Precision, Accuracy, and Rounding Error Propagation in Exascale Computing

Invited Paper

Marius Cornea, Intel Corporation
marius.cornea@intel.com

Abstract

Exascale level computers might be available in less than a decade. Computer architects are already thinking of, and planning to achieve such levels of performance. It is reasonable to expect that researchers and engineers will carry out scientific and engineering computations more complex than ever before, and will attempt breakthroughs not possible today. If the size of the problems solved on such machines scales accordingly, we may face new issues related to precision, accuracy, performance, and programmability. The paper examines some relevant aspects of this problem.

I. INTRODUCTION

Supercomputers have evolved over the past several decades, and their performance has leapt many times by factors of 100x or more. Today, they are used for tasks of a great variety, ranging from stock trading to high energy physics to weather forecast to breaking encryption codes. Yet, there has never been so much discussion about the next level of computing as there is now, when that level is exascale (at ~10^{18} floating-point operations per second, or FLOPS). The reason is that many of the techniques and technologies that have allowed for progress so far are approaching barriers that seem hard to surpass, even when performance now has to increase by less than 40x in order to reach that next level. The U.S. Department of Energy (DOE) in particular, asked the industry to reach that goal by the end of this decade, while staying within a 20 MW power envelope. Of the many challenges in reaching that goal, those faced by the Computer Arithmetic community are not among the top ones at this point. Yet, we have to consider all aspects carefully in order to make sure floating-point computations as we know them will not become a limiting factor in just a few years.

Exascale computing will have applications in many areas. One example where progress is easy to grasp is weather prediction. A computer capable of sustaining 1 TFLOPS in 2005 was able to predict weather in useful time using a climate model with a 160-km spatial grid; in 2010 with a PFLOPS machine the grid granularity was reduced to 22 km; and before 2020 it is expected to become about 1 km with an EFLOPS machine (T, P, and E are abbreviations for Tera, Peta, and Exa, respectively).

We cannot think of exascale computing without taking a look at where we are now. This is best reflected in the current Top 500 supercomputer list, whose first few entries are shown in Table 1. Scaling the top-ranked system to the exascale level (theoretical performance) would result in more than 20 million cores and 26 PB of memory, consuming more than 300 MW in power.

TABLE I. TABLE 1. TOP 500 LIST – TOP ENTRIES (WWW.TOP500.ORG)

Top500 Rank, March 2013	Site	Computer	Theoretical Perf. (PFLOPS)	Linpack Perf. (PFLOPS)	Memory (GB)	Power (kW)
1	DOE/SC/ORNL	Titan - Cray XK7, AMD Opteron* 6274 16C 2.200GHz, Cray Gemini* interconnect NVIDIA Tesla* K20x; 560640 cores	27.11	17.59	710144	8209
2	DOE/NNSA/LLNL	Sequoia – BlueGene*/Q, Power BQC 16C 1.60 GHz, Custom; 1572864 cores	20.13	16.32	1572864	7890
3	RIKEN Inst. for Comp. Science (AICS)	K computer, SPARC64* VIIIfx 2.0GHz, Tofu interconnect, 705024 cores	11.28	10.51	1410048	12660
4	DOE/SC/ Argonne Nat. Laboratory	Mira – BlueGene*/Q, Power BQC 16C 1.60GHz, Custom, 786432 cores	10.07	8.16	N/A	3945
5	Forschungs-zentrum Juelich (FZJ)	JUQUEEN – BlueGene*/Q, Power BQC 16C 1.600GHz, Custom Interconnect, 393216 cores	5.03	4.14	393216	1970
6	Leibniz Rechen-zentrum	SuperMUC - iDataPlex DX360M4, Intel® Xeon® E5-2680 8C 2.70GHz, Infiniband* FDR, 147456 cores	3.19	2.90	N/A	3423
7	Texas Adv. Comp. Ctr. Uni. of TX	Stampede - PowerEdge C8220, Intel® Xeon E5-2680 8C 2.700GHz, Infiniband* FDR, Intel® Xeon Phi™ coprocessor, 204900 cores	3.96	2.66	184800	N/A

*Other brands and names are the property of their respective owners

II. Barriers to Surpass On the Way to Exascale

Among the many challenges toward the exascale goal, four stand out: power, extreme parallelism, reliability, and memory and storage capacity and bandwidth [1].

