CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3232

SPECIAL ISSUE PAPER

An investigation of the effects of hard and soft errors on graphics processing unit-accelerated molecular dynamics simulations

Robin M. Betz[1], Nathan A. DeBardeleben[2] and Ross C. Walker[3,*,†]

[1]*San Diego Supercomputer Center, La Jolla, CA 92093, USA*
[2]*Ultrascale Systems Research Center, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*
[3]*San Diego Supercomputer Center, Department of Chemistry and Biochemistry, UC San Diego, La Jolla, CA 92093, USA*

SUMMARY

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 high performance computing applications support error-correcting codes (ECC) that detect and correct single bit-flip soft 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 high performance computing 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 because of the lack of ECC. In an initial attempt to quantify these concerns, an investigation of the reproducibility of GPU-accelerated MD simulations using the AMBER software was conducted on the XSEDE supercomputer Keeneland, a cluster at Los Alamos National Laboratory, and a cluster at the San Diego Supercomputer Center. 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. However, a considerable amount of error originating from defective hardware was observed, which suggests that rigorous acceptance testing should be performed on new GPU-based systems by repeatedly running reproducible yet realistic calculations. Copyright © 2014 John Wiley & Sons, Ltd.

Received 7 November 2013; Revised 22 January 2014; Accepted 29 January 2014

KEY WORDS: GPUs; molecular dynamics; error-correcting code

1. INTRODUCTION

The field of computational science 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

*Correspondence to: Ross C. Walker, San Diego Supercomputer Center, Department of Chemistry and Biochemistry, UC San Diego, La Jolla, CA 92093, USA.
†E-mail: ross@rosswalker.co.uk

Copyright © 2014 John Wiley & Sons, Ltd.

results if left unchecked [2]. As these errors are transient, hardware independent, and in principle unpreventable without costly shielding, extra parity bits and Hamming codes were added to DRAM, CPUs, and other hardware memory in order to detect and correct for these errors . This approach has largely continued to the present day with server-class hardware; desktop hardware does not typically include error-correcting code (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 [3]. 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 [1]. It should be noted that the GeForce gaming-class hardware does not include any form of ECC support.

The molecular dynamics (MD) 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 [4, 5]. 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 performance.

AMBER was selected for this experiment as it is the only MD code that is fully deterministic on GPUs, and multiple simulations on the same hardware will produce bitwise-identical results in the absence of errors. This reproducibility is the result of considerable effort on the part of the developers to create completely deterministic code using fixed-precision arithmetic [6]. This allows for easy identification of problems over 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. AMBER does not contain any error-checking functionality, as this is both unnecessary (because of easy reproducibility of correct results) and would slow down the code.

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 occur during the timeframe of a calculation.

Finally, ECC events in RAM are exceedingly rare, requiring over 1000 testing hours to observe [7, 8]. The GPU-corrected error rate has not been successfully quantified by any study—previous attempts conducted over 10,000 h of testing without seeing a single ECC error event. 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. However, the baseline value for the rate of ECC events in GPUs is unknown.

The goal of this research was to run many identical simulations in hopes of observing an ECC event. This will enable quantification of these events and most importantly will give an idea of the effect ECC errors have on MD simulations.

2. SIMULATION SETUPS

We investigated these considerations by running repeated deterministic simulations on a variety of GPUs with and without ECC. A large-scale analysis was performed using the XSEDE high performance heterogeneous computing system Keeneland [9] and with several machines at Los Alamos National Laboratory (LANL) and the San Diego Supercomputer Center. Large, long simulations were run on all GPUs of many nodes of each machine with ECC both on and off. Analysis of the resulting trajectories 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.

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, fatal or otherwise.

