Partial Detectors Versus Replication To Cope With Silent Errors




Alix Tremodeux[1], Anne Benoit[1,2], Thomas Herault[3,4], Yves Robert[1]

1 ENS Lyon & Inria; 2 IUF & IDEaS, Georgia Tech;
3 Inria Bordeaux; 4 Univ. Tennessee Knoxville

19th Scheduling for large-scale systems workshop – March 17, 2026

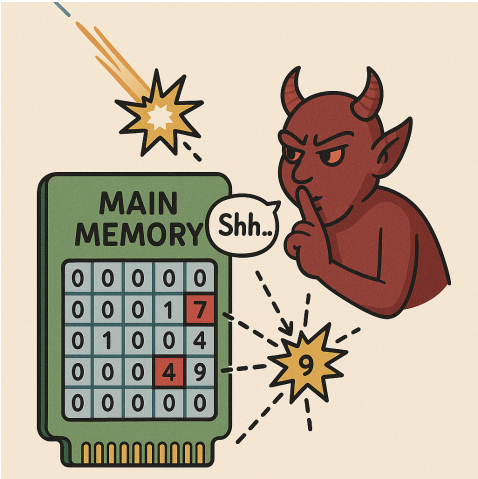
Outline

Introduction

Iterative algorithm executing on a large-scale platform

Silent errors may strike

Minimize expected cost per iteration



The Monster Under the Bed

Similarities:

- Hard to observe directly
- Fear may be overexaggerated?
- Hard to disprove

Probabilists: argument of scale (in time and/or in ressource)

"You have been promising ~~cold-fusion~~ hell for decades"

Very few studies in HPC/Supercomputers

Practical Studies on SDC in Supercomputers / HPC community

References

[1] Bautista-Gomez, Zyulkyarov, Unsal, & McIntosh-Smith (2016). Unprotected computing: A large-scale study of DRAM raw error rate on a supercomputer. In SC'16 pp. 645-655

Production logs from HPC systems
Deactivated ECC protection
1,000s of DRAM faults during the study, many simultaneous double-flips

[2] Schroeder, Pinheiro, & Weber (2009). DRAM errors in the wild: a large-scale field study. ACM SIGMETRICS Performance Evaluation Review, 37(1), 193-204.

100,000s DIMMs, Google Datacenters,
Months to Years of production
8% of DIMMs experience error(s) every year

AI to the Rescue!

Prompt

List published papers that studied the occurence of SDCs in HPC centers. I'm especially interested in papers that feature long term studies of applications that may have been corrupted because of SDCs, or that can show MTBFs or other statistics on the occurence of SDC in supercomputers.

[3] Fiala, D., Mueller, F., Engelmann, C., Ferreira, K. B., Brightwell, R., & Riesen, R. (2011). *Detection and Correction of Silent Data Corruption for Large-Scale High Performance Computing*, SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 2012, pp. 1-12

[4] Li, Z., Menon, H., Maljovec, D., Livnat, Y., Liu, S., Mohror, K., ... & Pascucci, V. (2020). *Spotsdc: Revealing the silent data corruption propagation in high-performance computing systems*. IEEE Transactions on Visualization and Computer Graphics

[5] Cavelan, A., Cabezón, R. M., & Ciorba, F. M. (2019, May). *Detection of silent data corruptions in smoothed particle hydrodynamics simulations*. In 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 31-40).

AI (Datacenters) to the Rescue! (1)

Component	Category	Interruption Count	% of Interruptions
Faulty GPU	GPU	148	30.1%
GPU HBM3 Memory	GPU	72	17.2%
Software Bug	Dependency	54	12.9%
Network Switch/Cable	Network	35	8.4%
Host Maintenance	Unplanned Maintenance	32	7.6%
GPU SRAM Memory	GPU	19	4.5%
GPU System Processor	GPU	17	4.1%
NIC	Host	7	1.7%
NCCL Watchdog Timeouts	Unknown	7	1.7%
Silent Data Corruption	GPU	6	1.4%
GPU Thermal Interface + Sensor	GPU	6	1.4%
SSD	Host	3	0.7%
Power Supply	Host	3	0.7%
Server Chassis	Host	2	0.5%
IO Expansion Board	Host	2	0.5%
Dependency	Dependency	2	0.5%
CPU	Host	2	0.5%
System Memory	Host	2	0.5%

Table 5 Root-cause categorization of unexpected interruptions during a 54-day period of Llama 3 405B pre-training. About 78% of unexpected interruptions were attributed to confirmed or suspected hardware issues.

