Research Products, "Exploiting Global View for Resilience"[*]

Andrew A. Chien, University of Chicago and Argonne National Laboratory
Pavan Balaji, Argonne National Laboratory

May 2015

1 Executive Summary

GVR research products include software releases, large-scale application demonstrations (and resulting modified application code versions), and a number of new research insights as documented in highly-refereed research publications and technical reports. Notably, the technical reports provide thorough documentation of both the GVR software itself, but also how it can be applied to large-scale applications, making the GVR research results, embodied and usable in the software release, accessible to mainstream HPC applications scientists and developers.

Global View Resilience (GVR), is a library the uses versioned distributed arrays to enable computational scientists to build portable, resilient applications. Beyond process/node crashes, GVR also enables resilience to more difficult *latent* or *silent* errors. Through application-controlled error checking and recovery.

Key novel features of GVR include:

- Multi-version distributed arrays that enable complex and latent error recovery.
- Multi-stream versioning that gives the programmer control of when versions are created for an array.
- Unified error signaling and handling, customized per GVR distributed array, that enable algorithm-based fault-tolerance (ABFT) error-checking and recovery.

We have applied the GVR approach to several large applications (PCG solver, OpenMC, ddc-MD, and Chombo). Based on this experience, we evaluate the programmer effort required (code changes) to adopt version-based resilience and its performance impact.

Our results show that:

- The design and implementations of the GVR version-based resilience model, including the API for multi-stream, versioned distributed arrays and flexible cross-layer error signaling and recovery enable flexible application resilience. This effort has culminated in an open source release in October 2014 that runs efficiently and scales well on a variety of Cray, IBM, and vanilla Linux systems. Extreme performance scaling experiments with GVR and

[*]Funded by Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Award DE-SC0008603 and Contract DE-AC02-06CH11357.

OpenMC that show excellent to 16,384 nodes, and in many other applications, experiments demonstrating over 8,192 nodes. **These studies show that the GVR API can be efficiently implemented in highly-scalable applications and run on multiple HPC hardware platforms.**

- Extensive, deep study of GVR with application proxies and large applications (PCG/Trilinos, OpenMC, ddcMD, and Chombo) show that only modest code changes (<2% LOC) are required in order to achieve application-controlled, portable, version-based resilience. **These studies show that resilience can be added to scalable, highly-tuned HPC applications in a portable fashion using GVR.** Because resilience is a moderate concern today, but expected to be a critical concern at exascale, **GVR provides and all-important "gentle slope" approach.**
- Application experiments that demonstrate varied approaches to application-level resilience by using a version-based distributed array model. These include recovery from immediately detected errors, latent or silent errors (where detection is delayed), and forward-error recovery techniques. The latter two particularly benefit from GVR's multi-version and multi-stream capability that enables a broad range of novel recovery techniques. **These GVR demonstrations show that application-based error checking and correction (ABFT) can be expressed and exploited using GVR, and the resulting applications can scale well.**
- Performance studies across the same applications documenting that the cost of running with versioning code generally results in <2% runtime overhead, at versioning frequencies much higher than needed for today's error rates. **These GVR results show forward-looking application teams can begin adding GVR to their codes today with low overhead to match existing reliable NERSC, ALCF, OLCF machine. Further, these modified codes can be used to tap lower-reliabilty platforms for more science.**

In short, our results show that GVR's version-based resilience using distributed arrays is a portable, gentle-slope resilience approach that handles both immediate and latent errors. It enables flexible error reporting and recovery and thus is promising for scaling to the higher error rates expected in extreme-scale hardware. These results are documented in great detail in our papers detailed in the bibliography below; with the relevant publications listed for each topic (note that several address multiple topics and thus are listed several times).

