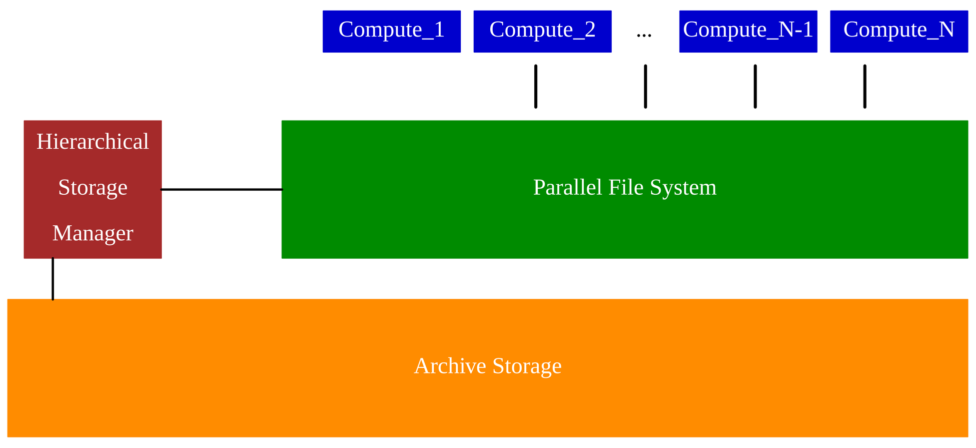
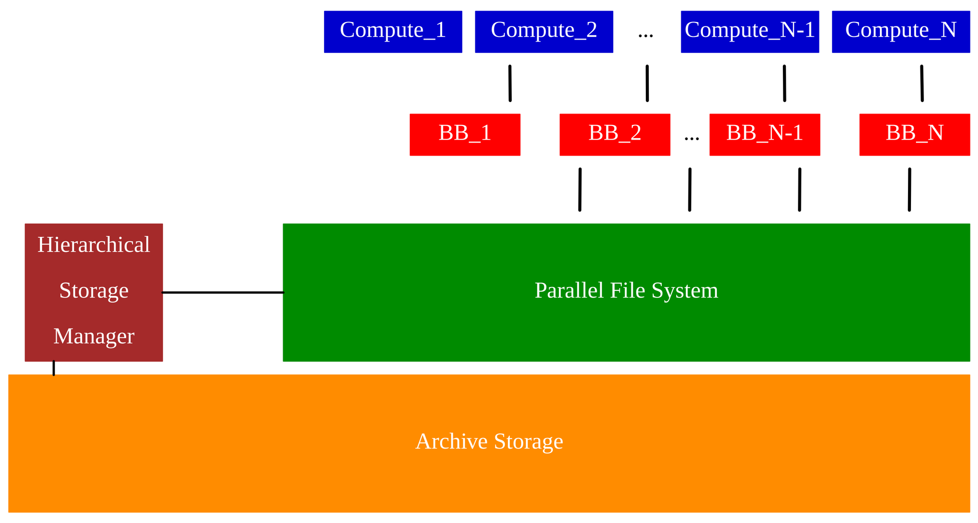
Campaign Storage: Erasure Coding With GPUs

Walker Haddock[1], Matthew L. Curry[2], Purushotham V. Bangalore[1], Anthony Skjellum[3]

