Numerical Reproducibility for Parallel Stochastic Simulation "Exascale Ready"

D.R.C. Hill, F.Y.P. Congo, T. Dao Van
ISIMA/LIMOS UMR CNRS 6158
Blaise Pascal University – Campus Universitaire des Cezeaux, 2, Rue de la Chebarde, TSA 60125 – CS 60026
63178 Aubière CEDEX FRANCE
david.hill@univ-bpclermont.fr

***Abstract*—How can we obtain correct parallel Monte Carlo results identical to what we obtain with the corresponding sequential simulation? How can we keep the statistical quality of our results whilst avoiding correlations between parallel computations at Exascale? In this short paper we present some elements of response which suppose the mastering of parallelization techniques for the best pseudo-random number generators.**

Keywords—Parallel Stochastic Simulation, Parallel Random Numbers, Reproducibility, Exascale Computing, High Performance Computing.

I. Introduction

When dealing with parallel stochastic simulations, the main cause of non-reproducibility can be linked to a lack of rigor in their design. Hellekaleck warned us a long time ago (Hellekaleck 1998). Research fields like nuclear safety, or nuclear medicine cannot live with an approximate quality of the final results due to poor distribution techniques of the parallel random numbers. In such fields the scientific method is a real concern and reproducibility is considered at the very beginning of computer experimental design. In less sensitive domains, numerical reproducibility has also been a concern as we can see from the significant work conducted by J.F. Claerbout and his colleagues (Schwab et al 2000) (Fomel and Claerbout 2009).

For reproducible simulations at Exascale, interested readers can find a set of advices in (Demmel and Nguyen 2013). In (Diethelm 2012) we also find interesting elements dealing with the limits of numerical reproducibility for simulations.

The introduction of new processors with multi and manycores twelve years ago impacted the obtaining of correct results because of the out-of-order execution of floating point instructions

Such out-of-order executions have been proposed to improve performance. If this approach is interesting when producing images and films, the introduction of this kind of hardware and software optimizations started to impact our scientific results more than ten years ago. This trend is increasing with the arrival of multicore CPUs and hardware accelerators such as FPGAs, GPUs and Intel Xeon Phi, which are now intensively used in high performance computing (Taufer al 2010) and impact the numerical reproducibility even on a run to run basis.

The last point reducing our chance to obtain reproducible results on first class computing machines is linked to silent date corruption and soft errors such as rare interactions of circuitry with stray subatomic particles. If those events are extremely rare for desktop and cluster computing, it is a fact that with our current technology, the Mean Time Between Failure (MTBF) is now measured in hours on the largest machines.

To cope with the previously cited points, there is a need for a deeper understanding and mastering of the techniques behind modern parallel computing and parallel stochastic simulations. In addition a real cultural change is needed to integrate computational reproducibility into the research process (Stodden et al 2013), into our publication system, and what seems more important to us, into our educational systems.

II. A New Software Crisis ?

Biologists, Physicists and all experimental researchers are used to filling laboratory books with extended details (equipment settings, raw results, processing techniques and procedures, statistical methods used, …). All these details are recorded in order to be able to reproduce the computational experiments under study. This scientific "reproducibility" is a key element of the scientific method, implying that with the details of the experiments, other scientists will be able to obtain similar results. Such reproducibility could be possibly achieved with different techniques and approaches. Scientific reproducibility implies various processes: scientific peer review, replication of experiments, confirmation of the results by other research teams, verification, and open research. In practice, it is not so easy for other scientists to find the same results as those presented in the literature for many reasons. This first reason could be linked to the publication race which leads to a lot of misbehaviour, for instance related to individual researchers not sharing their data, etc.

In the field of scientific computing, all the above mentioned facts and rules apply. Initially we have to focus on a subset of reproducibility named repeatability. The fact of being, at least, able to "replicate" the results (Drummond 2009). Finding the same results with the same input data and the same algorithm is an essential technique to test, verify and find errors, and to ensure the validity of computational experiments.

However, it is a fact and a pity that most computational experiments are not achieved as carefully as our colleagues from other research domains. Frequently, we are not recording the elements which are essential for a repeatability of our computer experiments. Even if the software source code, data and results are freely available (and this is rare), this is far from enough. Computer scientists should record the computer hardware and software configuration and versions, the compiling options, the parameter settings, the software architecture and design, the scientific workflow including the function invocation sequences... They should mandatorily use versioning software for their source code and keep track of the library dependencies, etc. This is really done for top scientific software but not for common development in software laboratories or companies. There is a human cost for applying all the best practices, and it is a fact that they are not applied in most scientific developments, leading to a lack of repeatability. Finally, the common practices observed affect the productivity of computer scientists and above all they affect the quality and credibility of their work. This participates in what Krishnamurthi and Vitek name the real software crisis (Krishnamurthi and Vitek 2015).

