Checkpointing strategies for parallel jobs

Marin Bougeret, Henri Casanova, Mikaël Rabie,
Yves Robert, and Frédéric Vivien

ENS Lyon & INRIA, France
University of Hawai'i at Mānoa, USA
University of Montpellier, France

Motivation

Framework

- **Very very** large number of processing elements (e.g., 2^{20})
- Failure-prone platform (like any realistic platform)
- Large application to be executed on the whole platform

$\Longrightarrow$ Failure(s) will certainly occur before completion!

- Resilience provided through coordinated checkpointing

Question

- When should we checkpoint the application?

State of the art

One knows that applications should be checkpointed periodically

State of the art

One knows that applications should be checkpointed periodically
Is this optimal?

State of the art

One knows that applications should be checkpointed periodically

Is this optimal?

Several proposed values for period

- Young: $\sqrt{2 \times C \times \text{MTBF}}$ (1st order approximation)
- Daly (1): $\sqrt{2 \times C \times (R + \text{MTBF})}$ (1st order approximation)
- Daly (2): $\eta \times \text{MTBF} - C$, where $\eta = \xi^2 + 1 + \mathbb{L}(-e^{-(2\xi^2+1)})$, $\xi = \sqrt{\frac{C}{2\times\text{MTBF}}}$, and $\mathbb{L}(z)e^{\mathbb{L}(z)} = z$
  (higher order approximation)

State of the art

One knows that applications should be checkpointed periodically
Is this optimal?

Several proposed values for period

- Young: $\sqrt{2 \times C \times \text{MTBF}}$ (1st order approximation)
- Daly (1): $\sqrt{2 \times C \times (R + \text{MTBF})}$ (1st order approximation)
- Daly (2): $\eta \times \text{MTBF} - C$, where $\eta = \xi^2 + 1 + \mathbb{L}(-e^{-(2\xi^2+1)})$, $\xi = \sqrt{\frac{C}{2 \times \text{MTBF}}}$, and $\mathbb{L}(z)e^{\mathbb{L}(z)} = z$
  (higher order approximation)

How good are these approximations?
Could we find the optimal value? At least for Exponential failures?
And for Weibull failures?

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Outline

1. Single-processor jobs
   - Solving Makespan
   - Solving NextFailure
2. Parallel jobs
   - Solving Makespan
   - Solving NextFailure
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Hypotheses

- Overall size of work: $\mathcal{W}$
- Checkpoint cost: C
  (e.g., write on disk the contents of each processor memory)
- Downtime: D (hardware replacement by spare, or software rejuvenation via rebooting)
- Recovery cost after failure: R
- Homogeneous platform
  (same computation speeds, *iid* failure distributions)
- History of failures has no impact, only the time elapsed since last failure does
- A failure can happen during a checkpoint, a recovery, but not a downtime (otherwise replace D by 0 and R by $R + D$).

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Problem statement

MAKESPAN

- Minimize the job's expected makespan, that is:
  - the expectation $\mathbb{E}$
  - of the time T needed to process
  - a work of size $\mathcal{W}$
  - knowing that the (single) processor failed τ units of time ago.
- Notation:
  - minimize $\mathbb{E}(T(\mathcal{W}|\tau))$
  - $\omega_1(\mathcal{W}|\tau)$: amount of work we *attempt* to do before taking the first checkpoint

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) =$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \overbrace{\mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)}^{\substack{\text{Probability}\\ \text{of success}}}(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C))$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\,(\overbrace{\omega_1 + C}^{\substack{\text{Time needed}\\ \text{to compute}\\ \text{the 1st chunk}}} + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C))$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\,(\omega_1 + C + \overbrace{\mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C))}^{\substack{\text{Time needed to}\\ \text{compute the remainder}}})$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\left(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C)\right) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau))\left(\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}|R))\right) \end{array}$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\left(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C))\right. \\ + \\ \underbrace{(1 - \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau))}_{\text{Probability of failure}}\left(\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}|R))\right) \end{array}$$

Recursive approach

$$\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\left(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C)\right) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau))\left(\underbrace{\mathbb{E}(T_{lost}(\omega_1 + C|\tau))}_{\substack{\text{Time elapsed}\\ \text{before the failure}\\ \text{occured}}} + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}|R))\right) \end{array}$$

Recursive approach

$$
\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\,(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C)) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau))\,(\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \underbrace{\mathbb{E}(T_{rec})}_{\substack{\text{Time needed}\\ \text{to perform}\\ \text{downtime}\\ \text{and recovery}}} + \mathbb{E}(T(\mathcal{W}|R))) \end{array}
$$

Recursive approach

$$
\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau)\,(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C)) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\omega_1 + C|\tau))\,(\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \mathbb{E}(T_{rec}) + \underbrace{\mathbb{E}(T(\mathcal{W}|R))}_{\substack{\text{Time needed} \\ \text{to compute } \mathcal{W} \\ \text{from scratch}}}) \end{array}
$$

