SAND2013-3965C

Realizing Exascale Performance for Uncertainty Quantification

Eric Phipps[1]*, H. Carter Edwards[2], Jonathan Hu[3], and Clayton Webster[4]

[1]Optimization and Uncertainty Quantification, Sandia National Laboratories†
[2]Multiphysics Simulation Technologies, Sandia National Laboratories†
[3]Scalable Algorithms, Sandia National Laboratories†
[4]Applied and Predictive Mathematics, Oak Ridge National Laboratory

Motivation

Exascale computing promises to address many scientific and engineering problems of national interest by facilitating computational simulation of physical phenomena at tremendous new levels of accuracy, fidelity, and scale, as well as unprecedented capabilities for high-level analysis such as uncertainty quantification for today's petascale computational simulations. Uncertainty quantification is a broad term for a variety of methodologies such as uncertainty propagation, model calibration, and error estimation, but here we are primarily concerned with the forward propagation of model input data uncertainty to simulation output quantities of interest. There are many uncertainty propagation approaches such as random sampling [4, 8, 12, 13, 14], stochastic collocation [2, 15, 16, 19], and stochastic Galerkin [5, 6, 20] that have been studied in the literature, most of which involve sampling simulations at numerous realizations of the uncertain input data. Unfortunately all of them generate extreme computational burdens when applied to many problems of interest when accurately capturing rare events in high dimensional uncertain input spaces exhibiting non-smooth behavior. It is not unusual to require thousands to millions of samples to achieve the level of accuracy desired in many uncertainty quantification problems, which is out of reach for most large-scale simulations on existing petascale computing architectures.

Nearly all existing uncertainty propagation methodologies wrap around deterministic simulation codes, a good example is sampling-based methods which repeatedly call a deterministic simulation code for different values of the model inputs. Since these samples are independent, they can easily be run in parallel on disjoint subsets of the available compute nodes. However it is unlikely an exascale computer will provide enough concurrency for a thousand-fold increase in concurrent sample evaluations for uncertainty propagation applied in this manner. Power and cooling limitations will favor compute nodes with dramatically increased floating-point capacity rather than substantially increased node counts [1], and thus increasing concurrent sample evaluation will require executing each sample on a smaller number of compute nodes or executing multiple samples simultaneously on each compute node. For reasons similar to those outlined below, the first approach is unlikely to be effective. The question for this paper is how to implement the second approach effectively given likely exascale architectural considerations, so that the rigorous, accurate uncertainty quantification capabilities promised by exascale can be achieved.

Approach

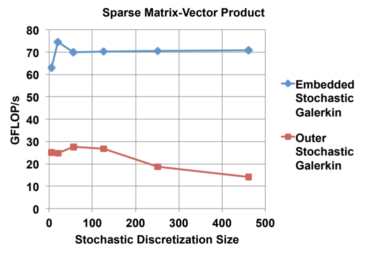
It is expected that node-level floating-point capacity will increase by a factor of 1,000 to 10,000 without increases in processor clock speed but rather increased concurrency through thousands of simple processor cores and with a memory capacity increase by only roughly a factor of 100 [1]. This suggests multiple samples should be executed simultaneously on each node by collections of threads sharing common data,

*corresponding author: etphipp@sandia.gov

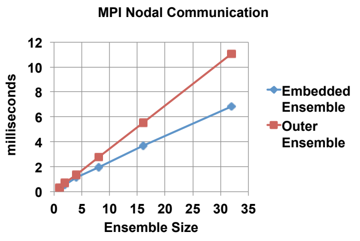
†Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

and implies that at least a portion of the "uncertainty propagation loop" must be *embedded* within the simulation code itself. Furthermore it is expected that memory latency will not decrease significantly and latency hiding through instruction-level parallelism and out-of-order execution will be replaced by hardware multi-threading and vectorization [1]. Thus it is imperative that calculations executing on these nodes exhibit good data locality as well expose sufficient fine-grained parallelism, which unfortunately is not the case for many simulations today (particularly those involving sparse linear algebra [3, 11]).

However we believe these challenges can be addressed by further embedding portions of the uncertainty loop at the lowest levels of the simulation code by replacing each scalar datum in a calculation with an array for the uncertainty representation of that datum, such samples in a sampling-based method or polynomial coefficients in a stochastic Galerkin-type method. Then any operations on that datum can be translated to parallel operations on the uncertainty array, both improving locality and exposing additional fine-grained parallelism. For example, Figure 1a displays floating-point throughput for sparse matrix-vector multiplication with scalars replaced by polynomial coefficients in a stochastic Galerkin method, compared to the standard approach of a sequence of scalar matrix-vector multiplications for increasing stochastic discretization size [18]. Since the working set dictated by the uncertainty array resides in L1 cache, much higher floating-point throughput is achieved. Similarly this approach enables amortization of expensive operations such as an MPI message across multiple pieces of contiguous data, for example Figure 1b displays the aggregate MPI communication time for multiple samples of an explicit dynamics calculation. Since the messages for multiple realizations are incorporated into one message for the ensemble, total communication time is reduced. Finally this approach enables new algorithmic approaches that reuse data and calculations across uncertainty representations to reduce aggregate simulation cost, e.g., reuse of mesh calculations that don't depend on uncertain input data or reuse of solvers and preconditioners across an ensemble [9, 10].



