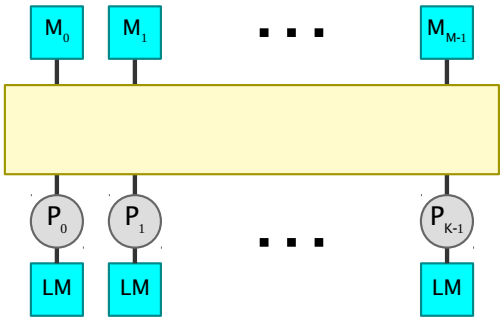
Parallel and Distributed Computing: Homework 1

Due April 10, 2020, in pdf format on Canvas .



System Definition: Consider a system, as the one depicted in the figure above, with K processors and a shared memory subsystem consisting of M single-ported memory modules. Processors and memory modules are interconnected by an interconnection network capable of pairing any processor to any memory module such that, at any given time, each processor connects to one, or none, free memory module. A free memory module is one that is not currently paired with any processor. During normal operation, processors run code and request access to data in the shared memory.

The purpose of this assignment is to use computer simulation to characterize the access time from processors to memory modules under different configurations, namely number of processors and memory modules, and under different workloads exhibiting different memory access patterns, namely the manner in which a concurrent program accesses memory.

Memory cycles: For the purposes of simulation, global time is divided into equal successive intervals called "memory cycles".

System processing: At the beginning of each memory cycle, processors request access to memory modules. Following a priority scheme, each requesting processor is connected to its requested memory module if and only if the module is free. A memory module is free if and only if no other processor has previously been connected to that module during the memory pairing mechanism at the beginning of the current memory cycle.

Processor access to data is assumed to occur during the remainder of the memory cycle. All memory modules become free again at the end of every memory cycle.

Processors that are not granted a connection to their requested memory modules wait for the next memory cycle to again request the same memory module.

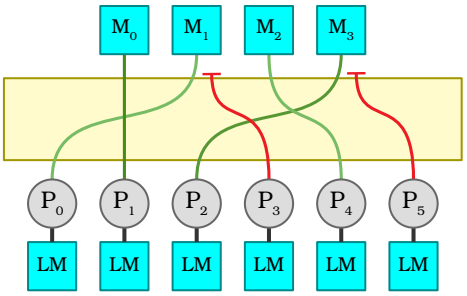
At the beginning of the next memory cycle, all processors that were granted access generate a new request while those who waited retry access to their same previously denied memory module.

Access-Time: If a processor x is connected to a memory module y, no other processor z can connect to y during the same memory cycle. **Processor z has to wait** and attempt access in the next memory cycle. The number of memory cycles that a processor waits for a memory module will be called "access-time". When a processor accesses a memory module on the first attempt (without waiting), the access-time is 0.

Memory Access Schemes: To pair processors and memory modules at the beginning of a memory cycle, any non-starving as-

signment scheme is acceptable. As an example, consider a priority scheme such processors are granted access to memory modules in their natural index order (processors with lower index are granted access first.) In the figure below, this is equivalent to scanning the processing elements from left to right and grant, in that order, memory access to those processors for which the requested memory module is free. In the example of the figure below, P_3 and P_5 will have to wait to another cycle to gain access to their requested modules. It is clear that this scheme suffers from starvation, some processors may never gain access to requested memory modules.

To fix the problem and avoid starvation, processors can be re-labeled at the beginning of a memory cycle such that the first "waiting" processor is re-labeled with index 0 and the remaining processors are cyclicly relabeled in the natural order. The relabeling can be done dynamically while scanning, or a circular ordered data structure can be kept with a dynamic pointer to the first processor to be considered (which is the first processor with a denied access in the current scan of the processor list.)



Assignment

You are asked to create a simulator in standard C, without external libraries, to characterize the access-time of processors to memory modules under different configurations and workload characteristics.