Recursive approach

$$
\mathbb{E}(T(\mathcal{W}|\tau)) = \begin{array}{l} \mathcal{P}_{\mathrm{succ}}(\omega_1 + C|\tau)\left(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C)\right) \\ + \\ (1 - \mathcal{P}_{\mathrm{succ}}(\omega_1 + C|\tau))\left(\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}|R))\right) \end{array}
$$

Problem: finding $\omega_1(\mathcal{W}, \tau)$ minimizing $\mathbb{E}(T(\mathcal{W}|\tau))$

Failures following an exponential distribution

Theorem

Optimal strategy splits $\mathcal{W}$ into K^* same-size chunks where

$$K^* = \max(1, \lfloor K_0 \rfloor) \text{ or } K^* = \lceil K_0 \rceil$$

(whichever leads to the smaller value)

where

$$K_0 = \frac{\lambda \mathcal{W}}{1 + \mathbb{L}(-e^{-\lambda C - 1})} \text{ and } \mathbb{L}(z)e^{\mathbb{L}(z)} = z$$

Optimal expectation of makespan is

$$K^* \left(e^{\lambda R} \left(\frac{1}{\lambda} + D \right) \right) \left(e^{\lambda \left(\frac{\mathcal{W}}{K^*} + C \right)} - 1 \right)$$

Arbitrary failure distributions

$$\mathbb{E}(T(\mathcal{W}|\tau)) =$$

$$\min_{0<\omega_1\leq\mathcal{W}} \left(\begin{array}{l} P_{suc}(\omega_1 + C|\tau)\Big(\omega_1 + C + \mathbb{E}(T(\mathcal{W} - \omega_1|\tau + \omega_1 + C))\Big) \\ \\ +(1 - P_{suc}(\omega_1 + C|\tau))\times \\ \qquad (\mathbb{E}(T_{lost}(\omega_1 + C|\tau)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}|R))) \end{array} \right)$$

Solve via dynamic programming

- Time quantum u: all chunk sizes ω_i are integer multiples of u
- Trade-off: accuracy versus higher computing time

Dynamic programming

Algorithm 1: $\textsc{DPMakespan}$ (x,b,y,τ_0)

```
if x = 0 then
    return 0
if solution[x][b][y] = unknown then
    best ← ∞;        τ ← bτ_0 + yu
    for i = 1 to x do
        exp_succ ← first(DPMakespan(x − i, b, y + i + C/u, τ_0))
        exp_fail ← first(DPMakespan(x, 0, R/u, τ_0))
        cur ← P_suc(iu + C|τ)(iu + C + exp_succ)
              +(1 − P_suc(iu + C|τ))( E(T_lost(iu + C, τ))
                                      + E(T_rec) + exp_fail )
        if cur < best then
            best ← cur;        chunksize ← i
    solution[x][b][y] ← (best, chunksize)
return solution[x][b][y]
```

Where:

$$best \leftarrow \infty; \quad \tau \leftarrow b\tau_0 + yu$$

$$exp\_succ \leftarrow first(\textsc{DPMakespan}(x - i, b, y + i + \tfrac{C}{u}, \tau_0))$$

$$exp\_fail \leftarrow first(\textsc{DPMakespan}(x, 0, \tfrac{R}{u}, \tau_0))$$

$$cur \leftarrow P_{suc}(iu + C|\tau)(iu + C + exp\_succ) + (1 - P_{suc}(iu + C|\tau))\Big(\mathbb{E}(T_{lost}(iu + C, \tau)) + \mathbb{E}(T_{rec}) + exp\_fail\Big)$$

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Problem statement

NEXTFAILURE

- Maximize expected amount of work completed before next failure
- Optimization on a "failure-by-failure" basis
- Hopefully a good approximation, at least for large job sizes $\mathcal{W}$

Approach

$$\mathbb{E}(W(\omega|\tau)) = P_{suc}(\omega_1 + C|\tau)(\omega_1 + \mathbb{E}(W(\omega - \omega_1|\tau + \omega_1 + C)))$$

Proposition

$$\mathbb{E}(W(\mathcal{W}|0)) = \sum_{i=1}^{K} \omega_i \times \prod_{j=1}^{i} P_{suc}(\omega_j + C|t_j)$$

where $t_j = \sum_{\ell=1}^{j-1} \omega_\ell + C$ *is the total time elapsed (without failure) before execution of chunk* ω_l*, and* K *is the (unknown) target number of chunks.*

Solving through dynamic programming

Algorithm 2: DPNextFailure (x, n, τ_0)

```
if x = 0 then
    return 0
if solution[x][n] = unknown then
    best ← ∞
    τ ← τ_0 + (W − xu) + nC
    for i = 1 to x do
        work = first(DPNextFailure(x − i, n + 1, τ_0))
        cur ← P_suc(iu + C|τ) × (iu + work)
        if cur < best then
            best ← cur;        chunksize ← i
    solution[x][n] ← (best, chunksize)
return solution[x][n]
```

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Failures following an exponential distribution

Theorem

Optimal strategy splits $\mathcal{W}(p)$ in $K^*(p)$ same-size chunks where

$$K^*(p) = \max(1, \lfloor K_0(p) \rfloor) \text{ or } K^*(p) = \lceil K_0(p) \rceil$$

