When is Multi-version Checkpointing Needed?

Guoming Lu[1,2], Ziming Zheng[1] and Andrew A. Chien[1,3]

University of Chicago[1]
University of Electronic Science and Technology of China[2]
Argonne National Laboratory[3]

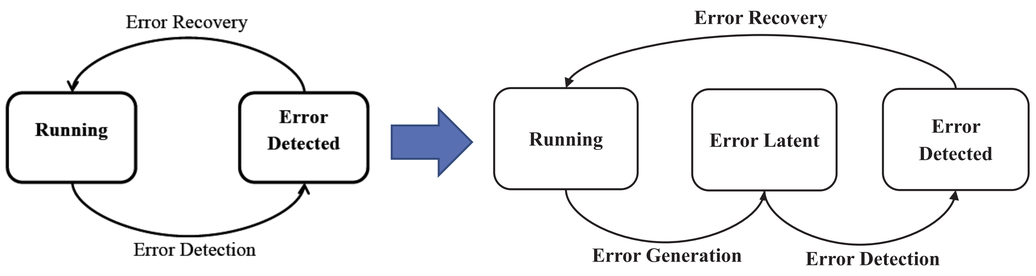
Goals

- Understand influence of latent error for exascale system and explore the importance of multi-version checkpoint systems.
  - Define a *Latent Error model* for future systems that captures the reality of *latent errors*
  - Derive optimal checkpoint intervals for systems with latent errors.
  - Evaluate the benefit of multi-version system for future system

Latent Error

- Latent error errors are likely to be a growing problem in future system.
  - Why? Increasing variety of errors, cost of checking.
    - More subtle hardware and software errors (small data perturbation, small data structure perturbation, minor divergence)
    - More expensive checks (scrubbing, x-structure, x-node, symmetry data structure, energy conserve, ...)
- Latent error make traditional checkpointing system low efficiency or even infeasible.

System Model Enhancement

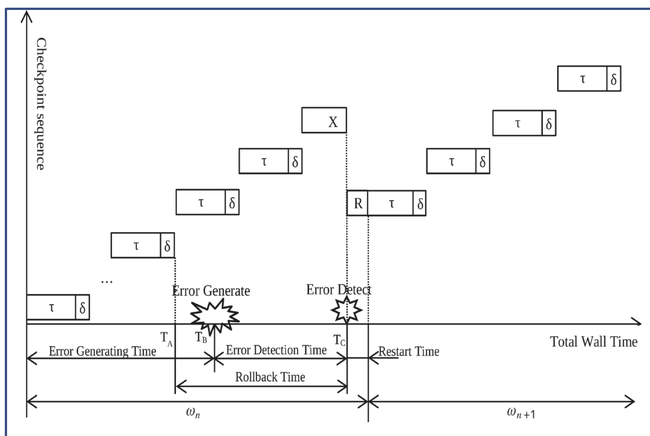


(a)Fail stop model (b)Latent Error model

Figure 1. System model enhancement

- Existing checkpointing systems mostly assume "Fail-stop" model.
- Multi-version system use "Error Latent" state to represent detection latency.

Multi-version Checkpointing Scheme



An error at time T_B is detected at time T_c. To recover, the application must rollback to T_A. Checkpoints between T_A and T_c are wasted.

Figure 2. Latent Errors and Multi-version Checkpoint Recovery.

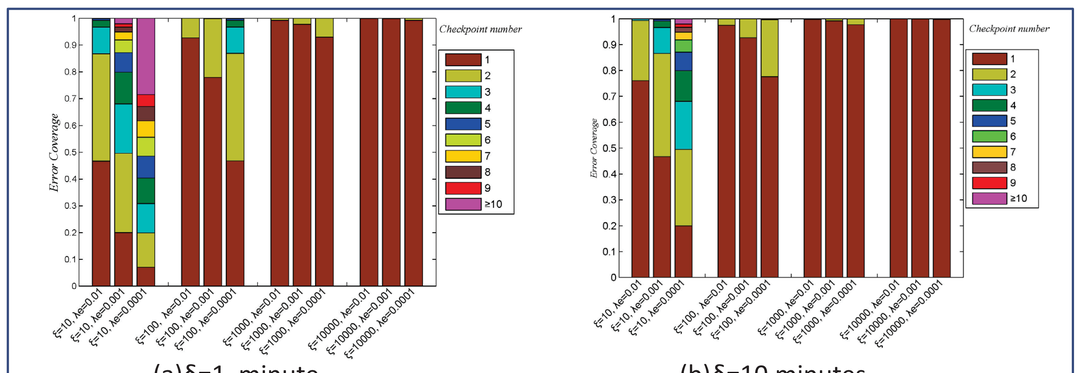
Optimal Checkpointing Interval

- Assumptions:
  - Error generation time follows exponential distribution with rate parameter λe,
  - Error detection time follows exponential distribution with rate parameter λd
- The optimal checkpointing with latent errors:

$$\tau_{opt} = \sqrt{2\delta(\frac{1}{\lambda_e} + \frac{1}{\lambda_d})} - \delta$$

δ is the checkpoint overhead.

