Resilience Assessment and Enhancement

SUPER — Institute for Sustained Performance, Energy, and Resilience

Pedro Diniz, Bob Lucas
USC-ISI

George Bosilca, Thomas Herault
University of Tennesse

Chunhua Liao, Bronis R. de Supinski, Todd Gamblin, Kathryn Mohror, Dan Quinlan
LLNL

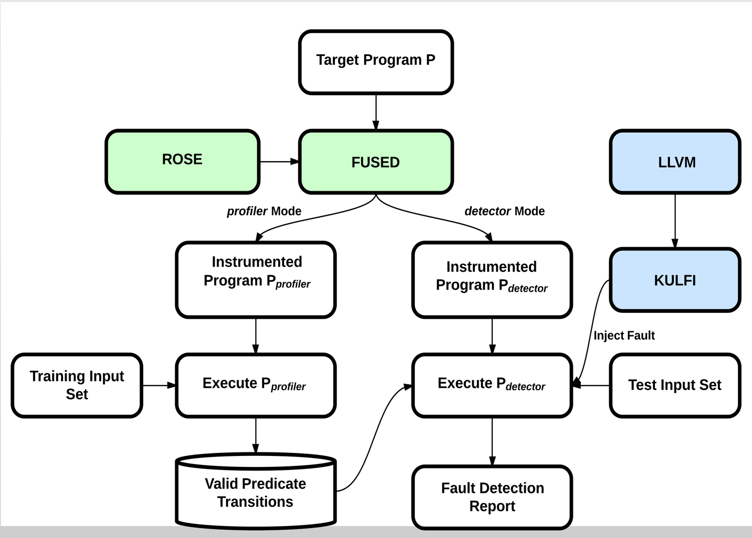
Ganesh Gopalakrishnan
University of Utah

U.S. Department of Energy — Office of Science

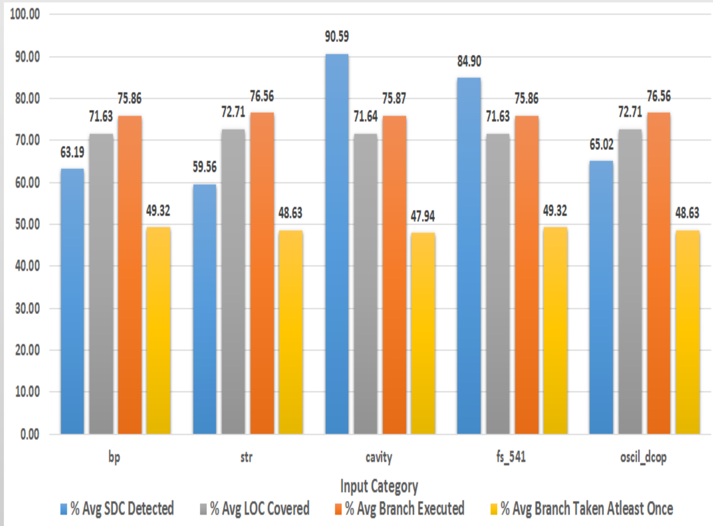
FUSED Framework

- Automatically synthesizes and inserts detectors
- Likely invariants are used for soft errors detection
- Uses profilers to generate likely invariants
- Our approach derives likely invariants using predicate transitions

- FUSED is evaluated using SuperLU Library
- Up to 90% of soft errors are detected
- Detectors only inserted into top-level LU factorization routine
- Average execution overhead of 15.7% due to the detectors



FUSED: An Error Detection Framework



Experimental Result

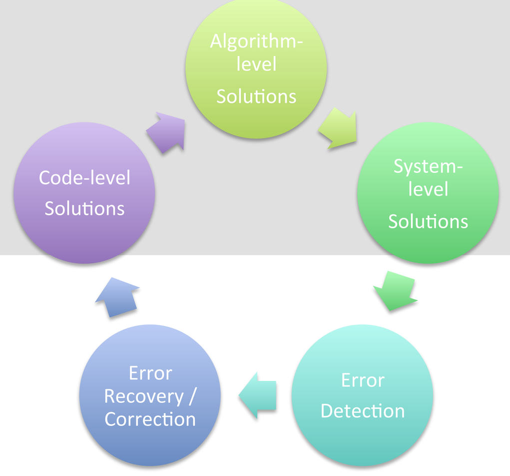
- Fault Injections done using Kontrollable Utah LLVM Fault Injector (KULFI)
  - https://github.com/soarlab/KULFI
- Active collaborations to promote usage of KULFI in other resilience studies
- Current collaborators - Greg Bronevetsky (LLNL), Sui Chen, Lu Peng (LSU)