(whichever leads to the smaller value)

$$\text{where } K_0(p) = \frac{\lambda \mathcal{W}(p)}{1 + \mathbb{L}(-e^{-p\lambda C - 1})} \text{ and } \mathbb{L}(z)e^{\mathbb{L}(z)} = z$$

Optimal expectation of makespan is

$$K^*(p) \left(\frac{1}{p\lambda} + \mathbb{E}(T_{rec}(p)) \right) \left(e^{\lambda \left(\frac{\mathcal{W}}{K^*(p)} + pC \right)} - 1 \right)$$

Arbitrary failure distributions

- Cannot solve analytically the recursion
- Cannot extend the dynamic programming algorithm DPMakespan designed for the single-processor case:
  - Would need to memorize all possible failure scenarios for each processor
  - Number of states exponential in p

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Dynamic programming

All τ variables evolve identically: recursive calls only correspond to cases in which no failure has occurred.

$$\mathbb{E}(W(\mathcal{W}|\tau_1,\ldots,\tau_p)) =$$
$$P_{suc}(\omega_1+C|\tau_1,\ldots,\tau_p)(\omega_1+\mathbb{E}(W(\mathcal{W}-\omega_1|\tau_1+\omega_1+C,\ldots,\tau_p+\omega_1+C)))$$

$\Rightarrow$ Same dynamic programming approach than previously

- Linear dependency in p (computation of P_{suc})
- Reduce complexity by recording only x most recent τ values and approximate the other values using y rounding values defined by x regularly-spaced quantiles

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Evaluated approaches

Heuristics

- Young [4]
- DalyLow [2]
- DalyHigh [2]
- Bouguerra [1]
- Liu [3]
- OptExp
- DPMakespan
- DPNextFailure

Theoretical bounds

- LowerBound (omniscient algorithm)
- PeriodLB

Synthetic failure distributions

	p_{total}	D	C, R	$MTBF$	$\mathcal{W}$
1-proc	1	60 s	600 s	1 h, 1 d, 1 w	20 d
Petascale	$45,208$	60 s	600 s	125 y, 500 y	$1,000$ y
Exascale	2^{20}	60 s	600 s	1250 y	$10,000$ y

Simulation parameters

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Sequential jobs under Exponential failures

Heuristics	MTBF 1 hour	1 day	1 week
LowerBound	0.62865	0.90714	0.979151
PeriodLB	1.00705	1.01588	1.02298
Young	1.01635	1.01590	1.02332
DalyLow	1.02711	1.01611	1.02338
DalyHigh	1.00700	1.01592	1.02373
Liu	1.01607	1.01655	1.02333
Bouguerra	1.02562	1.02329	1.02685
OptExp	1.00705	1.01611	1.02298
DPNextFailure	1.00785	1.01699	1.02851
DPMakespan	1.00737	1.01655	1.03467

Degradation from best, single processor, Exponential failures

Sequential jobs under Exponential failures

Heuristics	MTBF 1 hour	1 day	1 week
LowerBound	0.62865	0.90714	0.979151
PeriodLB	1.00705	1.01588	1.02298
Young	1.01635	1.01590	1.02332
DalyLow	1.02711	1.01611	1.02338
DalyHigh	1.00700	1.01592	1.02373
Liu	1.01607	1.01655	1.02333
Bouguerra	1.02562	1.02329	1.02685
OptExp	1.00705	1.01611	1.02298
DPNextFailure	1.00785	1.01699	1.02851
DPMakespan	1.00737	1.01655	1.03467

Degradation from best, single processor, Exponential failures

Sequential jobs under Weibull failures

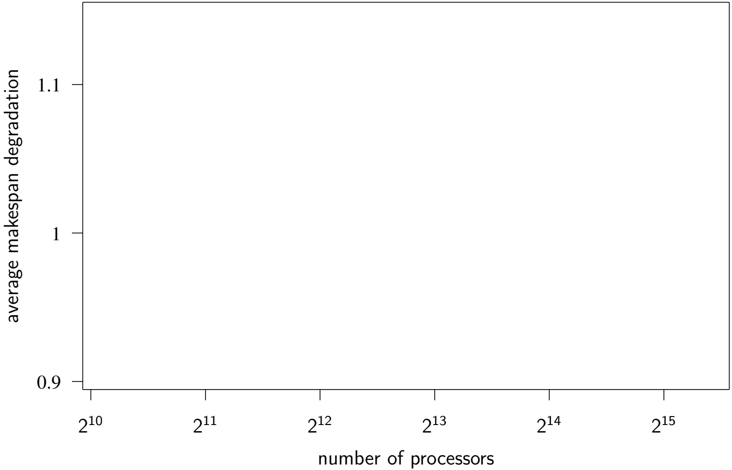
Heuristics	MTBF 1 hour	MTBF 1 day	MTBF 1 week
LowerBound	0.66417	0.90714	0.97915
PeriodLB	1.00960	1.01588	1.02298
Young	1.00965	1.01590	1.02332
DalyLow	1.01155	1.01611	1.02338
DalyHigh	1.01785	1.01592	1.02373
Liu	1.00914	1.01655	1.02333
Bouguerra	1.02936	1.02329	1.02685
OptExp	1.01788	1.01611	1.02298
DPNextFailure	1.01408	1.01699	1.02851
DPMakespan	1.00731	1.01655	1.03467

