An Investigation of the Effects of Error Correcting Code on GPU-accelerated Molecular Dynamics Simulations

Ross C. Walker
San Diego Supercomputer Center
Department of Chemistry and Biochemistry
UC San Diego
La Jolla, CA 92093
ross@rosswalker.co.uk

Robin M. Betz
San Diego Supercomputer Center
La Jolla, CA 92093
rbetz@ucsd.edu

ABSTRACT

Molecular dynamics (MD) simulations rely on the accurate evaluation and integration of Newton's equations of motion to propagate the positions of atoms in proteins during a simulation. As such, one can expect them to be sensitive to any form of numerical error that may occur during a simulation. Increasingly graphics processing units (GPUs) are being used to accelerate MD simulations. Current GPU architectures designed for HPC applications support error correcting codes (ECC) that detect and correct single bit-flip error events in GPU memory; however, this error checking carries a penalty in terms of simulation speed. ECC is also a major distinguishing feature between HPC NVIDIA Tesla cards and the considerably more cost-effective NVIDIA GeForce gaming cards. An argument often put forward for not using GeForce cards is that the results are unreliable due to the lack of ECC. In an initial attempt to quantify these concerns, an investigation of the effects of ECC on GPU-accelerated MD simulations using the AMBER software was conducted on 720 GPUs of the XSEDE supercomputer Keeneland with and without ECC. While the data collected are insufficient to make solid conclusions and more extensive testing is needed to provide quantitative statistics, the absence of ECC events and lack of any silent errors in all the simulations conducted to date suggest that these errors are exceedingly rare and as such the time and memory penalty of ECC may outweigh the utility of error checking functionality. This is particularly true in the case of large scale HPC runs where simulation is more likely to be interrupted by a node or storage failure and thus reducing the simulation wall clock time by turning ECC off may actually reduce the overall simulation failure rate.

Categories and Subject Descriptors

J.2 [**Computer Applications**]: Physical Sciences and Engineering

XSEDE 2013 San Diego, California USA

Keywords

XSEDE 2013, GPU-acceleration, ECC error

1. INTRODUCTION

The field of computational sciences uses the power of modern computers to gain insight into scientific systems. Researchers expend considerable time and effort using limited computational resources to create simulations that provide accurate results on reasonable timescales. As the goal of a simulation is to accurately represent real-world situations, elimination of mathematical error is a significant concern for computational scientists and both hardware and software are therefore routinely examined in-depth to identify and mitigate sources of error.

In the late 1970s, atmospheric radiation was found to cause bit-flips in random access memory (RAM), introducing small errors that could accumulate and lead to system malfunction or incorrect results if left unchecked [13]. As these errors are random, hardware-independent, and in principle unpreventable, extra parity bits and Hamming codes were added to DRAM modules in order to detect and correct for these errors [2]. This approach has largely continued to the present day with server-class hardware; desktop hardware does not typically include ECC checking of RAM.

The ability to run simulations on graphics processing units (GPUs) has opened up new frontiers for simulation complexity and timescale, but has correspondingly increased the opportunities for hardware errors [7]. In order to mitigate the anticipated effects of bit-flip errors in GPU memory, NVIDIA has implemented ECC functionality in its latest GPU architectures from the Fermi class onward, at a cost of approximately 10% of GPU memory and speed and correspondingly higher energy consumption [9]. It should be noted that the GeForce gaming class hardware does not include any form of ECC support.

The molecular dynamics code AMBER, which stands for Assisted Model Building with Energy Refinement, is a package of molecular simulation programs that is widely used within the computational chemistry and computational molecular biology communities [1, 10]. It includes a wide variety of programs that enable the simulation of molecular systems at the atomic level as well as tools for all stages of the simulation workflow. The AMBER project is focused on accurate and efficient simulation, and as such uses parallelization and GPU-acceleration to improve computation speed [4, 6].

AMBER was selected for this experiment as it is the only molecular dynamics code that is deterministic on GPUs,

and multiple simulations on the same hardware will produce bitwise-identical results in the absence of errors. This allows for easy identification of problems in multiple GPU simulations, and presents an environment to look for random, transient errors that are not possible to identify in the output of other simulation programs.

The size of the system that may be simulated by GPU-accelerated AMBER is limited by the amount of available GPU memory. As such, enabling ECC reduces the size of systems that may be simulated by approximately 10%. Enabling ECC also reduces simulation speed, resulting in greater opportunity for other sources of error such as disk failures in large filesystems, power glitches, and unexplained node failures to be introduced during the calculation.