16,000 GPUs
54-day snapshot of logs
47 planned interruptions
419 unexpected interruptions
$MTBF_{sys} :\approx 3h$
Instrumented NCCL:
- Perf. Improvement
- Stall detection

From *[A1] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... & Vasic, P. (2024). "The llama 3 herd of models." arXiv preprint arXiv:2407.21783.*

AI (Datacenters) to the Rescue! (1)

Component	Category	Interruption Count	% of Interruptions
Faulty GPU	GPU	148	30.1%
GPU			
Soft			
Netv			
Host			
GPU			
GPU			
NIC			
NCC			
Silen			
GPU			
SSD			
Power Supply	Host	3	0.7%
Server Chassis	Host	2	0.5%
IO Expansion Board	Host	2	0.5%
Dependency	Dependency	2	0.5%
CPU	Host	2	0.5%
System Memory	Host	2	0.5%

Zoom

GPU HBM3 Memory	GPU	72	17.2%
GPU SRAM Memory	GPU	19	4.5%
NCCL Watchdog Timeouts	Unknown	7	1.7%
Silent Data Corruption	GPU	6	1.4%

6 - 104 SDCs over 54 days, 16k GPUs

f logs
ptions

Instrumented NCCL:
- Perf. Improvement
- Stall detection

Table 5 Root-cause categorization of unexpected interruptions during a 54-day period of Llama 3 405B pre-training. About 78% of unexpected interruptions were attributed to confirmed or suspected hardware issues.

From *[A1] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... & Vasic, P. (2024). "The llama 3 herd of models." arXiv preprint arXiv:2407.21783.*

AI (Datacenters) to the rescue! (2)

Table 1. Statistics of training incidents collected over a three-month span, encompassing 778,135 LLM training jobs.

Category	Incident Symptom	Count	Percentage
Explicit	CUDA Error	19968	36.1%
	CPU Overload	6095	11.0%
	CPU OOM	5567	10.1%
	Insufficient Disk Space	2755	5.0%
	Infiniband Error	1599	2.9%
	Filesystem Mount	1176	2.1%
	HDFS [22] Error	1104	2.0%
	Container Error	781	1.4%
	OS Kernel Panic	203	0.4%
	GPU Memory Error	188	0.3%
	External Service Error	128	0.2%
	GPU Unavailable	76	0.1%
	Disk Fault	47	0.1%
Implicit	Job Hang	5506	9.9%
	MFU Decline	442	0.8%
	NaN value	148	0.3%
Manual Restart	Code/Data Adjustment	9582	17.3%

16,384 GPUs?
90-day period;
778,135 jobs;
39,687 “explicit” *errors*;
6,096 “implicit” *errors*;
9,582 planned interruptions;

148 - 6,096 SDCs?

From *[A2] Wan, B., Liu, G., Song, Z., Wang, J., Zhang, Y., Sheng, G., ... & Xiang, L. (2025, October). “Robust llm training infrastructure at bytedance.” In Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (pp. 186-203).*

Strategies

Replication: only general-purpose approach

Detectors: verified checkpoints (application-specific)

Need perfect detectors: no false negatives (recall $r = 1$)

Can we use partial detectors with recall $r < 1$?

If yes, how does it compare to replication?

Strategies

Replication: only general-purpose approach

Detectors: verified checkpoints (application-specific)

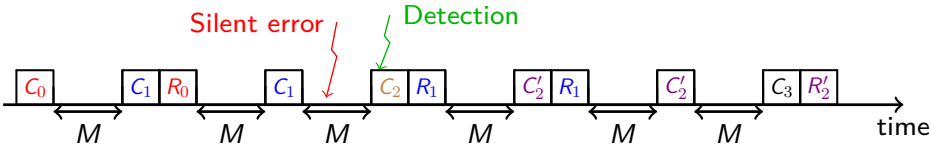
Need perfect detectors: no false negatives (recall $r = 1$)

Can we use partial detectors with recall $r < 1$?

If yes, how does it compare to replication?

Outline

Approach

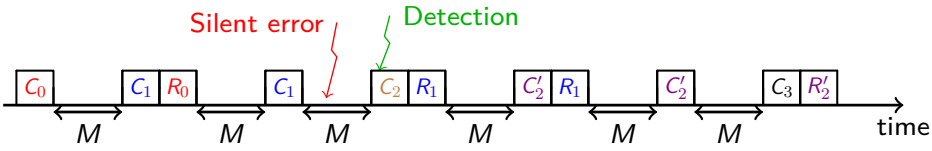


Execution is partitioned into *segments* of M iterations, each followed by a checkpoint

Execution of a new segment (after a checkpoint C):

