

The Next Level of Simulations: Extreme Computing

Anne C. Elster

Norwegian University of Science and Technology(NTNU), NO-7491 Trondheim, Norway

Tel: +47 7359-3674, elster@computer.org, <http://www.idi.ntnu.no/~elster>

KEY WORDS: Extreme computing, simulations, future predictions.

ABSTRACT

Computer simulations continue to play a vital role in science and engineering spanning areas from physical simulations based on numerical models (e.g. modelling weather systems, charged toner particles in photocopiers, oil wells, etc) to discrete event simulations modelling processes and complex systems (e.g. production planning and control, smart card phone system, network protocols, inventory systems, etc.)

On the one hand, with the power of "yesterday's" supercomputers now available on the desktop and in embedded systems, complex simulations can now be integrated with prototypes and real-world products blurring the borders between simulators and systems being simulated.

On the other hand, our yearning for even more computational power to solve even larger problems and/or doing increasingly more accurate simulations continues to grow. No doubt tomorrow's high-performance supercomputers will hence also play a central role in high-end simulations.

This paper introduces the new term: **Extreme Computing** – computing where computations no longer are done primarily at an HPC (High-Performance Computer) host system and/or cluster/grid, but where sensors, actuators, and system components themselves have dedicated processors that perform a good portion, if not the bulk of the computations.

The author gives some highlights of her experiences with various types of simulations and libraries, including PETSc, and provides some predictions on how Extreme Computing will take tomorrow's simulations to the next level. Her "Ant Theory" (even sugar ants display more complex planning and data processing than any human-built system of today) tells us we also have several more levels beyond that to conquer. Through newer technologies such as nanotubes and Bose-Einstein condensates, our future should even prove Moore's law wrong, and let the simulations tackle even bigger and more complex problems.

1 INTRODUCTION

Computer simulations have played and continue to play a vital role in science and engineering. Although my personal experiences are primarily in the area of physical simulations based on numerical models (e.g. charge

particle simulations) I have also touched on discrete event simulations which model processes and complex systems (e.g. smart card phone system and network protocols) as part of my experience at Schlumberger in the mid-1990's. Both types of simulations will continue to play vital roles in the future as simulations become increasingly more intertwined in not only research and development, but in our every-day lives.

Some of the highlights from my graduate work will be covered in the next section as an introduction to high-end physical simulations. Section 3 discusses current work related to simulations using computer clusters, MPI and PETSc – some of the "hottest" technology available today. I discuss grid computing and Extreme Computing in Sections 4 and 5 including several of my predictions for the future regarding computer simulations. Section 6 includes predictions regarding future processor technologies and how they will impact simulations. Finally, I end with my "Ant Theory" of Section 8 that tells us we have a long way to go.

2 HPC SIMULATIONS AND PARTICLE CODES

Simulations have always been central to High-performance computing (HPC). This section describes some of the systems and codes of the mid-1980's and early 1990's – the dawn of commercial supercomputing using distributed memory systems.

A nice overview of the early days of HPC including pioneering work by the Bell, Gray, Cray and others dating back to the 1950's was recently written by Bell and Gray [3].

2.1 Hypercubes, Butterflies and Transputers

Introduced in the mid-1980's, the IPSC (Intel Personal Super Computer) hypercube was one of the first commercial supercomputers based on distributed memory processing units interconnected via special hardware switches.

Around the same time, several parallel systems based on the Inmos Transputer, were also popular, especially in Europe. Unlike the Intel IPSC whose interconnection grew logarithmically with the number of processors, the Transputer consisted of CPUs with 4 serial connectors for each CPU, i.e. 4-way switches. [4]

Cornell University became in 1985, the year I joined as a graduate student, one of three NSF centers for supercomputing in the United States. Intel hypercubes, IBM 3090s (vector based supercomputers), and a BBN Butterfly were among their early high-end systems. The BBN Butterfly, named after the company's founders, MIT professors Richard Bolt and Leo Baranek and Bolt's student Robert Newman, was one of the first distributed memory machines with virtual shared-memory access. It had 32MB caches, but no primary memory and also provided direct hardware support for block transfers. BBN Technologies [5] is well known for designing the routers for ARPANET, the precursor to today's Internet.

In 1986, I got to work on the first Intel IPSC hypercube outside the United States at Christian Michelsen's Institute (CMI) in Bergen, Norway, where my first task was to write a matrix-vector multiplication routine

for their computational library aimed at petrochemical related codes [6]. Petrochemical simulations continue today to make heavy use of supercomputers. Directional drilling, a new technology where the drill bit is guided by real-time geophysical simulations based on sensory feed-back, is an example of how high-end simulations continue to push the envelope of technology.

