The Miracle, Mandate and Mirage of High Performance Computing

Das Wunder, der Auftrag und die Illusion des High Performance Computing

David Keyes*, King Abdullah University of Science and Technology, Saudi Arabia

* Correspondence author: david.keyes@kaust.edu.sa

Summary High performance computing boasts a steadily increasing number of adherents, for excellent reasons, traditional and innovative. However, the promises for which scientists and engineers have been drawn to the extreme of the spectrum are more elusive than the simple extrapolation of decades of exponential improvement suggests. Programming today's systems for high performance is already problematic for many scientists and the learning curve will inexorably steepen. Expectations should be brought in line with the increasing difficulty of pushing the high performance frontier, and, for once, the modeling, algorithms, and software advances required to use the emerging hardware effectively should not be left until the hardware begins to arrive. ►►► **Zusammenfassung** Das Hochleistungsrechnen rühmt sich einer – aus klassischen oder ganz neuen, aber immer aus sehr guten Gründen – stetig wachsenden Anhängerschaft. Allerdings sind die Verheißungen, die Wissenschaftler zum oberen Ende des Leistungspektrums gelockt haben, schwerer fassbar als die einfache Extrapolation des exponentiellen Wachstums an Rechenleistung, welche wir seit Jahrzehnten sehen, vorgaukelt. Bereits heute ist es für viele Wissenschaftler schwierig, Supercomputer voll auszureizen, und die Lernkurve wird in Zukunft unerbittlich flacher werden. Daher müssen die Erwartungen an Hochleistungsrechensysteme ins Verhältnis zum Aufwand gesetzt werden, der nötig ist, um deren Leistung bestmöglich auszunutzen. Hierbei darf man mit Fortschritten bei Modellierung, Algorithmen und Software, die zur effektiven Nutzung neuer Hardware notwendig sind, nicht bis zur flächendeckenden Verfügbarkeit dieser Hardware warten.

Keywords ACM CCS: Computing Methodologies → Parallel computing methodologies; High Performance Computing; Petascale; Exascale ►►► **Schlagwörter** Hochleistungsrechnen

A capability that offers vast improvements at predictable intervals justifiably draws attention and market share. As markets are reshaped, expectations grow and pressure increases to fulfill them. As a *gedanken* experiment, consider how the phenomenon of a price drop in a commodity by a factor of ten every three years could play out. Today, in 2013, peanut butter commands a price of approximately one thousand dollars per ton, and at this price it is a delicacy whose proper use is in select recipes and confections. However, if we were assured that by 2016, its price would be a mere one hundred dollars per ton, we would substitute peanut oil for other oils in a vast number of recipes where it is not practical today. If we could predict further a price drop to ten dollars per ton by 2019, we would consider how to exploit it as a feedstock in many other contexts, for instance, to make plastics. A price of one dollar per ton by 2022 would compel us to convert furnaces to burn peanut butter or to further engineer it into liquid fuels. Carrying the analogy one more stage, at a price of ten cents per ton by 2025, we would use peanut butter in place of asphalt to pave roads![1]

1 The "Miracle" of High Performance Computing

The cost of computing has been on a curve similar to this for more than two decades. Today's smartphones (of

[1] The author was inspired for this example by Dean R. Chapman's 1979 NASA Dryden lecture.

which IDC estimates 0.6 billion units were sold in 2012) surpass the processing power (1 Gflop/s) of the world's most powerful supercomputer of two decades ago, as documented in the Top500 list[2]. Though the Top500 list documents a specialized benchmark from dense linear algebra, which is artificial and of limited relevance at very large scales, it is a surprisingly good proxy for the Gordon Bell prize, awarded nearly annually since 1988, which, like the Top500 list, documents an impressive factor of one thousand in floating point performance on real scientific applications every decade, for a factor of more than ten million overall since the prize was first awarded. A related prize, awarded less frequently by the same ACM Gordon Bell Prize Committee, tracks price per performance and shows a reduction of almost a factor of one million in the cost per (double precision) Gflop/s of acquiring computers employed on a winning scientific application over a two decade period from 1989 to 2009, with the most recent large improvement coming from the advent of GPGPUs.