The DOE goal for maximum power of 20 MW leads to a performance efficiency requirement of 50 GFLOPS/W. Today's most efficient supercomputer (Table 2, top entry) achieves ~2.5 GFLOPS/W. A 20x gap needs to be closed. This will be difficult, because Moore's Law (the number of transistors on a chip doubles roughly every 18 months) which reliably predicted exponential growth in performance over the past 40+ years and is nowhere near the end of its life, is increasingly hard to apply due to physical limitations on the transistors that make up the CPUs. Today, power costs for the largest PFLOPS-level systems are in the range of $5-10 million annually [2]. To build an exascale system using current technology, the annual power cost would be above $2.5 billion per year. The power load would be over 1 GW - more than many power plants currently produce. The power efficiency challenge is considered the most serious one, because transistor switch scaling flattens with time as we go from TFLOPS to PFLOPS to EFLOPS, and supply voltage scaling is also too slow [3].

The extreme parallelism required for exascale computations is another barrier. This is also related to the applied mathematics challenges identified in [2]. There is a need for better mathematical models, e.g. for resolving fine structures in a rapidly changing environment, defining a self-teaching adaptive model, evaluating options in designing human-made systems, or analyzing the structure of uncertainties arising from imperfect measurements and incomplete knowledge of physical or biological phenomena.

Increasingly complex models will require new numerical methods capable of handling very large problem dimensions, nonlinear structures, mixtures of continuous and discrete variables, and drastically different scales. Computer arithmetic and numerical analysis along with several applied mathematics fields will play an important role in solving these problems. While exascale systems have yet to be defined, we will certainly need algorithms that are scalable and fast at very large levels of parallelism involving millions of processor cores. Scalability should be modeled and analyzed mathematically, using abstractions that represent key architectural ingredients. Simulations and experiments that indicate the effects of hardware and software perturbations on algorithmic efficiency can then guide the definition of methods that retain scalability under a variety of hardware scenarios.

Studies and practical applications in this area are facilitated by the advent of processors optimized for high-performance computing (HPC). Some are hardware accelerators for numerical computations. Others, such as the Intel® Many Integrated Core Architecture (Intel® MIC Architecture) are coprocessors, capable of running complete applications on their own albeit under control of a host processor. For example the first Intel MIC Architecture product, the Intel® Xeon Phi™ processor, has up to 61 cores implemented in Intel's 22 nm tri-gate process and is capable of sustaining computation rates of more than 1 TFLOPS.

The third big challenge is that of reliability [4]. With millions of processors running in concert, the probability of failures or other errors is expected to be relatively high. Even soft errors – chip-level or system-level - are considered to be a serious threat at this scale of computing. Greater tolerance to errors and resilience against failures will have to be embedded in operating systems, libraries, and applications alike.

Finally, memory and storage systems need to register major advances as well. Memory is one of the areas where the power reductions needed to reach exascale computing will be the most difficult to achieve.

TABLE II. GREEN500 LIST – TOP ENTRIES (WWW.TOP500.ORG)

Green500 Rank, March 2013	Performance/ Power Unit (GFLOPS/W)	Site	Computer	Total Power (kW)
1	2.50	NICS/Univ. of Tennessee	Intel Xeon proc. E5-2670 8C 2.600GHz, Infiniband* FDR, Intel Xeon Phi 5110P	44.89
2	2.35	King Abdulaziz City	Intel Xeon proc. E5-2650 8C 2.000GHz, Infiniband* FDR, AMD FirePro* S10000	179.15
3	2.14	DOE/SC/Oak Ridge Nat. Lab	Titan - Cray XK7 , AMD Opteron 6274 16C 2.200GHz, NVIDIA Tesla K20x	8,209.00
4	2.12	Swiss Scientific Computing Ctr.	Todi - Cray XK7 , Opteron* 6272 16C 2.100GHz, NVIDIA Tesla* K20 Kepler	129.00
5	2.10	Forschungszentrum Juelich	JUQUEEN – BlueGene*/Q, Power BQC 16C 1.600GHz, Custom Interconnect	1,970.00
6	2.10	Southern Ontario SCIC/ University of Toronto	BGQdev – BlueGen*e/Q, Power BQC 16C 1.600GHz, Custom Interconnect	41.09
7	2.10	DOE/NNSA/LLNL	rzuseq – BlueGene*/Q, Power BQC 16C 1.60GHz, Custom	41.09

