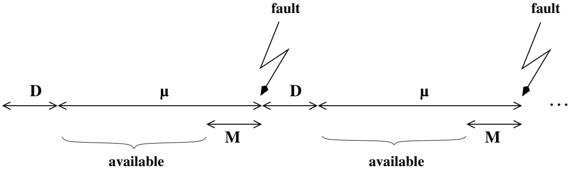
Probability of node being active

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

Global throughput

$$\rho_m = u_m \times (N - n) = \max\left(0, \frac{\mu - M}{\mu + D}\right) \times (N - n)$$

Migration (2/2)



No shortage of spare nodes?

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

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

Outline

Distribution (1/3)

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 $1 \leq j \leq Z = \log_2 N$

Distribution (1/3)

Number of processors required by typical jobs: *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 $1 \leq j \leq Z = \log_2 N$

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
  - β_j jobs that use 2^j processors exactly

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
  - β_j jobs that use 2^j processors exactly

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 β_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 β_j

Failures

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

- μ_j mean of the minimum of 2^j i.i.d. variables
- If the variables are exponentially distributed, with scale parameter λ, then

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

- If the variables are Weibull, with scale parameter λ 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?

- μ_j mean of the minimum of 2^j i.i.d. variables
- If the variables are exponentially distributed, with scale parameter λ, then

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

- If the variables are Weibull, with scale parameter λ and shape parameter a, then

$$\mu_j = \lambda\Gamma(1 + 1/(a2^j))$$

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

Outline

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" – $C = R = 10$, $D = 1$, $M = 0.33$

		Sequential Jobs		Parallel Jobs	
μ	N	$\varepsilon = 10^4$	$\varepsilon = 10^6$	$\varepsilon = 10^4$	$\varepsilon = 10^6$
1 day	2^{14}	1.19 (32)	1.16 (37)	3141.07 (32)	3140.08 (37)
	2^{17}	1.26 (164)	1.25 (177)	3086.92 (164)	3086.61 (177)
	2^{20}	1.28 (1086)	1.28 (1119)	3033.16 (1086)	3033.07 (1119)
1 week	2^{14}	0.14 (9)	0.12 (12)	3521.14 (9)	3520.47 (12)
	2^{17}	0.17 (35)	0.16 (40)	3511.74 (35)	3511.61 (40)
	2^{20}	0.18 (184)	0.18 (198)	3501.72 (184)	3501.67 (198)
1 month	2^{14}	0.02 (5)	0.00 (7)	1541.89 (5)	1541.69 (7)
	2^{17}	0.04 (13)	0.03 (17)	3354.95 (13)	3354.84 (17)
	2^{20}	0.04 (55)	0.04 (63)	3352.86 (55)	3352.83 (63)
1 year	2^{14}	-0.01 (2)	-0.01 (3)	69.22 (2)	69.21 (3)
	2^{17}	0.00 (4)	-0.00 (6)	1037.00 (4)	1036.99 (6)
	2^{20}	0.00 (11)	0.00 (13)	3381.52 (11)	3381.52 (13)

Scenario "2011" – $C = R = 5$, $D = 1$, $M = 0.33$

		Sequential Jobs		Parallel Jobs	
μ	N	$\varepsilon = 10^4$	$\varepsilon = 10^6$	$\varepsilon = 10^4$	$\varepsilon = 10^6$
1 day	2^{14}	0.48 (32)	0.45 (37)	1587.29 (32)	1586.78 (37)
	2^{17}	0.55 (164)	0.54 (177)	1573.40 (164)	1573.24 (177)
	2^{20}	0.57 (1086)	0.57 (1119)	1558.96 (1086)	1558.91 (1119)
1 week	2^{14}	0.04 (9)	0.02 (12)	1743.11 (9)	1742.77 (12)
	2^{17}	0.07 (35)	0.07 (40)	1741.00 (35)	1740.93 (40)
	2^{20}	0.08 (184)	0.08 (198)	1738.54 (184)	1738.52 (198)
1 month	2^{14}	-0.01 (5)	-0.02 (7)	734.36 (5)	734.26 (7)
	2^{17}	0.01 (13)	0.01 (17)	1656.28 (13)	1656.23 (17)
	2^{20}	0.02 (55)	0.02 (63)	1655.80 (55)	1655.78 (63)
1 year	2^{14}	-0.01 (2)	-0.02 (3)	25.16 (2)	25.15 (3)
	2^{17}	-0.00 (4)	-0.00 (6)	477.62 (4)	477.61 (6)
	2^{20}	0.00 (11)	0.00 (13)	1668.73 (11)	1668.73 (13)

Scenario "2015" – $C = 10R = 0.21$, $D = 0.25$, $M = 0.33$

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

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

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 = expected percentage of time lost, or "wasted":

$$W = \frac{C}{T} + \frac{T}{2\mu} \tag{1}$$

- First term in (1) by definition:
  C time-steps devoted to checkpointing every T time-steps
- Every μ 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 (1/3)

W = expected percentage of time lost, or "wasted":

$$W = \frac{C}{T} + \frac{T}{2\mu} \tag{1}$$

- First term in (1) by definition:
  C time-steps devoted to checkpointing every T time-steps
- Every μ 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)

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

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

Different from Daly:
target = steady-state operation of platform
target $\neq$ minimizing expected duration of a given job

Optimal period T (3/3)

$$W_{min} = \frac{R+D}{\mu} + \sqrt{\frac{2C}{\mu}} \qquad (2)$$

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

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

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

$$W^*_{min} = \min(W_{min}, 1)$$

Platform throughput

Sequential jobs

$$\rho = (1 - W^*_{min})N$$

Parallel jobs

$$\rho = \sum_{j=0}^{Z} (1 - W^*_{min}(j)) 2^j \beta_j$$

use μ_j instead of μ in (2) to derive $W^*_{min}(j)$

Numerical results: yield ρ/N for scenario "2015"

	$\mu = 1$ month		
N	per. chkpt.	prev. chkpt.	prev. mig.
2^8	96.04%	99.81%	98.99%
2^{11}	88.23%	98.50%	98.04%
2^{14}	62.28%	88.75%	86.41%
2^{17}	10.66%	40.04%	27.73%
2^{20}	1.33%	5.01%	3.47%

	$\mu = 1$ year		
N	per. chkpt.	prev. chkpt.	prev. mig.
2^8	98.89%	99.98%	99.59%
2^{11}	96.80%	99.88%	99.75%
2^{14}	90.59%	99.01%	98.79%
2^{17}	70.46%	92.41%	90.84%
2^{20}	15.96%	54.77%	45.46%

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 ρ/N for scenario "2015" and capped job sizes

	$N = 2^{20}$, $\mu = 1$ month		
max job size	per. chkpt.	prev. chkpt.	prev. mig.
2^{20}	1.33%	5.01%	3.47%
2^{19}	2.67%	10.01%	6.93%
2^{18}	5.33%	20.02%	13.87%
2^{17}	10.66%	40.04%	27.73%
2^{16}	21.32%	63.07%	55.46%
2^{15}	42.64%	79.04%	74.72%

	$\mu = 1$ year		
max job size	per. chkpt.	prev. chkpt.	prev. mig.
2^{20}	15.96%	54.77%	45.65%
2^{19}	31.92%	73.57%	68.13%
2^{18}	55.59%	85.54%	82.56%
2^{17}	70.46%	92.41%	90.84%
2^{16}	80.05%	96.11%	95.30%
2^{15}	86.36%	98.03%	97.62%

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!

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!