As a result of these dramatic successes in processing and corresponding improvements in data transmission and storage, fields of endeavor never considered in the realm of supercomputing decades ago, such as telecommunications, consumer products, finance, inventory and fleet management, entertainment, and so forth, have become highly dependent on cheap processing and flows of data. "Smart" worlds are aggressively under development: smart highways, smart farms, smart oil fields, smart utility grids, smart medicine, and even smart educational systems! A characteristic of "smartness" in this sense is the dynamic and custom delivery of services that used to be delivered steadily and monolithically. A common goal is to use a resource optimally, so that engineering margins can be minimized with guaranteed quality of service at the lowest cost. Enabling this smartness is the fusion of high performance simulation with vast amounts of data. For many users, a greater virtue than petascale processing may be access to petascale data sets stored in DRAM, on which relatively modest processing is required.

Not surprisingly, scientists and engineers, who collectively delivered this technology to the marketplace as a by-product of their own discipline-driven visions, have begun to rely on the "third paradigm" of simulation and the "fourth paradigm" of data analytics, alongside the first two paradigms of theory and experiment, in their own work. Understandably, given the standards for uncertainty quantification and reproducibility in scientific discovery and engineering design, the acceptance of the third and fourth paradigms relative to the first two has been slow and cautious. Nevertheless, it is today more common than not for a publication in an experimentally dominated field like chemistry to be backed by a simulation, and some fields, such as quantum chromodynamics or astrophysics, are essentially children of simulation.

2 The Mandate of High Performance Computing

The massive edifices of hardware, software, applications, organizational and educational infrastructure described in the previous articles of this issue, and the resulting happy ecosystem are, however, now exposed to a gulf between expectations and the ability to deliver. It is irresponsible to assume that the decadal improvements of factors of one thousand in general purpose applications will be sustainable. The purpose of this concluding discussion is to identify some vulnerabilities of the ecosystem. Properly done, we may avoid a crash of expectations and guide investment to critical issues.

In [1], following up on the arguments of 315 scientists and engineers engaged in the drafting of the 2003 report "A Science-based Case for Large-scale Simulation" [2], we summarized the major mandates for pushing to extreme scale simulation for scientific discovery and engineering design as follows:

1. Better resolve a model's full, natural range of length or time scales
2. Accommodate physical effects with greater fidelity
3. Allow the model degrees of freedom in all relevant dimensions
4. Better isolate artificial boundary conditions
5. Combine multiple complex models
6. Solve an inverse problem or perform data assimilation
7. Perform optimization or control
8. Quantify uncertainty

and we illustrated them for the oil industry. To these objectives, which presume the availability of physical models, such as conservation principles, we could add a ninth:

9. Accomplish predictions without physical models, using statistical models based on large data sets

Indeed, these are compelling reasons for continuing to push computational technology to its extremes, but it is disappointing to realize how many factors of one thousand, and therefore how many decades away from fulfillment (if we could extrapolate Top500 performance progress) many of these mandates are. Resolution must typically be refined in each of many dimensions, e. g., in a fairly isotropic problem in partial differential equations (PDEs). If the computation is explicit, Courant restrictions on the timestep force refinement in another dimension. A factor of one thousand spread over four dimensions in a wave propagation problem allows a factor of only approximately 5.6 in each dimension. This is a disappointing dividend for a ten-year wait. If we simultaneously want to replace constitutive relationships in the PDE coefficients with, say, first principles molecular dynamics, we may wait another decade. Wrapping the PDE inside an optimization loop for, say, forcing conditions on the boundary compounds the delay to the length of most individual scientific careers (!), if all is done in a brute force manner. Whereas optimal algorithms (e. g., multigrid, multipole, or fast transforms,

[2] http://www.top500.org

with operational complexity scaling linearly or at most log-linearly in data size) exist for many physics-based tasks, many scientific data mining techniques are not yet optimal in the size of the data, inasmuch as they require pairwise or higher-order operations (e. g., comparisons) of data elements with each other, in ways that are not yet apparently amenable to exploitation of hierarchy. The optimal algorithms listed above have in common the use of a hierarchy of scales from fine to coarse or high to low wavenumber, so that data at each physical range, in turn, enjoy locality of memory. We conclude from this paragraph that the most important scientific breakthroughs are not going to come from miraculous computer engineering alone, but from better mathematical formulations or more realistically posed objectives.

