Software-level Fault Tolerant Framework for Task-based Applications

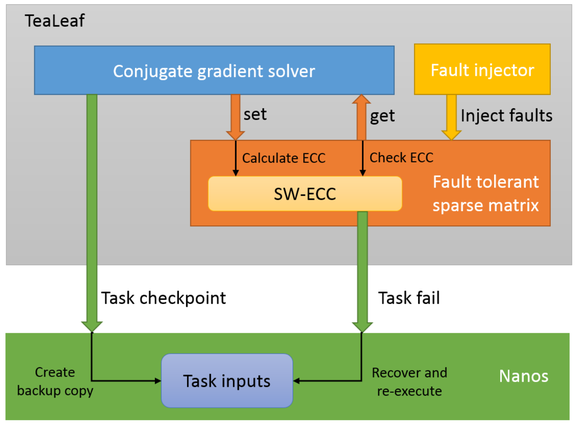
Joy Yeh, Grzegorz Pawelczak, James Sewart, James Price, Amaurys Avila Ibarra, Simon McIntosh-Smith, Ferad Zyulkyarov, Leonardo Bautista-Gomez, Osman Unsal

University of Bristol - Barcelona Supercomputing Center

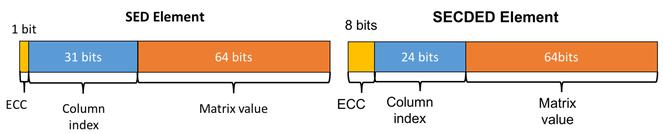
1 Introduction

Fault tolerance has been identified as one of the major challenges for exascale computing. In addition to fail-stop errors, silent data corruptions (SDCs) can perturb applications and produce incorrect results. Software-based fault tolerance mechanisms have the advantage of being capable of leveraging some of the properties of the applications to improve their reliability. In this poster, we present a fault tolerance framework that implements multiple resiliency schemes to cope with both fail-stop errors and data corruption. Our techniques are tested with two real scientific applications: BUDE, a molecular docking engine, and TeaLeaf, a heat conduction code. Using this framework we have successfully detected and recovered from real data corruptions. We have also performed error injection experiments, which clearly demonstrated the efficacy of our framework.

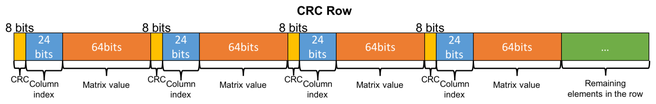
3 Fault Tolerance in TeaLeaf



- The sparse matrix is stored in Compressed Sparse Row (CSR) format.
- An earlier study within the Mont-Blanc2 project revealed that the top 8-bits from the 32-bit column index are typically unused (most matrices are smaller than 2^{24}) [1][2].
- Our software-level ECC and CRC implementations make use of these unused bits to store the error detection/correction data. This means that no extra memory is required to protect the critical data.



Either 1-bit (SED) or 8-bits (SECDED) are used for ECC data.

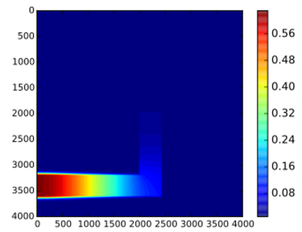


The first four elements from each matrix row store the CRC32 information - 8-bits per element.

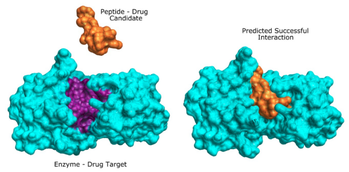
- The CRC integrity check leveraged instruction-level intrinsics where available (x86, ARMv8) and uses a software implementation as a fallback (ARMv7).
- Data integrity mechanisms were implemented inside the Conjugate Gradient solver.
- Successful recovery from real memory errors was observed on the Mont-Blanc 2 Prototype through using OmpSs tasks [3] with the Nanos runtime [4] task checkpointing [5][6].

2 Applications

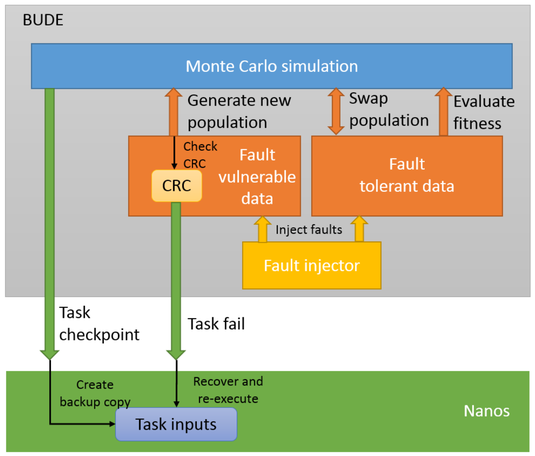
TeaLeaf [7] is a heat diffusion application, dominated by a sparse iterative CG solver on a structured grid; it is one of the Mantevo benchmarks [8].