Future Work

- Develop predicate-abstraction based heuristics for detector placement optimization
- Apply these heuristics for characterizing resilience properties of a progra

Abstract

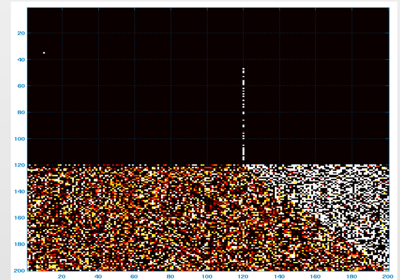
We are addressing the problems in software resilience with a holistic multifaceted approach that spans across software levels. One approach emphasizes tracking expected control flows or data invariants, and is aimed at detecting silent data corruption. Another explores language extensions and compiler technology to convey to compilers and run-time system resilience properties of code sections and algorithms. Additionally, we are investigating specific algorithmic properties of applications to develop fault tolerant extensions to dense and sparse methods. At the highest levels, we detect silent-data corruptions by replicating and comparing values across MPI processes and improve on the state of the art for checkpoint/restart with innovations in file systems and checkpoint compression.



Algorithmic-Based Approaches

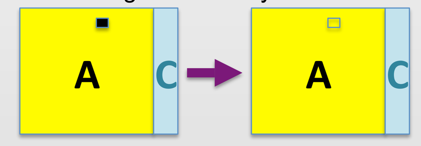
CHALLENGE

Detection and Correction of Errors using algorithmic properties



↑ The figure illustrates how a single **memory corruption** propagates to the entire matrix during a LU factorization due to the recursive nature of the algorithm.

Principle of Algorithm Based Fault Tolerance: the data is augmented with checksum columns that preserve a mathematical property, introducing redundancy in the data.



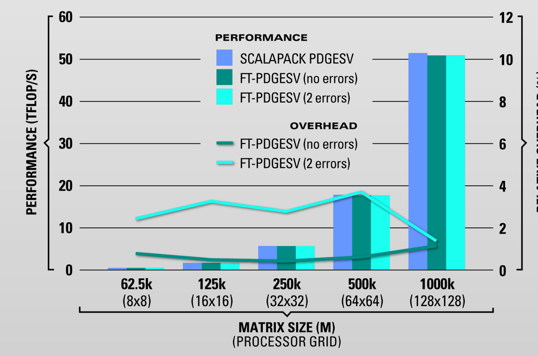
↑ In case of error(e.g. memory corruption), checksum inversion allows to restore the corrupted value.

Scope of ABFT: although ABFT techniques have been applied mostly for dense direct method in linear algebra (one-sided factorizations, two-sided factorizations, …), it is not limited to these: iterative methods (CG, …) and sparse operations are also target of ABFT techniques.

PERFORMANCE

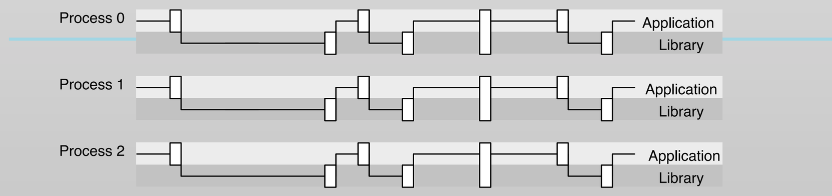
Kraken – Weak Scaling

LU Factorization (PDGESV)
Despite Memory Corruption Errors



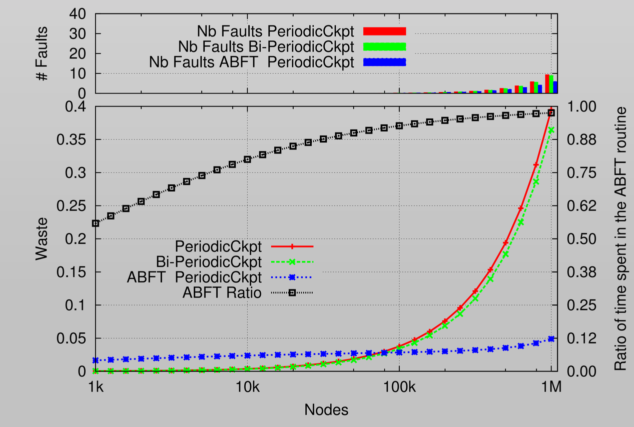
MODELS

Build comprehensive fault management models: iterative application with library phases protected by ABFT and application phases protected by checkpoint/restart.



