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**

![System model enhancement](Image)

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

**Multi-version Checkpointing Scheme**

![Multi-version Checkpointing Scheme](Image)

- An error at time \(T_e\) is detected at time \(T_e\). To recover, the application must rollback to \(T_e\). Checkpoints between \(T_e\) and \(T_f\) are wasted.

**Optimal Checkpointing Interval**

- Assumptions:
  - Error generation time follows exponential distribution with rate parameter \(\lambda_g\).
  - Error detection time follows exponential distribution with rate parameter \(\lambda_d\).
- The optimal checkpointing with latent errors:
  \[
  \tau_{opt} = \frac{1}{\lambda_g} + \frac{1}{\lambda_d} - \delta
  \]
  \(\delta\) is the checkpoint overhead.

**Distribution of Version Numbers Needed to Recovery Errors**

![Distribution of Version Numbers Needed to Recovery Errors](Image)

- If error detection latency is low (large \(\xi\), 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?**

![How many Versions to Achieve Efficiency?](Image)

- 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 \(\xi\) value).

**Exascale Reliability Scenarios**

![Exascale Reliability Scenarios](Image)

- 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

**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