Understanding Error Sensitivity in Checkpointing for Linear System Solvers

IOWA

Bohan Zhang, Yafan Huang, Jerry Li, Guanpeng Li (Advisor)

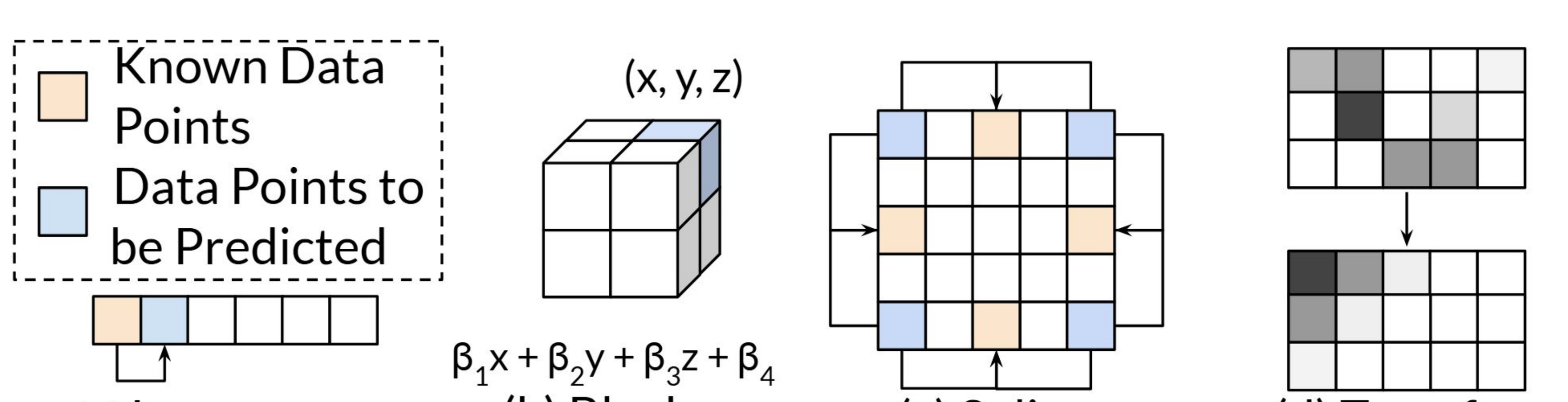
Computer Science Department, University of Iowa, Iowa City, IA, The United States

Introduction

- **Motivation**
  - Large-scale iterative linear solvers require fault tolerance due to long runtimes and high failure risks, but traditional checkpointing introduces significant storage overhead.
  - Error-bounded lossy compression can significantly reduce checkpoint sizes but introduces compression errors that may lead to additional computational overhead (extra iterations).
  - Unclear how compressor configuration influence extra iterations and compressed checkpoint size leaving users dependent on inefficient trial-and-error tuning.
- **Our goals**:
  - **Analysis of compressor configurations' impacts** on checkpoint size and compression error.
  - **Quantification of compression error's effect** on solver convergence and number of extra iterations.
  - **Investigation of solver behavior** when recovering checkpoints at various execution stages.

Error-bounded Lossy Compression

- **Definition:** A data reduction technique that offers aggressive compression ratios by allowing a controlled amount of data loss.
- **Error Bound:**
  - Maximum difference between original and decompressed data.
- **Error Modes:**
  - ***ABS***: Error capped by a fixed constant.
  - ***REL***: Error scaled by the data range.
- **Prediction Algorithms:**
  - ***Lorenzo***: Uses local neighbors to predict and reduce redundancy.
  - ***Block Regression***: Splits data into blocks for polynomial fitting.
  - ***Spline Interpolation***: Uses spline curves to approximate data.
  - ***Transforms***: Concentrate crucial low-frequency parts, discarding less critical high-frequency details.