2.2 The Communication Bottleneck

Communication overhead was easily traced on early hypercube models since it did not overlap at all with computations. This, of course, changed when separate communication processors were introduced on later models, and one then had to see if one could "eliminate" communication costs by doing alternate computations as one waited for data from other processors. In practice, this has become harder and harder to achieve since communication networks have not kept up with processor speeds. Today, communication times across multiple CPUs are several orders of magnitude slower than computation speeds.

Unfortunately, this communication bottleneck is hence also even more prevalent today both with respect to accessing different levels of the memory hierarchy on single processor or shared memory systems as well as for distributed memory systems.

2.3 Particle Codes

Particle simulations are fundamental in many areas of applied research, including astrophysics, plasma physics, semiconductor device physics, and Xerography. These simulations often involve tracking of charged particles in electric and magnetic fields. My dissertation work [7] involved modularizing and implementing a functional parallelized particle code.

At the time of my dissertation, these simulations had, due to their high demand for computer resources (especially memory and CPU power), been limited to investigating local effects – typically using up to an order of 1 million particles. By developing novel algorithmic techniques to take advantage of modern parallel machines such as the Kendall Square Research (KSR) machine and employ state-of-the-art particle simulation methods, my PhD work targeted simulations with 10-100 million particles in order to study interesting larger-scale physical phenomena. Other numerical codes that target parallel computers would also benefit from many of the techniques developed in this work.

Our application fell in the category of collisionless systems, where each simulation particle, often referred to as a "superparticle", represents millions of physical electrons or ions in a collisionless plasma. The numerical techniques used usually involve assigning charges to simulated particles, solving the associated field equations with respect to simulated mesh points, applying the field solution to the grid, and solving the related equations of motion for the particles. Codes based on these numerical techniques are frequently referred to particle-in-cell (PIC) codes.

2.4 Threads vs. Processes

One of the nice things about the KSR and other parallel systems in the 1990's was that they started to use threads (light-weight processes) rather than fork regular (heavy-weight) processes. This meant a lot of operating system (OS) kernel overhead was now avoided in parallel programs. A further discussion of thread vs. processes was discussed in the introduction of my PhD thesis [7].

2.5 Testing Large Simulation Codes

In addition to the standard test one would do for each numerical module as a computer science/math person, I took the testing of my PIC code one step further by plugging in real physics constants and running the code with initial conditions leading to known phenomena.

One such test involved the plasma frequency which is observed in real plasmas. A detailed description of the physics behind plasma frequencies and how we set up our tests were described in the thesis [7].

To further test whether our codes were able to simulate physical systems, we also performed a two-stream instability test [8]. In this case, two sets of uniformly distributed particles (in our case 2D particle grids) are loaded with opposite initial drift velocities. Detailed knowledge of the non-linear "eye" behavior associated with such simulations was developed in the 1960's.

3 THE NEXT STEP: Cluster Computing, MPI and Numerical Libraries

A recent trend in HPC – using a network of workstations and/or PCs rather than traditional supercomputers – is gaining momentum. Although current network speeds are even slower compared to available processing speeds, and implementing distributed memory programs is generally much harder than shared-memory programs, the price and flexibility of such systems should not be underestimated.

This view was also recently echoed by Bell and Gray [3] in their Feb. 2002 article entitled "What's Next in High-Performance Computing".

Satellite imaging systems and sensory device such as MRIs, radars, etc, etc, will no doubt continue to generate huge datasets that will not fit in memory of a high-end work station. For many large simulations, especially those that crunch on such large datasets, cluster computing hence has a lot to offer. By fitting these datasets on distributed RAMS, the clusters help eliminate the memory-disk bottleneck provided the data can be crunched fairly independently. A major challenge still is to find algorithms and techniques that minimize communications overhead.

Since programming on parallel and distributed systems such as clusters efficiently is much more complicated than on single CPUs, newer standards such as MPI as well as access to pre-parallelized numerical libraries such as PETSc, greatly improve the potential for the success of such systems.

Since I believe MPI and PETSc will play very central roles for high-end simulations in the future, I have described them in more detail in the next two subsections.

3.1 The MPI (Message Passing Interface) Standard

[10] [11] is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users in the mid-1990's.