[*] Other brands and names are the property of their respective owners

Both CPU and memory determine the size of the problems we can solve. Current projections indicate that by the year 2020, following past trends, processors' computing capacity will grow much faster – by about one order of magnitude - than memory capacity and access time, so current applications will have difficulty in scaling, and might have to be redesigned. It will take relatively longer to bring the data from memory than today, and meanwhile processors might remain idle. This will also lead to less memory per arithmetic functional unit, the need for finer-grained parallelism, and a programming model other than the currently used message passing or coarse-grained parallelism (e.g. with PThreads or OpenMP).

III. PRECISION

The floating-point formats specified in the IEEE Standard 754-2008 [5] are no doubt sufficiently flexible to cover the needs of exascale-level computing, and beyond. However, fast hardware implementations wider than 80-bit floating-point arithmetic do not exist today. Vector (SIMD) implementations do not exist beyond double precision. It is possible that in order to solve problems suited for exascale computing, we will need more precision than that available today. The U.S. Department of Energy identified high-precision arithmetic as one of nine areas in math and algorithms of interest for exascale computing, where significant progress needs to be made [6] (some of the others include multi-scale algorithms, combinatorial algorithms, adaptive mesh refinement, and solvers).

Double-extended precision would ensure better accuracy than double precision, but vector implementations do not exist today. On the Intel Xeon Phi processor for example, the double-extended floating-point operations are scalar, implemented in the x87 FPU, while double precision is implemented for 512-bit SIMD vectors. With the availability of fused multiply-add, one can execute 16 double precision floating-point operations in roughly the same time as one double-extended add or multiply.

Quad precision is often suitable for applications requiring more than double precision, however it is widely supported only in software and is slower by at least two orders of magnitude compared to the double precision floating-point implemented in hardware. Using it only where necessary may be a solution to potential precision limitations in the future.

The same can be said about multi-precision packages. Based on hardware-supported floating-point – typically double precision – in theory they allow for unlimited levels of accuracy. In practice, performance and memory limitations, now or at exascale, make them useful in limited situations.

IV. ACCURACY

Will there be accuracy issues on exascale systems? Many of those involved in the HPC field might be tempted to answer 'no'. That might even be correct – double precision may still ensure sufficient accuracy on such systems. Still, we should draw a definitive conclusion based on more study than done so far. We obviously cannot assume that double precision will be 'good enough' forever in most cases, as it is now (single precision is already insufficient for many problems solved on supercomputers today). We need to make sure that exascale computing will not be compromised by the lack of floating-point formats implemented in hardware, wider than double precision.

The larger the amount of computation, the larger the errors we can expect in the calculated results. This is acknowledged also in the pass/fail test used in the High Performance Computing Linpack Benchmark employed to rank Top500 supercomputers. The HPL benchmark [7] is a software package that solves, using LU factorization and an MPI implementation, a (random) dense linear system in double precision arithmetic on distributed-memory computers (it can thus be regarded as a portable implementation of the High Performance Computing Linpack Benchmark). It also provides a testing and timing program to quantify the accuracy of the solution as well as the time it took to compute it. The best performance of a system depends on a large variety of factors, but the algorithms used in the HPL benchmark are scalable in that their parallel efficiency is maintained constant with respect to the per processor memory usage.

The pass/fail test checks whether the following inequality holds:

$$||A \cdot x - B||_\infty / [\varepsilon \cdot (||x||_\infty \cdot ||A||_\infty + ||b||_\infty) \cdot N] < 16$$

where $\varepsilon = 2 \cdot 10^{-16}$ and N is the dimension of matrix A.

When N grows, at the denominator $||x||_\infty$ and $||b||_\infty$ can be expected to have roughly the same order of magnitude as before, but $||A||_\infty$ and N will grow. If N grows k times, the denominator will likely be about k^2 times larger. This allows for the residual at the numerator to be also k^2 times larger, while still having the HPL run pass. This ensures successful HPL runs as the accumulated errors grow for larger problems, but does not seem to guard against the case where these errors could be too large to be considered acceptable.

V. ROUNDING ERRORS

Errors can accumulate due to measurement uncertainty, conversion of input values, using a finite number of terms in computing the value of infinite series, and also due to round-off or overflow/underflow events during the computation.

