An Evaluation of Difference and Threshold Techniques for Efficient Checkpoints

Sean Hogan
University of Chicago
seanhogan@uchicago.edu

Jeff R. Hammond
Argonne National Laboratory
jhammond@anl.gov

Andrew A. Chien
University of Chicago and
Argonne National Laboratory
achien@cs.uchicago.edu

***Abstract*—To ensure reliability, long-running and large-scale computations have long used *checkpoint-and-restart* techniques to preserve computational progress in case of soft or hard failures. These techniques can incur significant overhead, consuming as much as 15% of an application's resources for the US DOE's leadership-class systems, and these overheads are projected to grow in exascale systems which are likely to have lower IO to compute ratios and higher failure rates.**

We explore the use of differenced checkpoint and cutoff techniques to increase the effectiveness of Lempel-Ziv (gzip), and thereby reduce the size of checkpoints. We apply these techniques to several types of scientific checkpoint data from NWChem, a widely-used computational chemistry code. Our results show that while standard compression techniques (and even those customized for floating point data) yield modest compression ratios ($\approx$ 1.2), differenced checkpoints and cutoffs are dramatically more successful, improving compression ratios by 50% to 1.55 to 3.15 for a variety of checkpoint data. If cutoffs in the differenced checkpoints are incorporated, these compression ratios can be increased further with cutoff of 10^{-7} yielding dramatic improvement in compression ratios greater than 100. These results suggest further exploration of these approaches are promising to reduce checkpoint (and resilience) overhead.

***Keywords*-NWChem, data compression, checkpoint, lempel-ziv, fpc, resilience, fault-tolerance**

I. INTRODUCTION

A range of daunting technology challenges are projected for exascale systems including extreme energy efficiency ($20 \cdot 10^{-12}$ joules (picojoules) / operation), extreme levels of parallelism (10^9-fold), and high error rates ($\approx 2 \cdot 10^9$ FITS/billion hours or 30 mins MTTI). These changes represent an extraordinary set of challenges for programming models and tools for exascale systems. The increased failure rate threatens the utility of exascale systems, creating tremendous pressure to improve the dominant model, checkpoint-and-restart as it faces critical challenges (growing size and frequency, poor I/O scaling, and synchronization cost). These challenges are well-documented [15].

To ensure reliability, long-running and large-scale computations have long used *checkpoint-and-restart* techniques to preserve computational progress in case of soft or hard failures. These techniques can incur significant overhead, consuming as much as 15% of an application's resources for the US DOE's leadership-class systems [32], and these overheads are projected to grow in Exascale systems which are likely to have lower IO to compute ratios and higher failure rates. These systems follow in the model of current generation systems [3], [23], that relegate checkpoints to target a parallel filesystem, hosted on nodes external to the compute array, and typically with a small fraction of the bandwidth. As a result, techniques to increase the efficiency of checkpointing are of great research and practical interest.

Data compression seeks to find compact representations for information, and in doing so to remove redundancy in encoding, making the data easier to transmit and store. While compression techniques have achieved dramatic compression results in some areas: text by 10-fold [34], images by 100-fold with JPEG, and video by 1000-fold [30]. In contrast, for scientific data, only modest progress has been achieved. Floating point numbers make efficient use of the available bits, producing much lower compression rates.

We explore two specific compression algorithms and two new techniques, differenced checkpoints and thresholding, which exploit application structure at two levels. These techniques, differenced checkpoint and cutoff, are used to increase the effectiveness of Lempel-Ziv (gzip), and thereby reducing checkpoint size. The differenced techniques exploit the observation that successive checkpoints often differ modestly, so differences between them may compress more effectively. The underlying assumption of course is that multiple checkpoints are useful (as they often are in uncoordinated checkpointing [7] or aggressive resilience techniques that exploit multiple snapshots). Cutoff techniques exploit the observation that checkpoint differences less than the computations numerical error are for all purposes equal to zero, and represent them accordingly. We measure the compressibility of multidimensional array objects used in quantum chemistry simulations, specifically, the coupled-cluster method implemented in NWChem [6], a well-known computational chemistry code. Our results support the following conclusions:

- standard compression techniques and even those customized for floating point data yield modest compression ratios ($\approx$ 1.2) for a range of scientific checkpoint data,
- a novel approach to capturing checkpoint data, differenced checkpoints that represent the "delta" between two checkpoints,
- a refinement of differenced checkpoints, differenced

with cutoffs, that further exploits application knowledge of precision to simplify the representation of deltas between two checkpoints

- evaluation of differenced checkpoints on NWChem scientific checkpoint data, which shows that they increase the effectiveness of compression as the computation converges, increasing compression ratios improvements by 50%, and achieving ratios ranging from 1.55 to 3.15, and
- evaluation of differenced checkpoints with cutoffs on NWChem scientific checkpoint data, which shows that adding cutoffs can further increase compression ratios with cutoff of 10^{-7}, corresponding to single-precision FP yielding greater than 100-fold compression ratios.

These results suggest further exploration of these approaches are promising to reduce checkpoint (and resilience) overhead.

The rest of the paper is organized as follows. Section II describes relevant compression and NWChem background. In Section III we present our experiments and results. Section IV discusses the work in context. Section V summarizes the results, and points out directions for future work.