Distribution of Version Numbers Needed to Recovery Errors

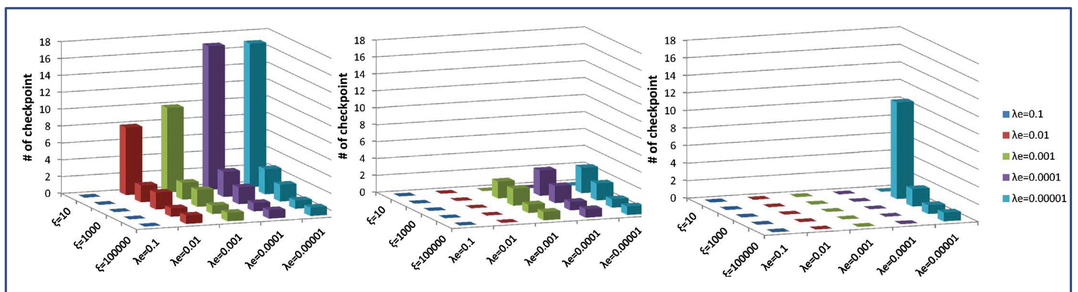


(a)δ=1 minute (b)δ=10 minutes

Figure 3. Fraction of errors covered by each checkpoint. Parts (a) and (b) vary the checkpoint overhead.

- If error detection latency is low (large ξ, fail stop"), 1-2 versions are sufficient.
- Higher latency, the number of versions increase significantly.
- Reduced checkpoint overhead increases need for more versions.

How many Versions to Achieve Efficiency?



(a) Efficiency=75% (b) Efficiency=90% (c) Efficiency=99%

Figure 4. # of Checkpoints needed for efficiency, with checkpoint overhead (δ) and restart time equal 1 minute. Zero represents infeasible efficiencies or the giving configuration.

- Number of version increases with the decreasing of error rate.
- Multi-version(more than two) help to achieve high efficiency in latent error case(smaller ξ value).

Exascale Reliability Scenarios

Table 1. Achievable efficiency and least version requirements compared with single version scheme(the checkpoint interval is set to K times of optimal interval). δ=10 for traditional checkpoint system. δ= 1 for optimized.

	ξ=10						ξ=500					
	Traditional(δ=10)			Optimized(SCR)			Traditional(δ=10)			Optimized(SCR)		
λe(per minute)	versions(K)	Ef of K-Version	Ef of 1-Version	versions(K)	Ef of K-Version	Ef of 1-Version)	versions(K)	K-Version	1-Version	versions(K)	K-Version (SCR)	1-Version (SCR)
0.3	NA	NA	NA	3	0.370	NA	2	NA	NA	2	0.404	NA
0.1	3	0.18	NA	4	0.566	NA	2	0.195	0.057	2	0.620	0.083
0.03	3	0.371	NA	5	0.695	NA	2	0.404	0.219	2	0.763	0.295
0.01	5	0.568	NA	9	0.787	NA	2	0.619	0.444	2	0.864	0.557
0.003	6	0.697	0.08	11	0.837	NA	2	0.764	0.647	2	0.919	0.736
0.001	10	0.792	0.272	16	0.866	0.261	2	0.865	0.793	3	0.956	0.853

- Multi-version required for usable efficiency at high error rates, many versions required
- Multi-version benefit increases with lower error rates (rework) and lower checkpoint cost (coverage)
- Multi-version much better, particularly at high error rates

Figure 5. Efficiency comparison. Single-version scheme VS. K-version scheme

Future Efforts

- To do real test of these ideas in the study of real exascale systems, and software libraries such as GVR
- To Investigate tradeoff of error detection and system efficiency
- To explore cases multiple latent errors

The Large-scale System Group(LSSG)

Global View Resilience Project (GVR) http://gvr.cs.uchicago.edu

ASCR X-Stack Awards DE-SC0008603/57K68-00-145