Degradation from best, single processor, Weibull failures

Outline

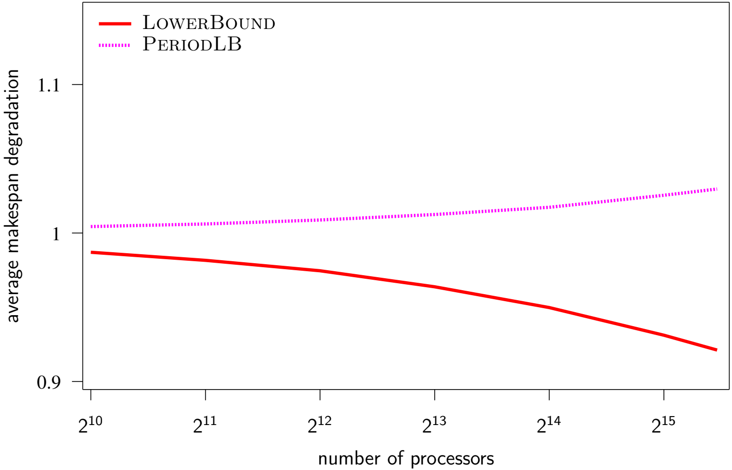
1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Parallel jobs under Exponential failures (1/2)



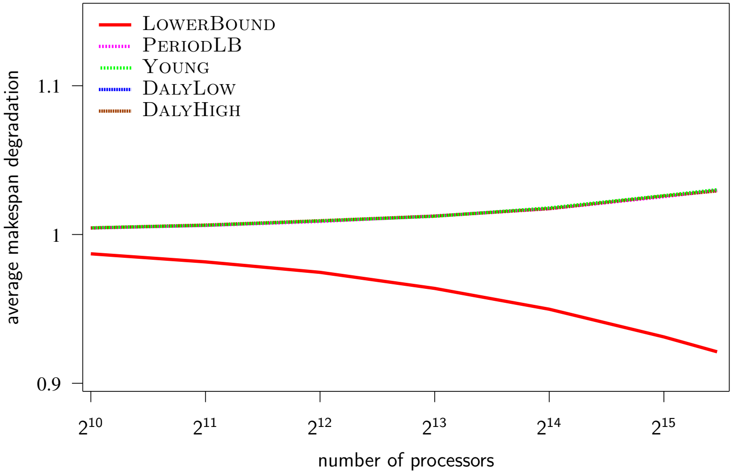
Petascale, MTBF $=$ 125 years

Parallel jobs under Exponential failures (1/2)



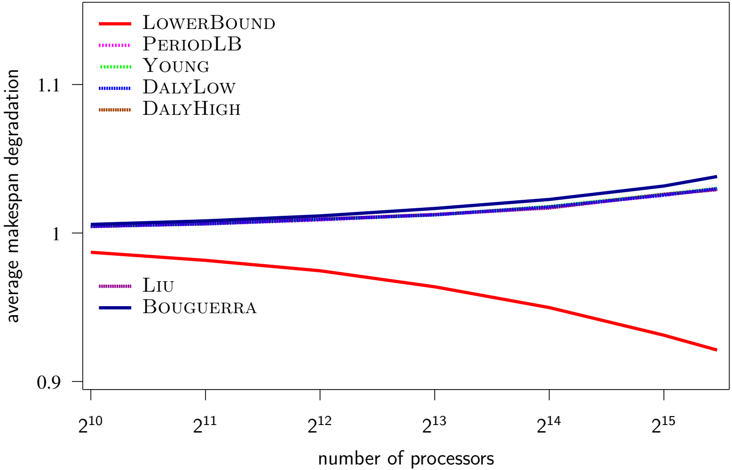
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



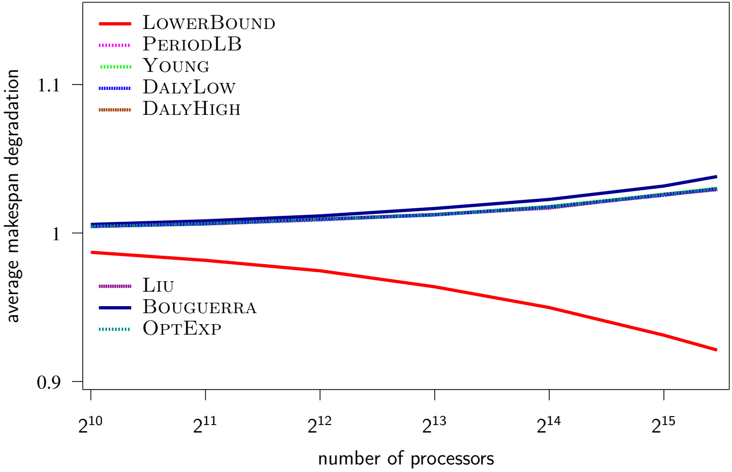
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



