On scheduling the checkpoints of exascale applications

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

INRIA, École normale supérieure de Lyon, France
Univ. of Hawai'i at Mānoa

Motivation and framework

Framework

- A **very very** large number of processing elements (e.g., 2^{20})
- A platform that may fail (like any realistic platform)
- A very large application to be executed

$\Longrightarrow$ a failure may occur before completion

Questions

- When should we checkpoint the application?
- Should we always use all processors?

Hypotheses and notations

- Overall size of work: $\mathcal{W}$
- Checkpoints of fixed cost: c
  (e.g., write on disk the contents of each processor memory)
- Recovery cost after failure: r
- Homogeneous platform
  (processing elements have same speed and same failure distribution)

State of the art

Applications should be checkpointed periodically

Several proposed values for the 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
  $\eta = \xi^2 + 1 + \text{Lambert}(-e^{-(2\xi^2+1)})$, and $\xi = \sqrt{\frac{c}{2 \times \text{MTBF}}}$
  (higher order approximation)

State of the art

Applications should be checkpointed periodically

Is that the optimal behavior?

Several proposed values for the 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
  $\eta = \xi^2 + 1 + \text{Lambert}(-e^{-(2\xi^2+1)})$, and $\xi = \sqrt{\frac{c}{2 \times \text{MTBF}}}$
  (higher order approximation)

How good are these approximations?

What is the optimal value?

What about failures not following an exponential distribution?

Presentation outline

Plan

Plan

Problem statement

MAKESPAN

- Minimize the job's expected makespan
- Minimize $\mathbb{E}(T(\mathcal{W}|\tau))$

Principle of recursive approach (1)

Notation

- $\mathbb{E}_{opt}(\mathcal{W}, t)$: optimal expectation of makespan, for a work of size $\mathcal{W}$, knowing that the last failure happened t units of time ago.
- $W_1(\mathcal{W}, t)$: size of first chunk, for a work of size $\mathcal{W}$, knowing that the last failure happened t units of time ago.
- $\mathcal{P}_{\text{succ}}(W, t)$: probability that a work of size W is completed before next failure, knowing that the last failure happened t units of time ago.

Underlying hypothesis

- The history of failures does not have any impact, only the time elapsed since the last failure does (renewal process).

Principle of recursive approach (2)

$\mathbb{E}_{opt}(\mathcal{W}, t) =$

Principle of recursive approach (2)

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \overbrace{\mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t)}^{\text{Probability of success}} \overbrace{(W_1(\mathcal{W}, t) + c}^{\substack{\text{Time needed} \\ \text{to compute} \\ \text{the 1st chunk}}} + \overbrace{\mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}, t), t + W_1(\mathcal{W}, t) + c))}^{\text{Time needed to compute the remainder}}$$

Principle of recursive approach (2)

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \overbrace{\mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t)}^{\text{Probability of success}} \overbrace{(W_1(\mathcal{W}, t) + c}^{\substack{\text{Time needed} \\ \text{to compute} \\ \text{the 1st chunk}}} + \overbrace{\mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}, t), t + W_1(\mathcal{W}, t) + c))}^{\text{Time needed to compute the remainder}}$$

$$+$$

$$\underbrace{(1 - \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t))}_{\text{Probability of failure}} (\underbrace{\mathbb{E}_{lost}(W_1(\mathcal{W}, t) + c, t)}_{\substack{\text{Time elapsed} \\ \text{before the failure} \\ \text{occured}}} + \mathbb{E}_{rec}(r) + \underbrace{\mathbb{E}_{opt}(\mathcal{W}, r)}_{\substack{\text{Time needed} \\ \text{to compute } \mathcal{W} \\ \text{from scratch}}})$$

Failures following an exponential distribution

General expression

$$\begin{aligned}\mathbb{E}_{opt}(\mathcal{W}, t) = \\ &\mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t) \\ &\qquad \times (W_1(\mathcal{W}, t) + c + \mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}, t), t + W_1(\mathcal{W}, t) + c)) + \\ &(1 - \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}, t) + c, t))\,(\mathbb{E}_{lost}(W_1(\mathcal{W}, t) + c, t) + r' + \mathbb{E}_{opt}(\mathcal{W}, r))\end{aligned}$$

Simplified with memoryless property

$$\begin{aligned}\mathbb{E}_{opt}(\mathcal{W}) = \\ &\mathcal{P}_{\text{succ}}(W_1(\mathcal{W}) + c) \\ &\qquad \times (W_1(\mathcal{W}) + c + \mathbb{E}_{opt}(\mathcal{W} - W_1(\mathcal{W}))) + \\ &(1 - \mathcal{P}_{\text{succ}}(W_1(\mathcal{W}) + c))\,(\mathbb{E}_{lost}(W_1(\mathcal{W}) + c) + r' + \mathbb{E}_{opt}(\mathcal{W}))\end{aligned}$$

with $r' = \mathbb{E}_{rec}(r)$

Remarks

- The first chunk successfully executed will be of size $W_1(\mathcal{W})$
- Whatever the scenario, the size of the "successful" chunks are known beforehand. There are $n_0(\mathcal{W})$ chunks.

Optimal checkpointing policy

$$\mathbb{E}_{opt}(\mathcal{W}) = \sum_{i=1}^{n_0(\mathcal{W})} \left(W_i(\mathcal{W}) + c + \frac{1 - \mathcal{P}_{\text{succ}}(W_i(\mathcal{W}) + c)}{\mathcal{P}_{\text{succ}}(W_i(\mathcal{W}) + c)} (\mathbb{E}_{lost}(W_i(\mathcal{W}) + c) + r') \right)$$

$$\mathbb{E}_{opt}(\mathcal{W}) = \left(\frac{1}{\lambda} + r' \right) \sum_{i=1}^{n_0(\mathcal{W})} (\mathrm{e}^{\lambda(W_i(\mathcal{W}) + c)} - 1)$$

Theorem

The expectation of the makespan is minimized when checkpoints are periodic of period $T_{opt} = \frac{1 + Lambert(-e^{-(1+\lambda c)})}{\lambda}$ *and* $n_0(\mathcal{W}) = \frac{\mathcal{W}}{T_{opt}}$, *with* $Lambert(x)e^{Lambert(x)} = x$. *Then,*

$$\mathbb{E}_{opt}(\mathcal{W}) = \frac{\mathrm{e}^{\lambda(T_{opt} + c)} - 1}{T_{opt}} \mathcal{W} \left(\frac{1}{\lambda} + r' \right)$$

Approximation and dynamic programming

Idea

- Time discretization: chunk sizes must be a multiple of a quantum u

Dynamic programming solution

$$\mathbb{E}_{opt}(\mathcal{W}, t) = \min_{\substack{W_1 = i.u \\ 1 \leq i \leq \frac{\mathcal{W}}{u}}} \begin{cases} \mathcal{P}_{\text{succ}}(W_1 + c, t)\,(W_1 + c + \mathbb{E}_{opt}(\mathcal{W} - W_1, t + W_1 + c)) \\ +(1 - \mathcal{P}_{\text{succ}}(W_1 + c, t))\,(\mathbb{E}_{lost}(W_1 + c, t) + r' + \mathbb{E}_{opt}(\mathcal{W}, r)) \end{cases}$$

Theorem

We have an algorithm in $O\left(\left(\frac{\mathcal{W}}{u}\right)^3 \left(1 + \frac{c}{u}\right)\right)$ *to compute* $\mathbb{E}_{opt}(\mathcal{W})$.

$\Longrightarrow$ numerical approximations

Plan

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

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

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.*

Dynamic programming

Algorithm 1: DPNextFailure (x, n, τ_0)

if $x = 0$ **then**
  return 0
if $solution[x][n] = unknown$ **then**
  $best \leftarrow \infty$;
  $\tau \leftarrow \tau_0 + (\mathcal{W} - xu) + nC$;
  for $i = 1$ *to* x **do**
    $work = first(\text{DPNextFailure}(x - i, n + 1, \tau))$;
    $cur \leftarrow P_{suc}(iu + C|\tau) \times (iu + work)$;
    if $cur < best$ **then**
      $best \leftarrow cur$; $chunksize \leftarrow i$
  $solution[x][n] \leftarrow (best, chunksize)$
return $solution[x][n]$

Plan

Scenarios

Checkpointing/recovery overheads

- Checkpoints synchronized over all processors
- Cost $C(p)$ and $R(p)$ with p processors
- Application's memory footprint: V bytes, with V/p per processor

- Proportional overhead: $C(p) \propto R(p) \propto V/p$
- Constant overhead: $C(p) \propto R(p) \propto V$

Parallel work

- $\mathcal{W}(p)$ time required for failure-free execution on p processors

- Embarrassingly parallel jobs: $\mathcal{W}(p) = \mathcal{W}/p$
- Amdahl parallel jobs: $\mathcal{W}(p) = \mathcal{W}/p + \gamma\mathcal{W}$
- Numerical kernels: $\mathcal{W}(p) = \mathcal{W}/p + \gamma\mathcal{W}^{2/3}/\sqrt{p}$

Plan

Exponential distribution

Aggregate p processors into a virtual "macro-processor":

- Exponential failure law of parameter $\lambda' = p\lambda$;
- Checkpoint/recovery overheads $C(p)$ and $R(p)$

Exponential distribution

Aggregate p processors into a virtual "macro-processor":

- Exponential failure law of parameter $\lambda' = p\lambda$;
- Checkpoint/recovery overheads $C(p)$ and $R(p)$

- Optimal solution (chunks) by previous Theorem
- Cannot analytically compute expected execution time ($\mathbb{E}(T_{rec})$ complicated)

Arbitrary failure distributions

- Cannot extend dynamic programming algorithm DPMakespan
- Would need to memorize all possible failure scenarios for each processor
- Number of states exponential in p

Arbitrary failure distributions

- Cannot extend dynamic programming algorithm DPMakespan
- Would need to memorize all possible failure scenarios for each processor
- Number of states exponential in p

Open problem?!