(a) *Sparse matrix-vector floating-point throughput for embedded stochastic Galerkin compared to a corresponding sequence of scalar matrix-vector multiplications (Intel Sandybridge with 32 cores).*



(b) *Aggregate MPI communication for multiple samples of an explicit dynamics calculation (Cray XK7 with 40 MPI ranks and 8 threads/rank).*

Figure 1

Challenges

While this approach could enable more concurrency in the aggregate uncertainty propagation calculation for likely exascale architectures, there are a number of challenges that must be overcome for these ideas to be useful for practical scientific problems. First incorporating these methods directly into simulations codes requires considerable programming effort, as well as cross-domain knowledge between simulation technologies and uncertainty quantification methodologies. However we and others have demonstrated that code transformation techniques based on automatic differentiation can alleviate much of this difficulty, particularly if they are incorporated early into the code design cycle [7, 17]. Second, the concept of propagating multiple samples simultaneously at the scalar level of the simulation is predicated on the assumption that the code paths for these samples do not diverge greatly, otherwise no or possibly negative benefit is achieved. However many realistic scientific problems exhibit some form of non-smooth behavior that put the simulation into different regimes depending on the values of the uncertain inputs. Thus careful research is needed connecting these ideas to high-level adaptive uncertainty propagation methods that decide when and how to group samples to be propagated together. Finally little research has been undertaken to develop solution and preconditioning algorithms that exploit the Kronecker-product problem structure inherently generated by this form of uncertainty propagation.

References

[1] *Scientific Grand Challenges: Architectures and Technology for Extreme Scale Computing*. US Department of Energy Workshop Report, December 2009.

[2] I. Babuska, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 1005–1034.

[3] N. Bell and M. Garland, *Implementing sparse matrix-vector multiplication on throughput-oriented processors*, in SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, New York, NY, USA, 2009, ACM, pp. 1–11.

[4] G. Fishman, *Monte Carlo*, Springer Series in Operations Research, Springer-Verlag, New York, 1996. Concepts, algorithms, and applications.

[5] R. Ghanem and P. D. Spanos, *Polynomial chaos in stochastic finite elements*, Journal of Applied Mechanics, 57 (1990).

[6] R. G. Ghanem and P. D. Spanos, *Stochastic finite elements: a spectral approach*, Springer-Verlag, New York, 1991.

[7] R. Giering and M. Vobeck, *Increasing memory locality by executing several model instances simultaneously*, in Recent Advances in Algorithmic Differentiation, S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, eds., vol. 87 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2012, pp. 93–101.

[8] J. Helton and F. Davis, *Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems*, Reliability Engineering and System Safety, 81 (2003), pp. 23–69.

[9] C. Jin and X. Cai, *A Preconditioned Recycling GMRES Solver for Stochastic Helmholtz Problems*, Communications in Computational Physics, 6 (2009), pp. 342–353.

[10] C. Jin, X. Cai, and C. Li, *Parallel Domain Decomposition Methods for Stochastic Elliptic Equations*, Siam Journal On Scientific Computing, 29 (2007), pp. 2096–2114.

[11] P. T. Lin and J. N. Shadid, *Towards large-scale multi-socket, multicore parallel simulations: Performance of an MPI-only semiconductor device simulator*, Journal of Computational Physics, 229 (2010), pp. 6804–6818.

[12] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), pp. 239–245.

[13] N. Metropolis and S. Ulam, *The Monte Carlo method*, Journal of the American Statistical Association, 44 (1949), pp. 335–341.

[14] H. Niederreiter, *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc, 84 (1978), pp. 957–1041.

[15] F. Nobile, R. Tempone, and C. G. Webster, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2309–2345.

[16] ——, *An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2411–2442.

[17] R. P. Pawlowski, E. Phipps, and A. G. Salinger, *Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part I: Template-based generic programming*, Scientific Programming, 20 (2012), pp. 197–219.

[18] E. Phipps, H. C. Edwards, J. Hu, and J. T. Ostien, *Exploring emerging manycore architectures for uncertainty quantification through embedded stochastic Galerkin methods*, International Journal of Computer Mathematics, (2013). Submitted.

[19] D. Xiu and J. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1118–1139.

[20] D. Xiu and G. Karniadakis, *The wiener-askey polynomial chaos for stochastic differential equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 619–644.