Checkpointing and fault prediction

Guillaume Aupy[1], Yves Robert[1,2],
Frédéric Vivien[1], & Dounia Zaidouni[1]

1 – ENS Lyon & INRIA
2 – University of Tennessee Knoxville

http://graal.ens-lyon.fr/~yrobert/slides/prediction.pdf

Joint Lab - Nov. 20, 2012

Overview

Context

- Failure-prone platform, small MTBF
- Very large number of processors ($N = 16K$ to $N = 1024K$)
- Fault predictor characterized by its recall and precision
- Resilience: combine coordinated & preventive checkpointing

Objective

- Design efficient checkpointing policies
- Compute expected waste
- Assess impact of predictions

Exascale platforms (courtesy Jack Dongarra)

Potential System Architecture with a cap of $200M and 20MW

Systems	2011 K computer	2019	Difference Today & 2019
System peak	**10.5 Pflop/s**	**1 Eflop/s**	**O(100)**
Power	**12.7 MW**	**~20 MW**	
System memory	1.6 PB	32 - 64 PB	O(10)
Node performance	128 GF	1,2 or 15TF	O(10) – O(100)
Node memory BW	64 GB/s	2 - 4TB/s	O(100)
Node concurrency	8	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	88,124	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	705,024	O(billion)	O(1,000)
MTTI	days	O(1 day)	- O(10)

Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores
- Failure-prone

MTBF – one node	1 year	10 years	120 years
MTBF – platform of 10^6 nodes	30sec	5mn	1h

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

Exascale platforms

- Hierarchica…
  - 10^5 or 10^6 node…
  - Each node equipped wi… 10^4 … 10^3 cores
- Failure-prone

MTBF – o… node	1 year	10… …ars	120 years
MTBF – …latform …of 10^6 nodes	30sec	5m…	1h

Mor… …odes = … …een …ilures)

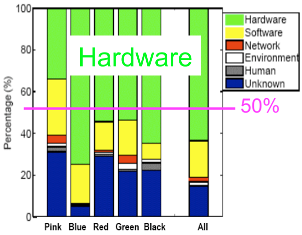
Exascale

$\neq$ Petascale $\times 1000$

Error sources (courtesy Franck Cappello)

Sources of failures

- Analysis of error and failure logs
- In 2005 (Ph. D. of CHARNG-DA LU) : "Software halts account for the most number of outages (59-84 percent), and take the shortest time to repair (0.6-1.5 hours). Hardware problems, albeit rarer, need 6.3-100.7 hours to solve."
- In 2007 (Garth Gibson, ICPP Keynote):



Relative frequency of root cause by system type.

- In 2008 (Oliner and J. Stearley, DSN Conf.):

Type	Raw Count	Raw %	Filtered Count	Filtered %
Hardware	174,586,516	98.04	1,999	18.78
Software	144,899	0.08	6,814	64.01
Indeterminate	3,350,044	1.88	1,832	17.21

Software errors: Applications, OS bug (kernel panic), communication libs, File system error and other.

Hardware errors, Disks, processors, memory, network

Conclusion: Both Hardware and Software failures have to be considered

A few definitions

- Many types of faults: software error, hardware malfunction, memory corruption
- Many possible behaviors: silent, transient, unrecoverable
- Restrict to faults that lead to application failures
- This includes all hardware faults, and some software ones
- Will use terms *fault* and *failure* interchangeably

Failure distributions: with several processors

- Processor (or node): any entity subject to failures
  $\Rightarrow$ approach agnostic to granularity

- If the MTBF is μ with one processor,
  what is its value with p processors?

- Well, it depends ☹

Failure distributions: with several processors

- Processor (or node): any entity subject to failures
  $\Rightarrow$ approach agnostic to granularity

- If the MTBF is μ with one processor,
  what is its value with p processors?

- Well, it depends ☹

With rejuvenation

- Rebooting all p processors after a failure
- Platform failure distribution
  $\Rightarrow$ minimum of p IID processor distributions
- With p distributions $Exp(\lambda)$:

$$\min\left(Exp(\lambda_1), Exp(\lambda_2)\right) = Exp(\lambda_1 + \lambda_2)$$

$$\mu = \frac{1}{\lambda} \Rightarrow \mu_p = \frac{\mu}{p}$$

- With p distributions $Weibull(k, \lambda)$:

$$\min_{1..p}\left(Weibull(k, \lambda)\right) = Weibull(k, p^{1/k}\lambda)$$

$$\mu = \frac{1}{\lambda}\Gamma(1 + \frac{1}{k}) \Rightarrow \mu_p = \frac{\mu}{p^{1/k}}$$