Two different errors never lead to the same (incorrect) result

Approach



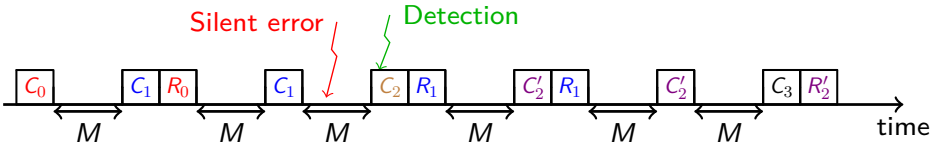
Execution is partitioned into *segments* of M iterations, each followed by a checkpoint

Execution of a new segment (after a checkpoint C):

Re-execute until getting the same result twice

Two different errors never lead to the same (incorrect) result

Approach



Execution is partitioned into *segments* of M iterations,
each followed by a checkpoint

Execution of a new segment (after a checkpoint C):

Execute segment for the first time and checkpoint result res_1

While results after $t \geq 1$ attempts are all different,

execute new attempt $t+1$:

- recover from checkpoint C
- redo the M iterations
- checkpoint result res_{t+1}

Keep the outcome of the two identical checkpoints

Proceed to next segment

Minimizing expected time per iteration

A silent error may strike each iteration independently and with fixed probability f (geometric law)

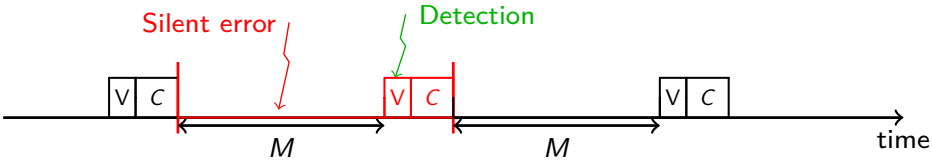
Given a segment:

- cost of first attempt: $M + C$
- cost of following attempts: $R + M + C$
- number of attempts until one is successful:
geometric law of parameter $p_S = (1 - f)^M$
- expected $cost(M) = (M + C) + \left(\frac{2}{p_S} - 1\right)(R + M + C)$
- expected slowdown $\mathcal{S} = \frac{cost(M)}{M} = \frac{2(R+C)}{Mp_S} + \frac{2}{ps} - \frac{R}{M}$
- differentiate for M and solve, find optimal M numerically

No closed-form solution unless $R = 0$ (then Lambert ☺)

Outline

Perfect detectors



Perfect detector of cost V

Segment: M iterations + detector V + checkpoint C

Recall $r = 1 \Rightarrow$ verified checkpoint

Optimal value of M well-known

First-order approximation à la Young-Daly

Perfect detectors

Do you believe it?

- Detectors are not perfect ☹
- High recall is expensive if at all achievable ☹

Can we use partial detectors?

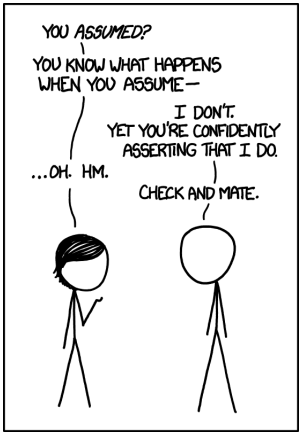
First-order approximation à la Young-Daly

Can we use partial detectors?

Can we use partial detectors?

NO!...unless we make some assumption ...

Can we use partial detectors?



NO!...unless we make some reasonable assumption ...

Partial Detectors

Intuition: online ABFT Scheme for PCG (Zizhong Chen et Al., 2013)

Compute $r^{(0)} = b - Ax^{(0)}, p^{(0)} = z^{(0)} = M^{-1}r^{(0)}, \rho^{(0)} = {r^{(0)}}^T z^{(0)}$ for some initial guess $x^{(0)}$;

Checkpoint A, M, b;

for $i = 0, 1, 2, \ldots$ until convergence do

$q^{(i)} = Ap^{(i)}$; $\alpha = \rho^{(i)} / {p^{(i)}}^T q^{(i)}$;

$x^{(i+1)} = x^{(i)} + \alpha p^{(i)}$; $r^{(i+1)} = r^{(i)} - \alpha q^{(i)}$;

$z^{(i+1)} = M^{-1} r^{(i+1)}$; $\rho^{(i+1)} = {r^{(i+1)}}^T z^{(i+1)}$;