II. BACKGROUND

A. Compression

For scientific data, compression is difficult. Floating point numbers make efficient use of the available bits, producing much lower compression rates. Furthermore, as we are exploring the compression of checkpoint data, the compression is often and online, as a integral part of the simulation. Both the speed (execution time) and the quality (compression ratio) matter, while the decompression effort is less important. This contrasts sharply with compression for media distribution, where arbitrarily large amounts of computation can be used to compress data, but decompression must happend subject to real-time constraints. We explore two specific compression algorithms and two techniques, differenced checkpoints and thresholding, which exploit application structure at two levels.

1) Lempel-Ziv: The Lempel-Ziv '77 algorithm (LZ77) is a well-known method of leveraging common prefixes in sets of data in order to gain compression. Much work has been done documenting LZ77, and two compression softwares, the standalone *gzip* and its library counterpart, *zlib*, use LZ77 and Huffman encoding in order to perform general purpose, lossless compression. We use both in conjunction with our study on compression of differenced checkpoints.

2) FPC: FPC [5] is a predictor-based compression algorithm specifically designed to be effective in compressing scientific data. As a result, it is among the most effective techniques for compressing scientific floating point data. Because its predictor-based, it is able to achieve very high data rates (low computational overhead).

3) Differenced Checkpoints: For most experiments, we used the differenced checkpoints method, done by our delta compression program. The program takes two checkpoints with 64-bit floating point data, at any timestep - but in these experiments, one at time t (the subtrahend) and another at time $t+1$ (the minuend). The checkpoints are read value-by-value, and a "delta" is calculated, based on the magnitude of their difference. The minuend's exponent is stored in the delta, and the mantissa is shifted right based on how close the values are. We operate under two assumptions: first, some loss is acceptable in these data sets at the benefit of compression, and second, with two close minuend and subtrahend values, we need less precision to get a close approximation of the minuend value when recovering from the subtrahend, which is why we shift the mantissa more when two values are near.

4) Differenced with Threshold: The used of differenced checkpoints presents an opportunity to exploit the fact that the magnitude of differences may be small. We explore a threshold (or cutoff) where small differences can be rounded to zero. In general, the choice of cutoff depends on application semantics, so we explore two sample cutoffs (single precision - 10^{-7} and a lower cutoff 10^{-13}). The single precision cutoff is used for two reasons: first, if the iterations are far from convergence, truncating the checkpoint has almost no impact on the ability to restart but improves I/O throughput substantially, and second, it has recently been demonstrated that single-precision can be exploited for performance in multiple ways within CC [33], [16], [10]. We believe that truncating at twice the precision of the overall solution, as measured by the residual norm, is potentially an effective compromise between checkpointing time and restart accuracy.

5) NWChem: NWChem [6] is a widely used computational chemistry package designed to run on distributed-memory clusters and supercomputers. It runs on a wide variety of supercomputer architectures, including Infiniband clusters, Cray XT and XE systems, and Blue Gene/P.

A wide variety of simulation capability is implemented in NWChem, including density-functional theory (DFT), many-body methods such as coupled-cluster theory (CC), and hybrid methods such as QM/MM. The implementation of the coupled-cluster methods in NWChem is of particular interest from a computational perspective because of the scalability [17] and performance on terascale [27] and petascale systems [2], [19], [18]. In this paper, we focus on the TCE module [14], which implements dozens of different many-body methods using automatic code generation [1]. Of all the iterative CC methods in NWChem, the CCSD method [29] is by far the most common, as it is the precursor to the "gold-standard" CCSD(T) method as well as widely used excited-state (i.e. EOM) methods [20].

The systems under investigation in this paper are water clusters, which represent an important application area, are well-behaved from a numerical perspective, and have been used in previous computational studies [13]. We consider two relatively small clusters (*w3* and *w4*) with a relatively large basis set (cc-pVTZ) and one larger clusters (*w13*) with a small basis set (3-21G) for expediency, but there is no mathematical or physical reason why the compression rations observed for these systems would not be observed for other systems, as the fundamental principle exploited for compression is the smooth convergence of the CC iterations, which has been observed by one of us across an incredibly diverse range of chemical systems, basis sets and CC trunctations (i.e. beyond CCSD, e.g. CCSDT and CCSDTQ).

Unlike many other iterative simulations, CC requires a relatively small number of iterations (usually 10-40) that frequently require many minutes each, and occasionally hours[1]. As such, a fault that occurs in the middle of a simulation is quite costly to the user. Regardless of the scale of the job, the loss of both human and computer time to faults must be minimized, and checkpointing is the most common, if not exclusive means to achieving this.

The compression of data-structures used in CC is relevant not only to fault-tolerance, but to other situations when one wishes to trade computation for storage or bandwidth. The common description of post-petascale computation (i.e. the exascale roadmap) includes a continuous increase in computational capability, but not a proportional increase in memory or bandwidth, both of which are limiting from both a power and cost perspective. If CC objects can be compressed to a useful degree, it opens the doors to compression of messages for communication as well as for storage in non-volatile memory, for example.