Without rejuvenation

- Rebooting only faulty processor
- Platform failure distribution
  $\Rightarrow$ superposition of p IID processor distributions
- Simple formula for arbitrary distributions:

$$\mu_p = \frac{\mu}{p}$$

  with p processors of MTBF μ
- Rejuvenation does not matter for Exponential
- Rejuvenation harmful for Weibull with $k < 1$

MTBF with p processors (1/2)

Theorem: $\mu_p = \frac{\mu}{p}$ for arbitrary distributions

With one processor:

- $n(F) =$ number of failures until time F is exceeded
- X_i iid random variables for inter-arrival times, with $\mathbb{E}(X_i) = \mu$
- $\sum_{i=1}^{n(F)-1} X_i \leq F \leq \sum_{i=1}^{n(F)} X_i$
- Wald's equation: $(\mathbb{E}(n(F)) - 1)\mu \leq F \leq \mathbb{E}(n(F))\,\mu$
- $\lim_{F \to +\infty} \frac{\mathbb{E}(n(F))}{F} = \frac{1}{\mu}$

MTBF with p processors (2/2)

Theorem: $\mu_p = \frac{\mu}{p}$ for arbitrary distributions

With p processors:

- $n(F)$ = number of platform failures until time F is exceeded
- $m_q(F)$ = number of those failures that strike processor q
- $n_q(F) = m_q(F) + 1$ = number of failures on processor q until time F is exceeded (except for processor with last-failure)
- Y_i iid random variables for platform inter-arrival times, with $\mathbb{E}(Y_i) = \mu_p$
- $\lim_{F \to +\infty} \frac{n(F)}{F} = \frac{1}{\mu_p}$ as above
- $\lim_{F \to +\infty} \frac{n(F)}{F} = \frac{p}{\mu}$ because $n(F) = \sum_{q=1}^{p} m_q(F)$
- Hence $\mu_p = \frac{\mu}{p}$

Values from the literature

- MTBF of one processor: between 10 and 125 years
- Shape parameters for Weibull: $k = 0.5$ or $k = 0.7$
- Failure trace archive from INRIA (`http://fta.inria.fr`)
- Computer Failure Data Repository from LANL (`http://institutes.lanl.gov/data/fdata`)

Outline

1. Young/Daly's approximation
2. Failure Prediction
   - Framework
   - Exact date predictions
   - Prediction windows
3. Experiments
4. Conclusion

Outline

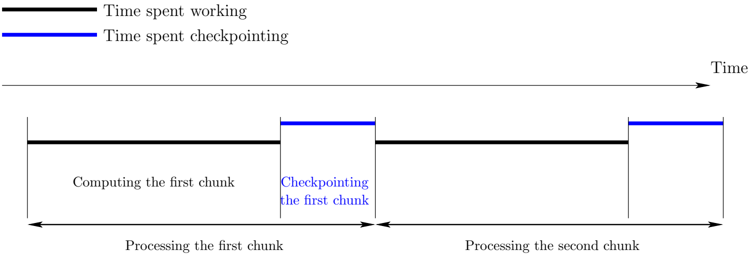
1. Young/Daly's approximation

2. Failure Prediction
   - Framework
   - Exact date predictions
   - Prediction windows

3. Experiments

4. Conclusion

Checkpointing cost



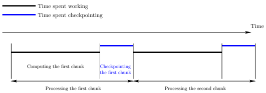
Blocking model: while a checkpoint is taken, no computation can be performed

Framework

- Periodic checkpointing policy of period T
- Independent and identically distributed failures
- Applies to a single processor with MTBF $\mu = \mu_{ind}$
- Applies to a platform with p processors with MTBF $\mu = \frac{\mu_{ind}}{p}$
  - coordinated checkpointing
  - tightly-coupled application
  - progress $\Leftrightarrow$ all processors available

Waste: fraction of time not spent for useful computations

Waste in fault-free execution



- $\text{TIME}_{\text{base}}$: application base time
- TIME_{FF}: with periodic checkpoints but failure-free

$$\text{TIME}_{\text{FF}} = \text{TIME}_{\text{base}} + \#checkpoints \times C$$

$$\#checkpoints = \left\lceil \frac{\text{TIME}_{\text{base}}}{T - C} \right\rceil \approx \frac{Time[base]}{T - C} \text{ (valid for large jobs)}$$

$$\text{TIME}_{\text{FF}} = \text{TIME}_{\text{base}} \frac{T}{T - C} \text{ and } \text{WASTE}[FF] = \frac{\text{TIME}_{\text{FF}} - \text{TIME}_{\text{base}}}{\text{TIME}_{\text{FF}}}$$

$$\text{WASTE}[FF] = \frac{C}{T}$$

Waste due to failures

- $\textsc{Time}_{\text{base}}$: application base time
- $\textsc{Time}_{\text{FF}}$: with periodic checkpoints but failure-free
- $\textsc{Time}_{\text{final}}$: expectation of time with failures

$$\textsc{Time}_{\text{final}} = \textsc{Time}_{\text{FF}} + N_{faults} \times T_{\text{lost}}$$

N_{faults} number of failures during execution
T_{lost}: average time lost par failures

$$N_{faults} = \frac{\textsc{Time}_{\text{final}}}{\mu}$$

T_{lost}?