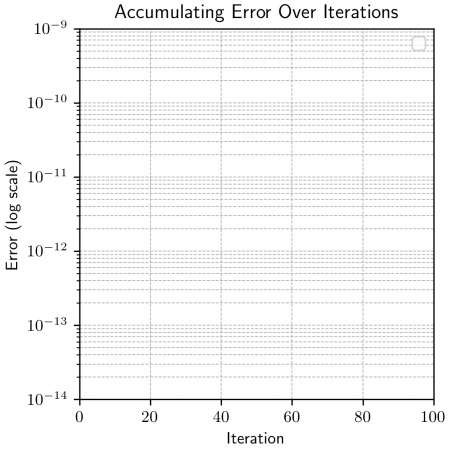
$p^{(i+1)} = z^{(i+1)} + (\rho^{(i+1)} / \rho^{(i)}) p^{(i)}$;

If $\mathcal{D}(A, b, x^{(i+1)}, p^{(i)}, q^{(i+1)}, r^{(i+1)}) > \varepsilon$ then

Recover from last checkpoint;

If $i = i - 1[D]$ then

checkpoint A, M, b;



Partial Detectors

Intuition: online ABFT Scheme for PCG (Zizhong Chen et Al., 2013)

Compute $r^{(0)} = b - Ax^{(0)}, p^{(0)} = z^{(0)} = M^{-1}r^{(0)}, \rho^{(0)} = {r^{(0)}}^T z^{(0)}$ for some initial guess $x^{(0)}$;

Checkpoint A, M, b;

for $i = 0, 1, 2, \ldots$ until convergence do

$q^{(i)} = Ap^{(i)}$; $\alpha = \rho^{(i)} / {p^{(i)}}^T q^{(i)}$;

$x^{(i+1)} = x^{(i)} + \alpha p^{(i)}$; $r^{(i+1)} = r^{(i)} - \alpha q^{(i)}$;

$z^{(i+1)} = M^{-1} r^{(i+1)}$; $\rho^{(i+1)} = {r^{(i+1)}}^T z^{(i+1)}$;

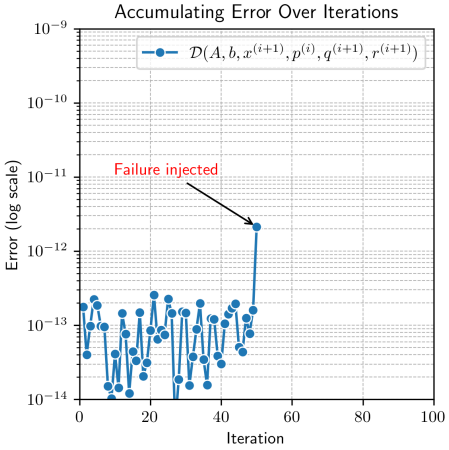
$p^{(i+1)} = z^{(i+1)} + (\rho^{(i+1)}/\rho^{(i)}) p^{(i)}$;

If $\mathcal{D}(A, b, x^{(i+1)}, p^{(i)}, q^{(i+1)}, r^{(i+1)}) > \varepsilon$ then

Recover from last checkpoint;

If $i = i - 1[D]$ then

checkpoint A, M, b;



Partial Detectors

Intuition: online ABFT Scheme for PCG (Zizhong Chen et Al., 2013)

Compute $r^{(0)} = b - Ax^{(0)}, p^{(0)} = z^{(0)} = M^{-1}r^{(0)}, \rho^{(0)} = {r^{(0)}}^T z^{(0)}$ for some initial guess $x^{(0)}$;

Checkpoint A, M, b;

for $i = 0, 1, 2, \ldots$ until convergence do

$q^{(i)} = Ap^{(i)}$; $\alpha = \rho^{(i)} / {p^{(i)}}^T q^{(i)}$;

$x^{(i+1)} = x^{(i)} + \alpha p^{(i)}$; $r^{(i+1)} = r^{(i)} - \alpha q^{(i)}$;

$z^{(i+1)} = M^{-1} r^{(i+1)}$; $\rho^{(i+1)} = {r^{(i+1)}}^T z^{(i+1)}$;

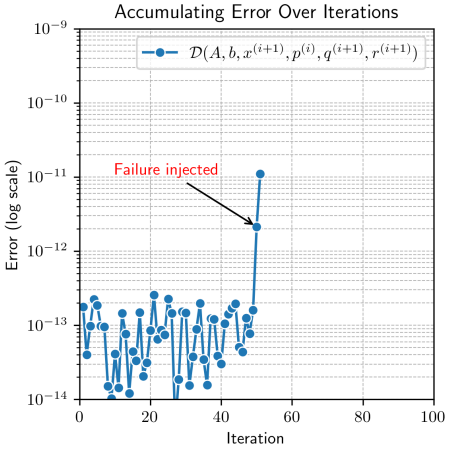
$p^{(i+1)} = z^{(i+1)} + (\rho^{(i+1)} / \rho^{(i)}) p^{(i)}$;

If $\mathcal{D}(A, b, x^{(i+1)}, p^{(i)}, q^{(i+1)}, r^{(i+1)}) > \varepsilon$ then