Workload assumptions To represent the memory access demands of a workload, different memory access patterns can be used. Access patterns of an application represent the extent to which locality of reference is present in the workload. For the purposes of this assignment, two distributions of memory requests will be assumed:

- **Uniform distribution** Each processor issues a random memory module request at the beginning of each memory cycle using a uniform distribution.
- **Normal distribution** In a system with M memory modules, before the main simulation execution, every processor π selects one uniformly distributed random memory module μ_π. All subsequent memory requests of processor π will be given by $mod(round(X), M)$, where X is a random variable normally distributed with mean μ_π and standard deviation $M/6$.

Input Your program will accept 2 command line arguments. You can assume that the given command line arguments will always be correct.

- A positive integer p specifying the number of processors to simulate.
- A lowercase character d specifying the distribution of the memory requests with the only possible values `'u'` (uniform) or `'n'` (normal).

Computation Let $S(p, m, d)$ be a system with p processors, m memory modules, and a workload distribution d. And let $S_c(p, m, d)$ be the system $S(p, m, d)$ at memory cycle c of its simulation.

- For each number of memory modules $m \in [1, 2048]$, a simulation will be run

and will be limited to a maximum of $C = 10^6$ cycles of $S(p, m, d)$.

- For each simulation run, you are to compute the time-cumulative average of the access-time for each processor, and the arithmetic average $\bar{W}\left(S_c(p, m, d)\right)$ of all processors' time-cumulative averages. The time-cumulative average of a processor's memory access at cycle c is defined as the total number of simulated memory cycles c divided by the total number of granted accesses so far.

- As it is expected that the average system memory access time $\bar{W}$ settles at some stable value, two termination conditions for the simulation of $S(p, m, d)$ will be considered.

  - The system average access time between the current $\bar{W}_c$ and previous average $\bar{W_{c-1}}$ is less than a certain tolerance ϵ

    $$Abs\left(1 - \frac{\bar{W}\left(S_{c-1}(p, m_i, d)\right)}{\bar{W}\left(S_c(p, m_i, d)\right)}\right) < \epsilon$$

  - or the maximum number of cycles is reached $c = C$

Output The program will produce 2048 lines ended in `\n` to standard output, one line per simulation. Each line will have the last $\bar{W}$ of that simulation with 4 decimal places.

Verification of feasibility You are free to design your simulation program in any which way you like, however, it suggested to use the following considerations/specifications:

- Using 3 equal size arrays to represent each processors' request, access counter, and priority of connection. Think of this as being a data structure containing K entries, one per each processor, and each entry having, for each processor, a request, access counter, and a priority of connection.

- Another array to represent the memory modules. Each element in this array has a 1 if the represented memory module is already connected to a processor or 0 if it is free.

- To avoid processing element starvation, it is suggested that you use the cyclic priority index assignment described above to effect processor-memory pairings.

Report

You are to digitally produce an individual report of two pages with two charts, and a one page discussion of the results. *Do not forget to include your name and student ID number on the top of the first page of your report.*

Charts The first page of your report will contain two charts, one each for the assumed uniform and normal distributions. Each chart consists of an X-Y cartesian drawing in which the results of the simulations are plotted. A plot is a curve showing the values on the Y-axis of a variable as a function of values in the X-axis.

- In each chart the X-axis represents the number of memory modules varying from 1 to 2048 while the Y-axis represents the values of $\bar{W}$ obtained in the simulations.

- Both axis have to be linear (not logarithmic or any other non-linear scale).

- Generate one plot (curve) for each fixed number of processors $\{2, 4, 8, 16, 32, 64\}$. There must be 6

curves per chart. Although the data for the plots is generated by your simulator as a set of $\bar{W}$ that can be captured on a file by redirecting the command line output, you can use any available tool to generate the plots, and hence the charts, such as LibreOffice, Google Sheets, Microsoft Office, etc.

In summary, parameterizing each chart on the number of processors, six plots (curves) will be superimposed into each chart.

Discussion In the second page, you will include the analysis and interpretation of the obtained results. How does the memory request distribution affects the behavior of the system? If this simulation is in the context of making a decision to buy expensive memory modules for a given number of powerful processors, what would you recommend? Why?

Submission

Submit a zip file named `x.zip`, where `x` is your UCInetID, containing 2 files:

- The well commented C source code of the simulator called `simulator.c`.
- The digitally produced, individual report of two pages called `report.pdf`.