Waste due to failures

- $\textsc{Time}_{\text{base}}$: application base time
- $\textsc{Time}_{\text{FF}}$: with periodic checkpoints but failure-free
- $\textsc{Time}_{\text{final}}$: expectation of time with failures

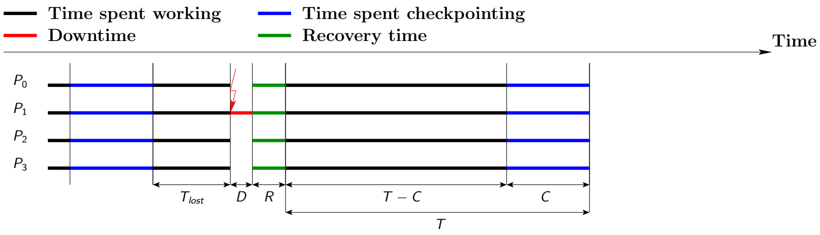
$$\textsc{Time}_{\text{final}} = \textsc{Time}_{\text{FF}} + N_{faults} \times T_{\text{lost}}$$

N_{faults} number of failures during execution
T_{lost}: average time lost par failures

$$N_{faults} = \frac{\textsc{Time}_{\text{final}}}{\mu}$$

T_{lost}?

Computing T_{lost}



$$T_{\text{lost}} = D + R + \frac{T}{2}$$

$\Rightarrow$ Instants when periods begin and failures strike are independent

$\Rightarrow$ Valid for all distribution laws, regardless of their particular shape

Waste due to failures

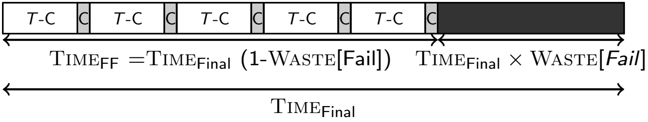
$$\textsc{Time}_{\mathsf{final}} = \textsc{Time}_{\mathsf{FF}} + N_{faults} \times T_{\mathsf{lost}}$$

$$\textsc{Time}_{\mathsf{final}} = \textsc{Time}_{\mathsf{FF}} + \frac{\textsc{Time}_{\mathsf{final}}}{\mu} \times \left(D + R + \frac{T}{2}\right)$$

$$\textsc{Waste}[fail] = \frac{\textsc{Time}_{\mathsf{final}} - \textsc{Time}_{\mathsf{FF}}}{\textsc{Time}_{\mathsf{final}}}$$

$$\textsc{Waste}[fail] = \frac{1}{\mu}\left(D + R + \frac{T}{2}\right)$$

Total waste



$$\text{WASTE} = \frac{\text{TIME}_{\text{final}} - \text{TIME}_{\text{base}}}{\text{TIME}_{\text{final}}}$$

$$(1 - \text{WASTE}[fail])(1 - \text{WASTE}[FF])\text{TIME}_{\text{final}} = Time[base]$$

$$1 - \text{WASTE} = (1 - \text{WASTE}[FF])(1 - \text{WASTE}[fail])$$

$$\text{WASTE} = \frac{C}{T} + \left(1 - \frac{C}{T}\right)\frac{1}{\mu}\left(D + R + \frac{T}{2}\right)$$

Waste minimization

$$\textsc{Waste} = \frac{C}{T} + \left(1 - \frac{C}{T}\right)\frac{1}{\mu}\left(D + R + \frac{T}{2}\right)$$

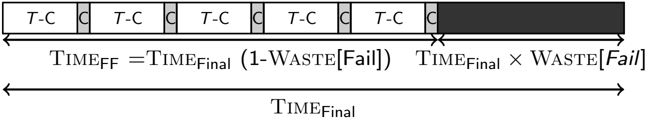
$$\textsc{Waste} = \frac{u}{T} + v + wT$$

$$u = C\left(1 - \frac{D+R}{\mu}\right) \qquad v = \frac{D + R - C/2}{\mu} \qquad w = \frac{1}{2\mu}$$

$\textsc{Waste}$ minimized for $T = \sqrt{\frac{u}{w}}$

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

Comparison with Young/Daly



$$\left(1 - \text{WASTE}[fail]\right)\text{TIME}_{\text{final}} = \text{TIME}_{\text{FF}}$$
$$\Rightarrow T = \sqrt{2(\mu - (D+R))C}$$

