SUPER
INSTITUTE FOR SUSTAINED PERFORMANCE, ENERGY, AND RESILIENCE

Resilience Assessment and Enhancement

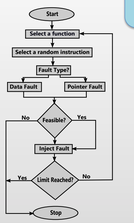
Bronis R. de Supinski, Marc Casas-Guix, Todd Gamblin, Kathryn Mohror
LLNL

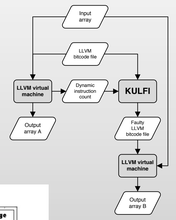
Pedro Diniz, Bob Lucas
USC-ISI

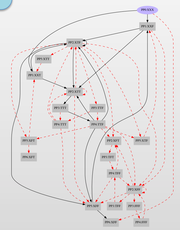
Ganesh Gopalakrishnan
University of Utah

George Bosilca, Thomas Herault
University of Tennesse

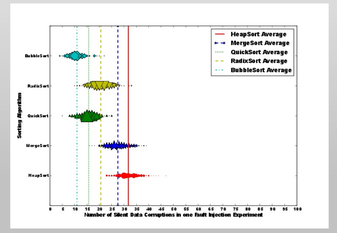
KULFI and PTD in Action







PTD for Bubble Sort (red shows faulty moves)

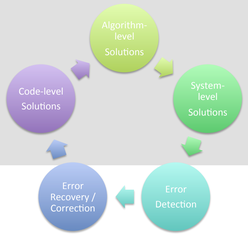



```
bubbleSort(array[], size)
 {
PP0: //state --> XXX
L0:  for (i = (size - 1); i > 0; i--)
     {
PP1:   //state --> TXX
L1:  for (j = 1; j <= i; j++)
     {
PP2:   //state --> TTX
L2:    if (array[j-1] > array[j])
       {
PP3:     //state --> TTT
L3:      swap(array[j-1],array[j])
PP4:     //state --> TTF
       }
       else
       {
PP5:     //state --> TTF
L4:      skip
       }
     }
PP6: //state --> TFX
   }
 }
```

- KULFI (Kontrollable Utah LLVM Fault Injector) helps inject faults at the LLVM level (joint work with Rakamaric' ; https://github.com/soar-lab/KULFI)
- Fault injection helps gauge inherent resilience of algorithm types
- Synthesizing control-flow invariants via Predicate Transition Diagrams (PTD) gives insight into basic reasons
- Future studies to cover realistic numerical routines

Abstract

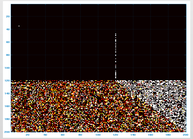
We are addressing the problems in software resilience with a holistic multifaceted approach that spans across software levels. One approach emphasizes tracking expected control flows or data invariants, and is aimed at detecting silent data corruption. Another explores language extensions and compiler 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 level, we detect silent-data corruptions by replicating and comparing values across MPI processes.



Algorithmic-Based Approaches

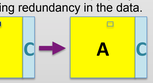
CHALLENGE

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.



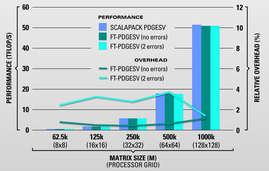
↑ In case of error(e.g. memory corruption), checksum inversion allows to restore the corrupted value.

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.

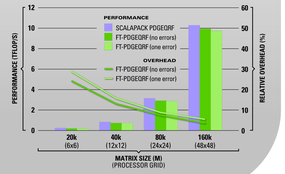
PERFORMANCE

Kraken – Weak Scaling

LU Factorization (PDGESV)
Despite Memory Corruption Errors



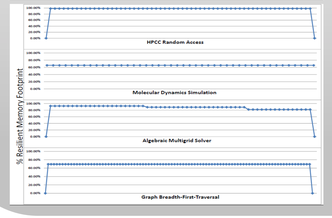
QR Factorization (PDGEQRF)
Despite Fail-Stop Errors

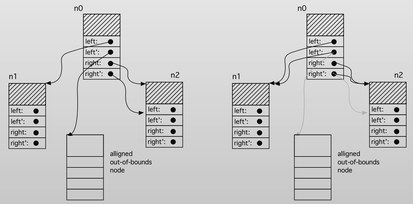


Language Extensions & Compiler Technology for Resilience

- 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>));
```



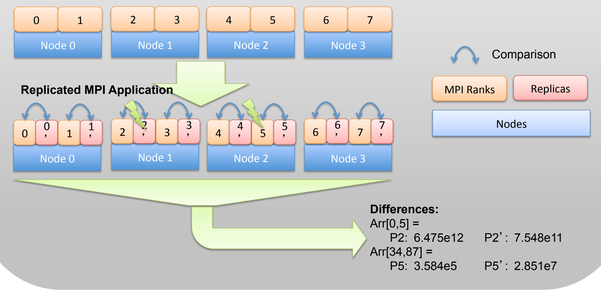


(a) Graph with mis-aligned and out-of-bounds addresses

(b) Graph with recovered node addresses

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

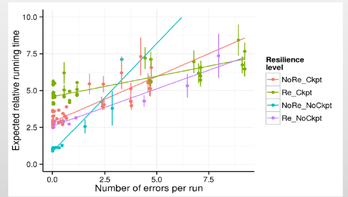


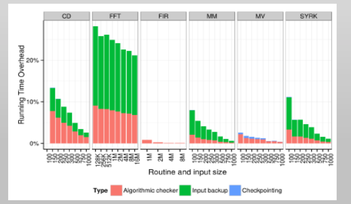
Comprehensive Algorithmic Resilience

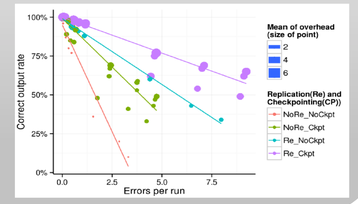
We demonstrate that is possible to protect scientific applications from many sources of errors combining three different resilience techniques:

- 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-641050

(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.