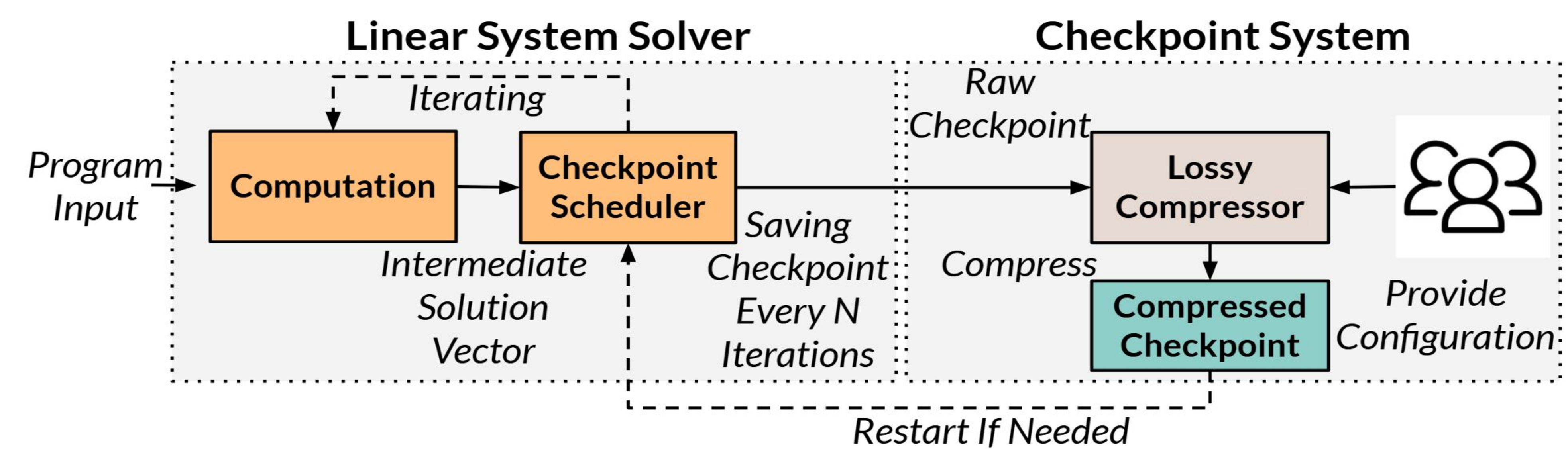


(a) Lorenzo Prediction (b) Block Regression (c) Spline Interpolation (d) Transform

Sensitivity Study Setup

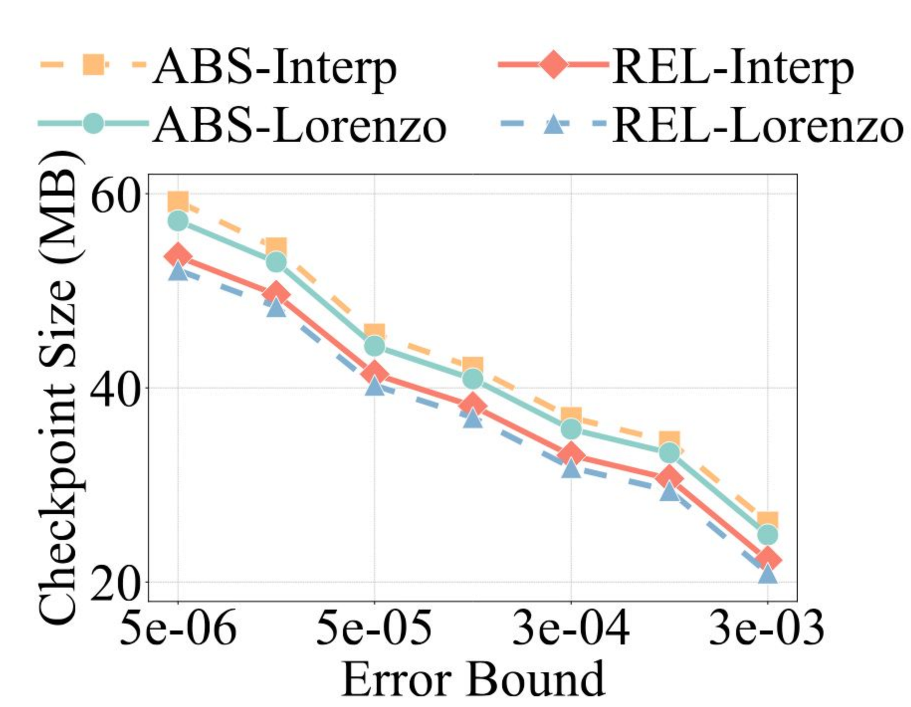
- We focus on the **Conjugate Gradient (CG)** method, a widely used solver for symmetric positive-definite matrices whose performance is representative of iterative approaches.
- We utilize the **SZ3 lossy compressor** to compress checkpoints, exploring error bounds from 3×10^{-3} to 5×10^{-6}, various error modes (ABS and REL), and prediction algorithms provided by SZ3 (Interpolation and Lorenzo).
- Metrics:
  - **Checkpoint Size:** Storage needed for compressed solution vector.
  - **Compression Error:** Measured by mean squared error (MSE) between original and decompressed solution vectors.
  - **Extra Iterations:** Additional iterations needed to converge due to compression-induced errors.