6) The coupled-cluster method: Coupled-cluster [4] is a variant of quantum many-body theory based upon a nonlinear parameterization of the wavefunction, $|\Psi\rangle = \exp(\hat{T})|\Phi\rangle$, where Ψ is the many-body CC wavefunction, $\hat{T}$ is an operator that excites particles from occupied to virtual states ($\hat{T}_i$ excites i particles), and Φ is a single-determinent reference wavefunction, usually defined by the converged Hartree-Fock orbitals. The CCSD level of theory considered here is defined by $\hat{T} = \hat{T}_1 + \hat{T}_2$, meaning that it includes all singly- and doubly- excited terms in the cluster operator, $\hat{T}$.

The coupled-cluster equations are solved by projection, rather than variational minimization, common for other quantum chemistry methods. Complete description is not possible here but NWChem computes the residuals from a few dozen tensor contractions between 2- and 4-dimensional arrays [8]. The solver Jacobi iteration with a diagonal preconditioner and DIIS [28] convergence acceleration.

Due to their nonlinearity, the CC iterations do not always converge, although for many chemical systems, they converge in less than 50 iterations when DIIS is employed. When the CC iterations do converge, however, they usually do so monotonically, such that the residual norm decreases each iteration, which indicates that the difference between the amplitudes at each iteration is also decreasing. Near convergence, the amplitudes will change by a relatively small amount and potentially at only a fraction of the array elements, which is the basis for the impressive compression ratios observed.

III. EXPERIMENTS

A. Platforms and Measurements

We ran the tests on a single node with two Intel Xeon E5620 4-core processors. We used a few pieces of software: FPC, *gzip*, and our Global Arrays- [22] (GA) and MPI-based differenced checkpoint/delta computing program. **CPU Time** includes measured cpu time, while Total Runtime includes file input/output overheads, which in the case of single checkpoint compression are a single checkpoint read and then write. In the case of differenced checkpoints, involves two checkpoint reads and then a write. Warm start conditions are emulated by doing repeated measurements and discarding measurements from the first run. **Throughputs** are computed by dividing processed data size by compute time, expcluding other overheads such as input/output. These measurements characterize the cost of the data compression and other processing, and are reported in megabytes/second. **Compression Ratio** is computed as uncompressed size divided by compressed data size. Timing measurements were taken using a variety of system-level routines, and specifically using system routines to measure user CPU time for the compression and other data processing, and wall-clock time to capture overall costs that include system and IO times.

Our experiments use NWChem checkpoints (64-bit floating point data) from three difference configurations: *w3_ccsd_cc-pvtz.t2* (45 MB, 21 timesteps), *w4_ccsd_cc-pvtz.t2* (138 MB, 21 timesteps). *w13_ccsd_3-21G.t2* (328 MB, 21 timesteps), and

B. Experiments

FPC and LZ Compression Our first experiment, show in Table I, explores the use of Lempel-Ziv and FPC compression to reduce the size of checkpoints, present four exemplars of the 21 checkpoints, as the compression performance varies little. *gzip* was used with default settings and FPC with table size 15. The LZ77 compression ratio is low, ranging from 1.12 to 1.21 for w3 and w13, with w4 doing better at 2.19. FPC is much faster– as expected given the design for rapid compression, but the compression ratio is even worse than for LZ77. Given these results, we focus on LZ77 for the remainder of this study.

Differenced Checkpoints We used differenced checkpoints implemented in our delta compression program; in

---

[1] This depends on the system policy. Supercomputer users try to use more nodes to reduce the iteration time to a few minutes, while cluster users have no choice but to tolerate multi-day simulations.

	w3				w4				w13			
	gzip		fpc		gzip		fpc		gzip		fpc	
	CR	TP	CR	TP	CR	TP	CR	TP	CR	TP	CR	TP
1	1.21	17.78	1.02	336.15	2.23	19.37	1.08	328.02	1.12	17.66	1.01	297.14
6	1.21	17.83	1.02	320.95	2.19	18.82	1.07	337.65	1.12	17.41	1.01	322.78
11	1.21	17.85	1.02	324.53	2.19	18.62	1.07	311.81	1.12	17.89	1.01	312.15
16	1.21	18.04	1.02	285.53	2.19	19.03	1.07	307.27	1.12	17.77	1.01	305.78
21	1.21	17.90	1.02	297.15	2.19	19.04	1.07	316.37	1.12	17.50	1.01	301.10