Efficient Versioning and Exploitation of NVRAM

1. Hajime Fujita, Kamil Iskra, Pavan Balaji, and Andrew A. Chien, "Empirical Characterization of Versioning Architectures", to appear in IEEE Cluster, September 2015, Chicago. (compares hardware-assisted and software-only versioning)
2. A. Chien, P. Balaji, N. Dun, A. Fang, H. Fujita, K. Iskra, Z. Rubenstein, Z. Zheng, J. Hammond, I. Laguna, D. Richards, A. Dubey, B. van Straalen, M Hoemmen, M. Heroux, K. Teranishi, A. Siegel. Exploring Versioning for Resilience in Scientific Applications: Global-view Resilience, submitted for publication, March 2015. (Best overall project summary, GVR programming approach, programmer effort, basic versioning performance)

3. Aiman Fang and Andrew A. Chien, “How Much SSD Is Useful for Resilience in Supercomputers, in IEEE Symposium on Fault-tolerance at Extreme-Scale (FTXS), June 2015. (use of NVRAM for resilience)

4. Aiman Fang, “How Much SSD Is Useful for Resilience in Supercomputers, Master's Thesis, Department of Computer Science, University of Chicago, April 2015. (use of NVRAM for resilience)

5. A. Chien, P. Balaji, P. Beckman, N. Dun, A. Fang, H. Fujita, K. Iskra, Z. Rubenstein, Z. Zheng, R. Schreiber, J. Hammond, J. Dinan, A. Laguna, D. Richards, A. Dubey, B. van Straalen, M Hoemmen, M. Heroux, K. Teranishi, A. Siegel, and J. Tramm, “Versioned Distributed Arrays for Resilience in Scientific Applications: Global View Resilience”, in International Conference on Computational Science (ICCS 2015), Reykjavik, Iceland, June 2015.

6. Hajime Fujita, Nan Dun, Zachary Rubenstein, and Andrew A. Chien. Log-Structured Global Array for Efficient Multi-Version Snapshots, IEEE CCGrid 2015, May 2015. (flat and log-structured versioning approaches)

7. Hajime Fujita, Nan Dun, Zachary Rubenstein, and Andrew A. Chien. Log-Structured Global Array for Efficient Multi-Version Snapshots, UChicago CS Tech Report 2014-16, Nov 2014. (similar to above, more extensive results)

8. Guoming Lu, Ziming Zheng, and Andrew A. Chien, When are Multiple Checkpoints Needed?, in 3rd Workshop for Fault-tolerance at Extreme Scale (FTXS), at IEEE Conference on High Performance Distributed Computing, June 2013, New York, New York. (silent/latent errors, need for versioning approach)

9. Hajime Fujita, Robert Schreiber, Andrew A. Chien, It's Time for New Programming Models for Unreliable Hardware, to appear in ASPLOS 2013 Provocative Ideas session, March 18, 2013. (silent/latent errors, need for versioning approach)

10. Wesley Bland, Aurelien Bouteiller, Thomas Herault, Joshua Hursey, George Bosilca, and JackJ. Dongarra. An evaluation of User-Level Failure Mitigation support in MPI. Computing, 95(12):11711184, 2013. (user-level MPI recovery)

Application and API Studies

1. A. Chien, P. Balaji, N. Dun, A. Fang, H. Fujita, K. Iskra, Z. Rubenstein, Z. Zheng, J. Hammond, I. Laguna, D. Richards, A. Dubey, B. van Straalen, M Hoemmen, M. Heroux, K. Teranishi, A. Siegel. Exploring Versioning for Resilience in Scientific Applications: Global-view Resilience, submitted for publication, March 2015. (Best overall project summary, GVR programming approach, programmer effort, basic versioning performance)

2. A. Chien, P. Balaji, P. Beckman, N. Dun, A. Fang, H. Fujita, K. Iskra, Z. Rubenstein, Z. Zheng, R. Schreiber, J. Hammond, J. Dinan, A. Laguna, D. Richards, A. Dubey, B. van Straalen, M Hoemmen, M. Heroux, K. Teranishi, A. Siegel, and J. Tramm, “Versioned Distributed Arrays for Resilience in Scientific Applications: Global View Resilience”, in International Conference on Computational Science (ICCS 2015), Reykjavik, Iceland, June 2015.