Plan

Dynamic programming

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

$$\mathbb{E}(W(\omega|\tau_1,\ldots,\tau_p)) = \begin{cases} \omega_1 + \mathbb{E}(W(\omega-\omega_1|\tau_1+\omega_1+C(p),\ldots,\tau_p+\omega_1+C(p)) \\ \quad \text{if none of the processors fails during a time } \omega_1+C(p) \\ 0 \quad \text{otherwise.} \end{cases}$$

- 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

Plan

Plan

Heuristics

- Young [4]
- DalyLow [2]
- DalyHigh [2]
- Bouguerra [1]
- Liu [3]
- OptExp
- DPMakespan
- DPNextFailure
- 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
Peta	$45,208$	60 s	600 s	125 y, 500 y	$1,000$ y
Exa	2^{20}	60 s	600 s	1250 y	$10,000$ y

Simulation parameters

Plan

Sequential jobs under Exponential failures

Heuristics	MTBF 1 hour	1 day	1 week
LowerBound	0.62852	0.90684	0.97870
PeriodLB	1.00743	1.01557	1.02259
Young	1.01746	1.01567	1.02319
DalyLow	1.02801	1.01607	1.02325
DalyHigh	1.00748	1.01569	1.02335
Liu	1.01734	1.05440	1.20767
Bouguerra	1.02640	1.02288	1.02662
OptExp	1.00743	1.01557	1.02259
DPNextFailure	1.00793	1.01666	1.02791
DPMakespan	1.00783	1.01564	1.03425

Degradation from best, single processor, Exponential failures

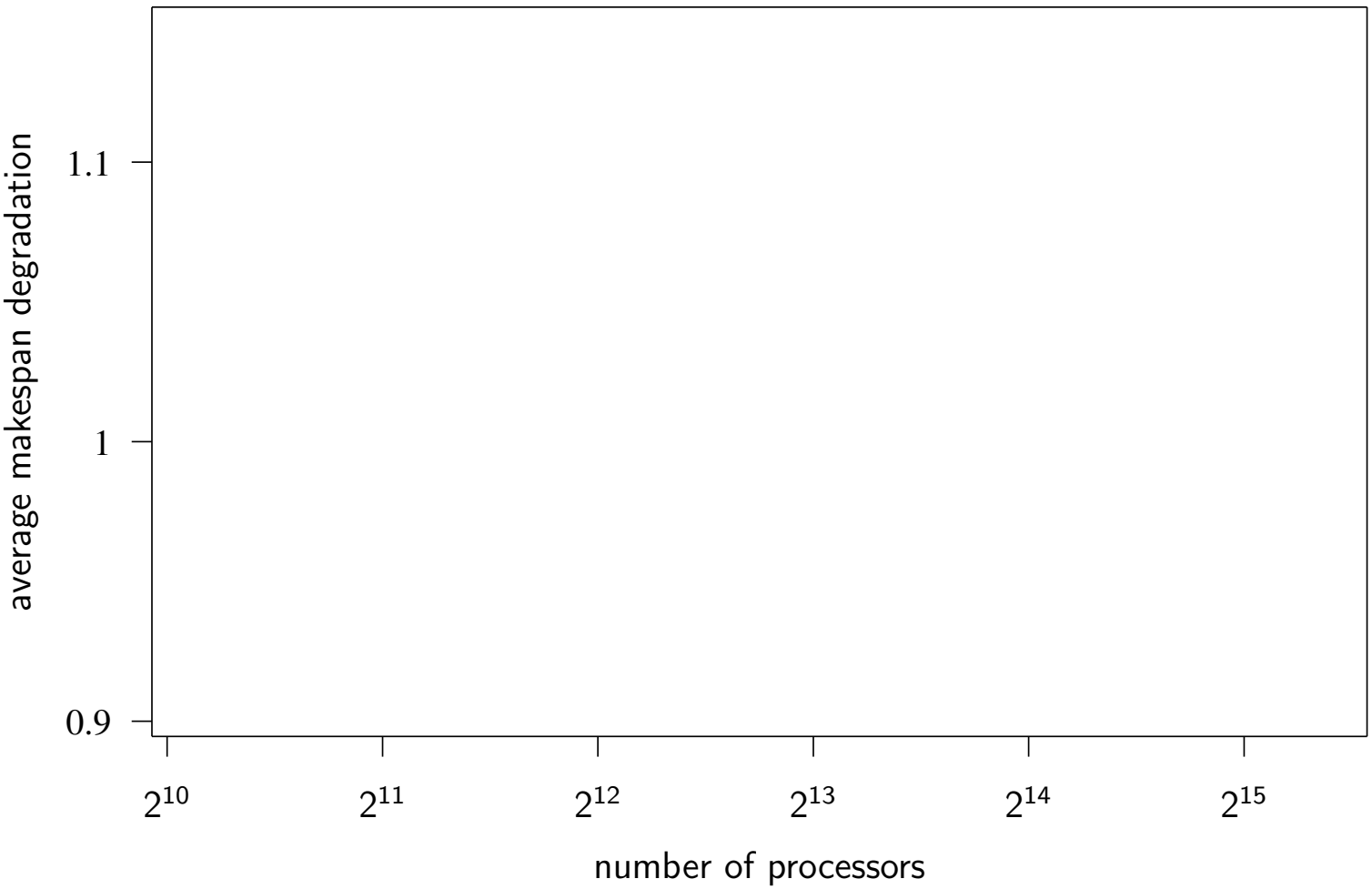
Sequential jobs under Weibull failures

Heuristics	MTBF 1 hour	1 day	1 week
LowerBound	0.66351	0.91012	0.97612
PeriodLB	1.00991	1.01596	1.02236
Young	1.00981	1.01643	1.02293
DalyLow	1.01182	1.01666	1.02298
DalyHigh	1.01741	1.01625	1.02286
Liu	1.01051	1.07015	1.19324
Bouguerra	1.02843	1.01873	1.02324
OptExp	1.01734	1.01646	1.02236
DPNextFailure	1.01360	1.01654	1.02716
DPMakespan	1.00739	1.01547	1.03459

Degradation from best, single processor, Weibull failures

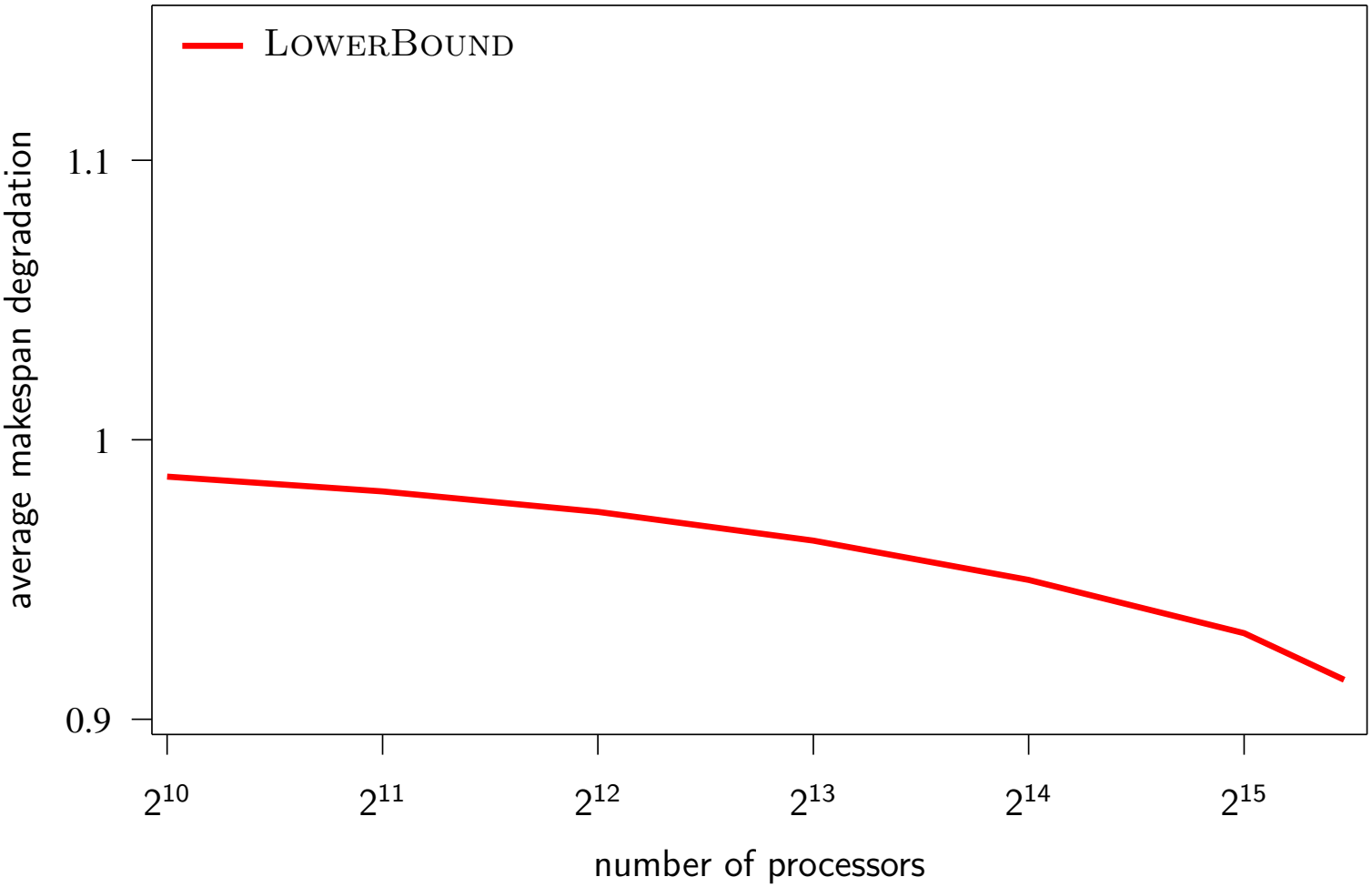
Plan

Parallel jobs under Exponential failures (1/2)



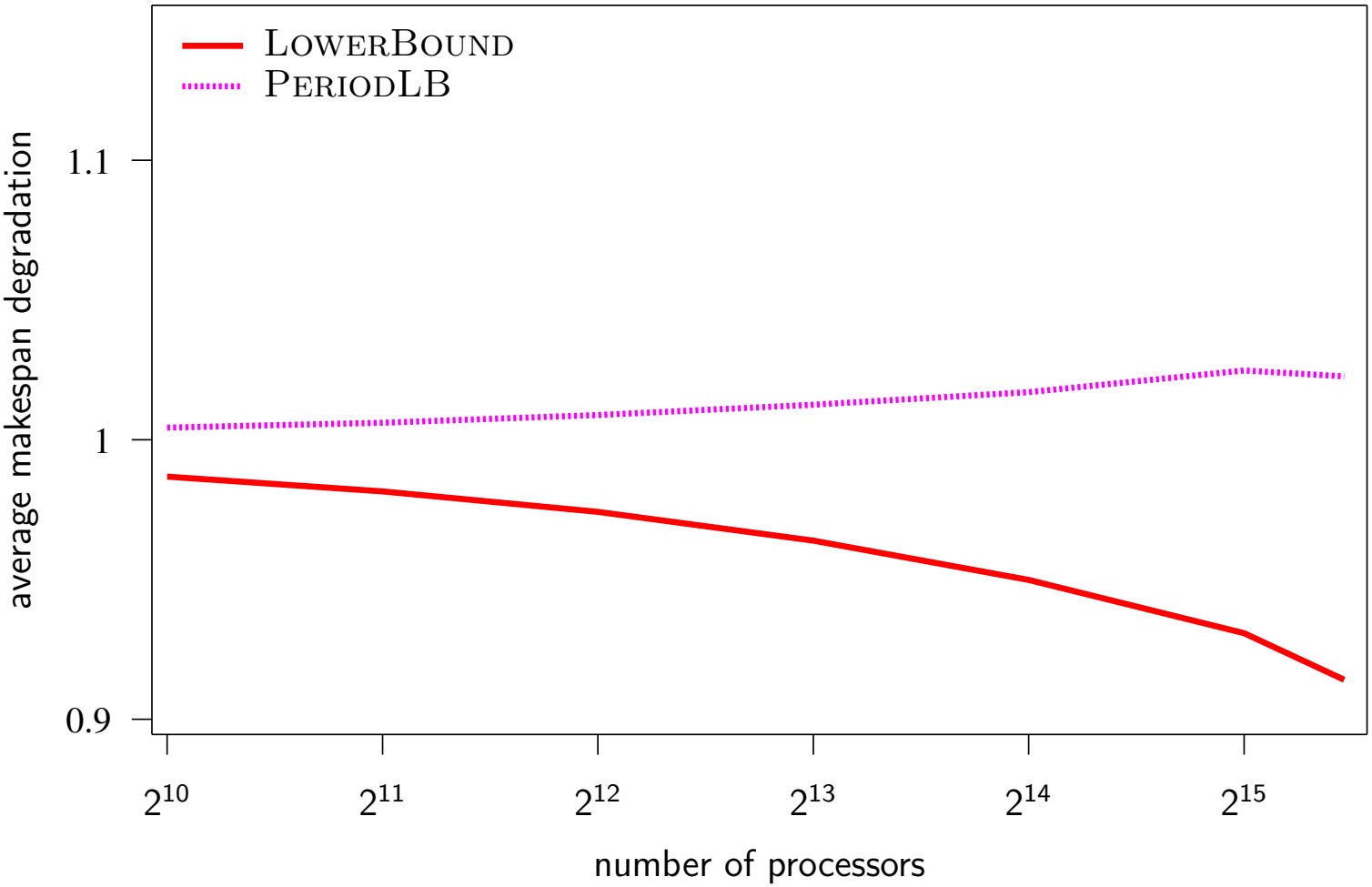
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