Lossy Checkpoint Scheme

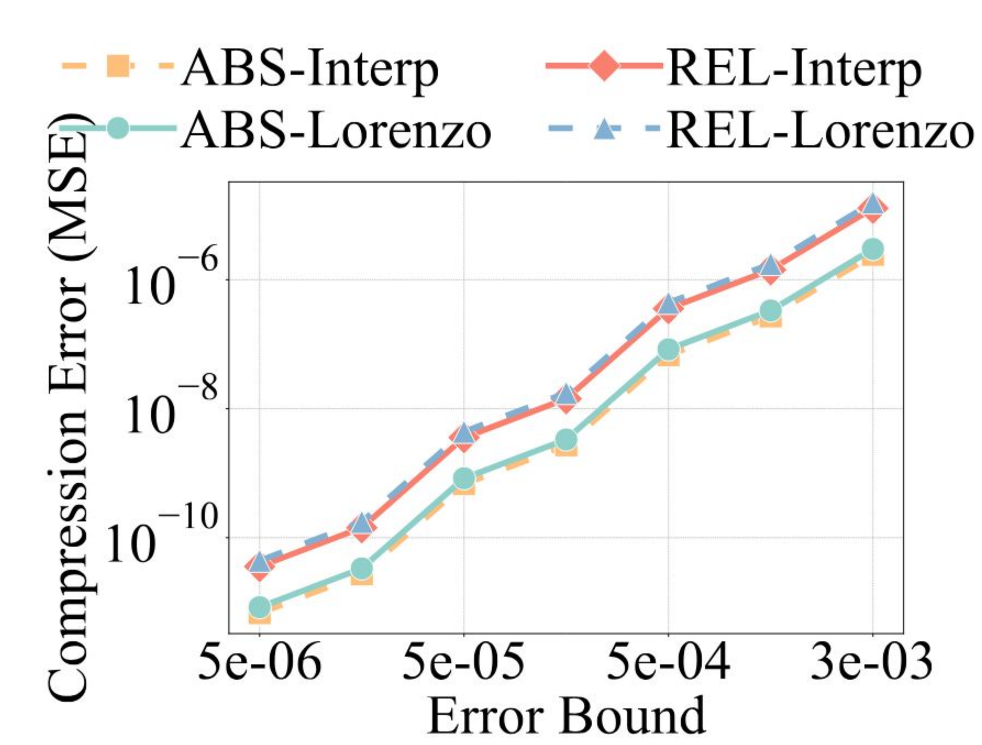


1. **Application-level Checkpointing**: The scheme employs lossy checkpointing at the application level, independent of external fault detection.
2. **Computation Component**: Performs core numerical operations each iteration and produces the intermediate solution vector.
3. **Checkpoint Scheduler**: Periodically saves the intermediate solution vector as an uncompressed raw checkpoint.
4. **Lossy Compression**: Apply compression configurations to raw checkpoint.
5. **Recovery**: On faults, the solver decompresses the checkpoint, restoring the solution vector and re-initializing variables.