3 The Mirage of High Performance Computing

Even assuming that scientific objectives could be satisfied with today's "brute force" methods by waiting for hardware advances, extrapolating exponentials is always unwise. Effects that are ignorable and are therefore almost forgotten over many e-folding times eventually undo the assumptions. Moore's Law, which anticipates an exponential increase in linear time with constant power density of digital gate density on CMOS chips, may well last into another (fifth or even sixth) decade. However, leakage currents and the resistivity of correspondingly miniaturized copper vias can no longer be neglected at smaller dimensions, so packing in more transistors is moot.

Similarly, core-level parallelism, which has risen to more than 1.5 million in the Blue Gene/Q "Sequoia" machine at Lawrence Livermore National Laboratory, has outpaced the programming model of the Message-Passing Interface (MPI), in which the international scientific community has an investment of an estimated billion dollars of software. Memory buffers and coordination overheads for that many interacting processes under arbitrary communication patterns are too great. The natural mitigation is to adopt hybrid programming models in which MPI is used on the outside, at the level of nodes, while within a node, cores sharing memory use one of the many possible models for multiple threads. Unfortunately, scientific codes for which machines like Sequoia are designed are often memory bandwidth-limited in critical kernels and expanding the number of cores within a distributed memory element saturates quickly for enhanced performance. Lack of portability of shared memory models is another discouragement.

Whether distributed or shared, programming models must progress beyond the current models, which typically employ pre-ordered loops and deep sequences of routines, and are often synchronized at each calling interface. This style of coding, which has many virtues, unfortunately both over-orders and over-synchronizes relative to more relaxed dynamic data flow implementations. Nearly all scalable code today follows the "bulk synchronous parallel" (BSP) model, observed in [3] to bridge software and hardware in the sense of providing a hardware-oblivious target for programmers and a software-oblivious target for architects. BSP programs are easy to understand since they express computation and communication as iteratively executed synchronized "supersteps". The BSP model is appropriate for applications in which high concurrency with good load balance can be achieved, and it has been refined and extended to many challenging applications, including those with time-varying loads and adaptivity. However, currently contemplated exascale architectures will lack *hardware performance* reliability, which means that precision load balancing is rendered moot by the wide variations in the rate at which instructions are processed. The lack of performance reliability comes from the need to operate the hardware very close to the noise level, and also very close to the thermal dissipation limit, meaning the computations may need to be redone or that portions of the chip may need to be dynamically slowed or powered off.

Throughout a century or more of numerical analysis, significantly predating digital computers, the premium has been on attaining a given level of accuracy while conserving floating point operations, which was a natural proxy for minimizing runtime and minimizing the energy of computation. Semiconductor industry roadmaps now show, however, that by the end of the decade, the energy cost of transmitting the data (from deep local memory or from another processor) will be an order of magnitude or more greater than the cost of the arithmetic, which it already exceeds even on current commodity architectures. The cost in time of finding and moving data is already greater than the cost of the arithmetic. By the most basic of engineering design principles – optimize the most expensive operations first – today's "optimal" algorithms have been designed with the yesterday's objectives in mind and should be completely revisited with the objective of minimizing data motion, even at the cost of considerable extra arithmetic. The popular mantra "flops are free" is deceptive because some flops require moving data, but it is certainly true that counting flops may soon have little to do with measuring the efficiency of an algorithm.

Another of the many implications of the hardware stresses at the exascale is that there will be pressure on today's default use of high precision in order to operate far from the stability limit of numerical algorithms in the presence of floating point rounding error. In order to conserve memory capacity, memory bandwidth, and power, algorithms will need to be rewritten to compute and communicate mainly the "deltas" between quantities that are updated, rather than the full quantities, themselves. This requires a major rethinking and certainly a recoding of numerical algorithms. Numerical analysts are not accustomed to negotiating the types of tradeoffs that effective use of emerging hardware may require, including tradeoffs of extra storage versus extra computations, extra communications versus extra computations, and waiting for delayed communications versus proceeding speculatively with the possibility of rollback.