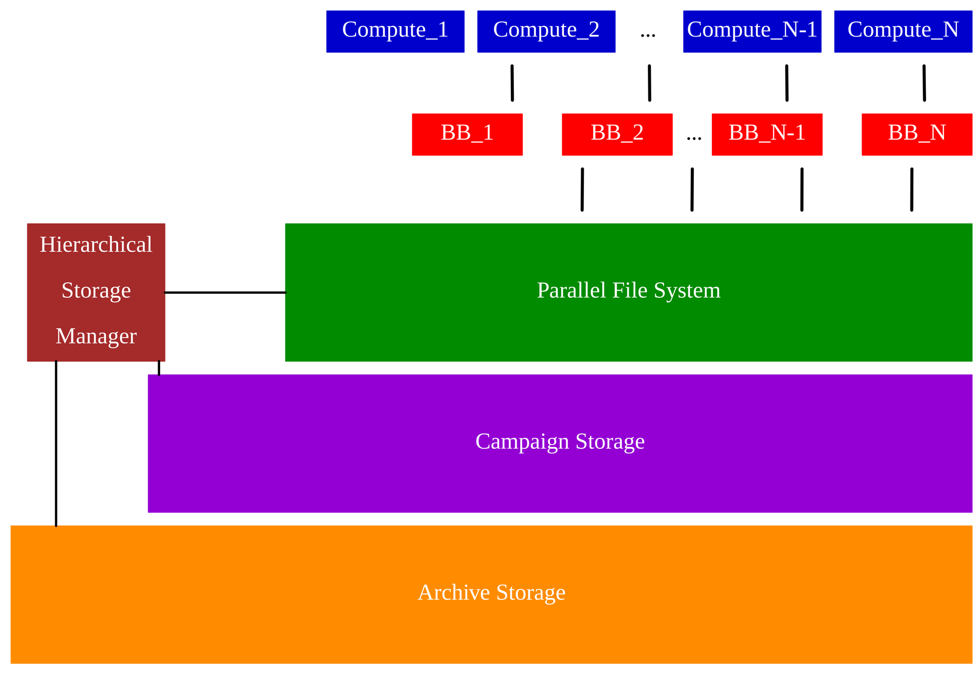
Evolution of HPC Storage Implementations



(a) Traditional Parallel File System. File system bandwidth requirements were met by sufficient spindles to meet capacity requirements.



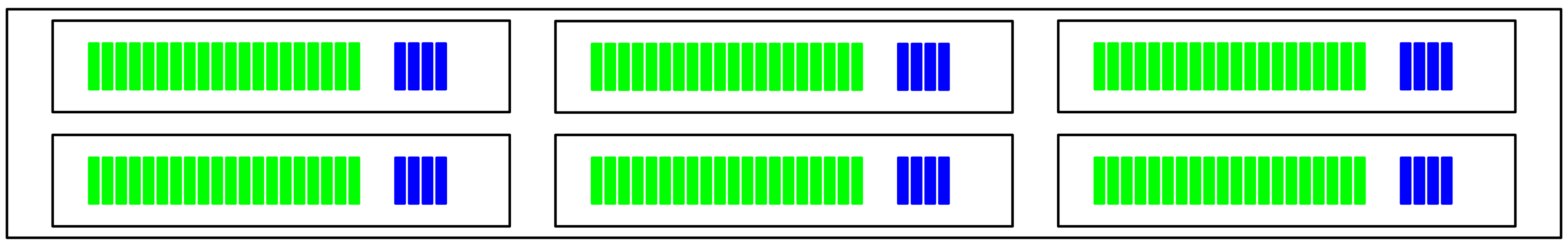
(b) Burst Buffers with Parallel File System. In order to meet the higher bandwidth requirements of Exascale computing, use more expensive Solid State Disks for higher bandwidth and then move data to/from PFS (2).



(c) Adding Campaign Storage to (b). Because of greater data storage requirements in Exascale computing, use cloud storage technologies with erasure coding to lower capacity requirements of PFS and lower costs (4).

Lazy Erasure Repair

- Damaged objects can be read with acceptable performance – does not harm user experience!
- Changing the stripe from 20+4 to 120+24 decreases the urgency to rebuild while storage overhead is constant.