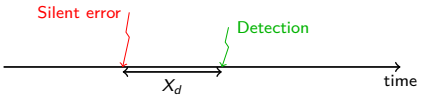
Recover from last checkpoint;

If $i = i - 1[D]$ then

checkpoint A, M, b;



Bounded detection latency



Error and detection latency

Assumption

If a silent error strikes at iteration I ...
...it will be detected at iteration $(I-1)+X$ or after

X obeys a probability distribution with bounded support $[1, D]$

Rationale

The impact of the silent error on the application data grows and becomes more and more *detectable*

For computation errors: numerical amplification as execution progresses

Case study

X truncated geometric R.V. with bounded support $[1, D]$

$X = \min(Y, D)$, with Y geometric R.V. of parameter θ

$\theta =$ probability that error is detected at each iteration $\sim$ recall

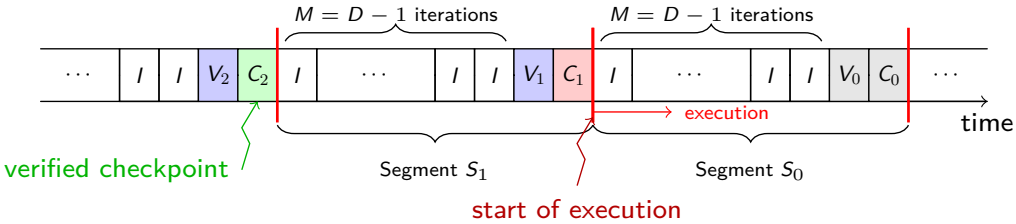
Typical values for maximum detection distance D

θ	$\min\{d \mid P(X \geq d) \leq 10^{-6}\}$	$\min\{d \mid P(X \geq d) \leq 10^{-9}\}$
0.2	62	93
0.4	28	41
0.9	6	9

Efficient partial detector $\theta = 0.9$:
distance detection never exceeds 10 in practice

Poor partial detector $\theta = 0.2$ (capturing only 20% of errors): distance detection never exceeds 100 in practice

Simple scheme

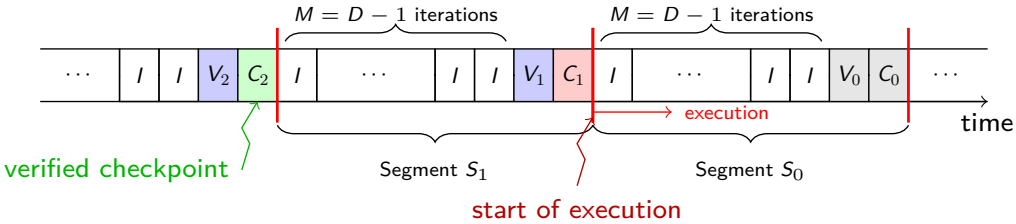


Completing the execution of segment S_0

Checkpoints C_1 and C_2 stored in memory

C_2 is verified (by induction) but C_1 is not (yet)

Simple scheme

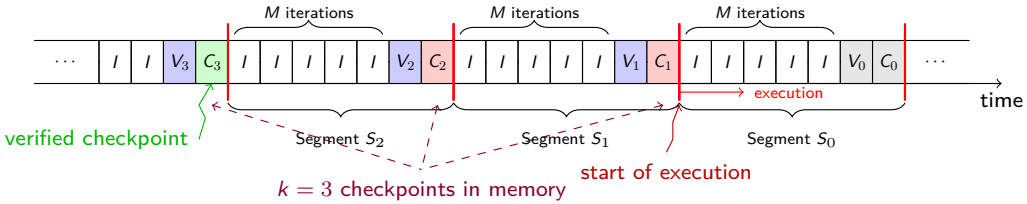


After V_0:

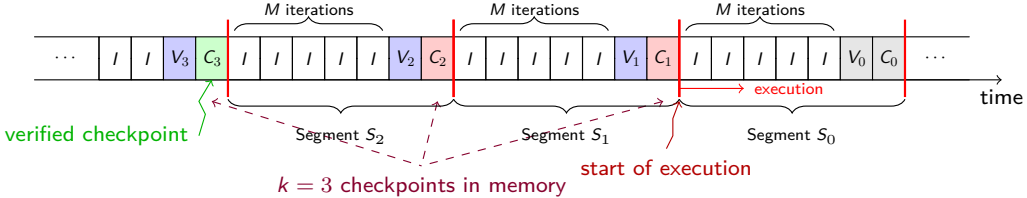
No error is detected
$\Rightarrow$ C_1 is verified (Why?)
$\Rightarrow$ take C_0 and overwrite C_2