BUDE [9] is a molecular docking application, which combines N-Body molecular mechanics with Monte Carlo search.



4 Fault Tolerance in BUDE



- The data describing the position, orientation and fitness of candidate drug positions was determined to be intrinsically fault tolerant, as the Monte Carlo-based genetic algorithm already perturbs this data to produce a subsequent generation of potential candidate drug positions.
- This fault tolerant data approaches 100% of the program's memory space.
- CRC was employed to protect critical data that was not fault tolerant, namely evolution parameters, program code segment, and all read-only pages.
- ABFT measures were implemented to defend against specific bit-flip cases:
  - A floating-point special value filter prevents error propagation in the rare case that an error resulted in a NaN or Inf.
  - The resulting "best" candidate drug positions were double checked to further reduce the possibility of incorrect results being returned.
- Memory size protected:
  - Only 16KB of code and 19KB of critical data needed protecting from errors, the rest was naturally fault tolerant data.
  - As only this critical data and code needed checkpointing, we reduced the size of the required checkpoints by over 98% compared to full-program checkpointing.
- Simulated memory errors in the critical data regions successfully triggered fault recovery in the Nanos task checkpointing framework.

5 Performance Results

- The overheads of task restarts is low as the Nanos framework stores the recovery data in RAM.
- In BUDE:
  - The overhead of the Nanos task checkpointing scheme does not increase with the population size, since the population data itself is naturally fault tolerant and thus is not checkpointed.
  - For the typical workload, checkpointing overheads were 7% for ARMv7, <1% for ARMv8, with variability amongst x86 machines.
  - Software CRC and defensive coding schemes measured negligible (unobservable) overheads.
- In TeaLeaf:
  - The Single Error Detect (SED) scheme experienced 0-1% overhead on x86 and 14% overhead on ARMv8.
  - The Single Error Correct and Double Error Detect (SECDED) schemes incurred a 72% overhead on x86 and 150% overhead on ARMv8.
  - The CRC scheme resulted in 17% overhead on x86 and 25% overhead on ARMv8. Note that this scheme would detect an arbitrary number of bit errors in the protected data, compared to the 1 or 2 bit errors that can be detected and/or corrected in the other two schemes.

6 Conclusions

- We have demonstrated an efficient task-based checkpoint-restart scheme for fail-stop errors and task-level recovery using the Nanos framework.
- Our solution combines a resilient runtime with ABFT techniques, including software ECC and CRC methods, to deal with SDCs.
- These techniques cause negligible overheads for most of the ABFT-based SDC detection mechanisms in both BUDE and TeaLeaf.
- Nanos' task recovery is triggered by trapping program exceptions, enabling it to handle affected memory regions without crashing the application.

7 References

[1] Rob Hunt and Simon McIntosh-Smith.
Exploiting spatial information in datasets to enable fault tolerant sparse matrix solvers.
In *Proceedings of IEEE Cluster 2015*, volume 2015-October, pages 543–551, 9 2015.

[2] Simon McIntosh-Smith, Rob Hunt, James Price, and Alex Vesztrocy.
Application-Based Fault Tolerance Techniques for Sparse Matrix Solvers.
International Journal of High Performance Computing Applications, 2016.

[3] Alejandro Duran, Eduard Ayguadé, Rosa M Badia, Jesús Labarta, Luis Martinell, Xavier Martorell, and Judit Planas.
OmpSs: a proposal for programming heterogeneous multi-core architectures.
Parallel Processing Letters, 21(02):173–193, 2011.

[4] Xavier Teruel, Xavier Martorell, Alejandro Duran, Roger Ferrer, and Eduard Ayguadé.
Support for OpenMP tasks in Nanos v4.
In *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, pages 256–259. IBM Corp., 2007.

[5] Omer Subasi, Javier Arias, Osman Unsal, Jesus Labarta, and Adrian Cristal.
Nanocheckpoints: A task-based asynchronous dataflow framework for efficient and scalable checkpoint/restart.
In *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 99–102. IEEE, 2015.

[6] Omer Subasi, Javier Arias, Osman Unsal, Jesus Labarta, and Adrian Cristal.
Programmer-directed partial redundancy for resilient HPC.
In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 47. ACM, 2015.

[7] Simon McIntosh-Smith, Matt Martineau, Michael Boulton, Wayne Gaudin, Paul Garrett, Wei Liu, and Richard Smedley-Stevenson.
The TeaLeaf heat diffusion benchmark from Mantevo.
https://github.com/UoB-HPC/TeaLeaf, 2016 (accessed Oct 7, 2016).

[8] Michael A Heroux, Douglas W Doerfler, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich.
Improving performance via mini-applications.
Sandia National Laboratories, Tech. Rep. SAND2009-5574, 3, 2009.

[9] Simon N McIntosh-Smith, James R Price, Richard B Sessions, and Amaurys Avila Ibarra.
High performance in-silico virtual drug screening on many-core processors.
International Journal of High Performance Computing Applications, 29(2):119–134, 5 2015.