Although MPI was initially intended as a standard for message passing calls (send, receive, broadcasts, etc) on commercial supercomputer systems, MPI is now often used on clusters of workstations. The author was one of the proponents on the MPI Forum lobbying for dynamic processing capabilities. This effort led to the MPI 2.0 standard (1997) that includes dynamic processes, parallel I/O and other features. MPI originally specified C and Fortran interfaces, but is today available for C++. OpenMP, an MPI-based implementations for shared-memory systems, and MPI Java, are recent additions.

Indeed, my own work is currently focusing on cluster computing using MPI. E.g. my recent work with Paul Sack [13] involves improving the implementations of broadcasts in MPI by using Multicasting instead of the standard broadcast tree. Current work also includes benchmarking models for clusters.

I've also used MPI on clusters successfully in several class settings when teaching students parallel programming.

MPICH, one of the most popular MPI implementations, is available on the Web for a variety of platforms from

<http://www-unix.mcs.anl.gov/mpi/index.html>

3.2 PETSc

PETSc is a versatile MPI-based software package from Argonne National Laboratory, that can be used to develop, implement and test numerical-based simulation codes. PETSc (Portable, Extensible Toolkit for Scientific Computation) [14] provides several modules for solving partial differential equations (PDEs) and related problems on high-performance computers.

PETSc is a large and complex package that is known to scare off several users for its apparent complexity and steep learning curve. However, it comes with an extensive User Manual [14] that is very well written once one "digs" into it. The support scripts for both the serial and parallel installations are fairly easy to modify and work well.

A non-optimized complex Chebyshev code [15] using PETSc was developed with my help by one of my students in a graduate class in 1-2 weekend! Note that this project involved making changes and additions to the PETSc source code as well as installing the package and running several of the Krylov routines for comparisons.

PETSc consists of a suite of data structures and routines that provide the building blocks for the implementation of large-scale physical simulation codes on both serial and parallel computers. PETSc is still under development and currently includes several parallel and non-linear equation solvers, unconstrained minimization modules, and lots of support routines. It provides many of the mechanisms needed within parallel application codes including parallel matrix and vector assembly routines. Although PETSc also support Fortran, the designer as well as this author recommends that one codes ones routines in C or C++ for maximum flexibility and portability on high-end parallel systems. The package uses routines from BLAS, LAPACK, LINPACK, MINPACK and SPARSPACK (the last 3 in C using ftoC) and provides interfaces to BlockSolve95 (parallel ICC(0), ILU(0) preconditioner), ESSL (IBM), Matlab (sockets interface for graphics and numerical post processing of data), ParMeTis (parallel graph partitioner), PVODE (parallel ODE integrator) and SPAI (Parallel sparse inverse preconditioner). Routines related to data structures include index sets (block-indexing and stride), a vector library, and storage routines for matrices (including compressed sparse row, block-compressed sparse row, block-diagonal and dense)

Krylov subspace methods already provided by PETSc include GMRES, Conjugate Gradient, CGS, Bi-CG-Stab, TFQMR, Richardson and Chebyshev. PETSc also provides several popular preconditioners including Additive Schwartz, Block-Jacobi, ILU, ICC, and LU (sequential only).

I hope that the reader gets inspired by this subsection to use PETSc as a tool to get started when developing their own implementations of research and production simulation codes using PDEs, or other routines provided by PETSc.

For a list of several current projects, products, applications and tools for scientific computing, check out the Webpage maintained by the PES (Problem Solving Environment) group at Purdue:

<http://www.cs.purdue.edu/research/cse/pses/research.html>

4 FROM GRID COMPUTING TO EXTREME COMPUTING

As can be seen from the previous sections, simulations have come a long way from their roots where monolithic computer systems sitting in a huge room and/or building were used for physical simulations. The Grid and related technologies, including wireless networks, are changing the way we think about computing.

4.1 Heterogeneous Clusters and Grid Computing

As clusters become heterogeneous networks of a variety of PCs, high-end workstations and more traditional supercomputer systems, as well as special-purpose systems such as graphics and visualization caves, our challenges for interoperability continue to increase.

There is no doubt in my mind that the computational grid where several of such local area networks extends across the high-end backbones of the Internet will provide a lot of challenges and opportunities for simulation students and researchers.

5 FUTURE CHALLENGES – EXTREME COMPUTING

No doubt, we will continue to refine our physical simulations of weather systems and other "Grand Challenge problems", however I predict we will also see emerging problems we have yet not fully defined.

