Introduction to High Performance Computing

Problem Set #3

---

1) Computational Science "Grand Challenges"

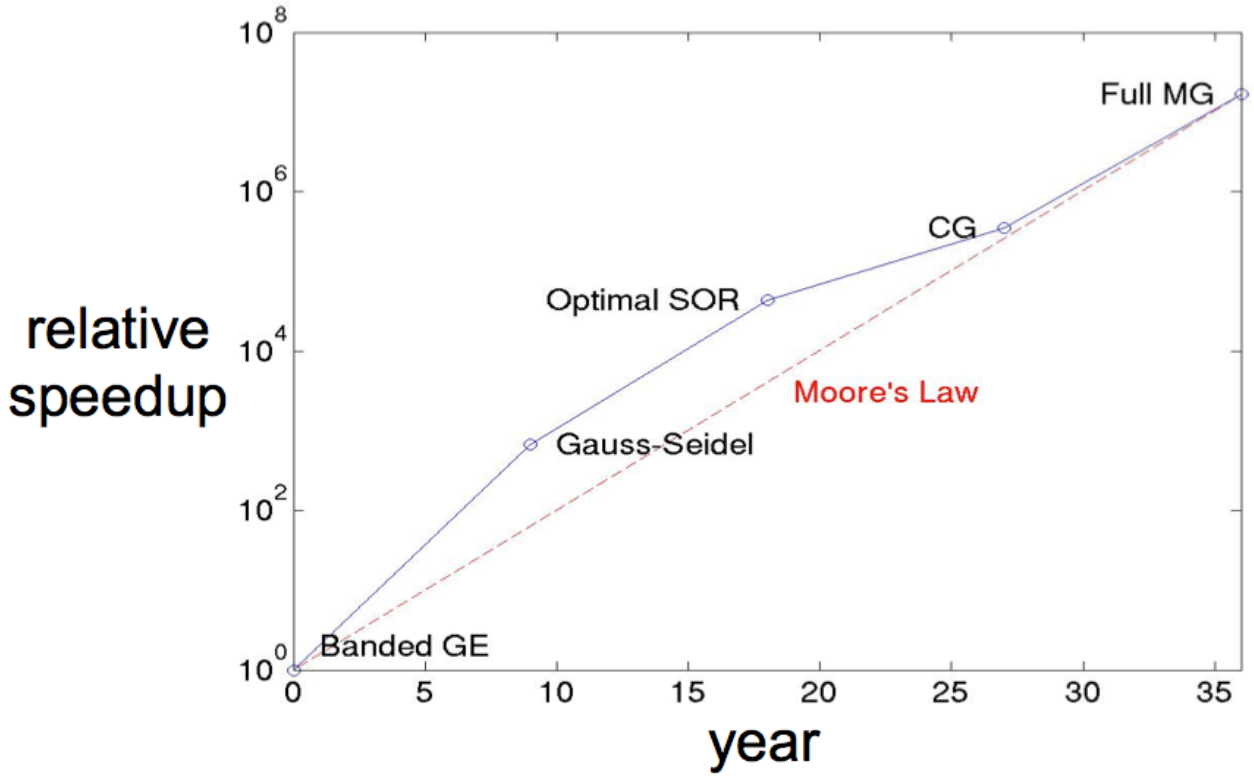
(a) List the six fields in science or engineering that Nobelist Kenneth G. Wilson in his seminal 1989 article (distributed in the course repository) singles out as examples for which a computational approach can pay big dividends. For each of the six fields, list one advantage that a computational approach has over an experimental/observational or a theoretical approach. (These advantages could be your own ideas or you could identify one of Wilson's.) Also, for each field, list one of the most significant limitations (at the time, at least) to a computational approach. (You can repeat a reason from one field to another if you feel that it is one of the most significant limitations to each.)

(b) List four concerns Wilson voices about the ability of computational science to live up to its potential.

(c) Thinking outside of Wilson's 1989 paper, what concerns do *you* have about the ability of computational science to live up to its potential in scientific discovery and technological advancement in 2016?

2) Algorithmic complexity and "Algorithmic Moore's Law" for Poisson's equation

Unit #4, Part 2, slide 41 contains a graph (reproduced below) schematically depicting the relative speed-up achieved over equal periods of 36 years, from algorithmic advances and from the architectural improvements known as Moore's Law [G. Moore, 1965] (dashed line). This figure, dubbed "Algorithmic Moore's Law" is from Volume 1 of the report *A Science-based Case for Large-scale Simulation*, by D. Keyes (ed.-in-chief) *et al.*, commissioned by the U.S. Department of Energy in 2003. It is based on a figure in the first HPC "bluebook" published by NITRD in 1992, as shown in Unit #3, slide 42. The graph concerns solving the Poisson equation on a uniformly gridded cubic domain, with n cells on each side. The algorithms are among the "Seven Dwarfs" of Berkeley, and/or the "Top 10" algorithms of the 20^{th} century, or the "Scalable, Hierarchical Five."