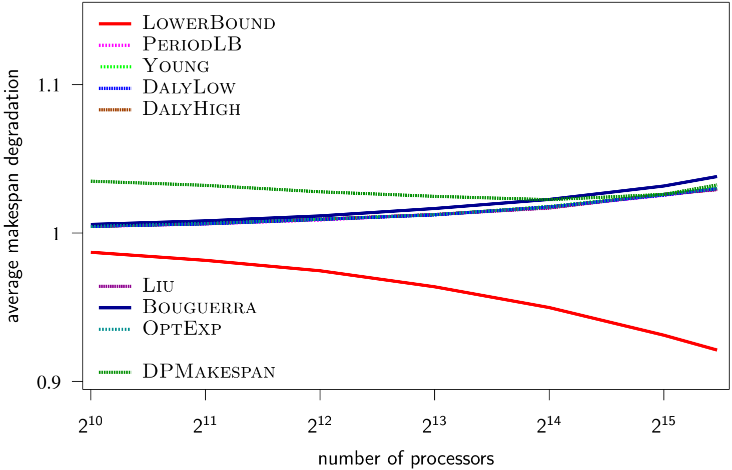
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



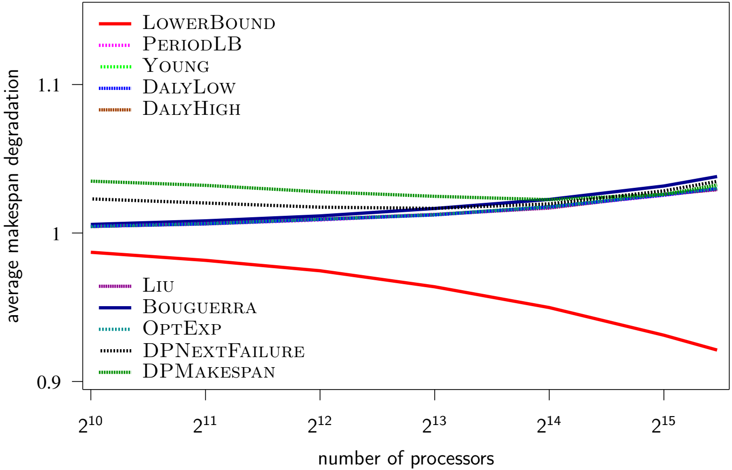
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



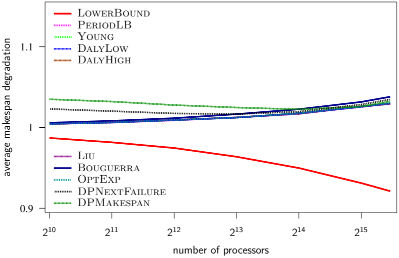
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



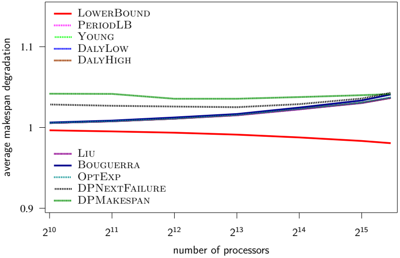
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (2/2)



Petascale

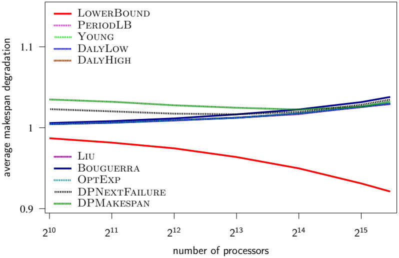
MTBF = 125 years



Petascale

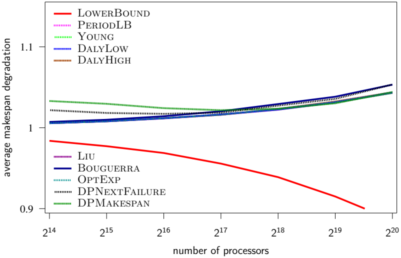
MTBF = 500 years

Parallel jobs under Exponential failures (2/2)