An error is detected
$\Rightarrow$ Roll back to C_2, re-execute S_1 then S_0

General scheme



General scheme



k segments, k checkpoints in memory ($k = 3$)

M iterations per segment ($M = 5$)

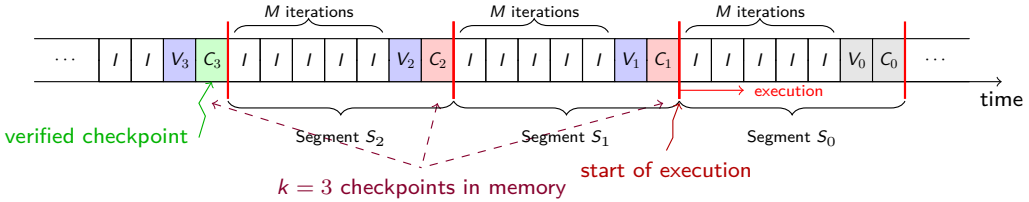
Need $(k-1)M \geq D - 1$

Given M, use $k = \left\lceil \frac{D-1}{M} \right\rceil + 1$

$M = 5$ and $k = 3 \Rightarrow D \leq 11$

Conversely for $D = 11$, $k = 3 \Rightarrow 5 \leq M \leq 9$

General scheme



After V_0:

No error is detected

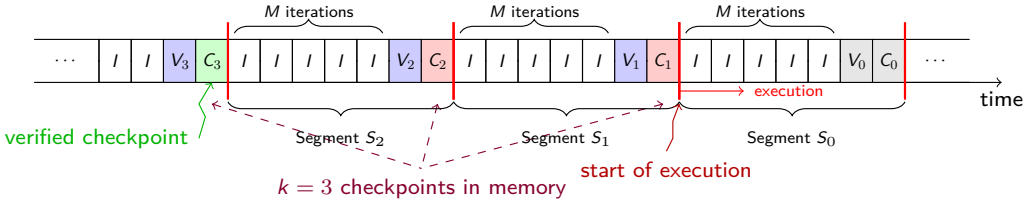
$\Rightarrow$ C_k is verified

$\Rightarrow$ take C_0 and overwrite C_k

An error is detected

$\Rightarrow$ Roll back to C_k, re-execute $S_{k-1}S_{k-2}\dots S_0$

General scheme



Compute E_0: expected time to process segment S_0 with a successful verification V_0 and take new checkpoint C_0
(then delete C_k from memory and move on to next segment)
Minimize $\mathcal{S} = \frac{E_0}{M}$ (with $k = \lceil \frac{D-1}{M} \rceil + 1$)

Computation of E_0

$$E_0 = a_k C + b_k (M + V) + c_k R$$

where

$$a_k = \left[1 + \left(\frac{1}{\Phi_{k-1}} - 1\right)\right] (1 + \sum_{m=0}^{k-3} \prod_{\ell=0}^{m} \frac{1}{\Phi_{k-2-l}}),$$
$$b_k = \left[\frac{1}{\Phi_{k-1}} + \left(\frac{1}{\Phi_{k-1}} - 1\right)\right] \sum_{m=0}^{k-2} \prod_{\ell=0}^{m} \frac{1}{\Phi_{k-2-l}}$$
$$c_k = \left(\frac{1}{\Phi_{k-1}} - 1\right) \prod_{\ell=0}^{k-2} \frac{1}{\Phi_\ell}$$

And

$$\Phi_j = \prod_{\ell=0}^{j} \prod_{i=1}^{M} \left(1 - \frac{f P_{i,\ell}}{(1-f) + f(P_{i,>\ell} + P_{i,\ell})}\right)$$

(Prob. no error detected up to and including V_j during S_j)

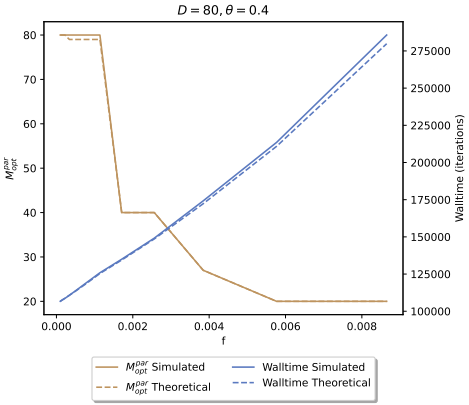
$P_{i,j}$: Prob. error at iteration i detected exactly j segments later

$P_{i,>j}$: Prob. error at iteration i detected more than j segments later

Outline

Validation

Partial Detector: Optimal values of (k, M_{opt}^{par}) and walltime, model vs. simulation



$D = 80$ (max detection distance)