Daly: $\text{TIME}_{\text{final}} = \left(1 + \text{WASTE}[fail]\right)\text{TIME}_{\text{FF}}$
$$\Rightarrow T = \sqrt{2(\mu + (D+R))C} + C$$

Young: $\text{TIME}_{\text{final}} = \left(1 + \text{WASTE}[fail]\right)\text{TIME}_{\text{FF}}$ and $D = R = 0$
$$\Rightarrow T = \sqrt{2\mu C} + C$$

Validity of the approach (1/3)

Technicalities

- $\mathbb{E}\left(N_{faults}\right) = \frac{\text{TIME}_{\text{final}}}{\mu}$ and $\mathbb{E}\left(T_{\text{lost}}\right) = \frac{T}{2}$
  but expectation of product is not product of expectations (not independent RVs here)
- Enforce $C \leq T$ to get $\text{WASTE}[FF] \leq 1$
- Enforce $D + R \leq \mu$ and bound T to get $\text{WASTE}[fail] \leq 1$
  but $\mu = \frac{\mu_{ind}}{p}$ too small for large p, regardless of μ_{ind}

Validity of the approach (2/3)

Several failures within same period?

- WASTE[fail] accurate only when two or more faults do not take place within same period
- Cap period: $T \leq \gamma\mu$, where γ is some tuning parameter
  - Poisson process of parameter $\theta = \frac{T}{\mu}$
  - Probability of having $k \geq 0$ failures : $P(X = k) = \frac{\theta^k}{k!} e^{-\theta}$
  - Probability of having two or more failures: $\pi = P(X \geq 2) = 1 - (P(X = 0) + P(X = 1)) = 1 - (1 + \theta)e^{-\theta}$
  - $\gamma = 0.27 \Rightarrow \pi \leq 0.03$
    $\Rightarrow$ overlapping faults for only 3% of checkpointing segments

Validity of the approach (3/3)

- Enforce $T \leq \gamma\mu$, $C \leq \gamma\mu$, and $D + R \leq \gamma\mu$
- Optimal period $\sqrt{2(\mu - (D + R))C}$ may not belong to admissible interval $[C, \gamma\mu]$
- Waste is then minimized for one of the bounds of this admissible interval (by convexity)

Wrap up

- Capping periods, and enforcing a lower bound on MTBF
  $\Rightarrow$ mandatory for mathematical rigor ☹
- Not needed for practical purposes ☺
  - actual job execution uses optimal value
  - account for multiple faults by re-executing work until success
- Approach surprisingly robust ☺

Outline

1. Young/Daly's approximation
2. Failure Prediction
   - Framework
   - Exact date predictions
   - Prediction windows
3. Experiments
4. Conclusion

Outline

Framework

Predictor

- Exact prediction dates (at least C seconds in advance)
- Recall r: fraction of faults that are predicted
- Precision p: fraction of fault predictions that are correct

Events

- *true positive*: predicted faults
- *false positive*: fault predictions that did not materialize as actual faults
- *false negative*: non-predicted faults

$$r = \frac{True_P}{True_P + False_N} \quad \text{and} \quad p = \frac{True_P}{True_P + False_P}$$

Fault rates

- μ: mean time between failures (MTBF)
- μ_P mean time between predicted events (both true positive and false positive)
- μ_{NP} mean time between unpredicted faults (false negative).
- μ_e: mean time between events (including all three event types)

$$\frac{(1-r)}{\mu} = \frac{1}{\mu_{NP}}$$

$$\frac{r}{\mu} = \frac{p}{\mu_P}$$

$$\frac{1}{\mu_e} = \frac{1}{\mu_P} + \frac{1}{\mu_{NP}}$$

Hypotheses

Regular (coordinated) checkpoints

- Checkpoint cost: C
- Downtime: D
- Recovery cost after failure: R

Two scenarios

① Exact date predictions

② Window-based predictions

Lead times

- Predictions must be provided at least C seconds in advance

Outline

Algorithm

1. While no fault prediction is available:
   - checkpoints taken periodically with period T
2. When a fault is predicted at time t:
   - take a checkpoint ALAP (completion right at time t)
   - after the checkpoint, complete the execution of the period

Computing the waste

1. **Fault-free execution:** $\textsc{Waste}[FF] = \frac{C}{T}$
2. **Unpredicted faults:** $\frac{1}{\mu_{NP}}\left[D + R + \frac{T}{2}\right]$
3. **Predictions:** $\frac{1}{\mu_P}\left[p(C + D + R) + (1 - p)C\right]$

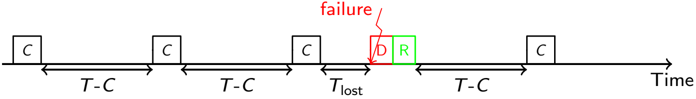
$$\textsc{Waste}[fail] = \frac{1}{\mu}\left[(1 - r)\frac{T}{2} + D + R + \frac{r}{p}C\right]$$

$$T_{opt} \approx \sqrt{\frac{2\mu C}{1 - r}}$$

Algorithm (v2)