Table I
COMPRESSION RATIOS (CR) AND THROUGHPUTS (TP) OF GZIP (*LZ*) AND FPC COMPRESSION SCHEMES

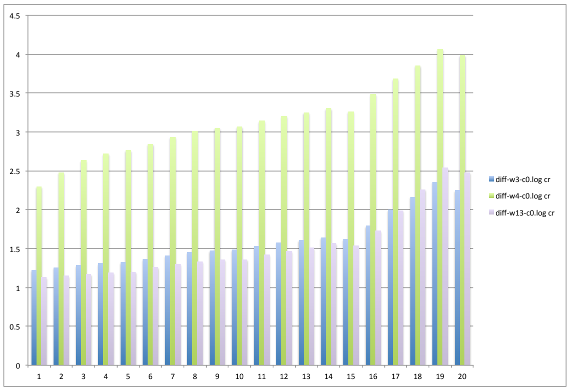

Figure 1. Compression Ratios for Differenced Checkpoints

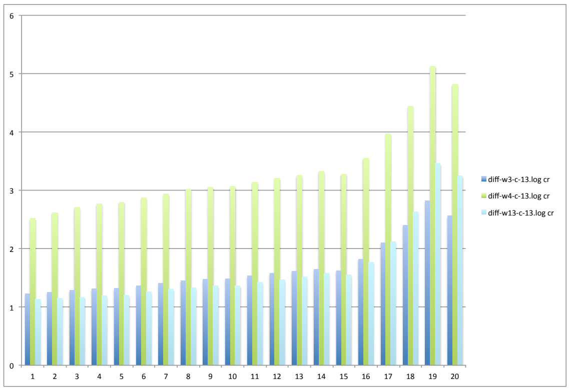

Figure 2. Compression Ratios for Diff Checkpoints, 10^{-13} threshold

these experiments (in Figures 1, 2, and 3) data is read into memory as a global array, distributed among multiple threads, differenced, compressed, and written to disk. We measured difference and compression times, throughput rate, and overall runtimes for the experiments. Our results confirm the hypothesis that differenced checkpoints can improve compression ratios (see Figure 1). On three different NWChem runs consisting of 21 checkpoints the average compression ratios are increased by $\sim 50\%$, from 1.21 to 1.61, from 2.19 to 3.15, and from 1.12 to 1.55. It's also clear compression effectiveness increases as the computation converges. This corresponds to the intuition behind our ideas – as the simulation converges, the deltas decrease in size producing increasing compression ratios.

Differenced with Threshold Further experiments, shown in Figures 2 and 3 add a threshold below which values in the differenced checkpoint are rounded to zero. Even with a low cutoff value of 10^{-13} our experiments show some improvement in compression ratios, but the improvement is small, and a similar trend of improvement as the simulation progresses is demonstrated. However, using a larger threshold 10^{-7} which corresponds to the precision afforded by single precision, produces a dramatic effect. In Figure 3, we see that beyond the initial simulation period which produces compression ratios similar to other experiments to over 1000. This number is presumed to be limited by the zlib implementation block size and metadata structures.

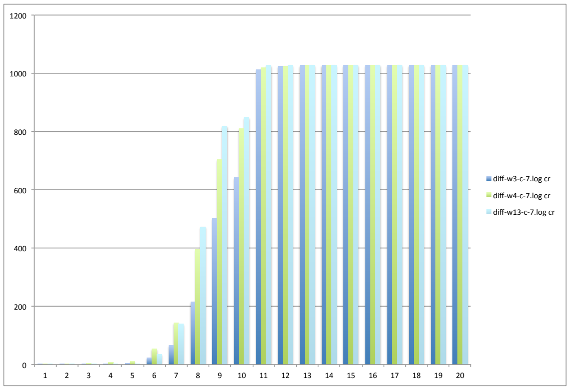

Figure 3. Compression Ratios for Diff Checkpoints, 10^{-7} threshold

CPU Times and Throughputs for Differenced Checkpoints and Thresholds Because checkpoints are computed and stored online in a computation, minimizing their overhead is important. Our experiments (see full data in Appendix) show that checkpointing time, even for our differenced checkpoint technique, is dominated by compression time, not file IO. As shown in Figure 4, the CPU time for compression using LZ77 can be significant, as much as 20 seconds per checkpoint for the larger (350MB) data sets. This number is representative of CPU times for the varied differenced checkpoint and thresholding schemes we explored. As compression effectiveness increases, the

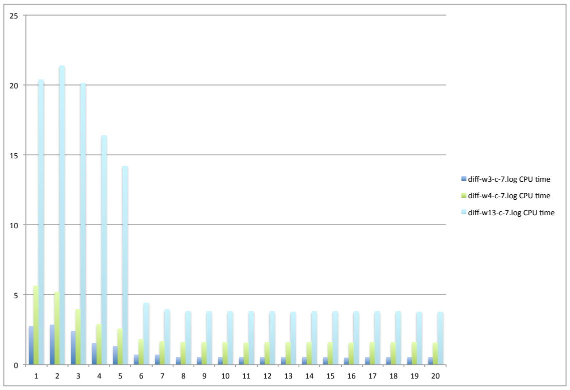

Figure 4. CPU Time for Diff. Checkpoints with 10^{-7} threshold (seconds)

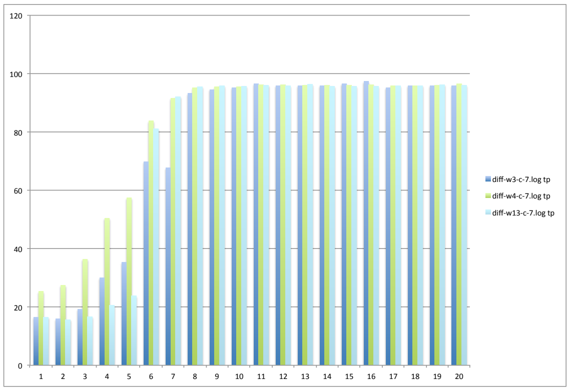

Figure 5. Throughput for Dif. Checkpoints with 10^{-7} threshold (MB/s)

cost of LZ77 per megabyte decreases. The corresponding throughput rate numbers are shown in Figure 5.

