Your name and student number: _____

PHYD57. MIDTERM EXAM 18 OCT 2024 - PROBLEMS

Write the solutions in the exam booklet. This part is worth 9% of the course mark.

1 [30 pts.] Biased random walk of electron in a conductor

Write in C(++) or Fortran (or metacode very closely resembling one of those languages; strict adherence to C/F syntax is not required, and insignificant syntax errors will be ignored), a procedure (subroutine) that simulates one random walk in 1-D of an electron inside a conductor, subject to a small voltage gradient. Spatial coordinate x is in the interval [0,n], n=1000, and is discretized into unit-length intervals: each step adds to x either +1 or -1, with probabilities p = 0.502 and 1-p = 0.498, correspondingly. Integer n, floating-point probability p, and the starting integer value x0 are the 3 input parameters to the procedure.

The only random number generator at your disposal is an intrinsic function randU(), which returns a uniformly-distributed pseudorandom, double precision, floating point value between 0 and 1. Explain how you can produce a biased random walk of the electron, using this generator.

RAM memory is unlimited. The procedure is meant to be run on art-1, which can run 12 concurrent, independent program threads. You should parallelize the loop(s) via OpenMP multithreading directives (please include them in the code). Time much longer than one call to randU() is needed to create every thread.

Parallelization should make use of the fact that non-overlapping sections of a long random walk are mutually independent (Markov chain steps do not remember their prior history) and can be computed concurrently. The random walk ends when the electron reaches either x=0 or x=n. The number of steps done, let's call it N, and the endpoint value of x are then returned to the calling program.

What value of N do you expect, on average, if x0 = n/2? (Order of magnitude answer suffices.)

The outline of the algorithm and its implementation should be briefly described separate from the code itself. The code should include comments explaining the names of variables, and what each important line or section of your code does.

2 [10 pts.] Amdahl's law(s)

State Amdahl's law, its implicit assumptions, and derive it (denote by f the parallelizable fraction of a sequential code). For example, if f=0.97, what maximum speedup factor can one achieve employing 16, 64, or 256 processors? Discuss the law's consequences.

SOLUTION

Let $f \leq 1$ be the fraction of the single-processor code's workload that can be parallelized, and assume that the parallelization is ideal, i.e. speeds up the execution of that part of code by a factor n, the number of processors (i.e. assuming that communication between threads, overhead of starting and closing them and any other possible factors are not present). In some units, execution time of a single-processor version is $f+(1-f)=1$, and of the time of n-processor execution is $f/n+(1-f)$. The speedup factor is thus

$$S = \frac{1}{1-f+f/n} \quad \text{(Amdahl's law)}$$

which tends to a limit $S \to 1/(1-f)$ with $n \to \infty$.

For example, if $n = \{16, 64, 128, \infty\}$, and $f = 0.97$, then the speedup factor is 11, 22.1, 26.6, and 33.3. Quadrupling the computational resourses from 16 to 64 processors only doubles the speed, and it gets even worse with further increase of n.

Comment: What is the best n to choose, given that $n = 1$ gives too slow an execution, and $n \to \infty$ uses too many resources (processors, watts) without much improvement in speed? Maybe $n_{opt} = 1/(1-f)$ (rounded to integer), which gives $S_{opt} \approx 1/(1-f^2)$, (in this example, $n_{opt} = 33$ and $S_{opt} = 16.8 \approx 16.9$), are the optimum numbers. But the price:performance ratio (let "price" be equal to n) is $n(1-f+f/n) = n(1-f)+f$. It is just getting worse, compared with $n = 1$ case, as n increases (unless $f = 1$), so you can't optimize price:performance by multiprocessing! Except when $1-f$ is negligibly small, the best performance per watt or $ is obtained on a single processor. The proposed optimum gives a noticeable speedup S_{opt}, at a 2 times worse price:performace ratio than a single-processor calculation.