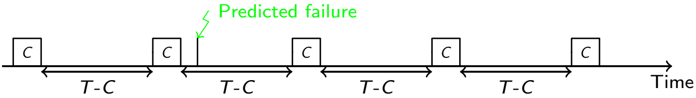
- While no fault prediction is available:
  $\Rightarrow$ Periodic checkpointing with period T
- When a fault is predicted:
  $\Rightarrow$ Decide whether to take prediction into account or not
  - With probability $1 - q$: ignore prediction
  - With probability q: trust prediction
    - If enough time before prediction date, checkpoint ALAP
    - Otherwise, start new period

Algorithm (v2)

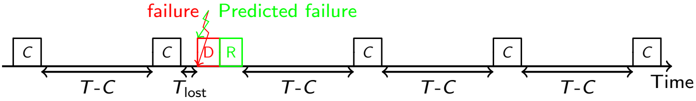


(a) Unpredicted fault

Algorithm (v2)

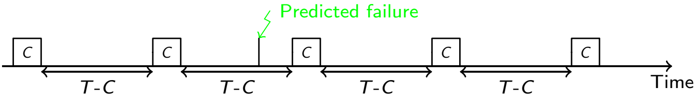


(b) Prediction cannot be taken into account - no actual fault

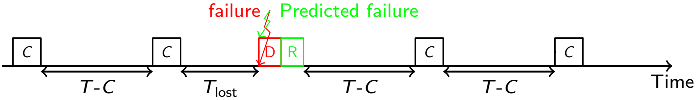


(c) Prediction cannot be taken into account - with actual fault

Algorithm (v2)

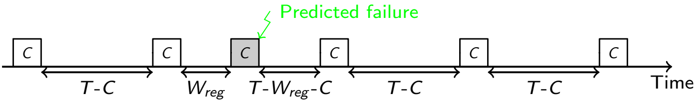


(d) Prediction not taken into account by choice - no actual fault

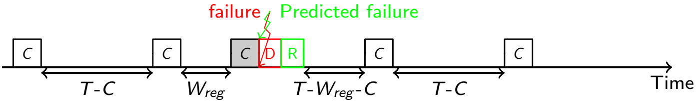


(e) Prediction not taken into account by choice - with actual fault

Algorithm (v2)



(f) Prediction taken into account - no actual fault



(g) Prediction taken into account - with actual fault

Waste minimization

- $\textsc{Waste}(q)$ minimized either for $q = 0$ or for $q = 1$
  - either never trust the predictor ...
  - ... or always trust it!

Optimal period:

$$T_{opt} \approx \sqrt{\frac{2\mu C}{1 - rq}}$$

Capping: $T_{opt} \leq \gamma \mu_e$

Outline

Strategies

Hypotheses

- Predictor gives a time window for each prediction
- Predictor generates predictions at least C seconds before beginning of time window

Description of strategies

Two modes for scheduling algorithm:

- Regular: Periodic checkpointing with period T_R
- Proactive (Several variants):
  - INSTANT: Ignore time-window ($\Leftrightarrow$ exact date)
  - NOCKPTI: No checkpoint during time window
  - WITHCKPTI: Several checkpoints during time window

Algorithm

Algorithm 1: WithCkptI.

if *fault happens* **then**
  After downtime, execute recovery;
  Enter *regular* mode;

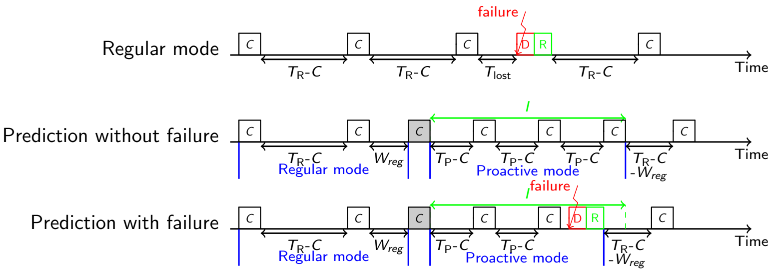
if *in proactive mode for a time greater than or equal to I* **then**
  Switch to *regular* mode

if *Prediction made with interval* $[t, t+I]$ **and** *prediction taken into account* **then**
  Let t_C be the date of the last checkpoint under *regular* mode to start no later than $t - C$;
  if $t_C + C < t - C$ **then** (enough time for an extra checkpoint)
    Take a checkpoint starting at time $t - C$
  else (no time for the extra checkpoint)
    Work in the time interval $[t_C + C, t]$
  $W_{reg} \leftarrow \max(0, t - C - (t_C + C))$;
  Switch to *proactive* mode at time t;

while *in regular mode and no predictions are made and no faults happen* **do**
  Work for a time T_R-W_{reg}-C and then checkpoint;
  $W_{reg} \leftarrow 0$;

while *in proactive mode and no faults happen* **do**
  Work for a time T_P-C and then checkpoint;