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe

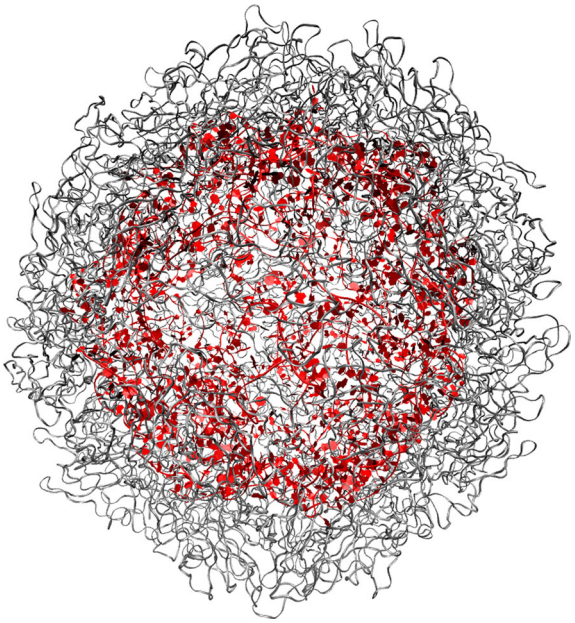

Figure 1. Satellite tobacco mosaic virus. The protein coat is in grey, and the viral RNA is shown in red. Solvent is omitted from this rendering.

2.1. *Keeneland*

We were allocated the entirety of Keeneland's 240 nodes for enough time to conduct two 10-h runs, one with ECC on and the other with ECC off. AMBER was used to simulate the satellite tobacco mosaic virus (STMV) shown in Figure 1, in explicit solvent. STMV 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 that only occupies a small amount of memory.

The system was equilibrated, and an NPT simulation at 300 K 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.

The simulation was run with exactly the same starting conditions on each of the three M2090 GPUs on 240 nodes of Keeneland for a total of 720 identical runs. Once the approximately 10-h run was 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.

2.2. *Los Alamos National Laboratory*

Reproducible MD runs were tested on a production cluster with queuing at LANL. Because of the number of users on the machine and difficulty of getting long walltime allocations with a queuing system, a smaller system was simulated for a shorter period.

An NPT simulation of the enzyme dihydrofolate reductase (DHFR) (Figure 2) was run in explicit solvent at 300 K with a timestep of 2 fs for a total of 2 ns of simulation. The system has 23,558 atoms and was chosen because of its frequent usage in the benchmarking of MD codes. Trajectories of all atoms, including solvent, were written every 2 ps of simulation.

Runs were performed independently on both M2090 GPUs on a node for a total of 500 runs with ECC on and 90 runs with ECC disabled. Because disabling ECC requires a reboot and the use of ECC is currently regarded as a best practice, for purely political reasons, it is difficult to get nodes without ECC enabled by cluster administrators. By special arrangement, one node was obtained

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe

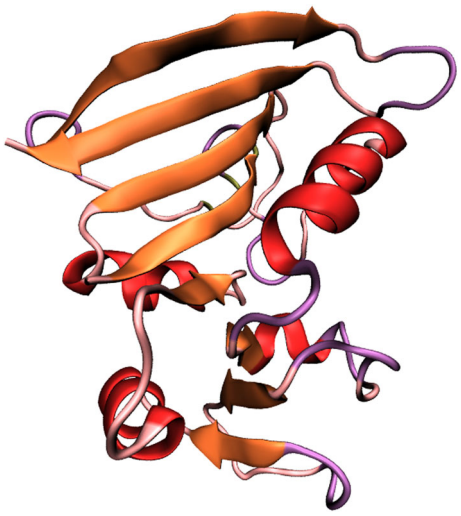

Figure 2. Dihydrofolate reductase colored according to secondary structure. Solvent is not shown.

without ECC, but the sample size of ECC disabled runs is clearly much smaller as the rest of the cluster was available for running with ECC on.

2.3. San Diego Supercomputer Center

The Walker group at the San Diego Supercomputer Center has recently constructed a cluster of eight nodes with four GTX680s per node, called Dante. As part of initial system testing, reproducibility runs were conducted on all GPUs, in order to check for hardware defects or other sources of error.

GTX680s are a more affordable card for running simulations and were marketed as a high-end gaming card, and as such do not feature ECC capabilities. Nevertheless, these reproducibility runs proved illuminating with regard to other sources of error that can occur on GPU clusters.

A simulation of DHFR with the same parameters as that run at LANL was performed for 8 ns on each GPU, and the results compared across all 10 independent runs. GPUs whose reproducibility test gave nondeterministic results were placed into another machine and the test re-run for a somewhat shorter time of 1 ns per trial, in order to check if errors were due to GPU construction or problems with the node.

3. RESULTS

3.1. Keeneland

The time penalty incurred from activating ECC was evident: jobs without it ran 8.8% faster. Interestingly, ECC made walltime usage considerably more variable. It is unknown what the origin of this variability is. A summary of timing information may be found in Table I.