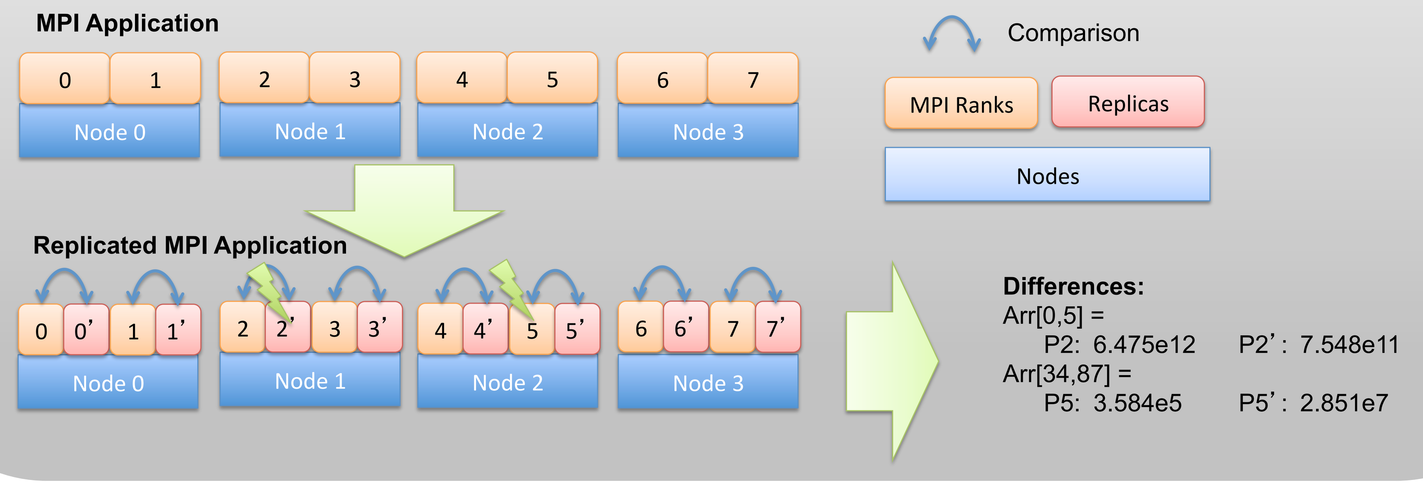
Weak scaling scenario for Exa-scale
- Number of components, x, increases
- Memory per component M_{ind} remains constant
- Problem size n increases in $O(x^{1/2})$

- MTBF at $x = 10^5$: 1 day, is $O(1/x)$
- Checkpoint (= restart) at $x=10^5$, is 1 minute (remains $O(x)$)
- Application spend 80% of execution time in the ABFT protected library



Silent Error Detection

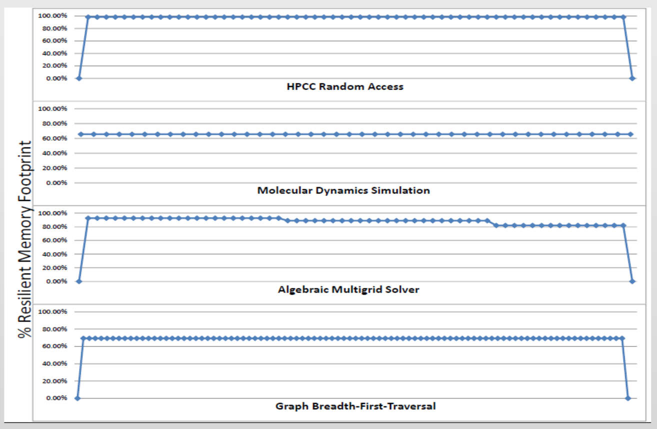
- We have seen unexpected behavior in jobs at scale on the LLNL Sequoia machine
  - Certain high-performance LINPACK runs have high residual
- Currently have no way to detect silent memory corruption
  - Conducting a detailed characterization of memory error rate of BG/Q, Cray
- Developing a tool, Dragnet, that finds memory errors through MPI replication
  - Replicate MPI processes on-node, do shared-memory comparison of arrays
  - Can convert any MPI program into a silent error detector

