CSC 4322: Parallel Computing

Homework # 3 (Due on 10/25)

Point: 100

**Java Program** Suppose the run-time of a serial program is given by T(serial) = n^2, where the units of the run-time are in microseconds. Suppose that a parallelization of this program has run-time T(parallel) = n^2/p + log2(p). Write a program that finds the speedups and efficiencies of this program for various values of n and p. Run your program with n = 10, 20, 30, 40, … ,320, and p = 1, 2, 4, … ,128. What happens to the speedups and efficiencies as p is increased and n is held fixed? What happens when p is fixed and n is increased?

* 1. When P is increased and N is fixed box speed ups and efficiencies are increasing.
  2. When N is increased and P is fixed speed up increases slightly and efficiencies decrease.

1. **Understand lecture 6(2,3,4)** A parallel program that obtains a speedup greater than *p*—the number of processes or threads—is sometimes said to have **superlinear speedup**. However, many authors don’t count programs that overcome “resource limitations” as having superlinear speedup. For example, a program that must use secondary storage for its data when it’s run on a single processor system might be able to fit all its data into main memory when run on a large distributed-memory system. Give another example of how a program might overcome a resource limitation and obtain speedups greater than *p*.
   1. A program can obtain speedups greater than p if, one by one pass through the data set is performed, computation is not required for all elements, processing a specific set of data elements completes operation, and the result that completes the analysis of the data is clumped.
2. A sequential application with a 20% part that must be executed sequentially, is required to be accelerated five-fold. How many CPUs are required for this task?
   1. To be increased 5-fold a CPU increase extending to infinity are required for this task.
3. In slide 12 of Lecture 6, we see that as the number of processors (or threads or cores) increases, the speedup increases. a. Why the curve of “double size” seems better than the other two? b. If we keep increasing the number of processes (i.e. the x-axis) what do you think the rest of the curve will look like?
   1. Double size is better because more cores with more data takes advantage of their being more cores. Less data could use more time communicating when unnecessary.
   2. The curve would continue to flatten out for double. The curve of of original would flatten out before double size.
4. **Understand the entire lecture 7**What happens in the greetings program if, instead of strlen(greeting) + 1, we use strlen(greeting) for the length of the message being sent by processes 1, 2, : : : , comm\_sz-1? What happens if we use MAX\_STRING instead of strlen(greeting) + 1? Can you explain these results?
   1. If you use strlen(greeting) you could the size of the message being send would be a CHAR short of the actual message. There is a value only read by the C compiler that must be accounted for when calculating the elements.
   2. If MAX\_STRING is used the number of elements allotted in the msg\_buf would be 100.