The algorithms, with their asymptotic arithmetic and storage complexity, are:

- Banded Gaussian elimination, a sparse direct method, arithmetic: $\mathcal{O}(n^7)$, memory: $\mathcal{O}(n^5)$
- Gauss-Seidel iteration, a sparse iterative method for structured or unstructured grids, arithmetic: $\mathcal{O}(n^5 \log n)$, memory: $\mathcal{O}(n^3)$
- Optimal successive over-relaxation (SOR), another sparse iterative method, arithmetic: $\mathcal{O}(n^4 \log n)$, memory: $\mathcal{O}(n^3)$
- Modified incomplete Cholesky preconditioned conjugate gradient (CG), a Krylov sparse iterative method, arithmetic: $\mathcal{O}(n^{3.5} \log n)$, memory: $\mathcal{O}(n^3)$
- Full multigrid (MG), an optimal algorithm, arithmetic: $\mathcal{O}(n^3)$, memory: $\mathcal{O}(n^3)$

These complexities for the iterative methods assume that we converge algebraic error of the discrete problem down the truncation error of the discretization. In other words, as the grid has more resolution, we work harder to take advantage of it, keeping in mind the triangle inequality (Unit #3) for overall error.

(a) Draw a new version of the graph depicting the progress of algorithms for solving the Poisson equation and the progress of Moore's Law. Instead of basing algorithmic speedup in the chart on a cube of $n = 64$ uniform mesh cells on each edge, base it on a cube of size $n = 1024$ cells on each edge. Does the architectural (Moore's Law) speedup baseline change as the problem size changes?

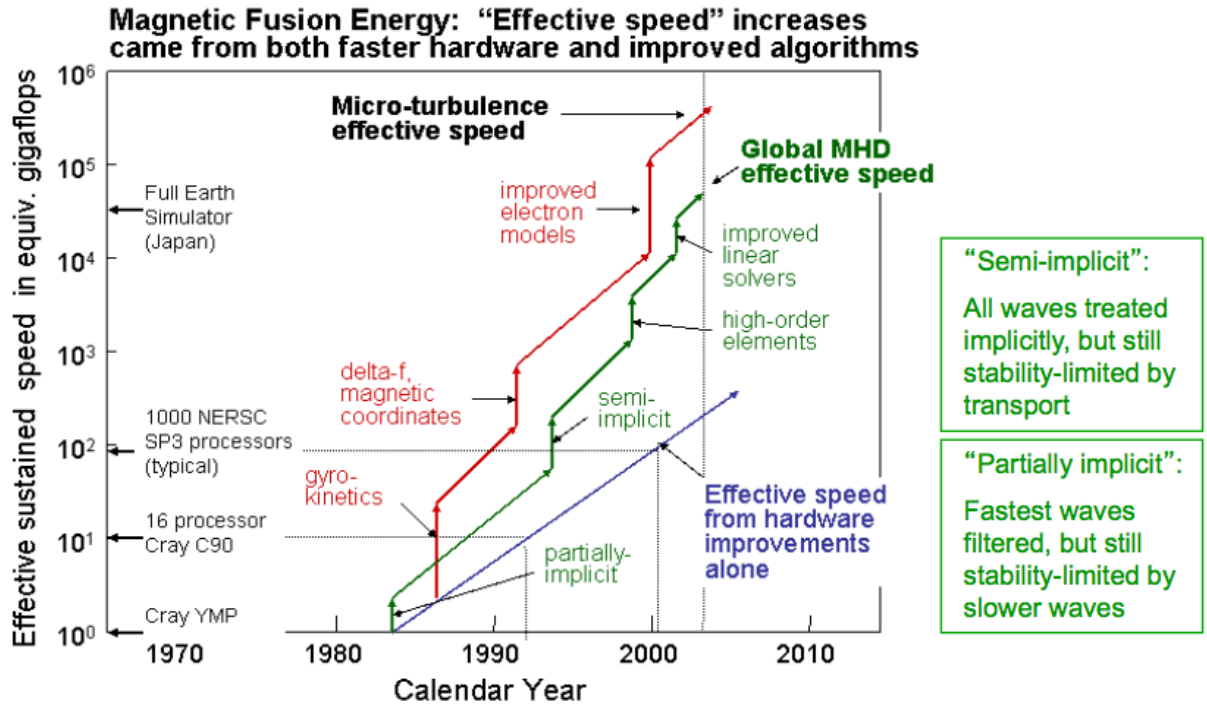
(b) How much memory (in Bytes) is required to store a 64-bit floating point number in each cell of a cube 1024 cells on each edge?

