High-Precision Arithmetic and Exascale Computing

David H. Bailey[*] Edward A. Baron[†] Hemant Shukla[‡] Xiaoye S. Li [§]

February 9, 2012

1 Introduction

Virtually all present-day computer systems today implement the IEEE 64-bit floating-point arithmetic standard, which provides 53 mantissa bits, or approximately 15 decimal digit accuracy. For most scientific applications, 64-bit arithmetic suffices (and a few can get by with 32-bit), but for a rapidly expanding body of applications, 64-bit arithmetic is not sufficient:

1. *Petascale and exascale simulations.* Computations that are well-behaved on modest-sized problems may exhibit significant numerical errors when scaled up to the huge sizes typical of those now being run on large petascale and future exascale systems.

2. *Numerically sensitive calculations.* Some scientific computations include sensitive portions that produce inaccurate results when performed using 64-bit arithmetic. Some of these can be remedied by using more sophisticated algorithms, but others cannot.

3. *Long-time simulations.* Almost any kind of physical simulation, if performed over many time intervals, will eventually depart from reality, due to cumulative round-off error.

4. *Large dynamic range simulations.* Some physical problems give rise to linear systems with large dynamic ranges, often due to exponentials in the problem definition. These systems are numerically singular in 64-bit arithmetic, but are well-behaved in higher precision.

Many applications of this type have been described in published literature: supernova simulations, climate modeling, planetary orbit calculations, Coulomb n-body atomic systems, scattering amplitudes of subnuclear particles, nonlinear oscillator theory, experimental mathematics, quantum field theory, dynamical systems and others. Several of these applications, such as the supernova simulations, are in SciDAC-funded activities. A detailed overview is given in [1].

[*]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, `DHBailey@lbl.gov`. Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231.

[†]Dept. of Physics and Astronomy, University of Oklahoma, `baron@ou.edu`. Supported in part by US DOE Grant DE-FG02-07ER41517.

[‡]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, `HShukla@lbl.gov`.

[§]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, `XSLi@lbl.gov`.

It is important to keep in mind that few computational scientists have completed a rigorous course in numerical analysis, and even those who have typically seek a simple and easy-to-implement remedy when numerical problems are encountered. High-precision arithmetic is an attractive option for such users, because even in situations where numerically better behaved algorithms are known in the literature, it is often both easier and more reliable to simply increase the precision used for the existing algorithm. What's more, in other cases there is no real alternative to using high-precision arithmetic. Some simple examples are given in [1].

2 High-precision software

In the past few years, several high-precision software packages have been produced. Two examples are QD, which performs "double-double" (31 digits) and "quad-double" (62 digits) arithmetic [3], and ARPREC, which performs an arbitrarily high level of precision [2]. C++ and Fortran high-level language interfaces are available in each case, supporting real and complex datatypes. These interfaces greatly simplify usage — one merely declares high-precision variables using a special type statement, and then the low-level routines are called automatically.

These two software packages were both developed at LBNL 10 years ago, with funding provided by an LDRD grant. In the most recent calendar year (2011), the QD package was downloaded 2502 times, and the ARPREC package was downloaded 2859 times.

These facilities greatly increase run times. For example, computations using double-double precision arithmetic typically run five to ten times slower than with 64-bit arithmetic, and quad-double applications run 25 to 50 times slower. On the plus side, high-precision arithmetic meshes well with modern highly parallel computer technology, since such calculations are highly memory-local.

3 High precision software and GPUs

Graphics processing units (GPUs) are rapidly gaining acceptance as a standard feature for computing hardware. Apple computer now supplies a Radeon 5770 GPU card as standard equipment on its workstations, and a more advanced Radeon 5870 card is available as an option.

Even more importantly, GPUs are now being provided on some high-end petascale computer systems, and there is an emerging consensus that GPUs are the only foreseeable option to achieve exascale computing within 10 years. Oak Ridge National Laboratory's recently announced "Titan" supercomputer utilizes NVIDIA Tesla GPUs.

Given the recent rise in the number of scientific applications requiring higher precision arithmetic, combined with the explosive increase in the size and scope of petascale and exascale applications, it is clear that high-precision software must be modified for GPU hardware. The principal issues that are the following:

- *Level of implementation.* It does *not* appear feasible to use GPU hardware to accelerate a single high-precision arithmetic operation, certainly not for precision levels of interest to Department of Energy-sponsored research projects. Instead, the best approach is to utilize GPU hardware to perform a long vector of high-precision operations.

- *Prototype implementation.* One issue is what would be the "first step" of such a conversion. It is clear that these initial developments must be done in the absence of the high-level software interfaces that are currently available for these packages.
- *Vector-style design.* GPUs are designed to operate in lock-step fashion on a vector of operands. However, existing high-precision software typically involves numerous conditional branches. Thus all of this code needs to be converted to SIMD style.
- *High-level interface.* As noted above, an easy-to-use high-level interface is a major advantage of a high-precision software package. For a GPU library, however, it will be necessary to craft a library of routines that operate on vectors of data, and then provide an even higher-level interface for the usage of these vector routines.
- *Initial precision levels.* Initial implementations will be limited to "double-double" datatype (approximately 31 digits precision). Higher-precision operations, such as quad-double (approximately 62 digits precision) or higher will need to wait. Eventually, though, at least the 256-bit facility will be needed.

4 Conclusion

GPU hardware, incorporated into very highly parallel systems, provides an attractive option for those who require high-precision arithmetic, because such operations greatly multiply the computational cost, and GPU hardware holds the potential of significantly reducing the amount of time expended in such operations. However, given the difficulty of custom programming with GPUs, it is most unlikely that any scientists will attempt such GPU programming on their own. Advanced and easy-to-use software facilities are required.

References

[1] David H. Bailey, Roberto Barrio, and Jonathan M. Borwein, "High precision computation: Mathematical physics and dynamics," SIAM Meeting on Emerging Topics in Dynamical Systems and Partial Differential Equations (DSPDEs'10), 31 May 2010, Barcelona, Spain, available at `http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/hpmpd.pdf`.

[2] David H. Bailey, Yozo Hida, Xiaoye S. Li and Brandon Thompson, "ARPREC: An arbitrary precision computation package," Sept 2002, available at `http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/arprec.pdf`.

[3] Y. Hida, X. S. Li and D. H. Bailey, "Algorithms for Quad-Double Precision Floating Point Arithmetic," 15th IEEE Symposium on Computer Arithmetic (ARITH-15), 2001, available at `http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/arith15.pdf`.