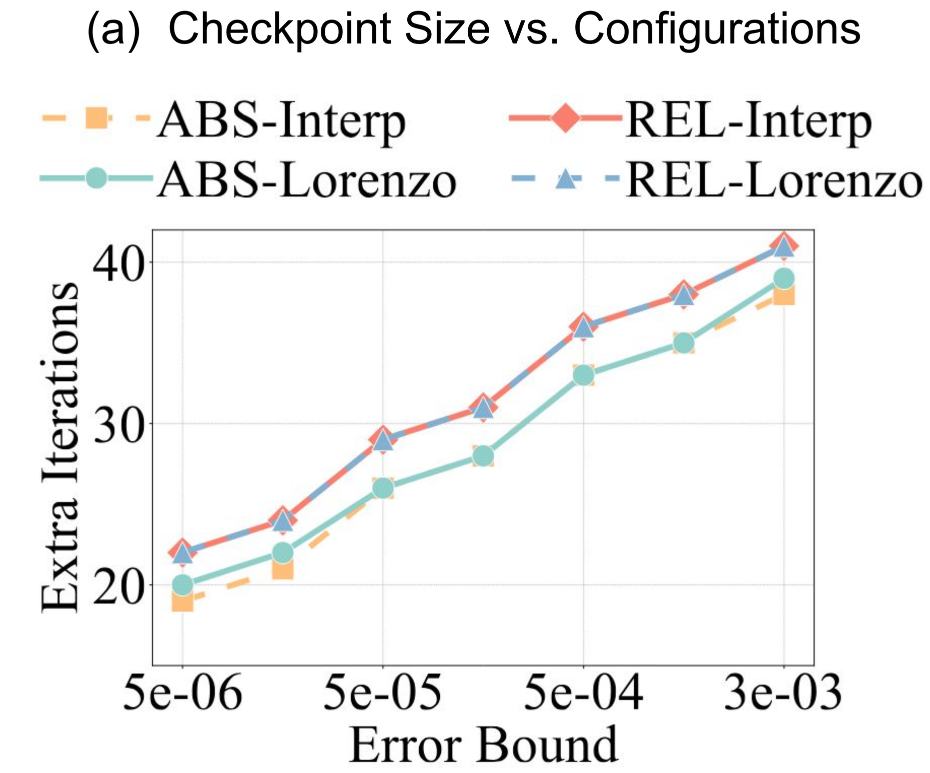
Key Observations



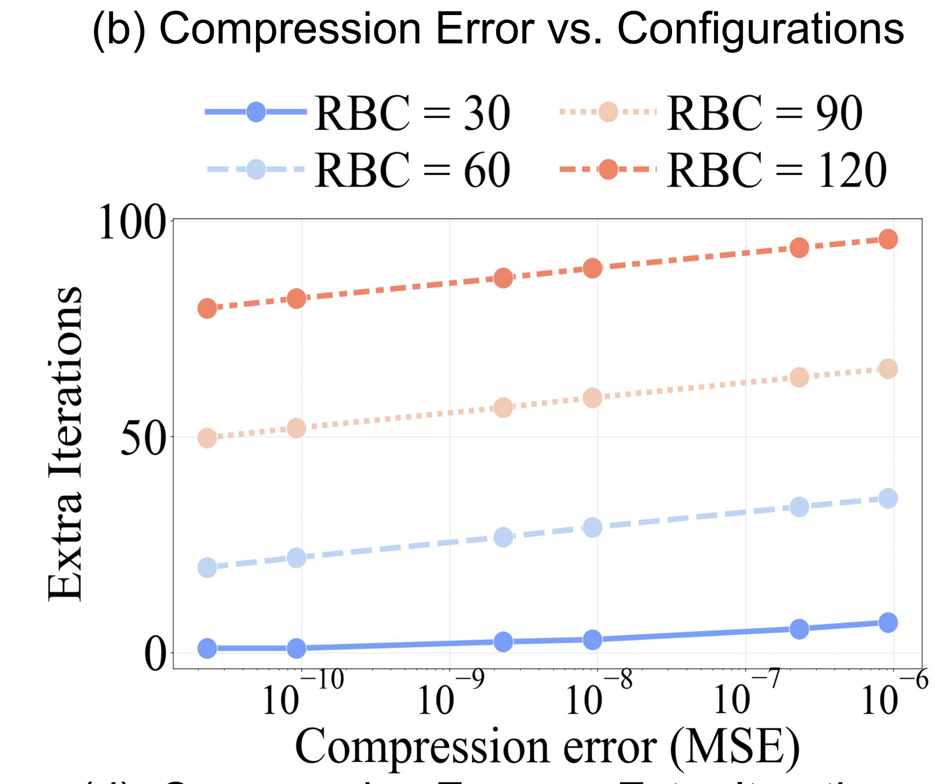
(a) Checkpoint Size vs. Configurations



(b) Compression Error vs. Configurations



(c) Extra Iterations vs. Configurations



(d) Compression Error vs. Extra Iterations at Rollback Checkpoints (RBC)

Takeaway 1: By increasing the error bound and using the Lorenzo algorithm with relative error mode, we can effectively reduce checkpoint size.

Takeaway 2: Using lower error bounds (especially with ABS mode) keeps compression error small, thereby minimizing the number of extra iterations.

Takeaway 3: In early solver stages, larger compression errors are acceptable, as the iterative process can correct them. Near convergence, smaller errors are essential to avoid excessive extra iterations.

Future Works

- Extend our study to additional iterative solvers (such as GMRES and Jacobi).
- Broader range of fault models, thereby validating and generalizing our findings.