CSCI-UA.0480-003

Parallel Computing

Homework Assignment 1

1. In the global sum problem that we discussed in class (slides 30 to 37 in lecture 1), if we assume that there is a variable called *my_rank* (local to each core) that gives each core a unique rank from 0 to p-1 (for p cores), devise an expression to calculate *my_first_i* and *my_last_i* assuming:

a. n is divisible by p and n > p.
b. n is not divisible by p.

2. We have seen two ways of calculating the final sum in the global sum example we have studied in class. In one of them, the master core receives the partial sums from the other cores and calculates the final sum. The other method is the tree-method. Assume that the master core is core 0.

a. Derive a formula for the number of receives and additions that core 0 does in the first (non-tree) method.
b. Repeat for the tree-method.
c. Make a table showing the number of receives and additions done by core 0 for each method when the number of cores is 2, 4, 8, ..., 1024.
d. Which operation do you think is more expensive: receive or addition? and why?

3. If you are given a sequential program that you are required to parallelize, first you need to find the parts that are parallelizable. However, in some cases, it is not worth it to parallelize those parts, why?

4. There are several types of parallelism. What are they? For each one, given a definition, and specify whether exploiting that type needs programmer involvement or the hardware/compiler are enough to exploit it.

5. What is the reason we have distributed memory architectures?

6. Suppose we have the algorithm (assume N is a large even number):

```
for(i = 0; i < N/2; i++)
        a[i] += a[i+ N/2 ];
```

a. [3] Can we parallelize the above algorithm? If no, why not? If yes, explain.
b. [2] What Is the maximum number of cores after which no performance enhancement can be seen? Justify