Tomorrow's high performance hardware is predicted to fail (unrecoverably) more frequently, reducing the average runtime between failures relative to today. Coordination libraries like MPI presume that all processes are up, and failure of one process generally implies restart from the last checkpoint. In contrast, tomorrow's algorithms and programming models will have to be able to execute resiliently past multiple random hardware failures, either by reconstructing missing data and rebalancing work, or by bounding the errors due to what is missing. There are interesting prospects for the latter in many computations, but algorithms and analyses will need to be developed by a generation of computational scientists not being trained with such contingencies in mind.

To make up for the increasing gap between a high-level expression of a scientific or engineering task and its dynamic execution on unreliable, massively distributed hardware, the software stack will need to add layers. The additional layers will be required for programmability, portability, and performance. More source-to-source translations will occur between the layers and more autotuning will be required within and between layers. The complexity of this layer decomposition and tuning is not yet understood nor fully taken into account, if predicting that exascale performance on real applications will quickly follow the availability of exascale hardware.

Computer scientists have approached the next computational frontier with great trepidation before, only to find that the radical reinventions thought to be required to attain it were unnecessary. Thus, for instance, the Petaflops workshops conducted in 1995–1997 under the leadership of Thomas Sterling, raised a Hybrid Technology MultiThreaded (HTMT) candidate along with a Commodity Off-The-Shelf (COTS) candidate for the petascale, out of concern that the COTS approach might fail one or more of the numerous challenges: buildability, programmability, affordability, etc. HTMT, itself, depended upon four high-risk technologies: post-CMOS superconducting rapid single flux quantum (RSFQ) logic capable of approaching a THz cycle time, processors-in-memory (PIM) DRAM storage, a "data vortex" optical interconnect network that could store data on the fly, and 1 PetaByte of holographic store. However due to its powerful elements, it did not require discovering and coordinating as much concurrency in scientific programs as did the COTS approach. (See [4; 5] for interesting projections for the first petascale computers predating their arrival by more than a decade.) Ultimately, the petascale was attained handily by three separate designs from IBM *alone* (let alone a plethora of designs from other vendors): the Roadrunner based on the SIMD Cell processor, the BlueGene based on massively parallel low-frequency cores, and the Power series based on a smaller number of high-frequency, but still conventional, cores. While the programming model for the Roadrunner appeared radical, attached processors with their own code and their own compilers are now commodity. Other petascale designs required no major departures from the contemporary message-passing canon. These successes, however, may have been the last under *Pax MPI-BSP*. The exascale is fundamentally harder than the petascale in that there are electrical power and heat dissipation limits that will not permit simple multiplication by a factor of one hundred of today's 10 Petaflop/s technology, even if all programmability and performance issues can be accommodated along that trajectory.

4 Conclusion

Many "grand challenges" of computational science, in simulating complex systems, await the increasing power of high performance computing. It can be argued that any system that can deterministically and reproducibly simulate another system must be at least as complex as the system being simulated. Therefore, the high performance computational environment is, itself, the most complex system to model. The grand challenge of *computer science* is making this complex simulation system sufficiently manageable to use that scientists who are expert in something else (e. g., quantum chemistry, geophysics, electromagnetics) can employ it effectively. The greatest mirage of HPC may be in the assumption that we can bridge the gap between machine complexity and human users. Certainly, this is the challenge that remains after all of the components in the chain are delivered.

As with a naturally occurring mirage, it will continue to be difficult to distinguish between wish and reality until we get closer.

References

[1] David E. Keyes, "The Exascale: Why and How", Comptes Rendus Acad. Sci. Mechanique, **39**:70–77 (2011).

[2] David E. Keyes, editor-in-chief, "A Science-based Case for Large-scale Simulation", U.S. Department of Energy, http://www.pnl.gov/scales (Volume 1, 2003 and Volume 2, 2004).

[3] Leslie G. Valiant, "A bridging model for parallel computation", Communications of the ACM, **33**:103–111 (1990).

[4] Thomas Sterling and Ian Foster, "Proceedings of the Petaflops Systems Workshops", Technical Report CACR-133, California Institute of Technology, Oct. 1996.

[5] David H. Bailey, "Onward to Petaflops Computing", ACM Communications **40**:90–92 (1997).

Received: January 24, 2013