The high cost of compression is largely due to (*zlib*), and this library is not parallelized at present. In future work, we will explore use of parallelized compression methods to reduce the time for delta computation + compression, and given these are independent block-based methods, we expect close to linear speedup on multicore processors. Of course, we will also explore more computation efficient compression algorithms – but with the requirement that they achieve equally high compression ratios.

IV. Discussion and Related Work

There is a long history of research in efficient checkpointing for reliability in scientific computations, with optimization of interval, the data collected, and so on [9], [26], [25]. In addition, in parallel computations, researchers have explored a range of coordinated and uncoordinated techniques [7], [12] to reduce the cost of synchronization and message logging for coherent state capture. Differenced checkpoints and thresholding which reduce the cost of capturing a series of checkpoints can be applied directly to these and other advanced resiliency – recovery and rollback approaches. Recently, several groups have explored in-memory [11] and flash-based [21] techniques which reduce checkpoint cost, and could benefit directly from our techniques if they capture multiple checkpoints.

Data compression is a long studied area, and the difficulty of compressing scientific data is well-documented. Recent work has indicated some promise for improvement [5], [24], [31] by customizing the compression algorithm or by selectively applying compression at the block level. These approaches are promising and may well be fruitfully combined with the techniques presented in this paper. However, our approach differs from these approaches, exploiting the structure of the data across multiple check points, and the scientific application data precision to create a new opportunity for data compression.

V. Summary and Future Work

Reliability is an increasingly critical problem for Exascale computing systems, and checkpointing continues to play an important role in a growing array of resilience techniques. Our experiments show two promising new methods to reduce the cost of checkpoints – differenced checkpoints and thresholding – which increase the effectiveness of data compression techniques by as much as three-fold. Future directions include broader experiments with a range of application data and larger systems, integration of our new techniques for in-memory compression, and parallelization of compression for higher performance.

References

[1] A. A. Auer et al. Automatic code generation for many-body electronic structure methods: the tensor contraction engine. *Molecular Physics*, 104(2):211–228, 2006.

[2] E. Aprà et al. Liquid water: obtaining the right answer for the right reasons. In *Proceedings of the ACM/IEEE SC Conference on High Performance Networking and Computing*, pages 1–7, New York, NY, USA, 2009. ACM.

[3] Argonne National Laboratory. Mira, Accessed: 3/22/2012. http://www.alcf.anl.gov/mira.

[4] R. Bartlett and M. Musiał. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, 79(1):291–352, 2007.

[5] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Trans. Comput.*, 58:18–31, January 2009.

[6] E. J. Bylaska et. al. NWChem, a computational chemistry package for parallel computers, version 6.0, 2010.

[7] C. Coti, T. Herault, P. Lemarinier, L. Pilard, A. Rezmerita, E. Rodriguez, and F. Cappello. Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant mpi. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[8] T. Crawford and H. Schaefer. An introduction to coupled cluster theory for computational chemists. In K. B. Lipkowitz and D. B. Boyd, editors, *Reviews in Computational Chemistry*, volume 14, chapter 2, pages 33–136. VCH Publishers, New York, 2000.

[9] J. Daly. A model for predicting the optimum checkpoint interval for restart dumps. In *Proceedings of the 2003 international conference on Computational science*, ICCS'03, pages 3–12, Berlin, Heidelberg, 2003. Springer-Verlag.

[10] A. DePrince III and J. Hammond. Coupled cluster theory on graphics processing units I. The coupled cluster doubles method. *Journal of Chemical Theory and Computation*, 7:1287–1295, 2011.

[11] L. A. B. Gomez, N. Maruyama, F. Cappello, and S. Matsuoka. Distributed diskless checkpoint for large scale systems. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pages 63–72, Washington, DC, USA, 2010. IEEE Computer Society.

[12] A. Guermouche, T. Ropars, E. Brunet, M. Snir, and F. Cappello. Uncoordinated checkpointing without domino effect for send-deterministic mpi applications. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 989–1000, Washington, DC, USA, 2011. IEEE Computer Society.

[13] J. R. Hammond, S. Krishnamoorthy, S. Shende, N. A. Romero, and A. D. Malony. Performance characterization of global address space applications: a case study with NWChem. *Concurrency and Computation: Practice and Experience*, 24(2):135–154, 2012.

[14] S. Hirata. Tensor contraction engine: Abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *Journal of Physical Chemistry A*, 107:9887–9897, 2003.

[15] D. S. Katz, J. Daly, N. DeBardeleben, M. Elnozahy, B. Kramer, L. Lathrop, N. Nystrom, K. Milfeld, S. Sanielevici, S. Cott, and L. Votta. Fault tolerance for extreme-scale computing workshop, 2009. Technical Memorandum ANL/MCS-TM-312.

[16] G. Knizia, W. Li, S. Simon, and H.-J. Werner. Determining the numerical stability of quantum chemistry algorithms. *Journal of Chemical Theory and Computation*, 7(8):2387–2398, 2011.