Finally, ECC events in RAM are exceedingly rare, requiring over 1000 testing hours to observe [3, 8]. The ECC error rate has not been successfully quantified by any study– previous attempts conducted over 10,000 hours of testing without seeing a single ECC error event [11]. Testing of GPUs for any form of soft error found that the error rate was primarily determined by the memory controller in the GPU, and that the newer cards based on the GT200 chipset had a mean error rate of zero [5]. However, the baseline value for the rate of ECC events in GPUs is unknown.

The XSEDE high performance heterogeneous computing system Keeneland was used to investigate these considerations [12]. A large, 10-hour simulation was run on all GPUs of the machine with ECC both on and off. An analysis of the resulting trajectory was conducted to identify all sources of error in order to investigate the effects of ECC on AMBER simulations, and describe the rate and effects of ECC error events.

2. SIMULATION SETUP

AMBER was used to simulate the tobacco mosaic virus (STMV) shown in figure 1, in explicit solvent. SMV was chosen because at 1,067,095 atoms it represents the larger end of the simulation size range AMBER users typically run. This also means that it uses approximately 2.6 GB (48% of the GPU card's memory) making it in principle more susceptible to memory corruption errors than a system which only occupies a small amount of memory.

The system was equilibrated and a NPT simulation at 300K was run with a timestep of 2 fs for a total of 0.3 ns of simulation. All atoms including solvent were saved to the output trajectory every 100 steps giving a total binary trajectory size of 19 GB per simulation.

AMBER simulations are ideal to test the effects of potential ECC errors as the output of correct runs is bitwise identical, enabling easy comparison between GPUs that return successful results and those that have errors.

The simulation was run with exactly the same starting conditions on each of the 3 M2090 GPUs on 240 nodes of Keeneland for a total of 720 identical runs. Once the approximately 10-hour run completed, the machine was rebooted in order to turn ECC off for each GPU, and the run was repeated. Issues with scheduling following the reboot resulted in only 447 of the ECC off runs executing, however there was still ample data to conduct an error analysis.

3. RESULTS

The time penalty incurred from activating ECC was evident: jobs without it ran 8.8% faster. Interestingly, ECC made walltime usage considerably more variable. A summary of timing information may be found in table 1.

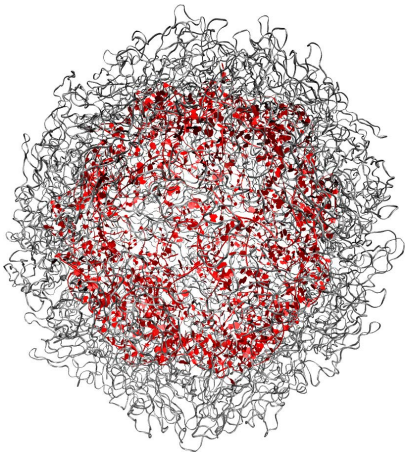

Figure 1: Satellite tobacco mosaic virus. The protein coat is in grey, and the viral DNA is shown in red. Solvent is omitted from this rendering.

Table 1: Walltime information in hours for successful runs.

ECC State	Mean Walltime	Standard Deviation
ON	9.951 hours	0.079 hour
OFF	9.096 hours	0.006 hour

Three runs on the same node with ECC on failed while reading the input coordinates, most likely due to a problem with I/O on the node. The remaining 717 completed without error. A single run with ECC off hung during the middle of the calculation while the remainder completed successfully yielding 446 bitwise identical (19 GB each) trajectory and output files.

The biggest concern with uncorrected memory errors is not that a calculation might hang or crash occasionally but that it could conceivably, with low bit errors, give what appear to be reasonable but scientifically invalid results due to silent data corruption. In these tests however there was no divergence observed in any of the simulations with or without ECC.

All runs that completed successfully returned the same trajectory– there was no divergence observed in any GPU with or without ECC. Additionally, none of the cards with ECC enabled reported correcting any ECC errors, indicating that most likely no ECC events occurred during the runs. It should be noted that the 720 GPUs utilized in these tests, when the hardware counters were checked, had reported no ECC error detection events within their entire installation life, which has been approximately 6 months.

4. CONCLUSION

Although the ability of ECC to detect and correct single bit errors is undeniably useful in theory, the practical application of this technology may not be in the interests of the molecular dynamics community. This test showed that the ability of ECC to correct extremely rare errors did not

outweigh the costs in terms of system size and calculation speed. These errors appear to be so rare in production GPU calculations that their rate of incidence could not be quantified with this experiment.