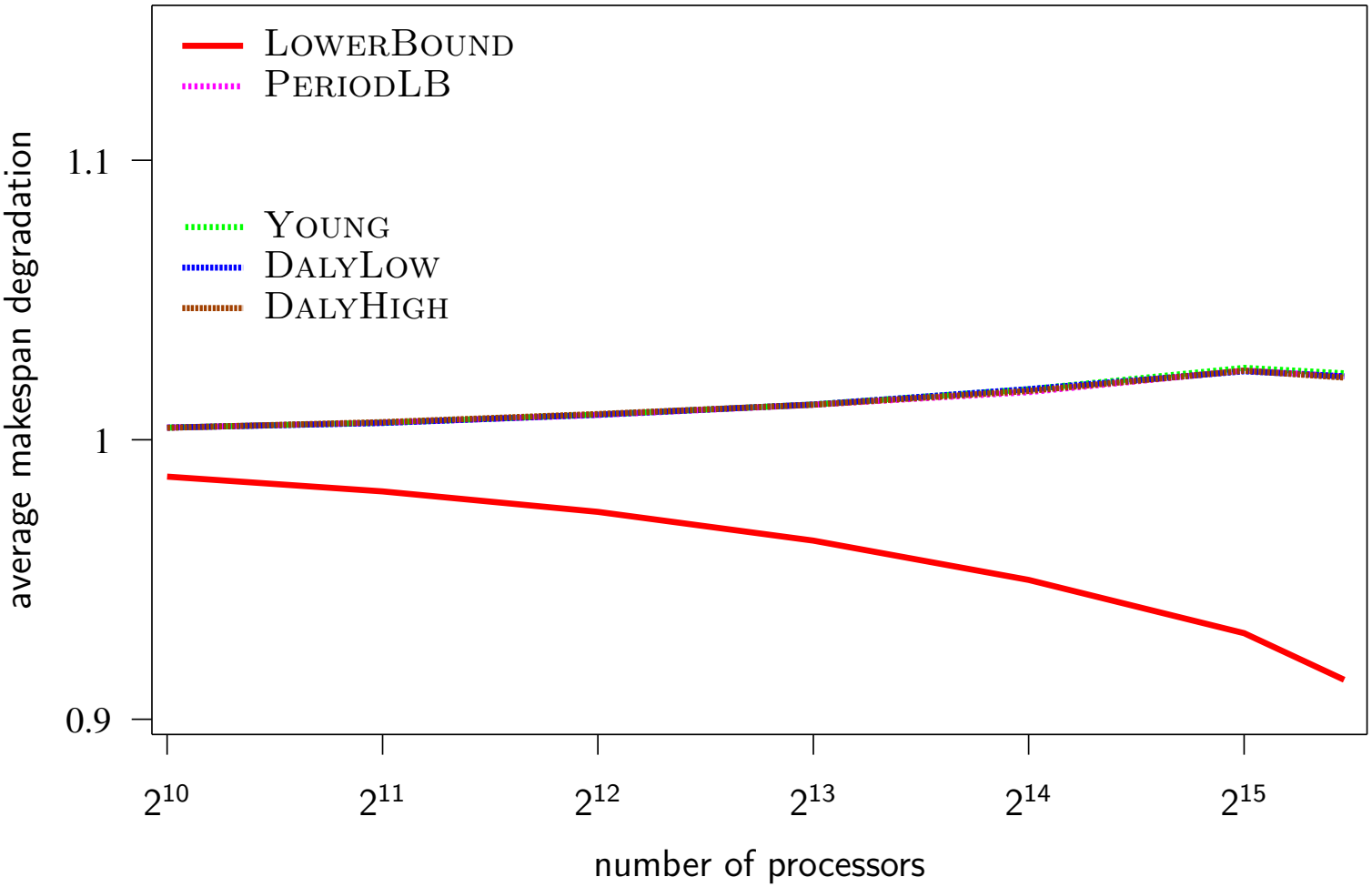
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



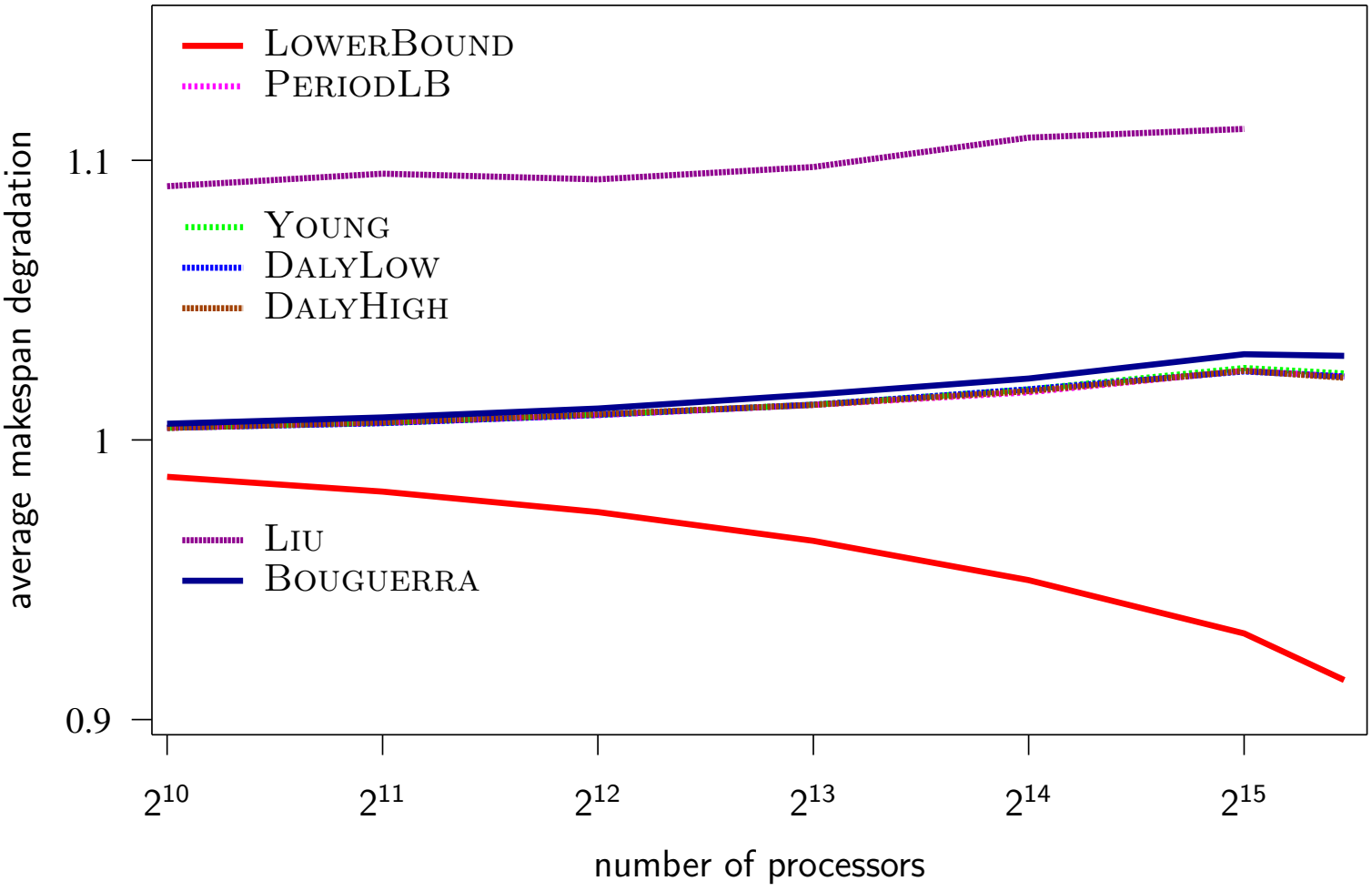
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