[17] K. Kowalski, J. R. Hammond, W. A. de Jong, P.-D. Fan, M. Valiev, D. Wang, and N. Govind. Coupled-cluster calculations for large molecular and extended systems. In J. R. Reimers, editor, *Computational Methods for Large Systems*, pages 167–200. John Wiley & Sons, Inc., 2011.

[18] K. Kowalski, S. Krishnamoorthy, R. Olson, V. Tipparaju, and E. Apra. Scalable implementations of accurate excited-state coupled cluster theories: Application of high-level methods to porphyrin-based systems. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1 –10, nov. 2011.

[19] K. Kowalski, R. M. Olson, S. Krishnamoorthy, V. Tipparaju, and A. E. Role of many-body effects in describing low-lying excited states of π-conjugated chromophores: High-level equation-of-motion coupled-cluster studies of fused porphyrin systems. *Journal of Chemical Theory and Computation*, 7(7):2200–2208, 2011.

[20] A. I. Krylov. Equation-of-motion coupled-cluster methods for open-shell and electronically excited species: The hitchhiker's guide to Fock space. *Annual Review of Physical Chemistry*, 59(1):433–462, 2008.

[21] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *IEEE/ACM Supercomputing Conference (SC)*, November 2010.

[22] J. Nieplocha, R. J. Harrison, and R. J. Littlefield. Global arrays: A portable "shared-memory" programming model for distributed memory computers. In *Supercomputing (SC)*, 1994.

[23] Oak Ridge National Laboratory. TITAN: Paving the way to exascale, Accessed: 3/22/2012. http://www.olcf.ornl.gov/titan/system-configuration-timeline/.

[24] M. A. O'Neil and M. Burtscher. Floating-point data compression at 75 gb/s on a GPU. In *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-4, pages 7:1–7:7, New York, NY, USA, 2011. ACM.

[25] J. S. Plank, M. Beck, G. Kingsley, and K. Li. Libckpt: transparent checkpointing under unix. In *Proceedings of the USENIX 1995 Technical Conference Proceedings*, TCON'95, pages 18–18, Berkeley, CA, USA, 1995. USENIX Association.

[26] J. S. Plank, J. Xu, and R. H. Netzer. Compressed differences: An algorithm for fast incremental checkpointing, 1995. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.1812.

[27] L. Pollack, T. L. Windus, and W. A. de Jong. Thermodynamic properties of the C5, C6, and C8 n-alkanes from ab initio electronic structure theory. *Journal of Physical Chemistry A*, 109(31):6934–6938, 2005.

[28] P. Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters*, 73(2):393 – 398, 1980.

[29] G. Purvis III and R. Bartlett. A full coupled-cluster singles and doubles model: The inclusion of disconnected triples. *Journal of Chemical Physics*, 76(4):1910–1918, 1982.

[30] I. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley Press, 2010.

[31] E. Schendel, Y. Jin, N. Shah, J. Chen, C. Chang, S.-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. Samatova. ISOBAR hybrid compression-I/O interleaving for large-scale parallel I/O optimization. In *The 21st International ACM Symposium on High Performance Parallel and Distributed Computing*, 2012.

[32] B. Schroeder and G. A. Gibson. Understanding failures in petascale computers. *Journal of Physics: Conference Series*, 78(1):012022, 2007.

[33] V. P. Vysotskiy and L. S. Cederbaum. Accurate quantum chemistry in single precision arithmetic: Correlation energy. *Journal of Chemical Theory and Computation*, 7(2):320–326, 2011.

[34] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transacations on Information Theory*, IT-23(3), May 1977.

APPENDIX

Full Measurement Data

	w3				w4				w13			
	gzip		fpc		gzip		fpc		gzip		fpc	
	CR	TP	CR	TP	CR	TP	CR	TP	CR	TP	CR	TP
1	1.21	17.78	1.02	336.15	2.23	19.37	1.08	328.02	1.12	17.66	1.01	297.14
2	1.21	18.04	1.02	288.22	2.19	18.80	1.07	333.91	1.12	17.74	1.01	296.79
3	1.21	17.73	1.02	328.01	2.19	19.05	1.07	309.40	1.12	17.65	1.01	294.46
4	1.21	17.80	1.02	329.97	2.19	18.98	1.07	321.84	1.12	17.36	1.01	307.05
5	1.21	17.92	1.02	314.39	2.19	18.30	1.07	338.86	1.12	17.73	1.01	320.15
6	1.21	17.83	1.02	320.95	2.19	18.82	1.07	337.65	1.12	17.41	1.01	322.78
7	1.21	18.04	1.02	328.01	2.19	18.87	1.07	324.58	1.12	17.61	1.01	313.31
8	1.21	17.64	1.02	324.53	2.19	18.91	1.07	331.23	1.12	17.27	1.01	336.01
9	1.21	17.17	1.02	335.75	2.19	18.98	1.07	324.39	1.12	17.92	1.01	316.83
10	1.21	17.90	1.02	473.70	2.19	18.97	1.07	330.44	1.12	17.58	1.01	297.30
11	1.21	17.85	1.02	324.53	2.19	18.62	1.07	311.81	1.12	17.89	1.01	312.15
12	1.21	17.99	1.02	285.53	2.19	18.74	1.07	333.78	1.12	17.70	1.01	313.52
13	1.21	17.80	1.02	452.63	2.19	18.93	1.07	336.38	1.12	17.71	1.01	302.52
14	1.21	17.48	1.02	377.38	2.19	18.87	1.07	311.39	1.12	17.63	1.01	304.61
15	1.21	18.07	1.02	311.55	2.19	18.87	1.07	316.05	1.12	17.07	1.01	315.20
16	1.21	18.04	1.02	285.53	2.19	19.03	1.07	307.27	1.12	17.77	1.01	305.78
17	1.21	17.97	1.02	316.39	2.19	18.89	1.07	327.01	1.12	17.00	1.01	303.93
18	1.21	17.71	1.02	318.51	2.19	18.96	1.07	311.39	1.12	17.40	1.01	314.38
19	1.21	17.75	1.02	401.19	2.19	19.11	1.07	342.89	1.12	17.64	1.01	312.64
20	1.21	18.09	1.02	333.70	2.19	18.74	1.07	315.94	1.12	17.74	1.01	317.68
21	1.21	17.90	1.02	297.15	2.19	19.04	1.07	316.37	1.12	17.50	1.01	301.10