The fact that other sources of hardware error were observed during the experiment regardless of ECC status indicates that there are much more probable ways for simulations to fail and that such failures most likely cause the simulation to crash rather than to produce bad data. The concept of ECC or other events producing "silent errors" in a seemingly successful run was not supported by this experiment.

The improved performance without ECC may actually imply higher reliability than with error checking enabled, as it reduces the time window that the simulation runs and is therefore vulnerable to network, I/O, or other failures. As a result of this performance gain, representing error rates as errors per nanosecond of simulation may indicate that simulations with ECC off are potentially more reliable.

Future work with longer simulations on more GPUs in the hopes of observing an ECC event will hopefully produce an estimate of the baseline ECC error rate. Failing that, the effects of random memory corruption may be simulated with either programs or an external radiation source in order to determine whether silent errors do occur. Currently, their absence in this and other tests makes an argument for considering the elimination of ECC in favor of reduced costs, increased system size support and simulation speed.

5. ACKNOWLEDGMENTS

This research used resources of the Keeneland Computing Facility at the Georgia Institute of Technology, which is supported by the National Science Foundation under Contract OCI-0910735.

This work was funded in part by the National Science Foundation through the Scientific Software Innovations Institutes Program - NSF SI2-SSE (NSF1047875 and NSF1148276) grants to R.C.W and also by the University of California (UC Lab 09-LR-06-117792) grant to R.C.W. The work was also supported by a CUDA fellowship to R.C.W from NVIDIA Inc.

6. REFERENCES

[1] D. Case, T. Darden, T. C. III, C. Simmerling, J. Wang, R. Duke, R. Luo, R. Walker, W. Zhang, K. Merz, B. Roberts, S. Hayik, A. Roitberg, G. Seabra, J. Swails, A. Goetz, I. Kolossváry, K. Wong, F. Paesani, J. Vanicek, R. Wolf, J. Liu, X. Wu, S. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, G. Cui, D. Roe, D. Mathews, M. Seetin, R. Salomon-Ferrer, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, and P. Kollman. Amber 12. 2012.

[2] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda, and K. Mashiko. A built-in Hamming code ECC circuit for DRAMs. *Solid-State Circuits, IEEE Journal of*, 24(1):50–56, 1989.

[3] M. Gordon, K. Rodbell, D. Heidel, C. Cabral Jr, E. Cannon, and D. Reinhardt. Single-event-upset and alpha-particle emission rate measurement techniques. *IBM Journal of Research and Development*, 52(3):265–274, 2008.

[4] A. W. Götz, M. J. Williamson, D. Xu, D. Poole, S. Le Grand, and R. C. Walker. Routine microsecond molecular dynamics simulations with AMBER on GPUs. *Journal of Chemical Theory and Computation*, 8(5):1542–1555, 2012.

[5] I. S. Haque and V. S. Pande. Hard data on soft errors: A large-scale assessment of real-world error rates in GPGPU. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 691–696. IEEE, 2010.

[6] S. Le Grand, A. W. Götz, and R. C. Walker. Spfp: Speed without compromise - a mixed precision model for GPU accelerated molecular dynamics simulations. *Comp. Phys. Comm*, 184:374–380, 2013.

[7] J. Nickolls and W. J. Dally. The GPU computing era. *Micro, IEEE*, 30(2):56–69, 2010.

[8] E. Normand. Single event upset at ground level. *Nuclear Science, IEEE Transactions on*, 43(6):2742–2750, 1996.

[9] D. Patterson. The top 10 innovations in the new NVIDIA Fermi architecture, and the top 3 next challenges. *NVIDIA Whitepaper*, 2009.

[10] R. Salomon-Ferrer, D. Case, and R. Walker. An overview of the Amber biomolecular simulation package. *WIREs Comput. Mol. Sci.*, 2012.

[11] G. Shi, M. Showerman, and V. Kindratenko. On testing GPU memory for hard and soft errors. In *Proc. Symposium on Application Accelerators in High-Performance Computing*, 2009.

[12] J. S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford, et al. Keeneland: Bringing heterogeneous GPU computing to the computational science community. *Computing in Science & Engineering*, 13(5):90–95, 2011.

[13] J. Ziegler and W. Lanford. Effect of cosmic rays on computer memories. *Science*, 206(4420):776, 1979.