Outline of Algorithm



Outline of strategy WITHCKPTI

Outline

1. Young/Daly's approximation
2. Failure Prediction
   - Framework
   - Exact date predictions
   - Prediction windows
3. Experiments
4. Conclusion

Prediction and failure distributions

- Failure traces (predicted and non predicted failures):
  - Exponential failure distribution
  - Weibull distribution law with shape parameter 0.5 and 0.7
- False predictions:
  - Same distribution as failure trace
  - Uniform distribution

Number of processors	D	C, R	μ_{ind}	W
$16,384$ to $524,288$	60 s	600 s	125 y	400 y

Simulation parameters

Job execution times for a Weibull distribution ($k = 0.7$)

$I = 300$	Execution time (hours) ($p = 0.82, r = 0.85$)		Execution time (hours) ($p = 0.4, r = 0.7$)	
	2^{16} procs	2^{19} procs	2^{16} procs	2^{19} procs
YOUNG	81.3	30.1	81.2	30.1
EXACTPREDICTION	65.9 (19%)	15.9 (47%)	69.7 (14%)	19.3 (36%)
NOCKPTI	66.5 (18%)	16.9 (44%)	70.3 (13%)	20.5 (32%)
INSTANT	66.5 (18%)	17.0 (44%)	70.3 (13%)	20.7 (31%)

$I = 3,000$	Execution time (hours) ($p = 0.82, r = 0.85$)		Execution time (hours) ($p = 0.4, r = 0.7$)	
	2^{16} procs	2^{19} procs	2^{16} procs	2^{19} procs
YOUNG	81.2	30.1	81.2	30.1
EXACTPREDICTION	66.0 (19%)	15.9 (47%)	69.8 (14%)	19.3 (36%)
NOCKPTI	71.1 (12%)	24.6 (18%)	75.2 (7.3%)	28.9 (4.0%)
WITHCKPTI	70.0 (14%)	22.6 (25%)	75.4 (7.1%)	27.2 (9.7%)
INSTANT	71.2 (12%)	24.2 (20%)	75.0 (7.6%)	28.3 (6.0%)

Comparing job execution times for a Weibull distribution ($k = 0.7$), and reporting gain when comparing to YOUNG.

Job execution times for a Weibull distribution ($k = 0.7$)

$I = 300$	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{16} procs	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{19} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{16} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{19} procs
YOUNG	81.3	30.1	81.2	30.1
EXACTPREDICTION	65.9 (19%)	15.9 (47%)	69.7 (14%)	19.3 (36%)
NOCKPTI	66.5 (18%)	16.9 (44%)	70.3 (13%)	20.5 (32%)
INSTANT	66.5 (18%)	17.0 (44%)	70.3 (13%)	20.7 (31%)

$I = 3,000$	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{16} procs	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{19} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{16} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{19} procs
YOUNG	81.2	30.1	81.2	30.1
EXACTPREDICTION	66.0 (19%)	15.9 (47%)	69.8 (14%)	19.3 (36%)
NOCKPTI	71.1 (12%)	24.6 (18%)	75.2 (7.3%)	28.9 (4.0%)
WITHCKPTI	70.0 (14%)	22.6 (25%)	75.4 (7.1%)	27.2 (9.7%)
INSTANT	71.2 (12%)	24.2 (20%)	75.0 (7.6%)	28.3 (6.0%)

Comparing job execution times for a Weibull distribution ($k = 0.7$), and reporting gain when comparing to YOUNG.

Job execution times for a Weibull distribution ($k = 0.5$)

$I = 300$	Execution time (hours) ($p = 0.82, r = 0.85$)		Execution time (hours) ($p = 0.4, r = 0.7$)	
	2^{16} procs	2^{19} procs	2^{16} procs	2^{19} procs
Young	125.4	171.8	125.5	171.7
ExactPrediction	75.8 (40%)	39.4 (77%)	82.9 (34%)	51.8(70%)
NoCkptI	77.3 (38%)	44.8 (74%)	84.6 (33%)	58.2 (66%)
Instant	77.4 (38%)	45.1 (74%)	84.7 (33%)	59.1 (66%)

$I = 3,000$	Execution time (hours) ($p = 0.82, r = 0.85$)		Execution time (hours) ($p = 0.4, r = 0.7$)	
	2^{16} procs	2^{19} procs	2^{16} procs	2^{19} procs
Young	125.4	171.9	125.4	172.0
ExactPrediction	76.1 (39%)	39.4 (77%)	83.0 (34%)	51.7 (70%)
NoCkptI	90.0 (28%)	71.8 (58%)	98.3 (22%)	84.5 (51%)
WithCkptI	87.8 (30%)	66.6 (61%)	98.0 (22%)	82.2 (52%)
Instant	89.8 (28%)	70.9 (59%)	98.2 (22%)	83.2 (52%)

Comparing job execution times for a Weibull distribution ($k = 0.5$), and reporting gain when comparing to Young.