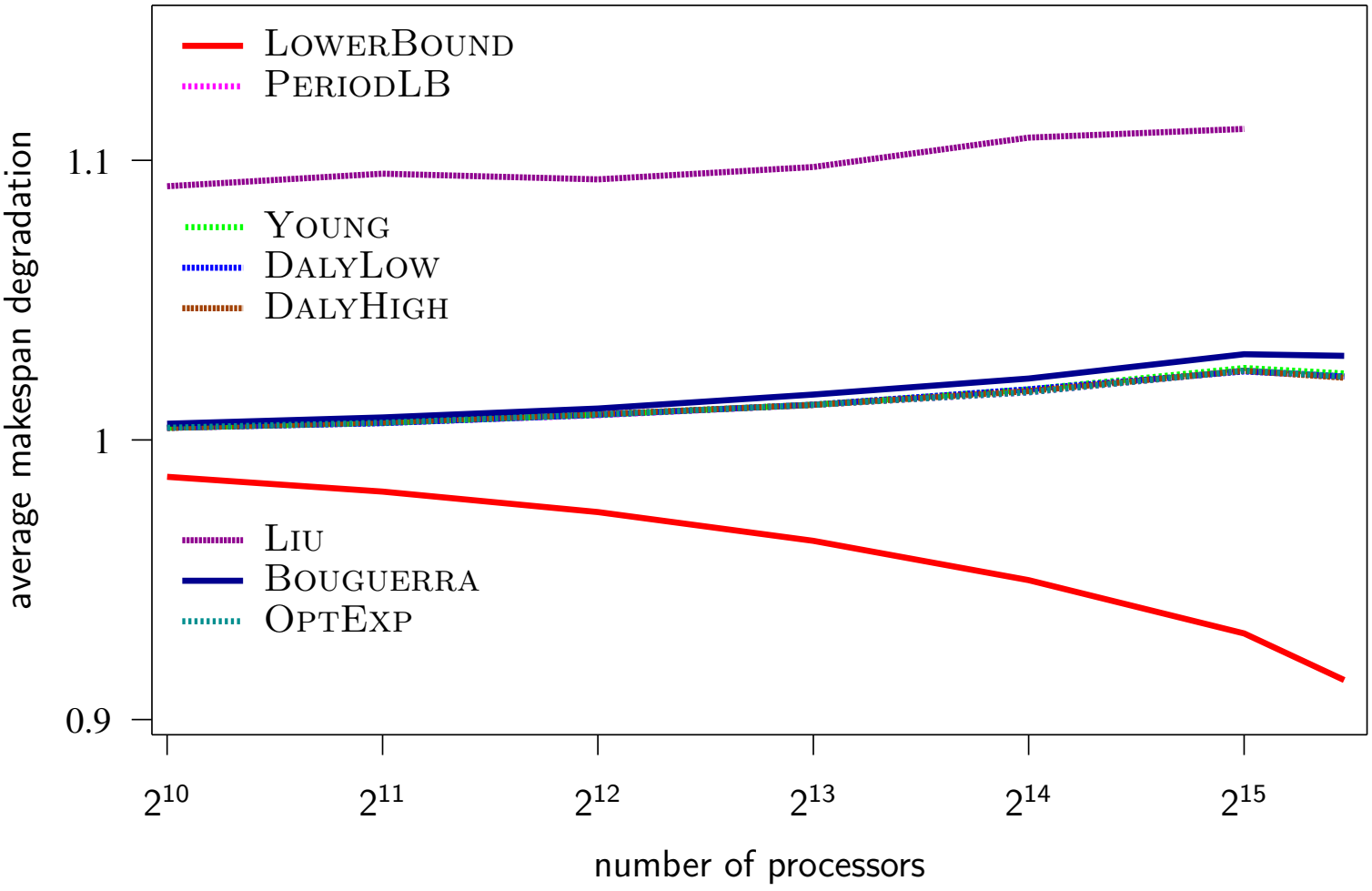
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



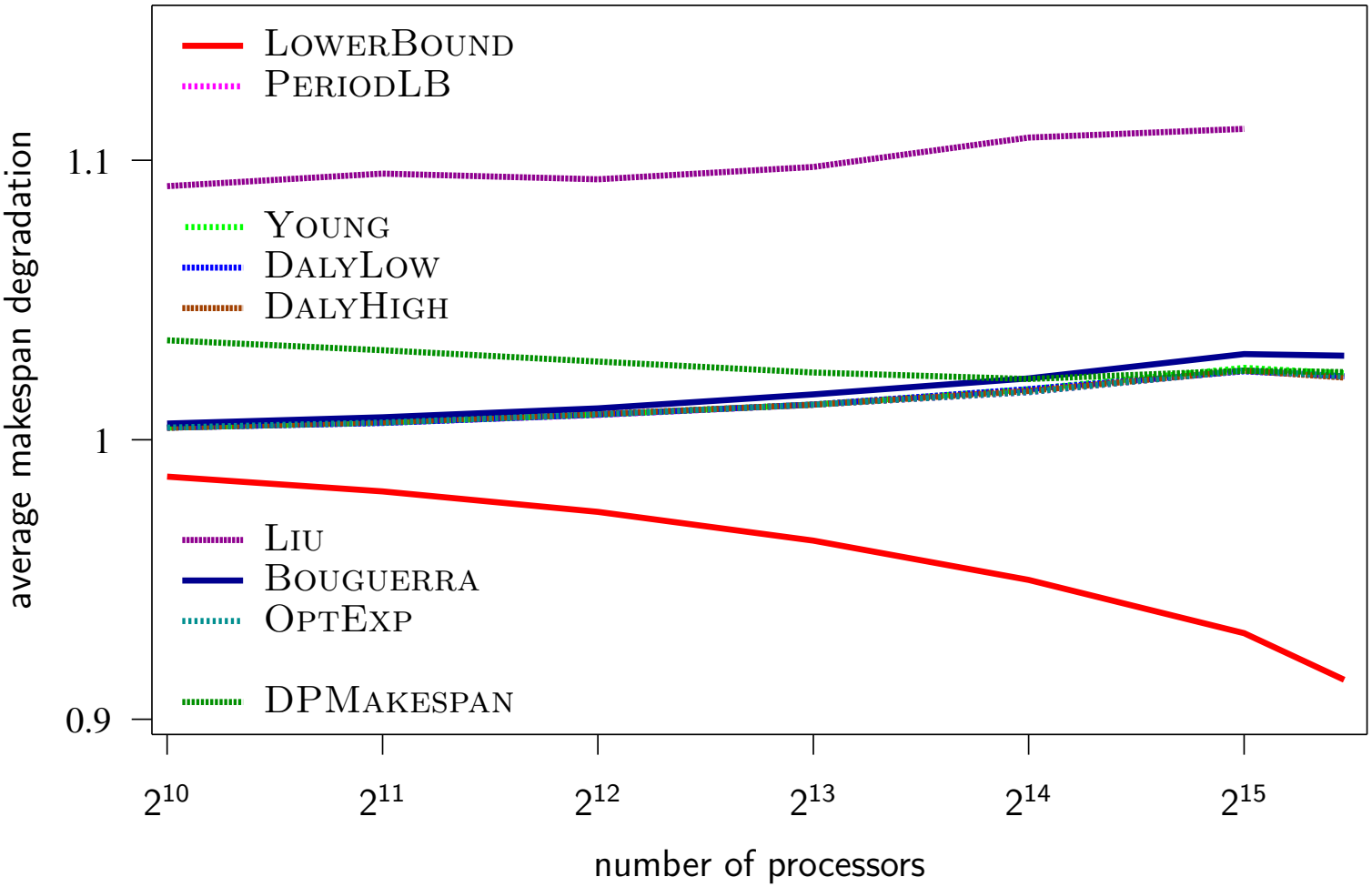
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



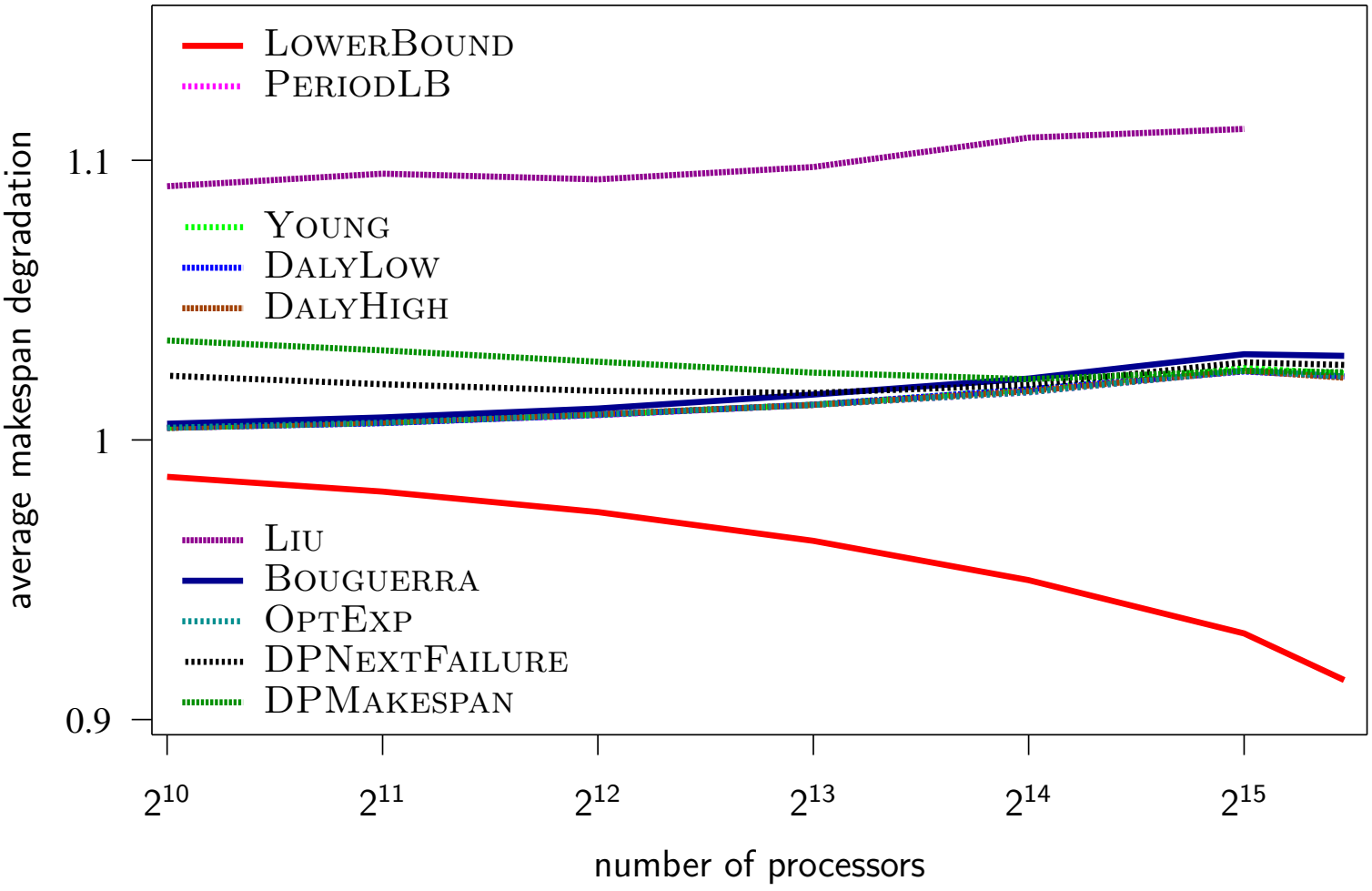
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)



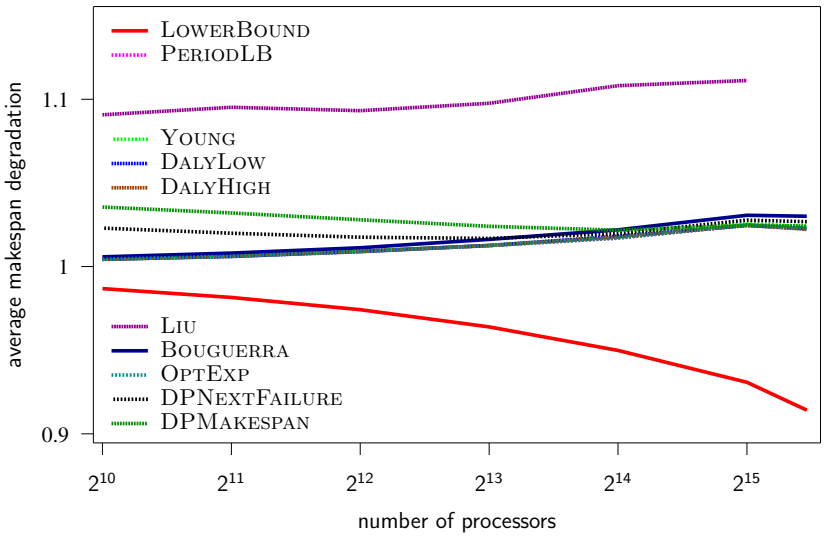
Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (1/2)