Three runs on the same node with ECC on failed while reading the input coordinates, most likely because of a problem with I/O on the node. The remaining 717 completed without error. A single run

Table I. Walltime information in hours for successful runs of satellite tobacco mosaic virus on Keeneland.

ECC state	Mean walltime (h)	Standard deviation (h)
On	9.951	0.079
Off	9.096	0.006

ECC, error-correcting code.

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe

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.

3.2. Los Alamos National Laboratory

No errors of any kind were observed in either the 500 runs with ECC on or the 90 runs with ECC off, as all simulations completed successfully with identical output.

The time penalty for ECC on was approximately 4.6%, with the simulations with ECC on finishing in a mean 77.18 min and those with ECC off finishing in 73.78 min. This time, penalty is less than that observed on Keeneland and is most likely due to the difference in system size between STMV and DHFR.

3.3. San Diego Supercomputer Center

Surprisingly, 11 of 32 GPUs in the cluster failed to reproduce all 10 DHFR simulations. We observed two modes of failure—either there would be minor differences in the final system energy for a few of the 10 trials, or massive error would be present in nearly every other run. Frequently, simulations with this massive error would crash or exit with an error due to extreme forces or atom positions.

To further determine the source of errors, four GPUs from one especially problematic node (that would output no video) were placed into another node and the test re-run. In the original node, two GPUs passed and the other two failed, but upon re-test in the other node, three passed and one failed. This indicates that one of the four GPUs is definitely defective, and the remaining may be giving errors due to a bad motherboard.

All of the problematic 11 GPUs were re-tested in a machine with known good hardware on which the group has been running correct simulations for several years. Six of these GPUs failed again in this machine, but five returned normal results on all 10 shorter 1-ns simulations in the re-test. However, several of these five GPUs continue to run nondeterministically when returned to the cluster, suggesting that this may be sensitivities to motherboard, CPU, BIOS, or some other variation external to the GPU.

In a further attempt to narrow down the source of the error, we returned four of the GPUs that gave correct results on the group machine to Dante and re-ran the test; however, at this point, the problem node would not boot or enter BIOS, indicating a likely problem with the motherboard.

However, installing a known good NVIDIA GPU from a different manufacturer in one of the worst-performing slots on another node resulted in no errors observed on re-test, implying that a faulty GPU caused the observed errors on this node.

We hypothesize that the errors observed on this cluster were caused by a complex interplay of defects in the motherboard and in the GPUs. Variations in manufacturing or driver problems may have caused the observed problems, and we are continuing to investigate the nature and source of errors.

4. DISCUSSION

Detection and mitigation of error is critical to ensuring the accuracy of MD simulations on GPUs. Errors may originate from transient bit-flips, termed ‘soft’ error, or may be from defects in hardware or driver bugs.

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe

As we detected no ECC events in any of our trials with cards supporting ECC, the rate of these errors remain unquantified, but our results are in agreement with previous suggestions that the rate is extremely low. Given that the nature of these errors remains unknown and could in theory produce silent differences in calculation, simulations should be checked for reproducibility before publication.

The hard error rate observed on the Dante cluster was alarmingly high and further underscores the need for reproducibility testing on new installations in order to identify and replace defective components. The number of points vulnerable to failure is high, as we saw problems with both GPUs and motherboards at a minimum, and this complicates error checking. Nonetheless, without our 10-run duplication experiment, these issues may have remained undetected, as even the most defective cards completed the simulation normally approximately half of the time.

Hard errors such as the ones we observed may never be detected in production environments without repeated runs of bitwise reproducible codes, and systems can pass standard acceptance testing while exhibiting considerable error on other calculations [10]. The crashes and unreasonable energies observed in the worst of the failed runs may be written off as a program bug, and more concerningly, the subtle errors in final energy may be accepted as valid results of a code that produces a different answer each time because of rounding differences. Although it took considerable effort on the part of AMBER developers to ensure reproducibility on GPUs, the detection of errors such as those that we observed completely validates this approach to GPU development.

It should be noted that AMBER users recently reported to us that a Replica Exchange run across 300 GPUs on Keeneland would consistently fail with errors, while the same calculations on Blue Waters are successful. The system administrators for Keeneland are now in the process of testing the entire machine using our AMBER validation package in order to identify GPUs with hard errors.