3. The GVR Team, Global View Resilience (GVR) Documentation, Release 1.0, University of Chicago, Computer Science Technical Report 2014-10. (The API reference)

4. Nan Dun, Hajime Fujita, John R. Tramm, Andrew A. Chien, Andrew R. Siegel, Data Decomposition in Monte Carlo Neutron Transport Simulations using Global View Arrays, International Journal of High Performance Computing Applications, March 2015. (OpenMC study, scaling to 16K ranks, forward recovery)

5. Nan Dun, Hajime Fujita, John Tramm, Andrew A. Chien, and Andrew R. Siegel. Data Decomposition in Monte Carlo Particle Transport Simulations using Global View Arrays, UChicago CS Tech Report 2014-09 May 2014. (similar to above, more extensive results)

6. Hajime Fujita, Nan Dun, Aiman Fang, Zachary A. Rubenstein, Ziming Zheng, Kamil Iskra, Jeff Hammond, Anshu Dubey, Pavan Balaji, Andrew A. Chien: Using Global View Resilience (GVR) to add Resilience to Exascale Applications, SC14, Nov 2014 (Best Poster Finalist)

7. Aiman Fang and Andrew A. Chien, "Applying GVR to Molecular Dynamics: Enabling Resilience for Scientific Computations", Tech Report, University of Chicago, Dept of Computer Science, CS-TR-2014-04, April 2014. (detailed GVR study with ddcMD)

8. Ziming Zheng, Andrew A. Chien, Keita Teranishi, "Fault Tolerance in an Inner-Outer Solver: a GVR-enabled Case Study", in Proceedings of VECPAR 2014, July 2014, Eugene, Oregon. Proceedings available from Springer-Verlag Lecture Notes in Computer Science. (detailed GVR study with Trilinos and GMRES)

Algorithm-based Fault Tolerance Using GVR

1. Nan Dun, Hajime Fujita, John R. Tramm, Andrew A. Chien, Andrew R. Siegel, Data Decomposition in Monte Carlo Neutron Transport Simulations using Global View Arrays, International Journal of High Performance Computing Applications, March 2015. (OpenMC study, scaling to 16K ranks, forward recovery)

2. Ziming Zheng, Zachary Rubenstein, and Andrew A. Chien, GVR-Enabled Trilinos: An Outside-In Approach for Resilient Computing, in the SIAM Conference on Parallel Processing, February 2014, Portland Oregon. (detailed GVR study with Trilinos and GMRES)

3. Ziming Zheng, Andrew A. Chien, Keita Teranishi, "Fault Tolerance in an Inner-Outer Solver: a GVR-enabled Case Study", in Proceedings of VECPAR 2014, July 2014, Eugene, Oregon. Proceedings available from Springer-Verlag Lecture Notes in Computer Science. (detailed GVR study with Trilinos and GMRES)

4. Ziming Zheng, Andrew A. Chien, Mark Hoemmen, Keita Teranishi, "Fault Tolerance in an Inner-Outer Solver: a GVR-enabled Case Study", available as Technical Report from University of Chicago Department of Computer Science, CS-TR-2014-01, January 2014. (similar to above, more extensive results)

5. Z. Rubenstein, "Error Checking and Snapshot-based Recovery in Preconditioned Conjugate Gradient Solver", Masters Thesis, University of Chicago, Department of Computer Science, March 2014. (error injection study for PCG/Trilinos)

6. Z. Rubenstein, J. Dinan, H. Fujita, Z. Zheng, A. Chien, "Error Checking and Snapshot-Based Recovery in a Preconditioned Conjugate Gradient Solver", University of Chicago, Department of Computer Science Technical Report 2013-11, December 2013. (extensive study of error injection study for PCG/Trilinos)

2 GVR Software Releases

The GVR software, Version 1.0.0 was released under BSD licensing on October 4, 2014. Prior to that, pre-release versions were shared with applications teams (Trilinos, ddcMD, OpenMC, Chombo, etc.). Most recently, we have made GVR Version 1.0.1 available, and this version is also in deployment at NERSC to make it generally available to the DOE community.