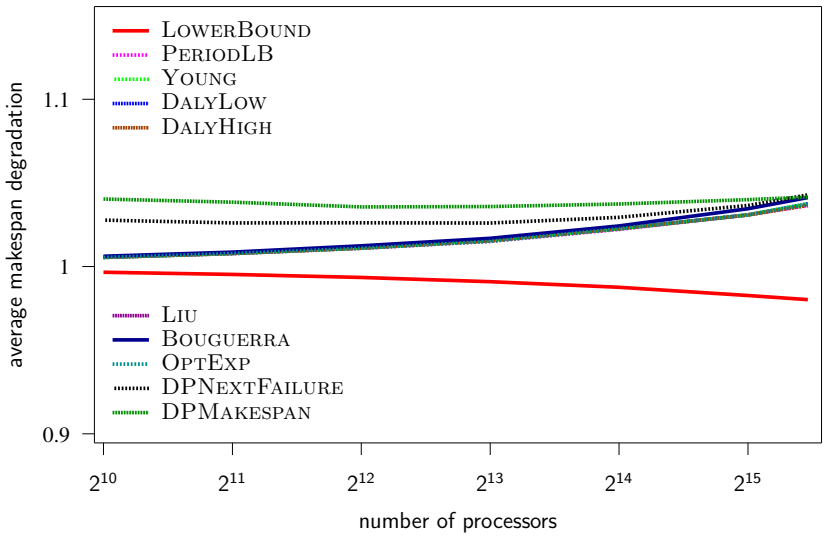


Petascale, MTBF = 125 years

Parallel jobs under Exponential failures (2/2)

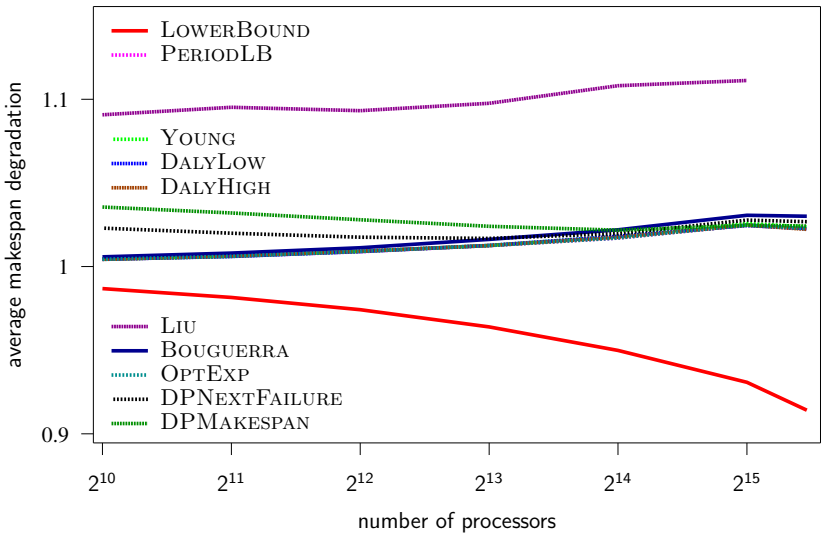


Petascale
MTBF = 125 years

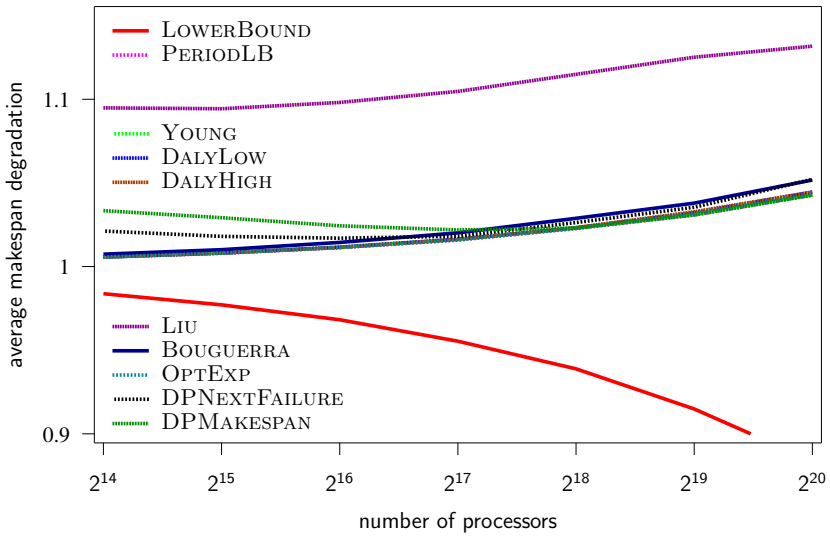


Petascale
MTBF = 500 years

Parallel jobs under Exponential failures (2/2)