(c) How much memory (in Bytes) is required to hold the working size of the computation (memory complexity) if banded Gaussian elimination is employed as the algorithm for the cube with 1024 cells on each edge?

(d) If a computer processor-memory system is capable of executing at the delivered rate of 1 Teraflop/s on a typical linear algebra problem, about how long will the Gaussian elimination algorithm of part (c) take to solve the Poisson problem for the 1024-cube?

3) Algorithmic "Moore's Law" for Fusion Energy

The graph below on Moore's Law (blue line) and algorithms for magnetically confined fusion energy simulations is from Volume 2 of *A Science-based Case for Large-scale Simulation.*



(a) The magnetohydrodynamics (MHD) simulations (green curve) represent an application of partial differential equations. Over a period of years, the formulation stayed roughly the same, but the numerical algorithms improved, on top of steady architectural improvements. Between the years 1984 and 2004 approximately what factor of the effective speedup of MHD was due to algorithmic advances? According to the labels on the graph, what were these advances?

(b) The micro-turbulence simulations (red curve) represent an application of particle methods. Over a period of years, the algorithms stayed roughly the same, but the formulation became more efficient for the physics at hand, on top of steady architectural improvements. Between the years 1984 and 2004 approximately what factor of the effective speedup of micro-turbulence was due to formulational advances? According to the labels on the graph, what were these advances?

(c) What (remaining) factor do we attribute to Moore's Law for semiconductors?

4) Floating Point Representability

The "machine epsilon" (also called *macheps*, machine precision, or unit roundoff, designated ϵ) is, for a particular processor offering floating point arithmetic, the smallest number that, when added to 1, gives a number greater than 1 when stored. The existence of a machine epsilon is a consequence of finite-precision floating point arithmetic. A computer cannot distinguish between two real numbers that differ by machine epsilon or less; their difference "falls off the end" of the mantissa, a consequence of one of the triple-finiteness aspects of computers (see Unit #3, slide 59). The machine epsilon depends on the rounding rules used in the floating point arithmetic. If the rounding is by truncation, then $\epsilon = b^{(1-p)}$, where b is the base (usually 2) and p is the number of bits in the mantissa. If rounding is to the nearest representable number, then a factor of one-half is saved: $\epsilon = \frac{1}{2} \cdot b^{(1-p)}$.

(a) A floating point number system offers a mantissa with 53 bits. To within a power of ten, what is the machine epsilon?

(b) A problem to be solved has a condition number that grows as the fourth power of the reciprocal of the mesh size; that is, $\kappa(h) \sim \frac{1}{h^4}$. At about what size h (or below) will the machine with the machine epsilon of part (a) computer be unsuitable for solving such an ill-conditioned problem? [Hint: Use the equation

$$\frac{||\tilde{x} - x||}{||x||} \leq \kappa(A) \frac{||\tilde{b} - b||}{||b||}$$

from slide 39 of the Unit #2 lecture notes to estimate how, in the worst case, an error in the machine representation of input b can be blown up to a unit-sized error in output x.]

(c) Assuming that the h from part (b) characterizes the cell size of a three-dimensional cubic mesh, about how many cells can the cube contain before the ill-conditioning makes it unsuitable to solve such a problem on this computer?

5) Convergence scalability of Krylov-Schwarz

Suppose a Schwarz-preconditioned domain decomposition method has a condition number of

$$\kappa = C(1 + \log^2(H/h)),$$

where H is the diameter of a subdomain and h is the diameter of a mesh cell. C is a constant that is independent of these partitioning and discretization parameters. To parallelize, we map one subdomain onto one distributed memory processor.

(a) How does this condition number behave in the limit of "weak" processor scaling, defined as when the mesh and the processor decomposition are refined together in proportion, to keep a fixed size algebraic problem on each processor?

(b) What does this imply for number of iterations required for a Krylov-Schwarz method to converge in the weak processor scaling limit?

6) Teamwork in Computational Simulation

In 200 words or less, present a problem for computational modeling in the style of D. O'Leary's 1997 article (distributed in the course repository) – that is, add a fifth example to her article, from a field of expertise of your own – or one in which you hope to become an expert. Use one figure to illustrate the central scientific issue and one piechart to suggest relative proportions of effort from different fields, Feel free to interview (and cite) an "expert," if you wish, but put the results in your own words and graphics.

7) Parallelization of a Top Ten Algorithm

Select one algorithm in *The Top 10 Algorithms of the 20th Century* by J. Dongarra and F. Sullivan (IEEE Computing in Science and Engineering **2**:22–23, 2000) or *Top 10 Algorithms in Data Mining* by X. Wu, V. Kumar, *et al.* (Knowledge and Information Systems **14**:1–37, 2008), both distributed in the course repository, and describe the four stages of its parallelization, according to the standard schema of Culler, Gupta & Singh.