2.1 GVR Software Release - Primary

GVR (Global View Resilience) is a user-level library that enables **portable, efficient, application-controlled resilience**. The primary target of GVR is HPC applications that require both extreme scalability and performance as well as resilience. GVR's key approaches include independent versioning of application arrays, efficient partial or whole restoration, open resilience to maximize the number of errors that can be handled (minimize fail-stop occurrences). Application knowledge can be exploited to control overhead, maximize error coverage, and maximize recoverable errors.

The GVR Version 1.0.x includes with following features, and **runs on IBM, Cray, and general Linux platforms.**

- Portable application-controlled resilience and recovery with incremental code change
- Versioned distributed arrays with global naming (a portable abstraction)
- Reliable storage of the versioned arrays in memory, local disk/SSD, or global file system
- Whole version navigation and efficient restoration
- Partial version efficient restoration (incremental "materialization")
- Independent array versioning (each at its own pace)
- Open Resilience framework to maximize cross-layer error handling
- application-defined error handling
- unified application and system error descriptors
- attribute based composition for easy extensibility at application, operating system, and hardware levels
- C native APIs and Fortran bindings

Current release requires only:

- an MPI library which is compatible with MPI-3 standard.

- Standard "autotools" preparation
- Requires no root privilege
- Runs on several platforms including x86-64 Linux cluster, Cray XC30 and IBM Blue Gene/Q

User documentation is available as "Global View Resilience (GVR) Documentation, Release 1.0", University of Chicago, Computer Science Technical Report 2014-10. The GVR team gratefully acknowledges the incorporation of elements of the Scalable Checkpoint-restart (SCR) system into the GVR system. The GVR source release also includes two subcomponents – not required for GVR use, but useful with it – LRDS and MPI with ULFM.

Local Reliable Data Store - LRDS An important element of the GVR project research is the Local Reliable data store that implements a variety of efficient change tracking and version creation mechanisms. The LRDS software was released as part of the GVR system release, and exploits a wide range of different hardware support situations, including:

- hardware TLB "dirty-bit" tracking
- virtual memory, page-protection based tracking
- classic, incremental checkpointing
- novel, decremental checkpointing

These approaches are documented in "Empirical Characterization of Versioning Architectures", to appear in IEEE Cluster, September 2015.

MPI with User-level Fault Mitigation - ULFM Several GVR application experiments (particularly those with Chombo), included recovery at the MPI-level, using the ULFM system. The corresponding ULFM software was also released as part of the GVR software release.

2.2 GVR System Documentation

- The GVR Team, Global View Resilience, API Documentation R0.8.1-rc0, University of Chicago, Computer Science Technical Report 2014-05.
- The GVR Team, How Applications Use GVR: Use Cases, University of Chicago, Computer Science Technical Report 2014-06.

3 GVR Team Acknowledgements

GVR Team Members include: Hajime Fujita, Zachary Rubenstein, Aiman Fang, Nan Dun, Yan Liu (UChicago), Pavan Balaji, Pete Beckman, Kamil Iskra, (ANL), and application partners Andrew Siegel (Argonne/CESAR), Ziming Zheng (UC/Vertica), James Dinan (Intel), Guoming Lu (UESTC), Robert Schreiber (HP), Jeff Hammond (Argonne/ALCF/NWChem-¿Intel), Mike Heroux, Mark Hoemmen, Keita Teranishi (Sandia), Dave Richards (LLNL), Anshu Dubey, Brian Van Straalen (LBNL)

We gratefully acknowledge the Scalable Checkpoint-restart team (SCR Team) at Livermore as we have incorporated some elements fo their SCR software in the GVR system.

This work was supported by the US Department of Energy, Office of Science, Advanced Scientific Computing Research DE-SC0008603 and DE-AC02-06CH11357.