PHYD57 MIDTERM EXAM 2024. QUIZ WITH SOLUTIONS

Quiz is worth 9% of the course. Statements may be tricky, so read carefully. A single word or number may be incorrect. **To gain 1 point of credit, write Y or N (yes or no) in the brackets. Any "[N]" answer MUST HAVE 1 or more wrong words or numbers correctly identified (please circle them).** Please disregard typos. Symbol % stands for a command-line prompt. To make up for unintended ambiguity or errors in wording of questions, 2 points will be added to your result.

[Y] TCP/IP packets sent over Internet contain detailed information, such as level 2 or the so-called MAC addresses, allowing unique identification of your sending equipment.

[N] Numpy is faster than Python because it is **multi-threaded** to use all CPU cores

[N] gcc is GNU compiler able to translate C++ into **Fortran and Python** code

[Y] OpenMP language assumes shared memory access, i.e. that all the created threads have access to the same operational memory (RAM).

[Y] In 1942, Atanasoff and Berry have built at Iowa State U. a specialized computer solving linear equations, which as the first used vacuum tubes.

[Y] The pre-WWII versions of German coding machine ENIGMA were deciphered and reverse-engineered by Polish cryptologists Rejewski, Zygalski and Rozycki.

[Y] 8086 was the name of a popular Intel 16-bit processor. Subsequent generations of Intel processors with similar instruction set have a name retaining part of that name. For instance, x86_64 means all modern 64-bit processors compatible with Intel's instruction set.

[N] **Assembly** program analyses the text of your C program for syntax errors and the correct use of elementary CPU instructions

[N] An unsigned 4-Byte integer can represent all numbers from 0 to 2^{32} **inclusive**. Since $1K = 2^{10} = 1024$, we have $2^{32} = 4G$.

[N] Nodes of a cluster are connected with a 10 Gbit/s ethernet network (theoretical transfer rate). The transfer of 1.6 gigabyte of data lasts about **0.16 seconds**.

[Y] Special option (-openmp in Intel compiler) enables OpenMP, whose C/Fortran directives are otherwise ignored

[N] Array indexes in Fortan are 1-based, and in C/C++ are 0-based. For instance, array declared in C(C++) as int A[100] has elements A[0] to **A[100]**

[Y] There can be 100G (1e11) transistors in a single integrated circuit in a laptop.

[N] Double semicolon in the C loop: for (i,j,k=0;;)(...) **is a syntax error**

[Y] IBM computers in 1960s were mostly leased at high cost to wealthy institutions, and had prorprietary design. In the 1970s, Digital Equipment Corp (DEC) disrupted that model by selling much cheaper minis: PDP-7 to PDP-11, with open architecture.

[Y] Antikythera Mechanism was the first known digital/analog personal computer of times of lunar and solar eclipses, positions of planets, olympiads etc.

[N] **C** is a relatively **recent** language created at **MIT**

[N] Operating system **Unix** was created by Apple Inc. for Macintosh personal computers

[Y] In Linux, a wave (tilde) means home directory of user; .. means parent of the current directory. Current directory, in turn, is printed by the pwd command.

[N] Command **%hostname** in Linux logs you to a remote computer for secure **file transfer**

[Y] Single precision floating point number is 4B long, and has 6 to 7 accurate decimal places

[N] Command %rm /home/user/output*2.dat removes files in /home/user, with a **single character** in place of *

[N] As a rule, in multiply nested loops, multithreading should apply to in the **innermost** loop, since there is an overhead cost of initiating threads.

[N] Variables listed as **public** in the OpenMP directive multithreading a loop will be made visible to each OMP thread by cloning their contents

[N] Zuse's Z3 computer is the 1st example of binary arithmetic in modern computers. Z3 used thousands of **vacuum tubes** and had a magnetic drum memory.

[N] After we throw **36 million** random points on unit square, and count points inside its inscribed circle to evaluate number π by MteCarlo method, the accuracy will **exceed 5 decimal digits**.

[Y] Reduction operation in OpenMP produces single sum from all threads, when "reduction(+:varname)" is present in the directive of loop parallelization, with varname replaced by actual name of the summation variable.