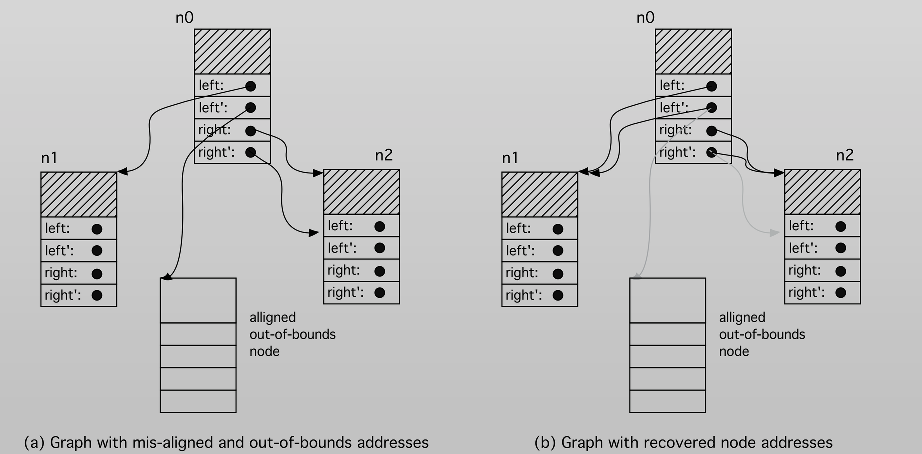


Language Extensions & Compiler Technology for Resilience

- Annotations that allow user to express fault-tolerant requirements and expectations: when and where errors matter and what to do about them.
- ROSE source-level resilience-oriented and user-guided transformations for array-based pointers and graph-based computations

tolerant int rgb[XDIM][YDIM];
tolerant<MAX.VALUE=...> unsigned int counter;
tolerant<precision.8f> double low_precision;
<type>* <var> = (<cast>)tolerant_malloc(sizeof(<type>));

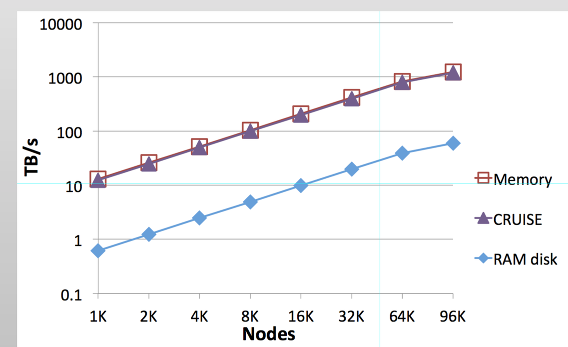




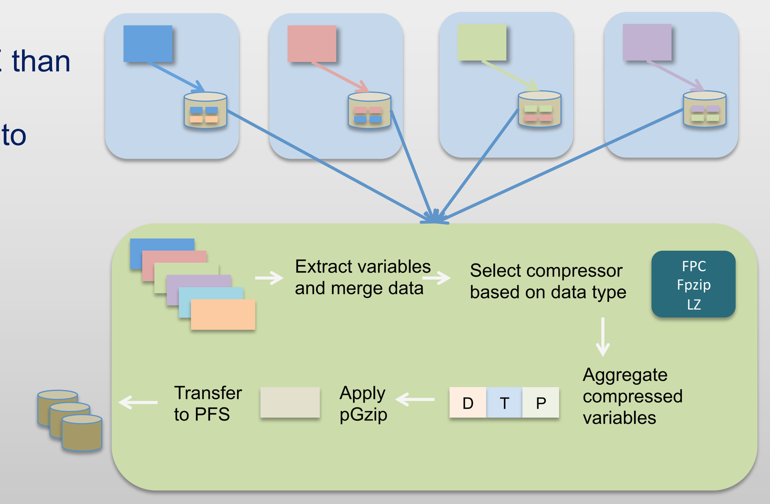
(a) Graph with mis-aligned and out-of-bounds addresses (b) Graph with recovered node addresses