Petascale

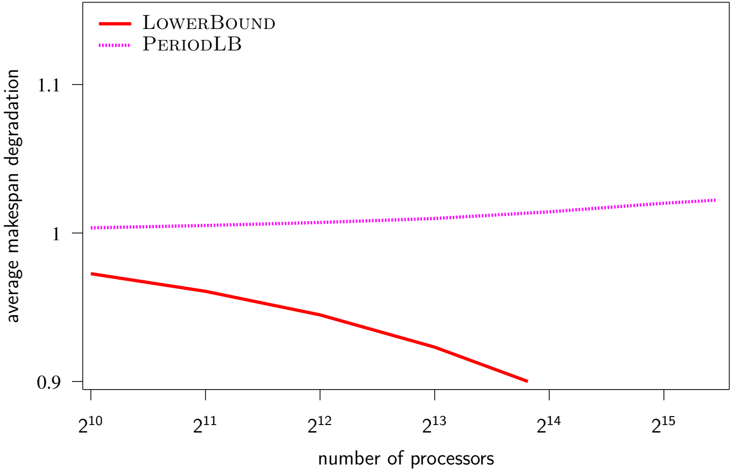
MTBF = 125 years



Exascale

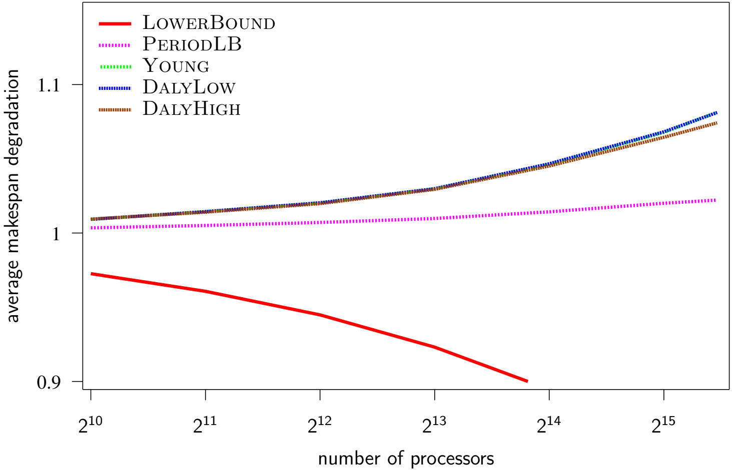
MTBF = 1250 years

Parallel jobs under Weibull failures (1/2)



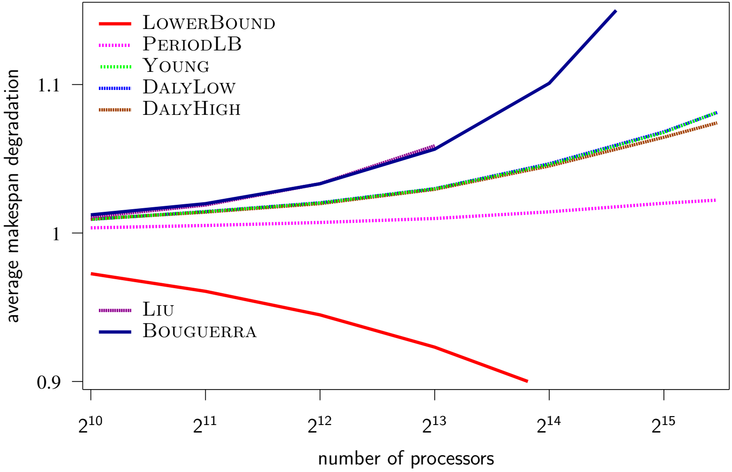
Petascale, MTBF = 125 years, k = 0.70

Parallel jobs under Weibull failures (1/2)



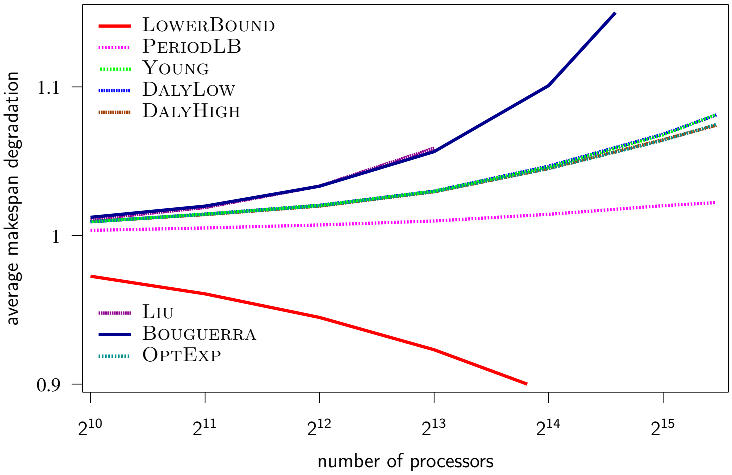
Petascale, MTBF = 125 years, k = 0.70

Parallel jobs under Weibull failures (1/2)



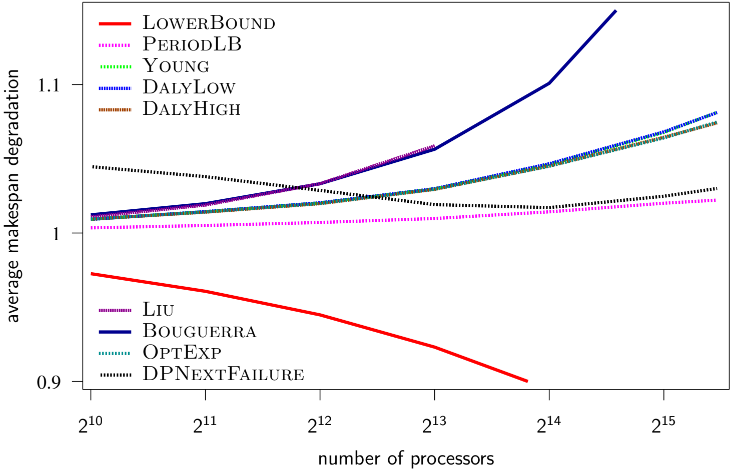
Petascale, MTBF = 125 years, k = 0.70

Parallel jobs under Weibull failures (1/2)