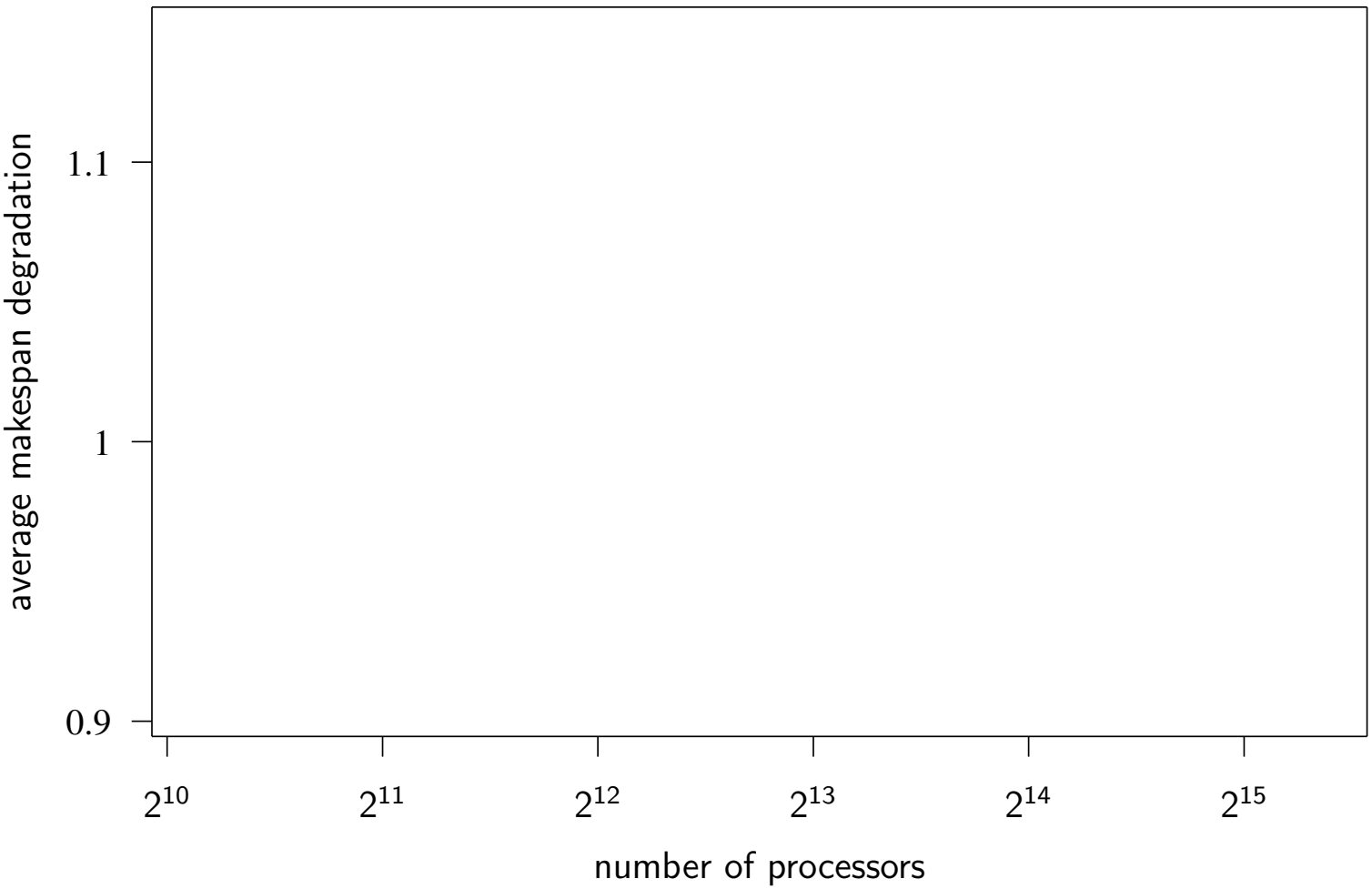


Petascale
MTBF = 125 years



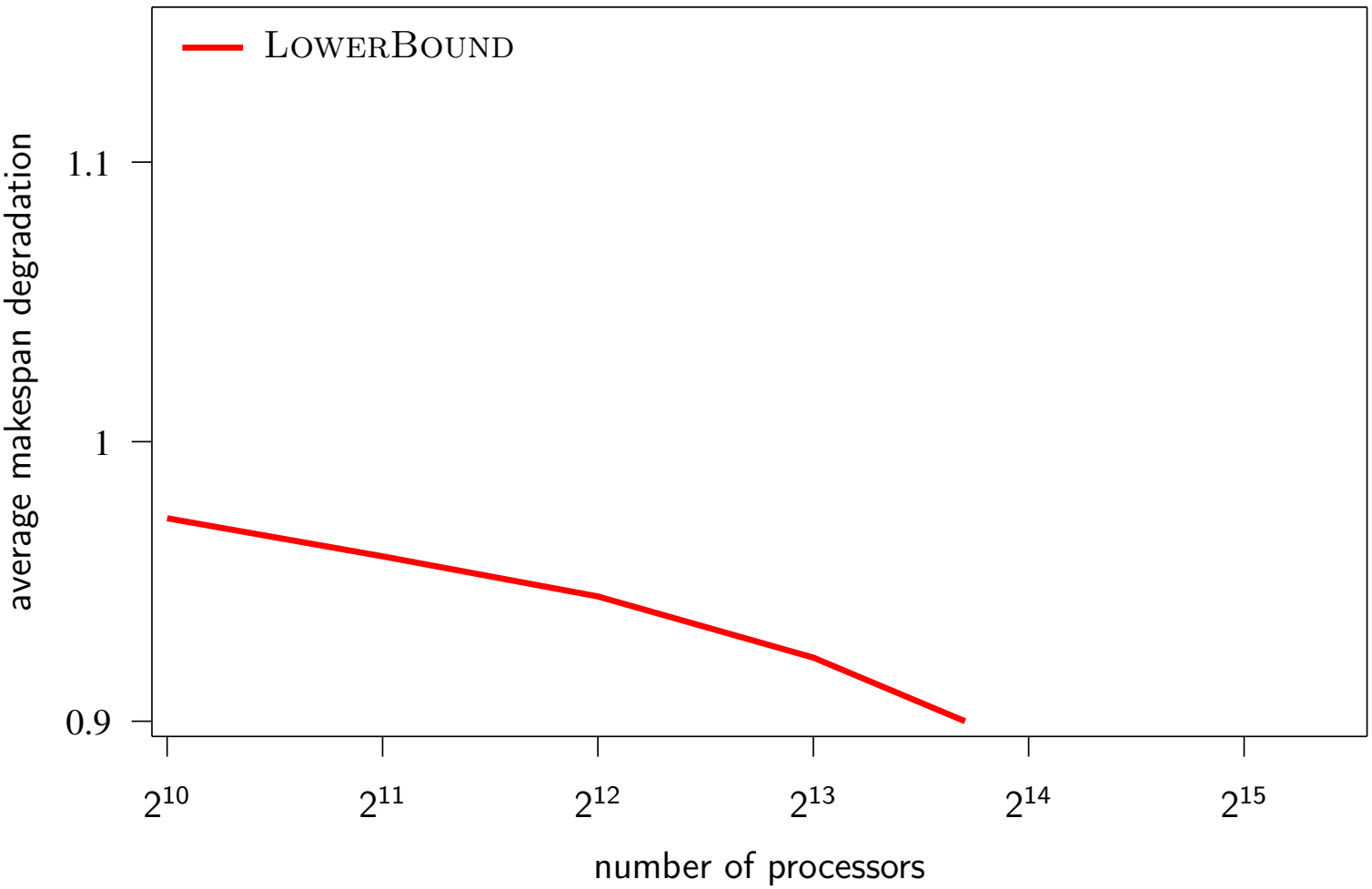
Exascale
MTBF = 1250 years

Parallel jobs under Weibull failures (1/2)



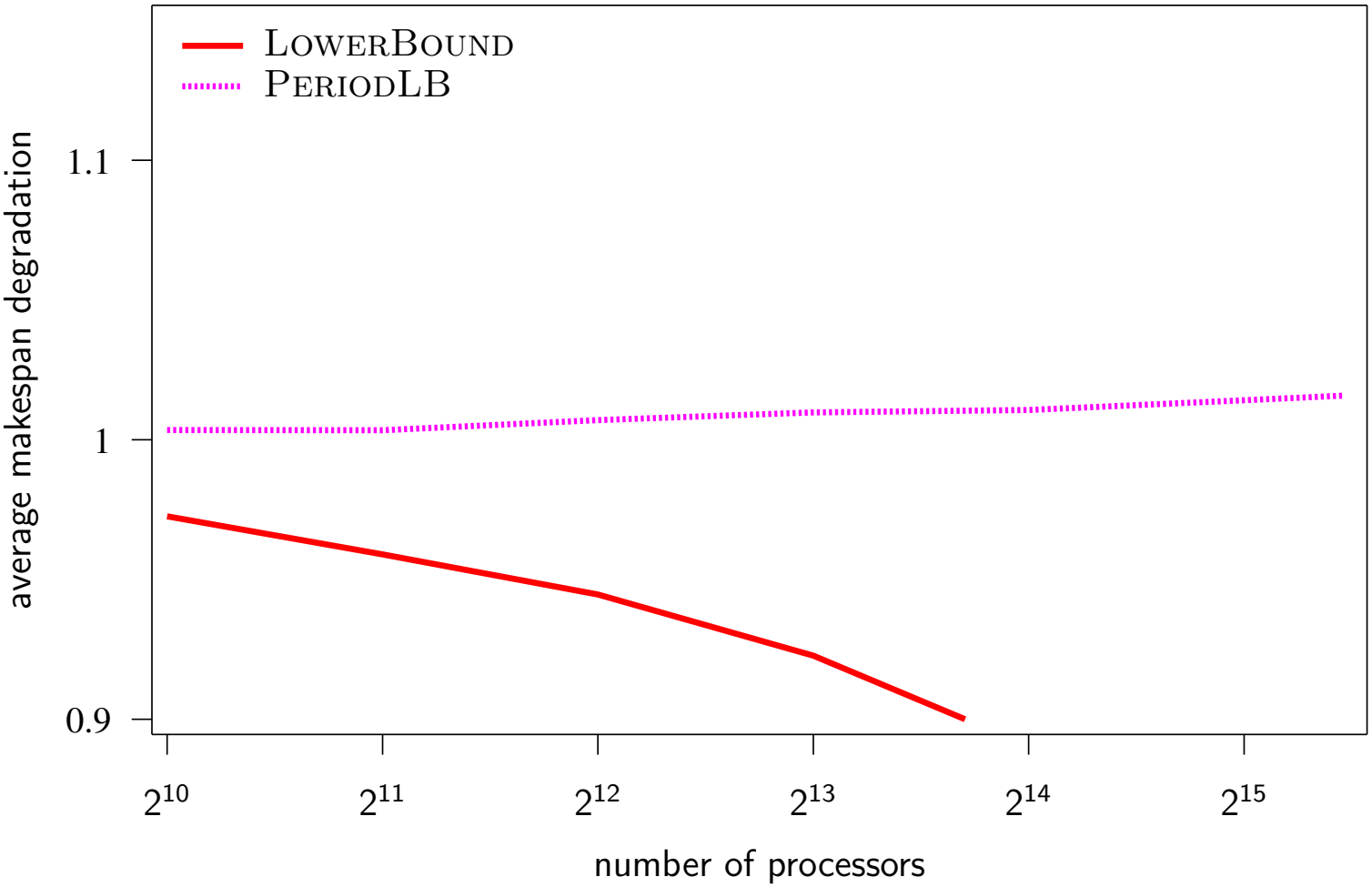
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)



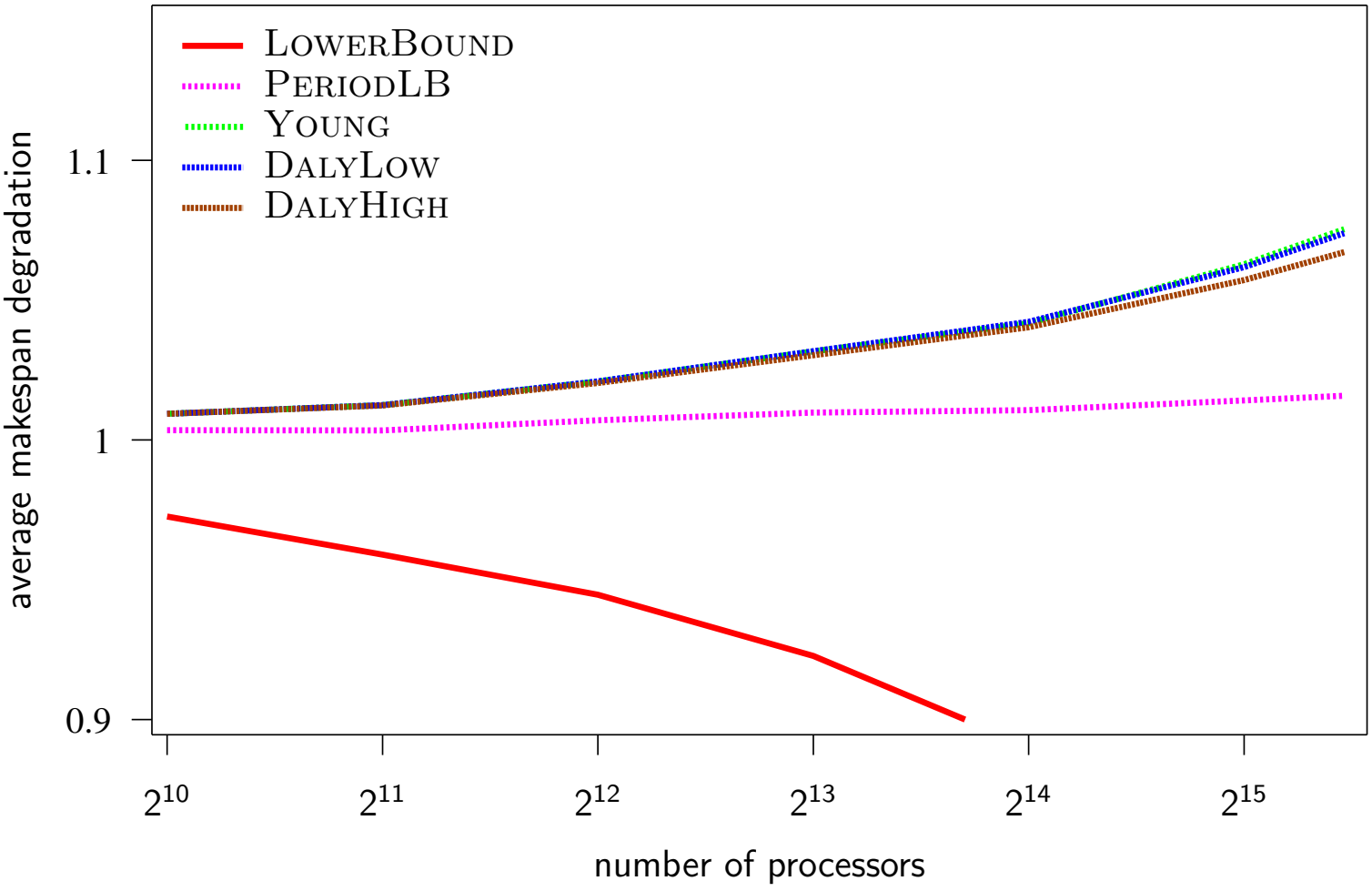
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)