5. 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 MD 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 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.

Our observation of high hard error rates in new clusters indicates that there are more frequent sources of error that manifest potentially extremely frequently and can escape detection in traditional validation tests. Without several runs of a code that gives completely reproducible results, these hard errors may go completely undetected. For this reason, at least one major hardware vendor (Exxact) has began utilizing AMBER to verify all hardware in their GPU systems before shipping.

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, the lack of ECC events observed in this and other tests makes an argument for allowing user space control of ECC in favor of reduced costs, increased system size support and simulation speed, given that other sources of hardware and software errors had a much greater impact on calculation correctness. Rigorous acceptance testing, hardware, software, and driver validation currently has a significantly greater ability to reduce error in calculations than does ECC error checking, and therefore should be the focus of error mitigation efforts.

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe

ACKNOWLEDGEMENTS

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, by the University of California (UC Lab 09-LR-06-117792) grant to R.C.W, and by a grant from the University of California Institute for Mexico and the United States (UC MEXUS) and the Consejo Nacional de Ciencia y Tecnología de México (CONACYT). The work was also supported by a CUDA fellowship to R.C.W from NVIDIA Inc.

REFERENCES

1. Maruyama N, Nukada A, Matsuoka S. Software-based ECC for GPUs. In *2009 Symposium on Application Accelerators in High Performance Computing (SAAHPC09)*, Vol. 107, Urbana, Illinois, July 28–30 2009. Available at http://saahpc.ncsa.illinois.edu/09.
2. Ziegler JF, Lanford WA. Effect of cosmic rays on computer memories. *Science* 1979; **206**(4420):776–788.
3. Nickolls J, Dally WJ. The GPU computing era. *IEEE Micro* 2010; **30**(2):56–69.
4. Case DA, Darden TA, Cheatham III TE, Simmerling CL, Wang J, Duke RE, Luo R, Walker RC, Zhang W, Merz KM, Roberts B, Hayik S, Roitberg A, Seabra G, Swails J, Goetz AW, Kolossváry I, Wong KF, Paesani F, Vanicek J, Wolf RM, Liu J, Wu X, Brozell SR, Steinbrecher T, Gohlke H, Cai Q, Ye X, Wang J, Hsieh M-J, Cui G, Roe DR, Mathews DH, Seetin MG, Salomon-Ferrer R, Sagui C, Babin V, Luchko T, Gusarov S, Kovalenko A, Kollman PA. Amber 12: University of California: San Francisco, 2012.
5. Salomon-Ferrer R, Case DA, Walker RC. An overview of the Amber biomolecular simulation package. *WIREs Computational Molecular Science* 2012; **3**(2):198–210.
6. Le Grand S, Götz AW, Walker RC. SPFP: speed without compromise - a mixed precision model for GPU accelerated molecular dynamics simulations. *Computer Physics Communications* 2013; **184**:374–380.
7. Gordon MS, Rodbell KP, Heidel DF, Cabral Jr C, Cannon EH, Reinhardt DD. Single-event-upset and alpha-particle emission rate measurement techniques. *IBM Journal of Research and Development* 2008; **52**(3):265–274.
8. Normand E. Single event upset at ground level. *IEEE Transactions on Nuclear Science* 1996; **43**(6):2742–2750.
9. Vetter JS, Glassbrook R, Dongarra J, Karsten S, Loftis B, McNally S, Meredith J, Rogers J, Roth P, Spafford K, Yalamanchili S. Keeneland: bringing heterogeneous GPU computing to the computational science community. *Computing in Science & Engineering* 2011; **13**(5):90–95.
10. DeBardeleben N, Blanchard S, Monroe L, Romero P, Grunau D, Idler C, Wright C. GPU behavior on a large HPC cluster. *6th Workshop on Resiliency in High Performance Computing (Resilience) in Clusters, Clouds, and Grids in conjunction with the 19th International European Conference on Parallel and Distributed Computing (Euro-Par 2013), Aachen, Germany*, Aachen, Germany, August 26–30 2013. Available at http://xcr.cenit.latech.edu/resilience2013.

Copyright © 2014 John Wiley & Sons, Ltd.

DOI: 10.1002/cpe