$\theta = 0.4$ (detection probability)

f (error probability per iteration)

M_{opt}^{par} (optimal segment size for partial detection)

Failure-free execution time: 100,000 iterations

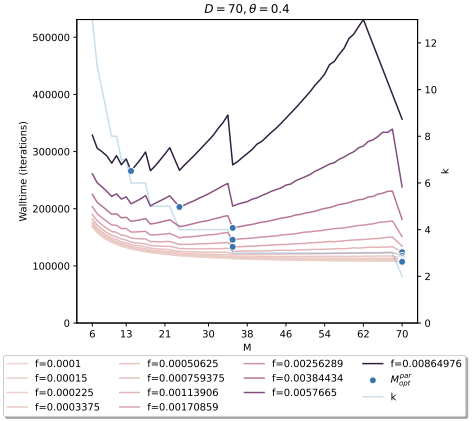
Close match between model and simulation

When f is very small, model has numerical issues

As failures become more frequent, model slightly underestimates completion time

Parameter exploration (1/3)

Partial Detector: Simulated walltime varying error risk f and segment size M



$D = 70$ (max detection distance)

$\theta = 0.4$ (detection probability)

f (error probability per iteration)

M (number of iterations between two checkpoints/verifications)

M_{opt}^{par} (optimal segment size for partial detection)

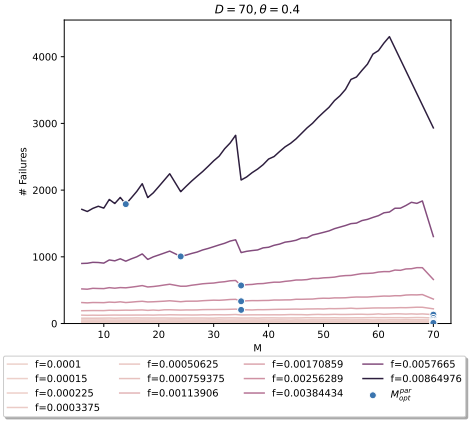
k (number of checkpoints in memory)

Failure-free execution time: 100,000 iterations

When f is very small, best strategy is to take M as large as possible (i.e., $k = 2, M = D - 1$)
For frequent failures, taking more checkpoints can save time (at the cost of more memory), but the strategy with $k = D - 1$ is not too bad

Parameter exploration (2/3)

Partial Detector: Number of errors varying error risk f and segment size M



$D = 70$ (max detection distance)

$\theta = 0.4$ (detection probability)

f (error probability per iteration)

M (number of iterations between two checkpoints/verifications)

M_{opt}^{par} (optimal segment size for partial detection)

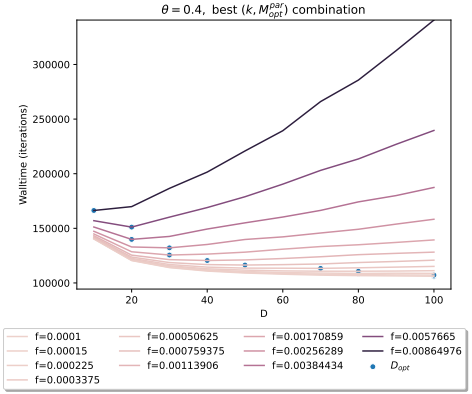
k (number of checkpoints in memory)

Failure-free execution time: 100,000 iterations

Number of failures linearly correlated to Walltime

Parameter exploration (3/3)

Partial Detector: Impact of latency bound D on walltime, varying f and M



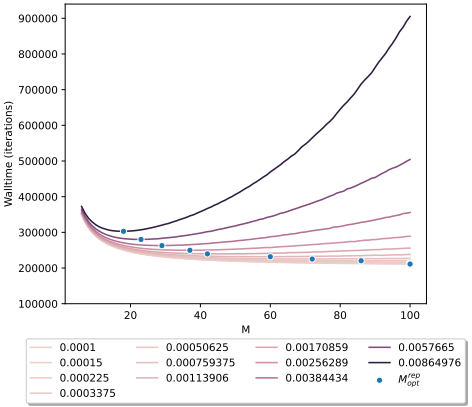
- D (max detection distance)
- $\theta = 0.4$ (detection probability)
- f (error probability per iteration)
- k (number of checkpoints in memory)
- M_{opt}^{par} (optimal segment size for partial detection)
- Failure-free execution time: 100,000 iterations

Optimal $D \neq 1$!

On second thoughts, this is expected: we force a checkpoint every $D - 1$ iterations. When f is small, it is more efficient to checkpoint less often.