Table II

COMPRESSION RATIOS (CR) AND THROUGHPUTS (TP) OF GZIP (*LZ*) AND FPC COMPRESSION SCHEMES O THE DATASETS. THREE TRIALS, SINGLE CORE.

	w3					w4					w13				
	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp
1	0.05	2.69	3.77	1.22	16.52	0.16	6.31	9.60	2.29	21.87	0.43	20.22	28.38	1.13	16.21
2	0.04	3.11	4.21	1.25	14.30	0.15	6.07	9.36	2.48	22.73	0.42	22.35	30.49	1.15	14.66
3	0.05	3.48	4.56	1.28	12.76	0.15	5.70	9.05	2.63	24.21	0.41	24.30	32.29	1.17	13.49
4	0.05	4.01	5.08	1.31	11.08	0.14	5.83	9.10	2.72	23.67	0.40	26.26	34.21	1.19	12.48
5	0.05	4.20	5.28	1.32	10.59	0.14	5.97	9.24	2.76	23.12	0.37	27.67	35.60	1.20	11.84
6	0.06	4.61	5.72	1.37	9.65	0.15	6.63	10.14	2.85	20.81	0.40	35.74	43.82	1.26	9.17
7	0.05	4.89	5.98	1.41	9.09	0.14	6.66	10.07	2.93	20.70	0.44	37.57	45.76	1.30	8.72
8	0.05	5.21	6.32	1.45	8.52	0.14	6.57	9.90	3.01	21.00	0.40	39.17	47.41	1.33	8.37
9	0.05	5.51	6.58	1.47	8.07	0.14	6.74	10.05	3.05	20.48	0.37	40.94	48.87	1.36	8.01
10	0.05	5.51	6.60	1.48	8.07	0.14	6.76	10.17	3.06	20.41	0.45	41.04	49.13	1.36	7.99
11	0.05	5.70	6.77	1.53	7.80	0.14	7.38	10.74	3.14	18.70	0.43	44.76	52.75	1.42	7.32
12	0.05	6.43	7.52	1.58	6.91	0.17	7.73	11.08	3.20	17.85	0.39	45.80	53.73	1.47	7.16
13	0.06	6.05	7.15	1.61	7.35	0.15	7.92	11.40	3.25	17.42	0.37	46.82	54.71	1.51	7.00
14	0.05	6.05	7.14	1.64	7.34	0.14	8.12	11.40	3.30	16.99	0.37	45.74	53.66	1.57	7.16
15	0.05	6.05	7.15	1.62	7.34	0.15	7.96	11.24	3.26	17.32	0.36	46.30	54.23	1.54	7.08
16	0.05	5.26	6.33	1.79	8.45	0.14	8.34	11.72	3.49	16.53	0.38	38.88	46.79	1.73	8.43
17	0.05	4.23	5.33	2.00	10.51	0.15	7.77	11.11	3.68	17.75	0.39	30.02	38.14	1.99	10.92
18	0.06	3.74	4.80	2.16	11.89	0.15	7.40	10.69	3.85	18.65	0.42	25.63	33.53	2.26	12.79
19	0.05	3.35	4.44	2.35	13.28	0.14	7.03	10.72	4.06	19.62	0.37	22.18	30.13	2.54	14.78
20	0.05	3.55	4.63	2.25	12.53	0.15	7.14	10.44	3.99	19.32	0.40	22.45	30.42	2.48	14.60