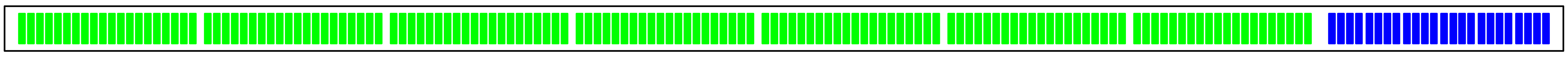
(a)

a) Trinity Erasure Coding Configuration (5)

- k=20
- m=4
- 24 disks per stripe set
- 6 stripe sets use 144 disks
- total storage available: 6 * k * disk size = 120 * disk size

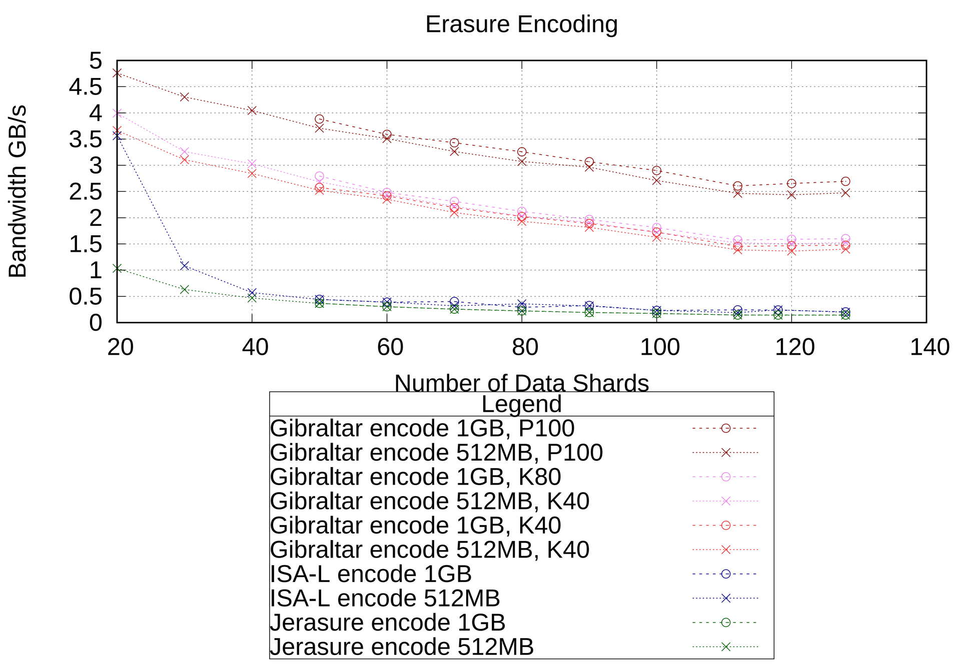
b) Erasure Coding Configuration with Gibraltar (3)

- k=120
- m=24
- 144 disks per stripe set
- one stripe set uses 144 disks
- total storage available: k * disk size = 120 * disk size

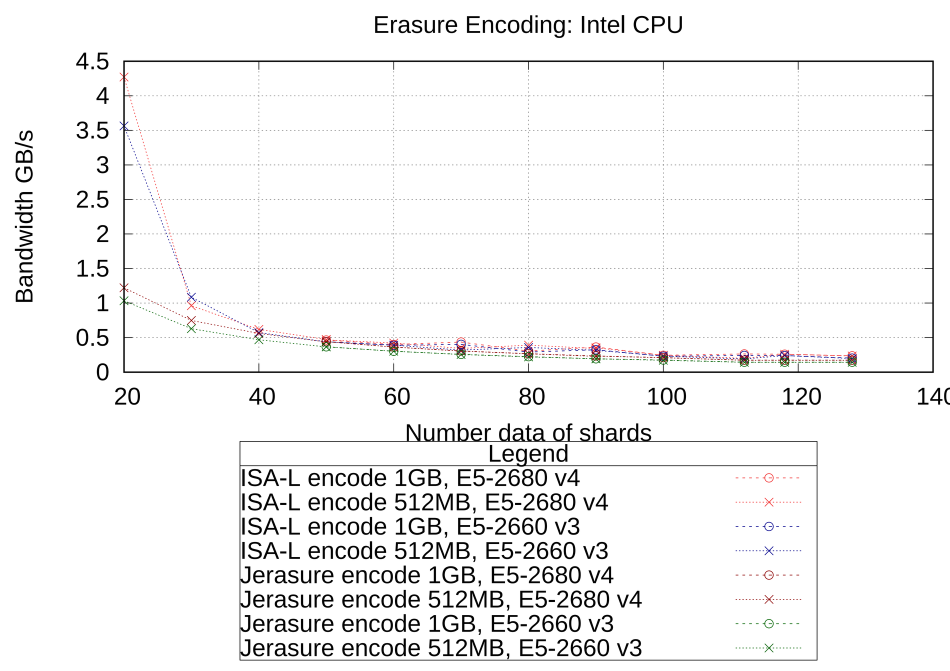


(b)