Replication (1/2)

Replication: Walltime, varying f and M



f (error probability per iteration)

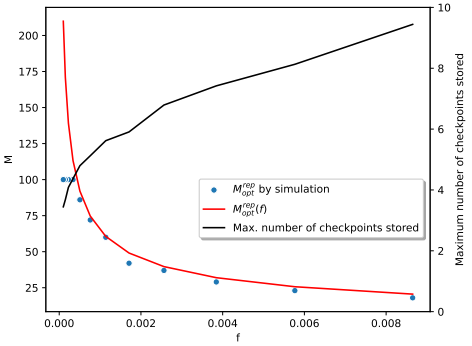
M_{opt}^{rep} (optimal segment size for replication)

Failure-free execution time: 100,000 iterations

Optimal M_{opt}^{rep} increases as f increases.
Same reasoning as above: when f is small, checkpoint less often.
Cost model similar to Young/Daly.

Replication (2/2)

Replication: Best segment size M for replication, model vs. simulation



f (error probability per iteration)

M_{opt}^{rep} (optimal segment size for replication, based on simulations)

$M_{opt}^{rep}(f)$ (optimal segment size for replication, based on model)

Failure-free execution time: 100,000 iterations

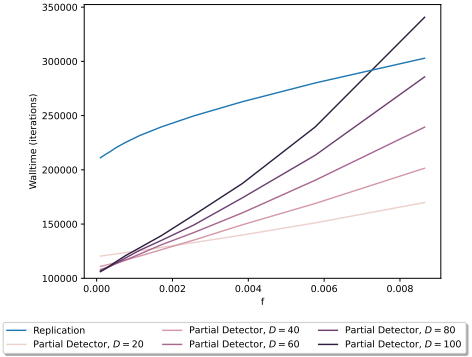
Replication model closely matches simulation results

Small f: didn't simulate with segments longer than 100 iterations

Number of checkpoints to compare to (= max number of segment restart) 3 to 10

Replication vs. Partial Detection

Comparing walltime, replication vs. partial detection with $\theta = 0.4$



D (max detection distance)

$\theta = 0.4$ (detection probability)

f (error probability per iteration)

Failure-free execution time: 100,000 iterations

Use optimal M_{opt} for all approaches.
Replication: at least x2 slower than failure-free
Need unreliable detector AND unreliable system for replication to outperform partial detection
Small D with small f implies loss of performance due to forced checkpoints

Conclusion

Synthesis

First comparison of replication with partial detection

Optimal solution for both approaches

Monte-Carlo simulations perfectly match model predictions

Partial detectors can massively outperform replication

Number of stored checkpoints:
fixed for partial detection, unknown for replication

Future work

Extend analysis to false positives (precision < 1):
extra rollbacks, recoveries, re-executions due to false alarms

Experimental validation with PCG

AI (Datacenters) to the rescue! (2bis)

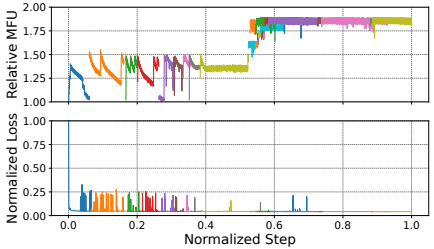


Figure 2. Normalized loss and relative MFU (ratio to the minimum MFU value) curves of an LLM training job running on 1000 GPUs in a production environment. Each color indicates one continuous, uninterrupted training period.

10,000 GPUs
10-day training span
28 planned+unplanned interruptions
Planned interruptions: rollback more than required to validate

From *[A2] Wan, B., Liu, G., Song, Z., Wang, J., Zhang, Y., Sheng, G., ... & Xiang, L. (2025, October). "Robust llm training infrastructure at bytedance." In Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles (pp. 186-203).*

AI studies quotes

[A2] "LLM training is akin to a scientific experiment"
[A1] [because of checkpointing] "tens of thousands of GPUs may increase or decrease power consumption at the same time"
[A3] "Since the necessary conditions occur within 2 training iterations after a failure occurs, the error detection latency of our technique is bounded."

[A3] He, Y., Hutton, M., Chan, S., De Gruijl, R., Govindaraju, R., Patil, N., & Li, Y. (2023, June). *Understanding and mitigating hardware failures in deep learning training systems*. In Proceedings of the 50th Annual International Symposium on Computer Architecture (pp. 1-16).

Conclusion

Synthesis

First comparison of replication with partial detection

Optimal solution for both approaches

Monte-Carlo simulations perfectly match model predictions

Partial detectors can massively outperform replication

Number of stored checkpoints:
fixed for partial detection, unknown for replication

Future work

Extend analysis to false positives (precision < 1):
extra rollbacks, recoveries, re-executions due to false alarms

Experimental validation with PCG