Table III

DIFFERENCED CHECKPOINTS PERFORMANCE FOR A SERIES OF NWCHEM CHECKPOINTS. CMPU = DIFFERENCE COMPUTATION TIME, CMPR = ZLIB COMPRESSION TIME, TOTAL = DISK I/O TIME + CMPU + CMPR, CR = COMPRESSION RATIO, TP = THROUGHPUT

	w3					w4					w13				
	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp
1	0.07	2.96	4.07	1.22	15.03	0.18	5.52	8.90	2.52	25.00	0.46	20.09	28.57	1.13	16.31
2	0.06	3.13	4.23	1.25	14.18	0.17	5.54	8.89	2.62	24.91	0.44	22.36	30.49	1.15	14.66
3	0.06	3.45	4.54	1.28	12.88	0.16	5.43	8.78	2.71	25.40	0.44	24.18	32.46	1.17	13.55
4	0.06	4.02	5.16	1.31	11.05	0.16	5.70	9.06	2.77	24.19	0.44	26.23	34.30	1.19	12.50
5	0.06	4.20	5.28	1.32	10.59	0.16	5.89	9.26	2.80	23.41	0.43	27.63	35.74	1.20	11.86
6	0.06	4.63	5.73	1.37	9.61	0.15	6.57	9.87	2.87	20.99	0.44	35.69	43.75	1.26	9.18
7	0.06	4.89	6.01	1.41	9.09	0.16	6.62	9.91	2.94	20.85	0.45	37.59	45.72	1.30	8.72
8	0.06	5.22	6.31	1.45	8.51	0.16	6.71	10.01	3.02	20.56	0.43	39.16	47.42	1.33	8.37
9	0.06	5.67	6.76	1.47	7.84	0.15	6.72	10.11	3.05	20.53	0.47	41.00	49.19	1.36	7.99
10	0.06	5.53	6.65	1.48	8.04	0.16	6.76	10.09	3.07	20.40	0.45	41.15	49.17	1.36	7.96
11	0.06	5.83	6.90	1.53	7.62	0.17	7.38	11.65	3.14	18.70	0.46	44.71	52.84	1.42	7.33
12	0.06	5.89	6.99	1.58	7.54	0.15	7.73	11.09	3.21	17.84	0.46	45.93	54.07	1.47	7.13
13	0.06	6.05	7.14	1.61	7.35	0.16	7.87	11.26	3.26	17.52	0.45	46.56	54.71	1.52	7.04
14	0.06	6.04	7.15	1.65	7.35	0.16	8.06	11.35	3.33	17.12	0.46	45.45	53.89	1.58	7.21
15	0.06	6.03	7.14	1.62	7.37	0.15	7.92	11.22	3.28	17.42	0.45	46.24	54.34	1.55	7.09
16	0.07	5.33	6.44	1.82	8.33	0.18	8.20	11.51	3.55	16.83	0.47	38.58	46.56	1.77	8.49
17	0.06	4.19	5.28	2.10	10.61	0.16	7.35	10.63	3.97	18.77	0.46	29.92	38.09	2.12	10.95
18	0.06	3.60	4.70	2.40	12.34	0.16	6.65	9.95	4.44	20.73	0.45	23.51	31.45	2.63	13.94
19	0.07	3.08	4.16	2.82	14.43	0.16	5.91	9.23	5.13	23.34	0.45	18.49	26.35	3.46	17.72
20	0.07	3.37	4.49	2.56	13.17	0.16	6.19	9.52	4.82	22.30	0.46	19.32	27.42	3.25	16.97

Table IV

DIFFERENCED CHECKPOINTS PERFORMANCE FOR A SERIES OF NWCHEM CHECKPOINTS, WITH A CUTOFF THRESHOLD OF 10^{-13}.

	w3					w4					w13				
	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp	cmpu	cmpr	total	cr	tp
1	0.07	2.69	3.81	1.25	16.50	0.18	5.44	8.83	2.60	25.34	0.46	19.92	28.12	1.16	16.45
2	0.06	2.79	3.90	1.44	15.91	0.17	5.04	8.36	3.02	27.39	0.45	20.93	28.98	1.25	15.66
3	0.07	2.31	3.41	2.05	19.21	0.17	3.79	7.17	4.39	36.43	0.45	19.68	27.93	1.46	16.65
4	0.06	1.48	2.59	3.61	30.03	0.16	2.74	6.07	7.97	50.41	0.47	15.92	24.02	2.01	20.59
5	0.06	1.26	2.33	4.66	35.27	0.16	2.40	5.68	10.53	57.40	0.44	13.77	21.79	2.43	23.79
6	0.05	0.64	1.71	23.61	69.80	0.15	1.65	5.00	54.23	83.77	0.39	4.04	11.93	35.47	81.12
7	0.06	0.66	2.04	65.41	67.67	0.15	1.51	4.81	143.11	91.56	0.38	3.56	11.42	139.25	92.06
8	0.05	0.48	1.53	215.30	93.23	0.15	1.45	4.79	395.35	95.13	0.40	3.43	11.42	472.10	95.55
9	0.05	0.47	1.56	501.84	94.55	0.15	1.44	4.74	703.99	95.57	0.40	3.42	11.17	818.38	95.83
10	0.05	0.47	1.54	642.13	95.23	0.14	1.44	4.76	810.12	95.57	0.39	3.42	11.39	849.06	95.73
11	0.06	0.46	1.52	1013.25	96.61	0.14	1.43	4.70	1019.21	96.24	0.39	3.41	11.33	1027.70	96.01

Table V

DIFFERENCED CHECKPOINTS PERFORMANCE FOR A SERIES OF NWCHEM CHECKPOINTS, WITH A CUTOFF THRESHOLD OF 10^{-7}. COMPARABLE TO DATA STORED AS SINGLE PRECISION FLOATING POINT.