Introduction to High Performance Computing

Problem Set #1

---

1) Implications of Amdahl's Law

In a three-page paper, *Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities*, AFIPS Conference Proceedings **30**:483–485, Gene Amdahl presented the very simple "law" that bears his name and gives an upper bound on the utility of increasing the size of the processor force employed in solving a fixed-size problem. Every student of parallel computing must begin with an appreciation for this law and should read Amdahl's original (as uploaded to the course website).

(a) Build a table that fleshes out Amdahl's Law in its simplest form as follows. Let all work in an algorithm for execution fall into two categories, a fraction a that cannot be parallelized at all, and a remainder that can be parallelized to an arbitrary degree. Across the columns list values of a, spanning the range from 0 to 1. Down the rows list the number of processors, increasing from 1 to some commercially mainstream number, say 1024. Fill in the table with the maximum possible speedup that is realizable. Use 11 rows and 11 columns and think about and *choose interesting values* for a and p (perhaps not equally spaced). Make a clearly labeled graph of two tabular cross-sections: one that reads down the middle column, and one that reads across the middle row. Interpret the results of this exercise in a few sentences.

(b) With a defined as above, show from Amdahl's Law that the parallel efficiency, defined as the speedup divided by the number of processors, $E = T_1/(pT_p)$, is $1/(ap + (1 - a))$.

(c) If the problem size is a constant (so that a does not change) but p increases, efficiency must usually degrade. If, however, a decreases as problem size increases, we may be able to add processors in proportion to some power of problem size and maintain a constant efficiency, a phenomenon called "scalability." Give an example of a possibly "scalable" computation. Answer in a few sentences — you do not have to give application specifics or implementation details, but just point out where the differential growth comes from. (Hint: think of problems whose parallelizable component of work grows like a high power of the problem size and whose nonparallelizable component of work grows like a smaller power. For instance, if parallel work grows like the volume of a region and nonparallel work grows like the surface of a region, you're all set, since volume grows like the $\frac{3}{2}$-power of enclosing surface for typical nonfractal objects.)

(d) Next, generalize Amdahl's Law to more than two classes of work. Assume, that there are P classes of work, for some integer $P > 1$, and fractions a_1, a_2, ..., a_P of work in each class, such that $\sum_{p=1}^{P} a_p = 1$, and such that fraction a_p can be parallelized perfectly up to p processors, but no more. (Note that for the simple form of Amdahl's Law, we in effect take $a_1 = a$, $a_P = 1 - a$, and $a_p = 0$, $p = 2, \ldots, P - 1$.) Provide formulae for the speedup and the efficiency on p processors, $p \leq P$.

2) Pipelining

(a) Following the example of the laundry room pipelining in the example of Unit #1, Part #1, with the specific execution latencies there described for washing, drying, and folding, what is the maximum parallel efficiency with which the three stations can be used, not counting pipeline fill and pipeline drain effects?

(b) Show how the asymptotic efficiency of part (a) is approached by tabulating the efficiency for one student using the three stages sequentially, two students using the three stages with maximal overlap, three students, and so forth. With how many students does the efficiency come within 10% of that which is achievable.

3) Asymptotic order estimates for optimal processor number

A very primitive model for wall-clock execution time of a program on a parallel computer with p processors is

$$T(p, n) = \frac{a(n)}{p} + c(p)$$

where $a(n)$ is a measure of the arithmetic complexity (typically a monotonically increasing function of problem size, n) and $c(p)$ is a measure of the communication complexity (typically a monotonically increasing function of the number of processors). By setting to zero the partial derivative of T with respect to p, and solving for p in terms of n, estimate how many processors should be used to minimize execution time as a function of problem size. For definiteness, assume $a(n)$ is linear in n. Consider for $c(p)$:

(a) a constant,

(b) a logarithm in p,

(c) a square root of p, and

(d) a first power of p.

4) Explicit stencil computations

(a) Following the example for the discretization of the law of conservation of mass in a fluid in the slides in Unit #1, Part #1 write down an explicit update for the density in a typical cell (i, j, k) at time level $\ell + 1$, based on the equation of continuity (conservation of mass) discretized on a uniform Cartesian lattice, using first-order differences and averages, assuming that the density is known at the center of every neighboring cell (those cells exactly one of whose i, j, or k values differs by ± 1) at time level ℓ, and assuming that each relevant velocity component is stored at the same locations.

(b) How does this rule have to be supplemented at cells on the boundary of the computational region? What is a reasonable assumption to make there, and how many types of such special cases are there, based on the three-dimensional geometry of the region, assuming that you are writing a code that loops over all cells performing this update?

5) Benchmarking the top-ranked computers

(a) The latest (as of this writing) lists of the Top 500 computers ranked by the HPL benchmark, the Top 223 computers ranked by their "Green-ness," and the Top 80 computers ranked by the HPCG benchmark were published in June 2016. The first two rankings are available at `top500.org` and the last at `hpcg-benchmark.org`. These lists rank all published global computers in terms of computational power (flop/s) for dense linear systems, and sparse linear systems, and electrical power efficiency (flop/W) for the Green500. Investigate these lists on-line and describe exactly what algorithm is executed to obtain the ratings.

(b) List all supercomputers which satisfy the constraint that the sum of their ranks on each of the HPL list and the Green list is less than 50. Present your answer in a list with six columns as follows: (1) the sum of the ranks, (2) the rank on the HPL list, (3) the rank on the Green list, (4) the manufacturer, (5) the model name, and (6) the country in which the computer is located. Rank your list by column (1) from lowest integer sum (highest rank).

6) Algorithmic challenges for High Performance Computing

(a) According to Dongarra in *High Performance Computing* (Princeton Companion of Applied Mathematics, 2016, distributed in the course repository), what are nine of the challenges for floating point algorithms on emerging extreme scale computers?

(b) Select one of these nine challenges and write a 150-word paragraph on what architectural feature or applications requirement increases the challenge, relative to today's petascale environment. For a specific algorithm (e.g., linear system solution, evolving particles dynamically in a self-interacting force field, etc.) describe an adaptation that the algorithm might make to overcome the challenge.

7) Application mandates for High Performance Computing

(a) According to Keyes in *The Miracle, Mandate and Mirage of High Performance Computing* (Information Technology **3**:110–114, 2013, distributed in the course repository), what are nine of the "mandates" for developing extreme scale simulation capabilities?

(b) Select one of these nine mandates and write a 150-word paragraph on what the opportunity of increased scale of simulation represents to some application (e.g., oil reservoir modeling, aerodynamics, etc.) and what is considered today to be a challenge for the application that awaits more computational power.