III. Numerical Reproducibility At Exascale

A. *Fine Approaches*

When working for the future exascale machines, we first have to realize that scalability is essential. Even with the current top systems (above O(10^7) compute cores) we are facing many of the previously mentioned problems. An exascale machine will propose O(10^9) compute cores. In (Ahn et al 2013), we can realize that if we want to meet numerical reproducibility at this scale, we need a common toolset with unified, targeted, and multilevel approaches. We agree that such a common toolset will be considerably valuable to control and eliminate many sources of non-determinism.

In addition to software tools, the numerical and floating-point aspects have to be highlighted. We have seen in the introduction that the numerical results depend not only on the computer arithmetic precision, but also on the order of arithmetic operations, and in addition we have stated that the emergence of hybrid computing (with hardware accelerators) have an impact on both aspects. In (Bailey et al 2002) we have potential solutions to master arbitrary precision in high performance computing and in (Jézequel et al 2014) and (Revol and Theveny 2014) we find the essential computer arithmetic approaches we need to increase trust and credibility in the results of our future simulations running at Exascale.

B. *The case of Parallel Stochastic Simulation*

In most cases parallel stochastic simulations lead to independent computing tasks which can be seen as naturally (some say embarrassingly) parallel. Moreover, the expected numerical results are obtained statistically and this is a very good point when you know you will lose the results of some computing tasks. While waiting for new technologies impacting the hardware of top systems, we may prefer Monte Carlo computing approaches because they are more fault tolerant, more resilient in a world where first class systems have an MTBF of a few hours. For instance, instead of experiencing repeated failures with deterministic parallel Partial Differential Equation (PDE) solver, we could select a parallel stochastic PDE solver able to recover and increase the precision of the final results from run to run between failures.

We have recently read in an HPC magazine that parallel stochastic simulations were not reproducible due to their use of random numbers. This absurdity is often mainly due to limited knowledge and also to the absence of education on this point in top universities. Specialists in stochastic simulation for High Energy Physics have mastered the reproduction of numerical results since the very beginning of parallel Monte Carlo Simulation. In (Urbatsch and Evans 1998) we have an example of C++ implementation with the common technique of keeping track for each particle of an ID independent random stream, with a specific status. Running this kind of approach at Exascale supposes modern generators used with the proper parallelization technique. Unfortunately the knowledge necessary to achieve this objective satisfactorily is not so well spread nowadays. To inform our younger colleagues, a survey of the best practice for parallel random simulations is proposed in (Hill et al. 2013), this survey also discusses the usage of GPUs and gives specific advice for modern hardware accelerators.

Though we already have a known set of best practices, we could not find any information to compare a parallel stochastic simulation with a sequential one (run low scales) for verification, validation and accreditation purposes. In the next section we give the main elements of a method which enables the numerical reproducibility of parallel stochastic simulations even when compared to its equivalent sequential implementation running when running at small scales. This method is detailed in (Hill 2015). A proof of concept has been implemented by P. Schweitzer, with C. Mazel and two colleagues in Physics. The application tested implements the simulation of atmospheric muons simulations which are used for the computed tomography of large edifices (volcanoes; pyramids,…) and which can also be used for scanning containers for security or customs applications.

IV. Reproducibility From Sequential To ParalleL

First of all, scientists using stochastic simulations have to remember that the generators mimicking randomness are fortunately deterministic. Top scientists know that this « feature » is essential for computational experiments. Mathematicians do their best to provide reproducible generators. We have recently tested many of them for numerical reproducibility on different platforms, with differents compilers, different operating systems (Dao et al 2014). If the results are often robust, some results are surprising and disappointing.

In the context of parallel stochastic simulations, obtaining the repeatability of our experiments (i.e. identical results) with different execution contexts (e.g. a different number of processors) is a challenge. To meet our expectations, and to satisfy our numerical reproducibility requirements, we propose using:

- A process or object oriented approach (with stochastic objects) ;
- modern and statistically sound generators according to the most stringent testing battery (TestU01);
- a fine parallelization technique adapted to the selected generator,
- a parallel design method starting from the design of the sequential program.

The last point is crucial and is detailed in (Hill 2015), we have to propose a sequential program that will serve as a reference, but which is designed by thinking "parallel" from the start and for all independent stochastic processes. This is the main key for a numerical reproducibility with a sequential validation at small scales.

V. Storing Simulation Metadata In Clouds For Better Reproducibility

The method shortly described previously is just one element in a set of approaches for numerical reproducibility. Run to run reproducibility and numerical repeatability implies keeping track of many other technical details: programming languages, versions of the compilers, of the libraries, of the operating system, of the virtual machine, pointer to data sets, etc. In a complementary work, we have created a reproducible process for the construction of eScience computing environments on top of cloud computing infrastructures. Our solution separates the construction of these environments into two distinct layers: the infrastructure layer and the software layer. Results have been obtained on two different computational clusters within two separate cloud computing environments to show how this framework facilitates the replication computational experiments (Congo 2015).