In describing taking HPC (High-Performance Computing) and simulations to the next level, I hereby introduce a new term – **Extreme Computing**. This is computing where computations no longer are done primarily at an HPC host system and/or cluster/grid, but where sensors, actuators, and system components themselves have dedicated processors that perform a good portion, if not the bulk of the computations.

5.1 Traffic Control System

Like railway systems linked with GPS, future HPC systems may be involved in controlling cars without input from drivers after input of desired destination. The challenges here are enormous as are the potential benefits considering how many accidents could be avoided.

Real-time simulations will no doubt be required in such systems both in each car along with sensory feedback as well as at other network control spots dealing with access, re-directions around hot-spots, etc. Today's automobiles already contain a lot of computing power. Extreme Computing will hence here be key since a lot of sensory data will need to be computed and simulated locally in order to satisfy the real-time constraints. The challenges lie in the accuracy, control and reliability of such distributed systems, as well as standardizations across regions (e.g. one would like to be able to drive seamlessly from one EU country to another.)

5.2 "Super Smart Cards" and Wireless Devices

Other areas that are already receiving increased attention are the use of the ultimate distributed processors: the Smart Cards and wireless network nodes.

One idea that has been proposed is to store biometric information such as fingerprints on the cards which can then be used as a password to a wireless net, or for other real-world identification purposes. This case is a great example where instead of accessing and comparing information from a huge database, one can instead provide the clients with enough processing power and storage to retrieve the data needed. Extending this to Extreme Computing – each "super-smart" card of the future could provide complex simulation tailored to the individual with respect to everything from medical history to financial models.

Taking this another step, these "Super Smart Cards" could be interfaced with wireless monitoring devices gathering biometric data as input to, say, automated medication delivering systems that optimize dosages in real-time.

With tie-ins to financial and government institution the challenges will not only be technical, but a question of who should get access to such detailed information.

5.3 Security

Security in itself also provides lots of challenges related to Extreme Computing as well as distributed computing and the Grid. The more we protect data and simulations, the harder it will become to make them inter-operate and work at the next level – across heterogeneous platforms, sensors and wireless devices.

Standards and protocols are still under development for several systems, including wireless devices, so much more work can be expected in this area.

6 FUTURE PROCESSOR TECHNOLOGIES

No doubt, the nature of Extreme Computing will be influenced by the technology available in the future. This section predicts some of the technology trends that the author predicts will have strong impact on HPC and Extreme Computing.

6.1 Simulations on COTS

We may see an interesting trend in high-end COTS (Commercial Off-The-Shelf) processors. Assuming the Wal-marts toy stores of the world continue to embrace high-end gaming and entertainment systems, we may very well see the main processor market going from PCs to Video-game stations where Playstation chips may actually out-power Pentiums.

In this scenario, I envision the Grid no longer as wired high-end PC's, but rather mostly networked video Playstation processors, possibly wirelessly connected. Just as going from Von Neuman (single CPU and Memory) systems to systems with several memory hierarchites (cache, on-chip memory, shared memory, and/or distributed memory systems, poses several challenges, moving future simulation to non-floating point based achitectures such as video cards and many DSP (digital signal processors), will no doubt introduce a lot of challenges.

6.2 Processor Break-Throughs

I also predict that we will continue to out-shine Moore's Law. Going from today's Pentium 4 chips with 42 million transistors using 130nm technology to using emerging technologies such as **nanotubes** [18] made from strands of single atoms, each a few nanometers, to make chips crammed with billions of transistors, will no doubt continue to impact HPC.

Even further out we may see applications of Bose-Einstein condensates in quantum computing and nanotechnology. Bose-Einstein condensation involves cooling atoms to near absolute zero (0 Kelvin). The first success in this area was done by 2001 physics Nobel laureates Weiman and Cornell who in 1995 cooled rubidium atoms to about 50 nanokelvin (50 billionth of a degree above zero Kelvin). A few months later 3rd winner Ketterle succeeded similarly with a larger condensate of sodium atoms. [19]

7 FUTURE CHALLENGES AND THE ANT THEORY

Physicists thought at the turn of the 20th century that they had "solved physics", but then came quantum theory. Today's physicists no longer dare to make such finite sounding claims, however, many scientists are starting to think that solid state will be reaching its physical limits in the near future and hence bar further advances in computing.

However, to those who think that getting down to the quantum level will stop us from making any further progress in computer simulations, I offer my "Ant Theory":

It stems from when I as a little girl, no more than 5 or 6 years old, used to sit in the grass and study the little critters making their way through the world of grass blades and debris. I was perpetually amazed at how well such tiny little creatures could navigate so well through a terrain that seems incredibly rough and large for their size.