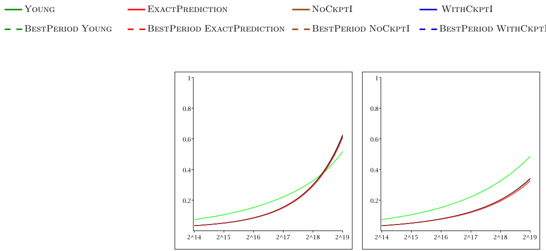
Job execution times for a Weibull distribution ($k = 0.5$)

$I = 300$	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{16} procs	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{19} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{16} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{19} procs
Young	125.4	171.8	125.5	171.7
ExactPrediction	75.8 (40%)	39.4 (77%)	82.9 (34%)	51.8(70%)
NoCkptI	77.3 (38%)	44.8 (74%)	84.6 (33%)	58.2 (66%)
Instant	77.4 (38%)	45.1 (74%)	84.7 (33%)	59.1 (66%)

$I = 3,000$	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{16} procs	Execution time (hours) ($p = 0.82, r = 0.85$) 2^{19} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{16} procs	Execution time (hours) ($p = 0.4, r = 0.7$) 2^{19} procs
Young	125.4	171.9	125.4	172.0
ExactPrediction	76.1 (39%)	39.4 (77%)	83.0 (34%)	51.7 (70%)
NoCkptI	90.0 (28%)	71.8 (58%)	98.3 (22%)	84.5 (51%)
WithCkptI	87.8 (30%)	66.6 (61%)	98.0 (22%)	82.2 (52%)
Instant	89.8 (28%)	70.9 (59%)	98.2 (22%)	83.2 (52%)

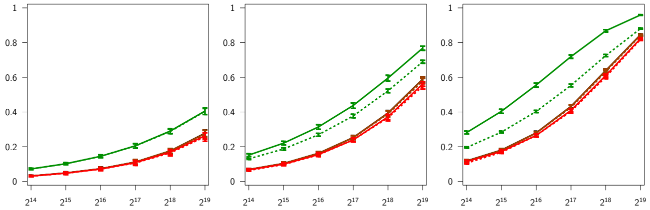
Comparing job execution times for a Weibull distribution ($k = 0.5$), and reporting gain when comparing to Young.