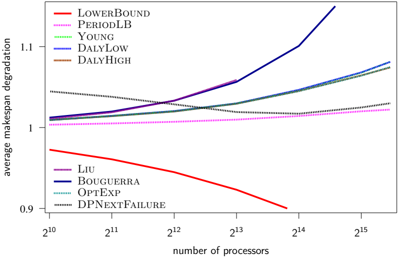
Petascale, MTBF = 125 years, k = 0.70

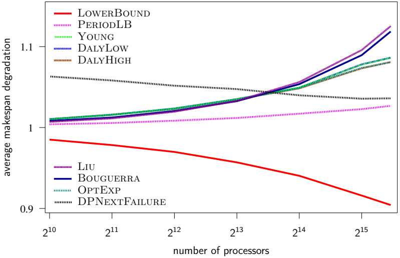
Parallel jobs under Weibull failures (1/2)



Petascale, MTBF = 125 years, k = 0.70

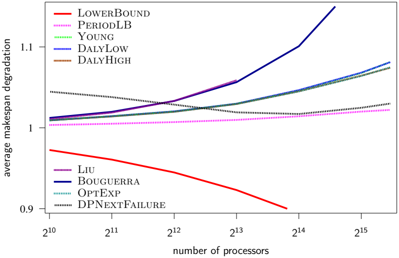
Parallel jobs under Weibull failures (2/2)



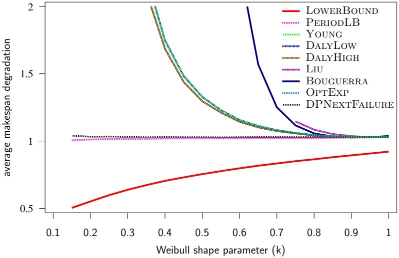


Petascale	Petascale
MTBF = 125 years	MTBF = 500 years
k = 0.70	k = 0.70

Parallel jobs under Weibull failures (2/2)

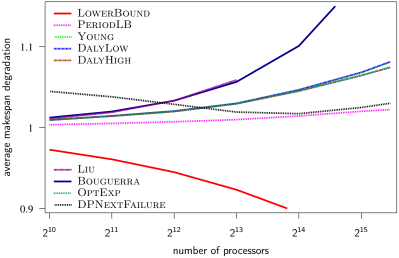


Petascale
MTBF = 125 years
k = 0.70

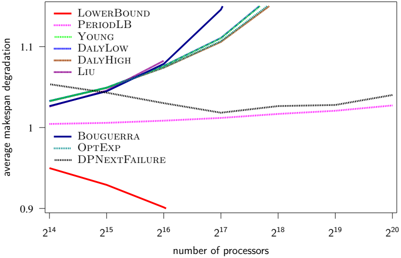


Petascale
MTBF = 125 years
45,208 processors

Parallel jobs under Weibull failures (2/2)



Petascale
MTBF = 125 years
k = 0.70

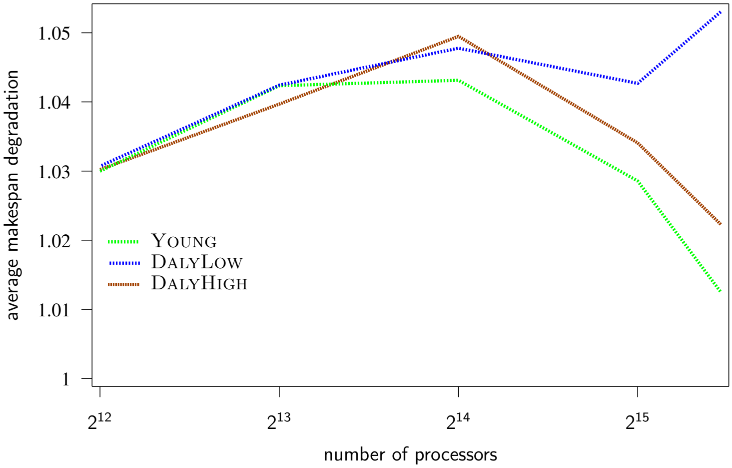


Exascale
MTBF = 1250 years
k = 0.70

Outline

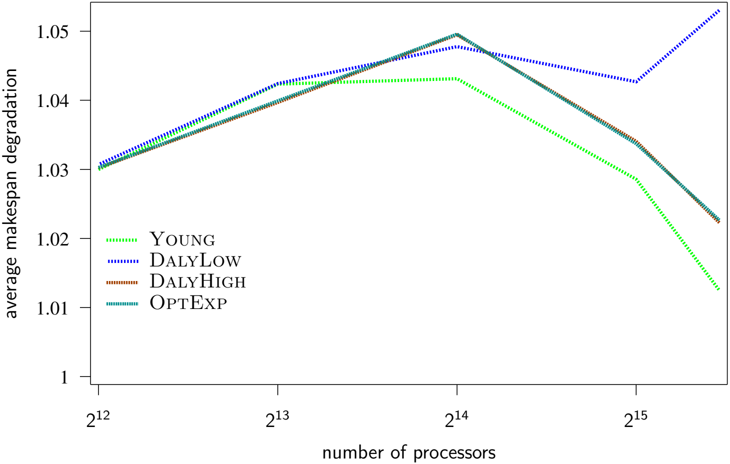
1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