VI. Conclusion

We have tried to show in this article the importance of caring about the numerical reproducibility of our stochastic simulation results. In the case of parallel stochastic parallel simulations, we have to handle rigorously the distribution of pseudo-random numbers. In this short paper, we give the key elements of a method which ensures that the results of parallel stochastic simulations are reproducible and second, it enables a comparison with a sequential implementation (before large scaling on the future exascale systems. This kind of numerical replication is very important for many scientists in many sensitive areas, finance, nuclear safety, medicine… Even if it does not solve the whole problem of scientific reproducibility, it is a key element for numerically reproducible research for parallel stochastic computing.

References

[1] Ahn D.H., Lee G.L., Gopalakrishnan G., Rakamarić Z., Schulz M., and Laguna I., "Overcoming extreme-scale reproducibility challenges through a unified, targeted, and multilevel toolset". *In ACM Proceedings of the 1st International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering*, 2013, pp. 41-44.

[2] Bailey D.H., Yozo H., Li X.S. and Thompson B., "ARPREC: An arbitrary precision computation package". 2002, Lawrence Berkeley National Laboratory, 2002, unpublished report ; http://escholarship.org/uc/item/2rt1p2vm

[3] Congo F.Y.P., "Building a Cloud Service for Reproducible Simulation Management", In Proc. Of The 14th Python In Science Conf. (Scipy), 2015, pp. 1-6.

[4] Dao V.T., Maigne L., Breton V., Nguyen H.Q., Hill D., "Numerical Reproducibility, Portability And Performance Of Modern Pseudo Random Number Generators", European Simulation And Modelling Conference, Oct. 22-24, 2014, Porto, Portugal, pp. 80-88.

[5] Demmel, J. and Nguyen H. D. 2013. "Numerical reproducibility and accuracy at exascale". In Proc. Of IEEE Symposium on Computer Arithmetic (ARITH), 2013, p. 235-237.

[6] Diethelm K. 2012. "The Limits of Reproducibility in Numerical Simulation". In Computing Science and Engineering, vol. 14, no. 1, pp. 64-72, Jan.-Feb. 2012, doi:10.1109/MCSE.2011.21

[7] Drummond C., "Replicability Is Not Reproducibility: Nor Is It Good Science," *In Proc. Evaluation Methods for Machine Learning Workshop, 26th Int'l Conf. for Machine Learning*, 2009 ; www.site. uottawa.ca/ICML09WS/papers/w2.pdf.

[8] Fomel S. and Claerbout J.F., "Reproducible Research", *Computing in Science & Eng.*, vol. 11, no. 5, 2009, pp. 5–7.

[9] Hellekalek P., "Don't Trust Parallel Monte Carlo!", *In Proc. Parallel and Distributed Simulation Conf.*, 1998, pp. 82–89.

[10] Hill D.R.C. et al., "Distribution of Random Streams for Simulation Practitioners", *Concurrency and Computation: Practice and Experience*, vol. 25, no. 10, 2013, pp. 1427–1442.

[11] Hill D.R.C., "Parallel Random Numbers, Simulation, Science and reproducibility". *IEEE/AIP - Computing in Science and Engineering*, vol. 17, no 4, 2015, pp. 66-71.

[12] Krishnamurthi S. and Vitek J, "The real software crisis: repeatability as a core value", *Communications of the ACM CACM*, vol. 58, no 3, 2015, pp. 34-36.

[13] Jézéquel F., Langlois P., and Revol, N. "First steps towards more numerical reproducibility". *In ESAIM: Proceedings and Surveys*, vol. 45, 2014, pp. 229-238.

[14] Revol N. and Théveny P., "Numerical reproducibility and parallel computations: Issues for interval algorithms.", *IEEE Transactions on Computers*, vol. 63, no 8, , 2014, pp. 1915-1924.

[15] Schwab M., Karrenbach and Claerbout J., "Making Scientific Computations Reproducible", *Computing in Science & Eng.*, vol. 2, no. 6, 2000, pp. 61–67.

[16] Stodden V., Bailey D.H., Borwein J., LeVeque R.J., Rider W. and Stein W., "Setting the default to reproducible: Reproducibility in computational and experimental mathematics", 2013, unpublished ICERM workshop synthesis; https://www.carma.newcastle.edu.au/ jon/icerm12.pdf

[17] Taufer M. et al., "Improving Numerical Reproducibility and Stability in Large-Scale Numerical Simulations on GPUs", *In IEEE Int'l Symp. Parallel and Distributed Processing*, 2010, pp. 1–9.

[18] Urbatsch T. J. and Evans T. M., "Parallel implicit Monte Carlo in C++", (No. LA-UR-98-2135; CONF-981207). 1998, Los Alamos National Lab., United States.