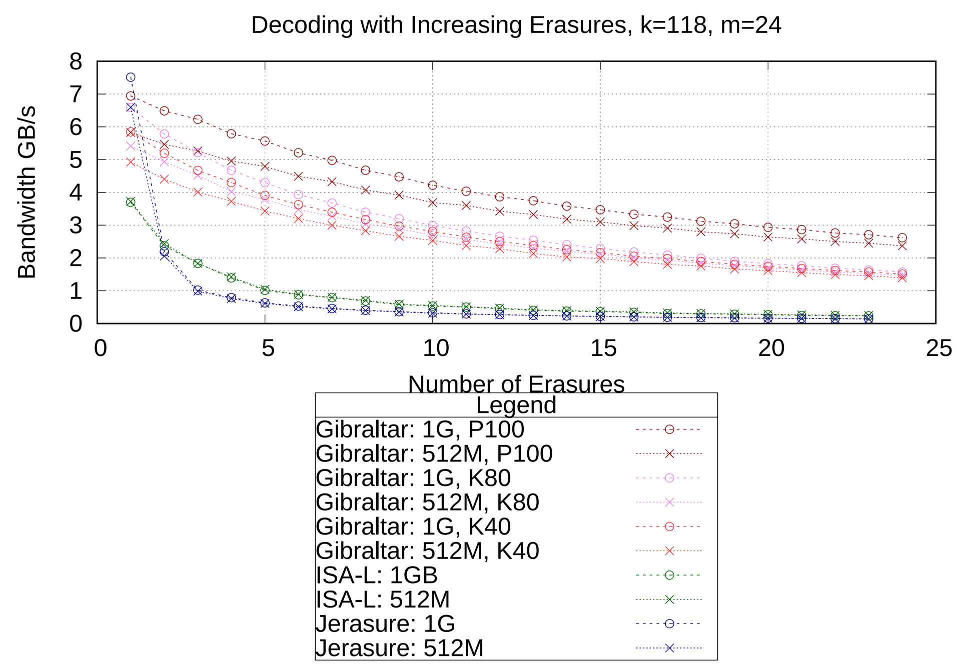
Results for GPU Assisted Erasure Coding



Comparison between Gibraltar (0; 1) GPU erasure coding on the NVIDIA K40 GPU with the Intel ISA-L (10) and Jerasure (7; 6; 8; 9) erasure encoding libraries. Gibraltar produces 1.5 GB/s at 128 data shards with 24 parity shards.