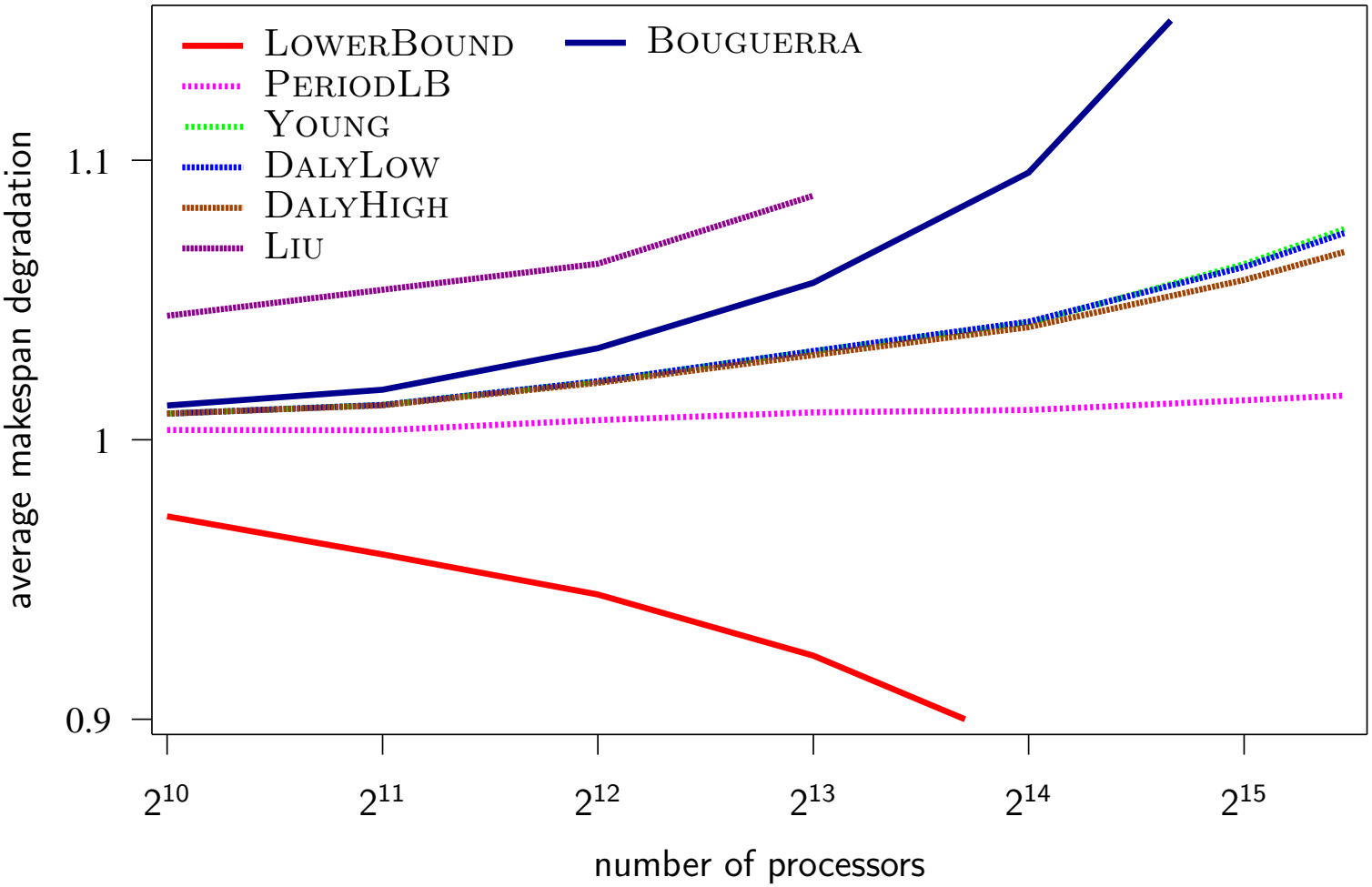
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)



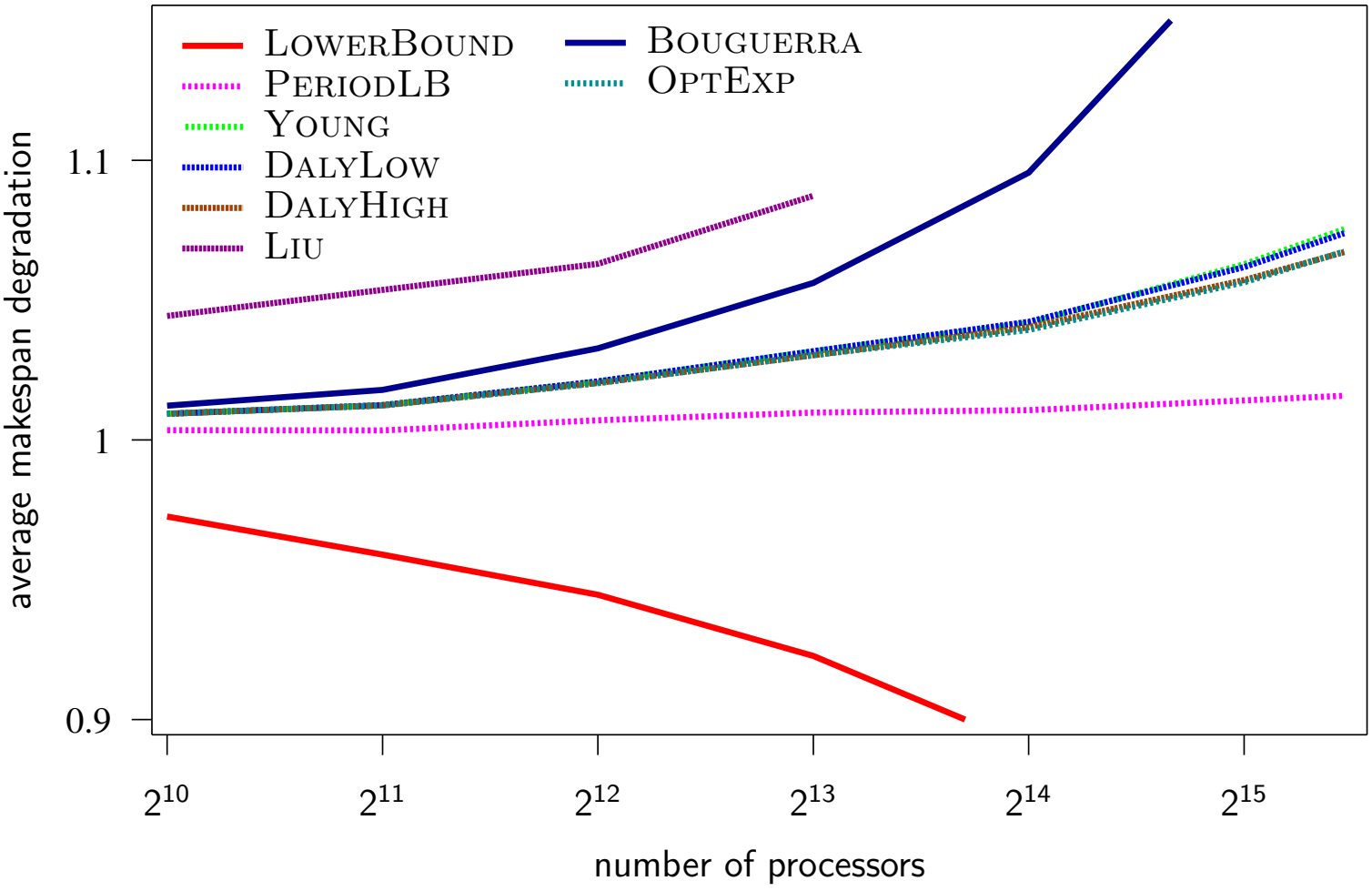
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)



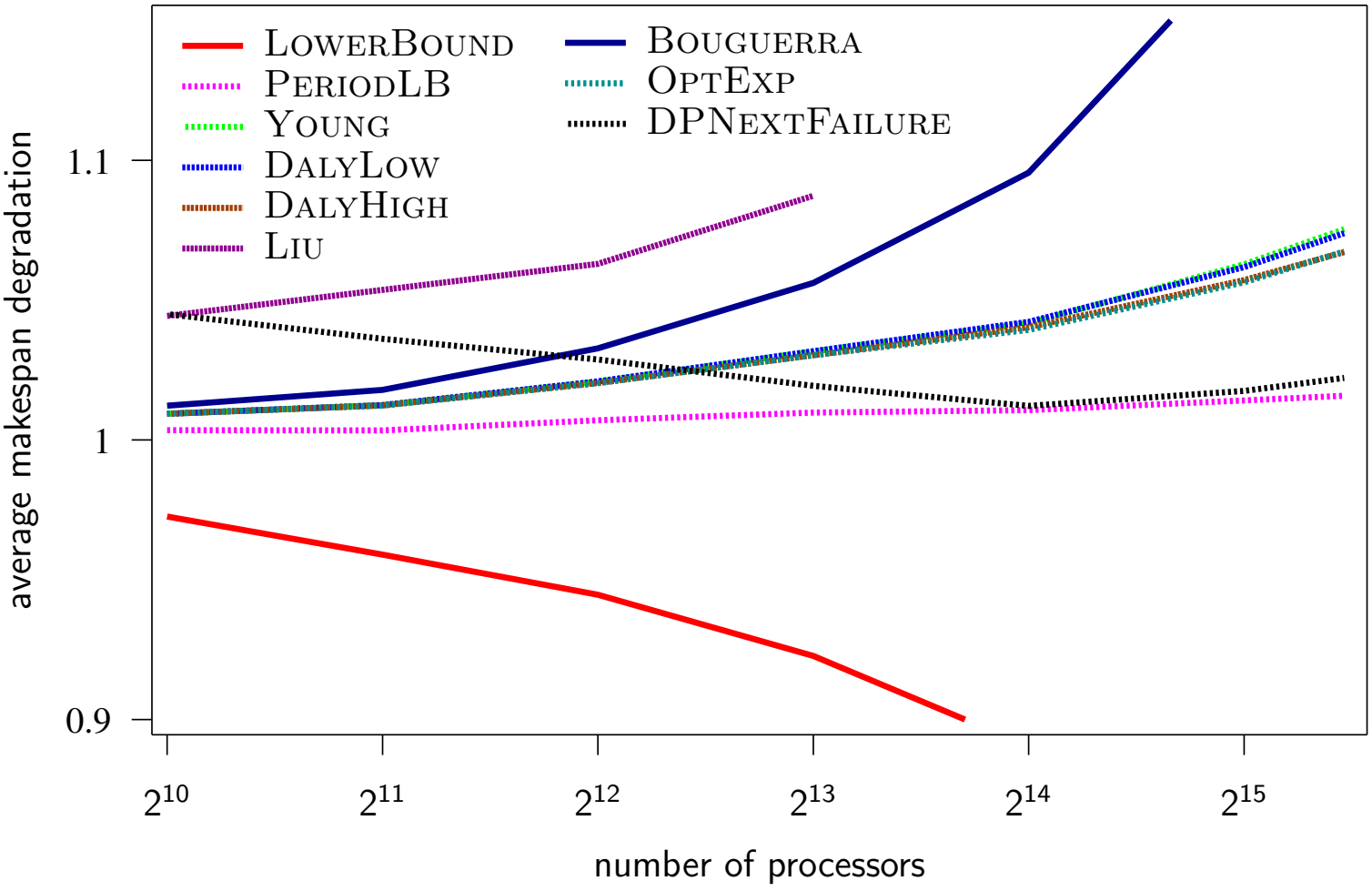
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)



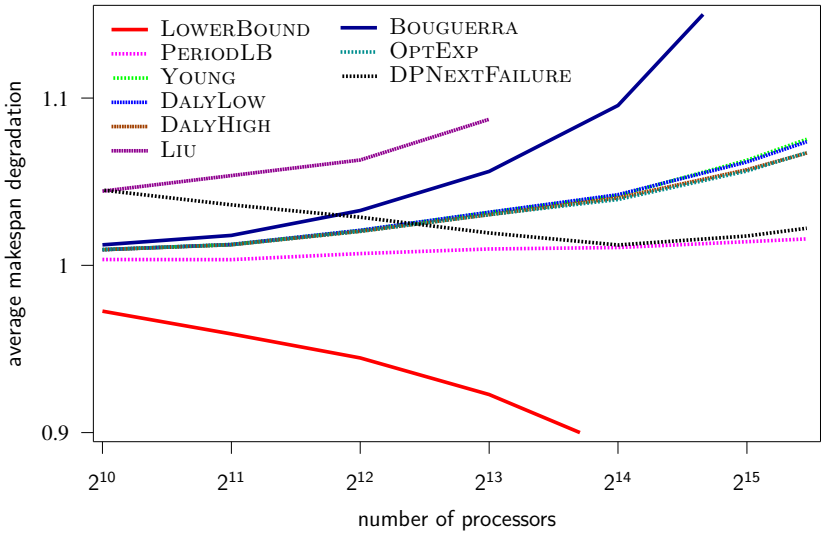
Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (1/2)