Recently, I had the same exact thoughts studying some sugar ants on my bathroom floor navigating between water drops and other obstacles sometimes pulling object larger than themselves back to their hills: How can such a tiny little creature see and navigate so well processing all that information in a system so minute?

The read answer to that question probably lies somewhere in biology and chemistry, and until we understand the connection between those fields and physics that enables such tiny organic creatures to perform so well, we have not come very far.

As the final version of this paper was prepared, *Communications of the ACM* published an interesting article referred to on the cover as "Swarm Technology" [20] which also uses the ant analogy. The article describes genetic algorithms as well as simulated annealing, and theorizes how one can achieve complex actions based on collections of simple primitives. It also provides several references to studies involving ants and other insects. However, I believe we are still a long way off from understanding what really makes organic life tick.

Moore's and other laws will continue to get broken.

References

1. "High Performance Simulators" – cover title on *Computer*, IEEE Computer Society, Feb. 2002.
2. Ruy Mesquita, Jos Dionsio, Pedro Cunha, Elsa Henriques, B. Janz: "Usability Engineering: VR interfaces for the next generation of production planning and training systems", *CG Topics*, 5/2000.
<http://www.inigraphics.net/publications/topics/2000/issue5/5_00a07.pdf>
3. Bell, G., Gray, J.: "What's Next in High-Performance Computing?" in *Communications of the ACM*, **45**, No. 2 (2002) 91–95.
4. Thiebaut, D.: *Parallel Programming in C for the Transputer*, Chapters 1 and 2, 1985 available on the WWW at:
<<http://cs.smith.edu/thiebaut/transputer/descript.html>>

5. BBN Timeline – A history of BBN Technologies available on the WWW at:
<<http://www.bbn.com/timeline>>
6. Elster, A.C. and Reeves, A. P.: "Block-Matrix Operations Using Orthogonal Trees", Proc. of the Third Conf. on Hypercube Systems and Applications, January 19-20, 1988 in Pasadena, CA, Ed. G. Fox, ACM, pp 1554-1561.
7. Elster, A.C.: "Parallelization Issues and Particle-in-Cell Codes", Ph.D. dissertation, Cornell University, August 1994. Abstract at: <<http://www.english.cornell.edu/thesesabstracts/August94/elster.html>>
8. Birdsall, C.K. and Langdon, A.B.: *Plasma Physics via Computer Simulations*, Adam Hilger, Philadelphia, 1991.
9. Hockney, R.W. and Jesshope, C.R.: *Parallel Computers*, Adam Hilger Ltd., Bristol, 1981.
10. The Message Passing Interface Standard (MPI): <<http://www-unix.mcs.anl.gov/mpi/index.html>>
11. Elster, A.C. and Presberg, David L.: "Setting Standards For Parallel Computing: The High Performance Fortran and Message Passing Interface Efforts", Theory Center SMART NODE Newsletter, May, 1993, **Vol.5**, No. 3, Cornell University.
12. MPICH - A Portable Implementation of MPI: <<http://www-unix.mcs.anl.gov/mpi/mpich/>>
13. Sack, P. and Elster, A.C.: "Fast MPI Broadcasts with Reliable Multicasting", PARA'02, in this volume of Springer Verlag's Lecture Notes on Computer Sciences, Fagerholm J. et al. (Eds), 2002.
14. PETSc (Portable, Extensible Toolkit for Scientific Computation): <<http://www-fp.mcs.anl.gov/petsc/>>
15. Elster, A.C. and Liang, C.: "Developing and Testing Linear Solvers Using PETSc", First SIAM Conference on Computational Science and Engineering, Washington, D.C., Sep 21-24, 2000. <<http://www.siam.org/meetings/cse00/cp25.htm>>
16. "Top 500 Supercomputers" – Website maintained by Univ. of Mannheim, Germany and Univ. of Tennessee at Knoxville, U.S.A.. Yearly updates presented at the Supercomputing conferences. <<http://www.top500.org>>
17. Slides summarizing trend of Top 500 Supercomputers 1993-2001:
<<http://www.top500.org/slides/2001/11/>>
18. Rotman, D: "The Nanotube Computer", *Technology Review*, **105**, No. 2, 36–45
19. "2001 Nobel Prizes", *Technology Review*, **105**, No. 2 (March 2002) 14–16
20. Schneider, D.: "Swarm Intelligence: Power in Numbers", *Communications of the ACM*, **45**, No. 8, August 2002