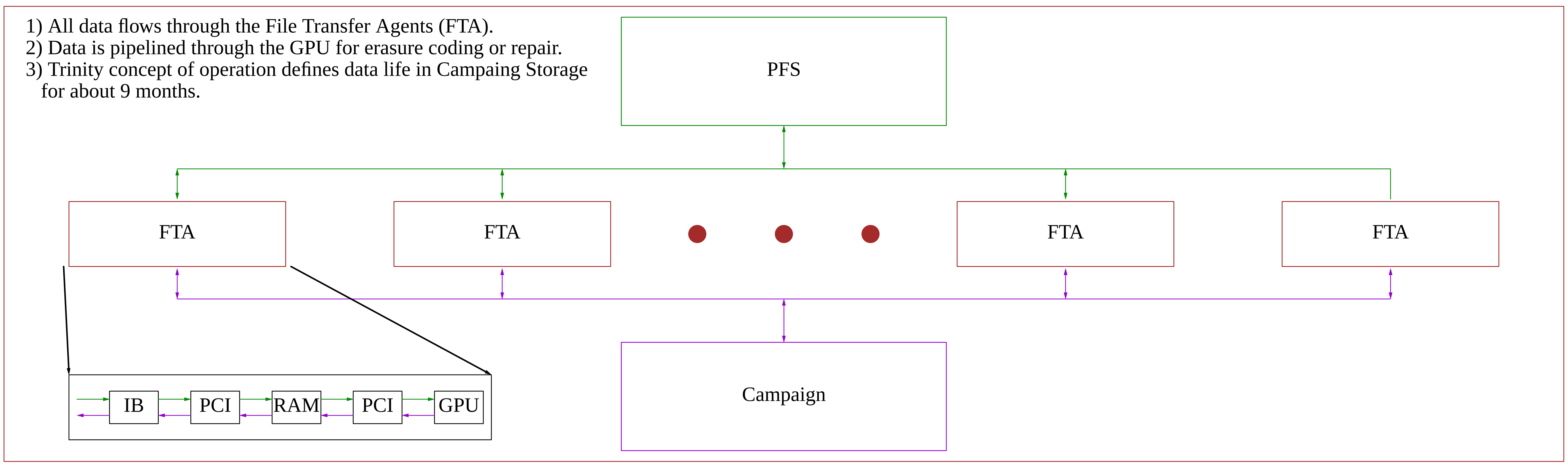


Benchmarks comparing two different generations of Intel CPUs erasure encoding with the Jerasure and ISA-L libraries. (These are all a single core of the CPU.)



Recovery rates while increasing erasures to 24. Gibraltar produces over 1.5 GB/s at 20 erasures while ISA-L has dropped to 267 MB/s at 20 erasures.

Erasure Coding Services on File Transfer Appliances



Bibliography

[0] Curry, M. L., Skjellum, A., Lee Ward, H., and Brightwell, R. (2008). Accelerating Reed-Solomon coding in RAID systems with GPUs. In *Proceedings of the 2008 IEEE international parallel & distributed processing symposium*, pages 1–6, Miami, FL, USA. IEEE.

[1] Curry, M. L., Skjellum, A., Lee Ward, H., and Brightwell, R. (2011). Gibraltar: A Reed-Solomon coding library for storage applications on programmable graphics processors. *Concurrency and Computation: Practice and Experience*, 23(18):2477–2495.

[2] Gary Grider (2016). HPC Storage and IO Trends and Workflows.

[3] Haddock, Walker, Curry, Matthew L., Bangalore, Purushotham V. and Skjellum, Anthony (2017). GPU Erasure Coding for Campaign Storage HPC-IODC ISC'17.

[4] Lamb, K. (2015). Trinity Campaign Storage and Usage Model.

[5] Morton, D. (2017). Trinity: DataManagement Scheme and Performance. American Institute of Aeronautics and Astronautics.

[6] Plank, J. S. (1997). A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems. *Software â Practice & Experience*, 27(9):995–1012.

[7] James S. Plank, Scott Simmerman, and Catherine D. Schuman (2008). Jerasure: A Library in C/C++ Facilitating Erasure Coding for Storage Applications. Technical Report Technical Report CS-08-627, University of Tennessee, Knoxville, TN 37996.

[8] Plank, J. S., Blaum, M., and Hafner, J. L. (2013). SD codes: erasure codes designed for how storage systems really fail. In *FAST*, pages 95–104, San Jose, CA, USA.

[9] Plank, J. S. and Thomason, M. G. (2004). A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In *Dependable Systems and Networks, 2004 International Conference on*, pages 115–124. IEEE.

[10] Tucker, G. (2014). ISA-L open source v2.14 API doc.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants Nos. OAC-1541310, CNS-0821497 and CNS-1229282. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

This material is based upon work supported by Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

[1]Department of Computer Science
University of Alabama at Birmingham
Alabama, US

[2]Center for Computing Research
Sandia National Laboratories
New Mexico, US

[3]SimCenter,
University of Tennessee at Chattanooga
Tennessee, US