Scalable Checkpoint/Restart with SCR

- The Scalable Checkpoint/Restart Library (SCR) caches checkpoints on compute nodes
- SCR caches checkpoints 20x faster when using CRUISE than when writing to RAM disk
- mcrEngine uses semantic information in checkpoint files to increase compression rates



CRUISE achieves a write bandwidth of 1 PB/s with 1 million tasks on Sequoia



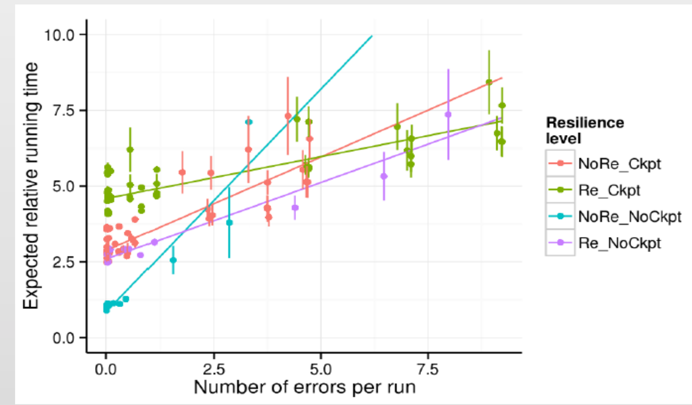
mcrEndgine can reduce data size as much as 70% and reduce I/O overhread by up to 87%

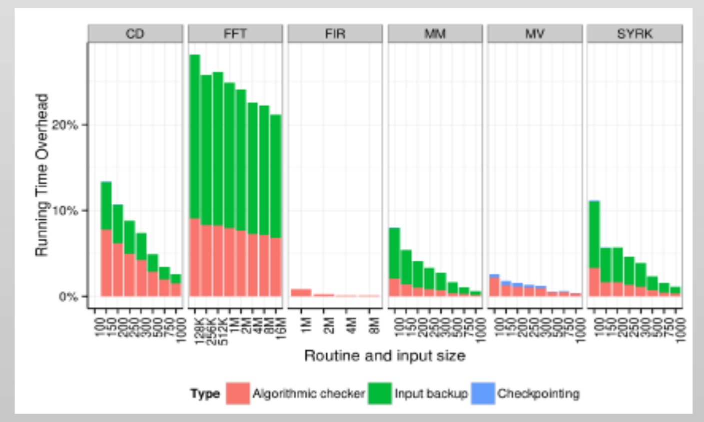
Comprehensive Algorithmic Resilience

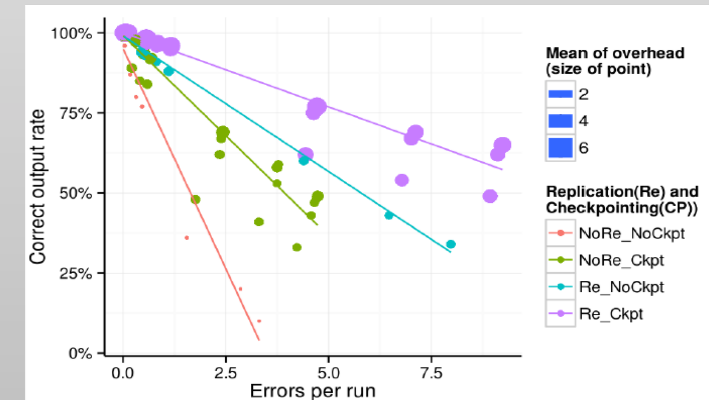
We demonstrate that is possible to protect scientific applications from many sources of errors combining three different resilience techniques:

- Algorithmic error checkers
- Replications of key data structures
- Checkpoint-restart mechanisms

Significant improvements can be achieved in terms of reduction of performance slowdown and output accuracy







(Partial) support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (and Basic Energy Sciences/Biological and Environmental Research/High Energy Physics/Fusion Energy Sciences/Nuclear Physics). LLNL-POST-657319.

(Partial) support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (and Basic Energy Sciences/Biological and Environmental Research/High Energy Physics/Fusion Energy Sciences/Nuclear Physics) under award numbers DE-SC0006844, DESC0006947.