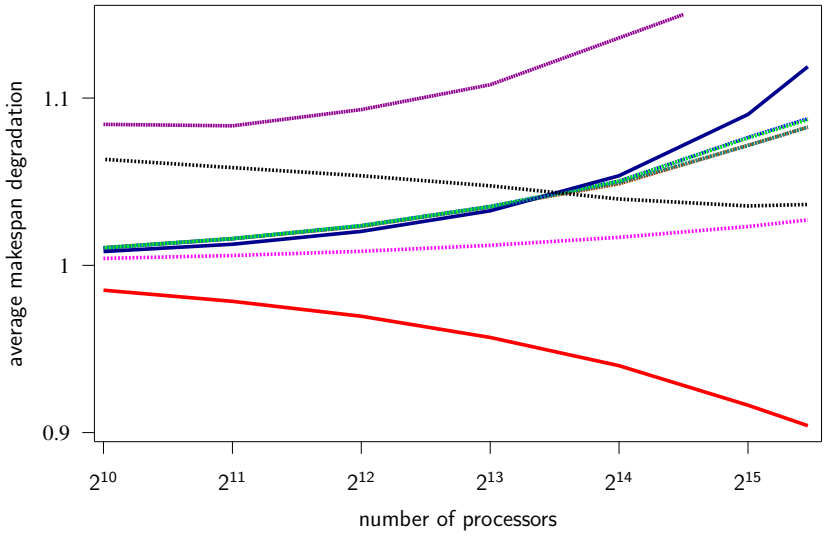


Petascale, MTBF = 125 years, k=0.70

Parallel jobs under Weibull failures (2/2)

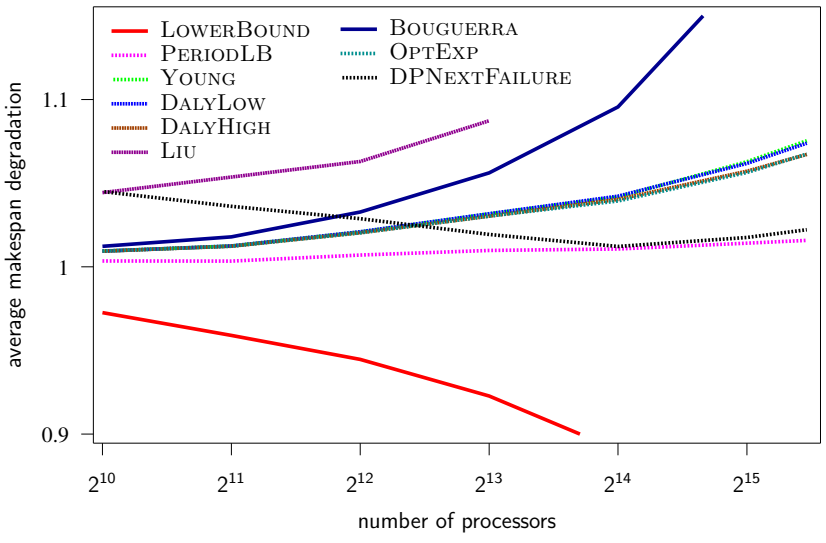


Petascale
MTBF = 125 years
k=0.70

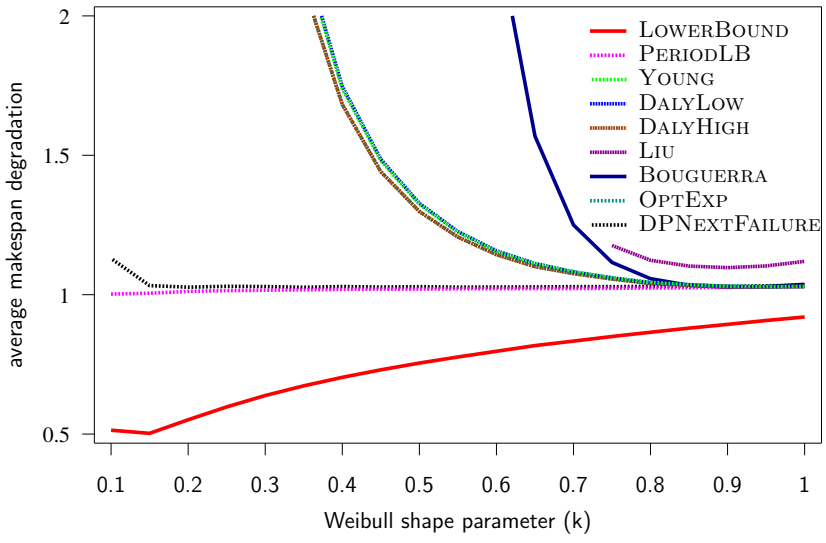


Petascale
MTBF = 500 years
k=0.70

Parallel jobs under Weibull failures (2/2)

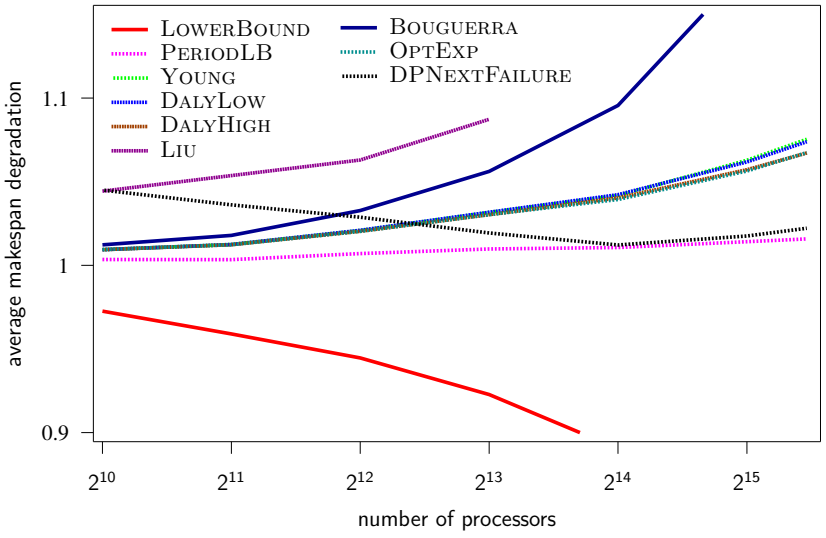


Petascale
MTBF = 125 years
k=0.70

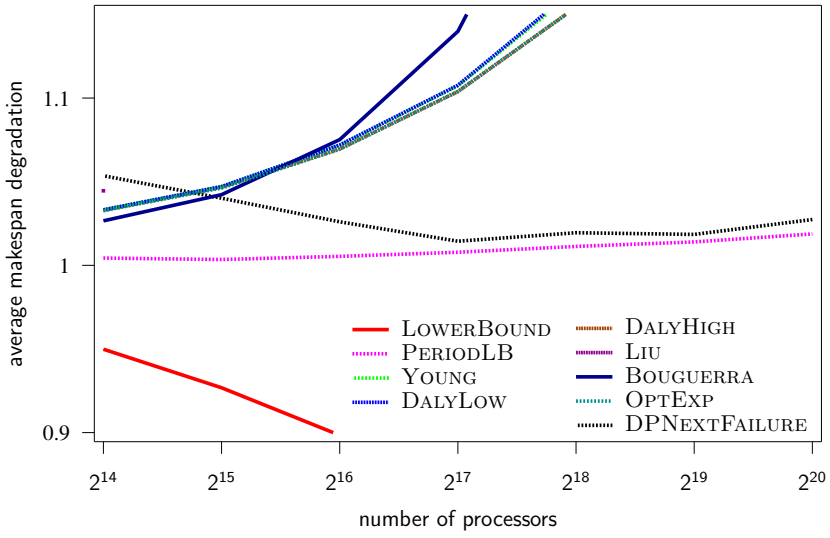


Petascale
MTBF = 125 years
45,208 processors

Parallel jobs under Weibull failures (2/2)



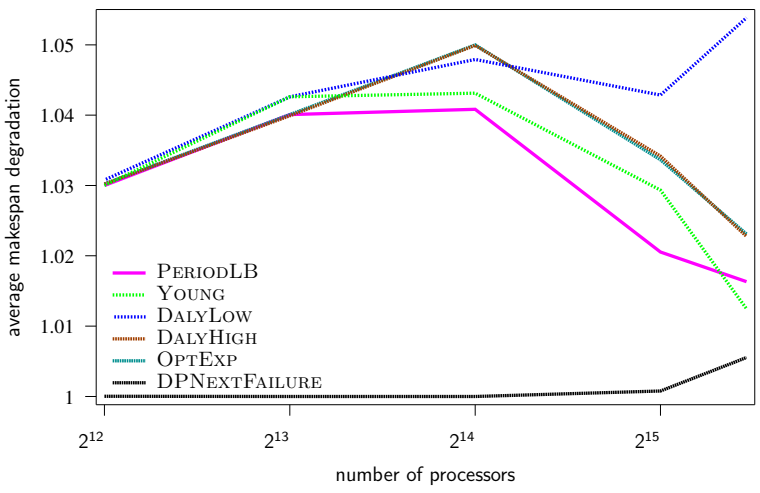
Petascale
MTBF = 125 years
k=0.70



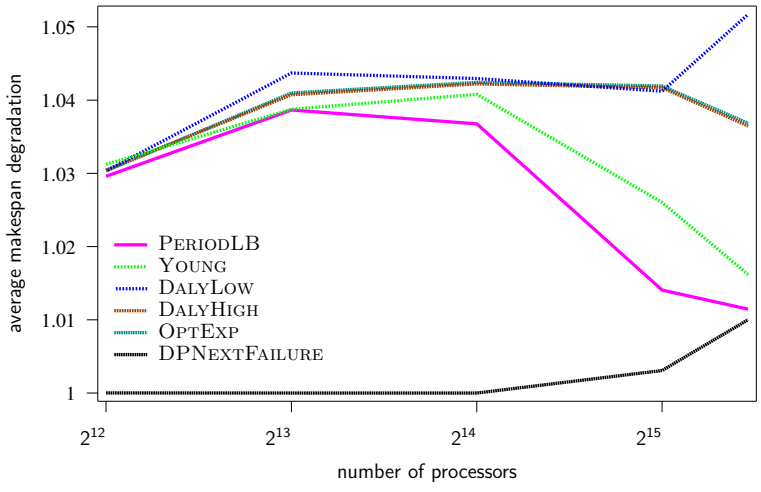
Exascale
MTBF = 1250 years
k=0.70

Plan

LANL trace set



Petascale / LANL Cluster 18



Petascale / LANL Cluster 19

Plan

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
  - Consider other actual traces
  - Replication and "group execution"
  - Multi-level checkpointing, uncoordinated 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