What can we expect as the final 'average' error (measured in units-in-the-last-place or ulps, relative, or absolute) after a given number of floating-point operations, in the absence of overflow or underflow? We know what the worst-case errors can be. We could also calculate at run time intervals that contain the exact result of a given computation (thus determining the maximum error - however this is often hard or practically impossible to apply). Knowing the 'average' expected error would be very useful, as it would help us determine whether double precision will suffice for exascale computing. In the 44-year history of ARITH conferences, only two papers covered related topics. In [8], error propagation was studied in some special cases, but a general conclusion was not drawn. In [9], the distribution of accumulated round-off errors was determined, but for one single floating-point operation, not for several (or many) consecutive operations. However, [10] gives us a “rule of thumb”: it states that the rounding error, in ulps, after n floating-point operations will have the same order of magnitude as $\sqrt{n}$. This is based on the assumption that rounding errors are independent random variables (not true, in fact).

In practice, it is often the case that the final error is due mostly to a small number of unusually large errors in the course of the computation, and not to the accumulation of many small rounding errors. Software tools and possibly hardware support to help detect more unusual errors could help detect and avoid such cases.

At exascale levels, we will want every aspect of our computations to be faster. In some cases however, we may have to settle for lower performance. In particular, this could be the case with libraries of mathematical functions where today we often increase performance by relaxing their accuracy. Many compiler implementations today provide several accuracy levels for LIBM-type functions: usually a version with very good accuracy, with maximum errors not much higher than 0.5 ulps in rounding-to-nearest mode; then one or more lower accuracy versions with maximum ulp errors bounded for example by 1 ulp, 4 ulps, or even $2^{p/2}$ ulps, where p is the precision of the floating-point format used. There are very few correctly rounded LIBM implementations today, but the advent of exascale computing should cause this number to increase, and it may also ensure proliferation of the already existing versions. We will have to seriously consider using correctly rounded versions of LIBM as the default case.

Using relaxed accuracy to gain performance also happens routinely with basic operations. For example, the IEEE 754 division or square root operations are often replaced by Newton-Raphson approximations. Such optimizations will have to be disabled in order to ensure better accuracy, and even reproducible results. Otherwise, every computation step will have errors of several ulps instead of at most 0.5 ulp per operation.

Because of more stringent accuracy requirements, existing object code for certain applications will not be ready to run on exascale machines. They will have to be recompiled with much stricter accuracy requirements. Other compiler optimizations, not mentioned here, may have to be disabled as well. Algorithms for accurate sums, products, and dot products such as those described in [11] should be useful.

VI. Conclusion

Compared to other major challenges faced by the hardware and software communities on the road to exascale computing, those related to precision, accuracy, floating-point computation errors and computer arithmetic issues in general seem relatively minor. Yet, the computer arithmetic, numerical analysis, and numerical algorithm designer and implementer community needs to exercise due diligence in making sure that no important aspects are overlooked, and that from a numerical computation point of view, everything will be ready for the challenges of exascale.

References

[1] S.S. Pawlowski, “Exascale Science: The Next Frontier in High Performance Computing”, 24th ACM International Conference on Supercomputing 2010

[2] The Opportunities and Challenges of Exascale Computing, U.S. DOE, 2010, http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf

[3] J. Rattner, “Extreme Scale Computing”, Keynote at ISCA 2012

[4] X. Yang & al, “The Reliability Wall for Exascale Supercomputing”, IEEE Transactions on Computers; Jun2012, Vol. 61 Issue 6, p767-779

[5] IEEE Standard 754-2008 for Floating-Point Arithmetic, 2008

[6] Modeling and Simulation at Exascale for Energy and the Environment, http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Townhall.pdf

[7] A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary, HPL -A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, 2008

[8] J. Marasa, D. Matula, A Simulative Study of Correlated Error Propagation In Various Finite Arithmetics, ARITH02, 2nd IEEE Symposium on Computer Arithmetic, 1972

[9] J. Barlow, On the Distribution of Accumulated Roundoff Error In Floating-Point Arithmetic, ARITH05, 5th IEEE Symposium on Computer Arithmetic, 1981

[10] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 1996

[11] J.M. Muller, N. Brisebarre, F. de Dinechin – Handbook of Floating-Point Arithmetic, Birkhauser, 2009