Waste with $p = 0.82$, $r = 0.85$ and $I = 300s$



(a) Capped period

(b) Uncapped period

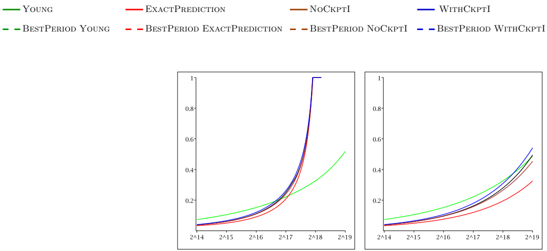


(c) Exponential

(d) Weibull $k = 0.7$

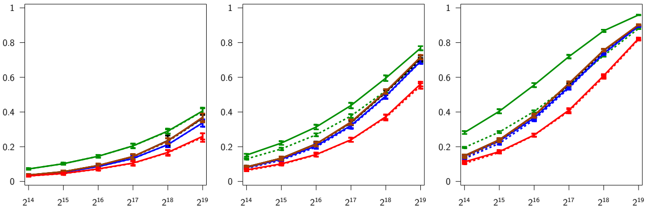
(e) Weibull $k = 0.5$

Waste with $p = 0.82$, $r = 0.85$ and $I = 3000$s



(a) Capped period

(b) Uncapped period

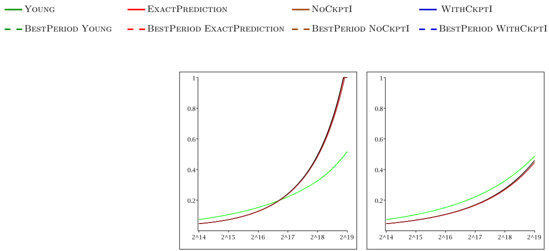


(c) Exponential

(d) Weibull $k = 0.7$

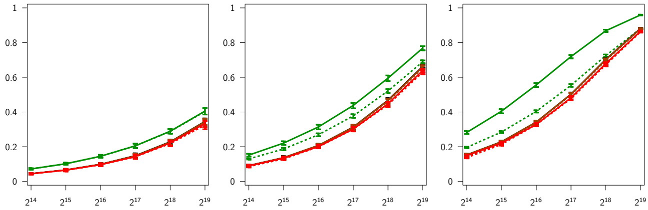
(e) Weibull $k = 0.5$

Waste with $p = 0.4$, $r = 0.7$ and $I = 300s$



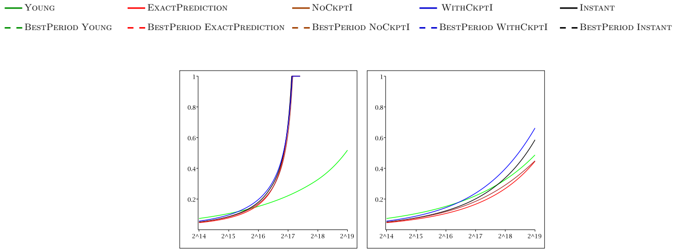
(a) Capped period

(b) Uncapped period

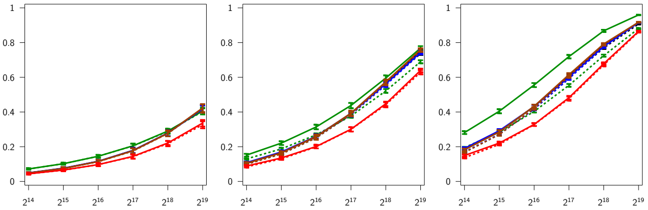


(c) Exponential (d) Weibull $k = 0.7$ (e) Weibull $k = 0.5$

Waste with $p = 0.4$, $r = 0.7$ and $I = 3000$s

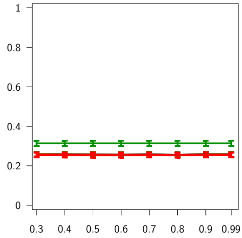


(a) Capped period (b) Uncapped period

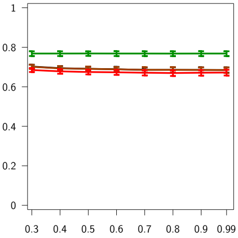


(c) Exponential (d) Weibull $k = 0.7$ (e) Weibull $k = 0.5$

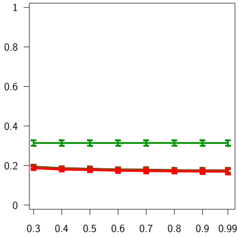
Impact of precision for a fixed recall (Weibull $k = 0.7$)



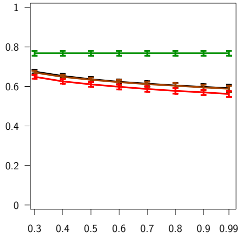
(a) $r = 0.4,\ N = 2^{16}$



(b) $r = 0.4,\ N = 2^{19}$

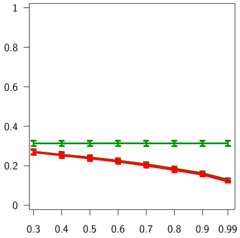


(c) $r = 0.8,\ N = 2^{16}$

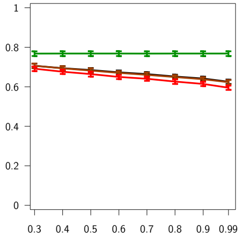


(d) $r = 0.8,\ N = 2^{19}$

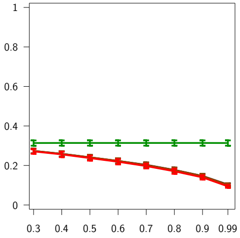
Impact of recall for a fixed precision (Weibull $k = 0.7$)



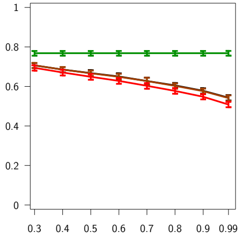
(a) $p = 0.4,\ N = 2^{16}$



(b) $p = 0.4,\ N = 2^{19}$



(c) $p = 0.8,\ N = 2^{16}$



(d) $p = 0.8,\ N = 2^{19}$

Outline

1. Young/Daly's approximation
2. Failure Prediction
   - Framework
   - Exact date predictions
   - Prediction windows
3. Experiments
4. Conclusion

Conclusion and perspectives

- Model is quite accurate
- Unified formula for optimal checkpointing period: $\sqrt{\frac{2\mu C}{1 - rq}}$
- Simulations fully validate the model:
  - Significant gain even for mid-range recall and precision
  - Best period always very close to one given by unified formula
- Recall has more impact on waste than precision

- Future work
  Use trace-based failure and prediction logs from current large-scale supercomputers