LANL trace set



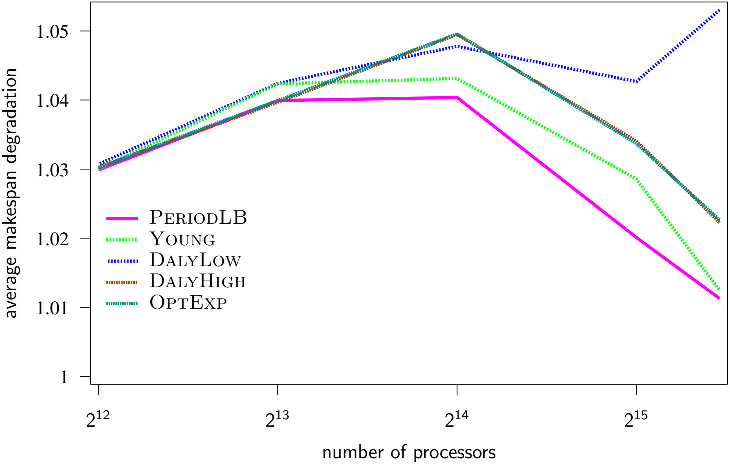
Petascale / LANL Cluster 18

LANL trace set



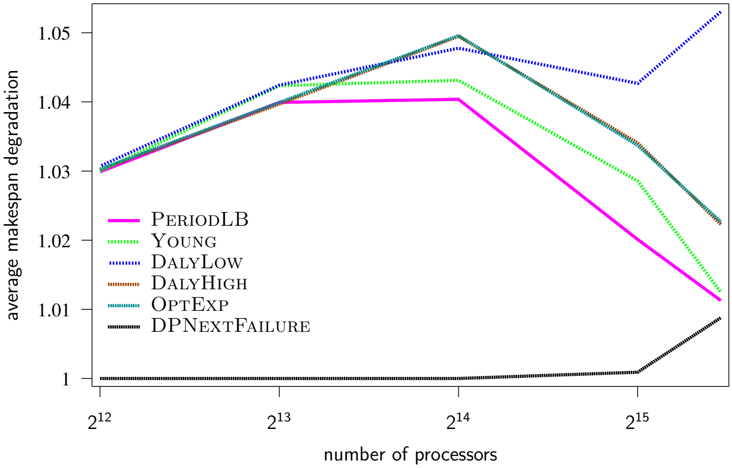
Petascale / LANL Cluster 18

LANL trace set



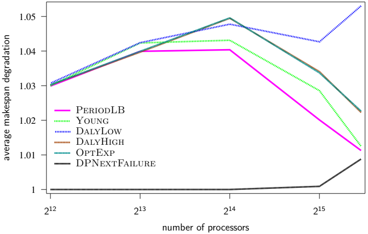
Petascale / LANL Cluster 18

LANL trace set

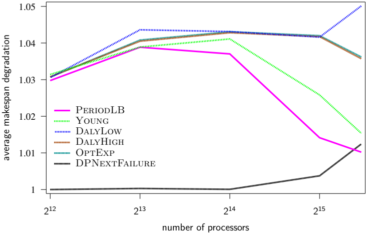


Petascale / LANL Cluster 18

LANL trace set



Petascale / LANL Cluster 18



Petascale / LANL Cluster 19

Outline

1. Single-processor jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
2. Parallel jobs
   - Solving MAKESPAN
   - Solving NEXTFAILURE
3. Experiments
   - Simulation framework
   - Sequential jobs under synthetic failures
   - Parallel jobs under synthetic failures
   - Parallel jobs under trace-based failures
4. Conclusion

Conclusion and perspectives

- Complete analytical solution for MAKESPAN/ Exponential
- Dynamic programming algorithms for NEXTFAILURE / Arbitrary distribution
- Makespan decreased by DPNEXTFAILURE (for the hardest cases)

- Future work
  Target non-coordinated checkpointing (e.g., hierarchical checkpointing with message logging)

Bibliography

M.-S. Bouguerra, T. Gautier, D. Trystram, and J.-M. Vincent.
A flexible checkpoint/restart model in distributed systems.
In *PPAM*, volume 6067 of *LNCS*, pages 206–215, 2010.

J. T. Daly.
A higher order estimate of the optimum checkpoint interval for restart dumps.
Future Generation Computer Systems, 22(3):303–312, 2004.

Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. Scott.
An optimal checkpoint/restart model for a large scale high performance computing system.
In *IPDPS 2008*, pages 1–9. IEEE, 2008.

J. W. Young.
A first order approximation to the optimum checkpoint interval