PHYS 5210: Computational Methods in Astrophysics, Assignment 2

Set Oct 19th due Nov 2nd

Notes: For the questions in which you are expected to write a code, please email me a copy.

Q1. Which of the following loops can be parallelized via an OpenMP parallel do, and which cannot? For each loop give a clear statement of why it does, or does not, parallelize. For bonus marks, if a loop cannot be parallelized suggest a way to make it parallel. **(3 marks)**
(a)

```
do i=1,n-1
a(i+1)=a(i)
end do
```

(b)

```
do i=1,n-1,2
a(i+1)=a(i)
end do
```

(c)

```
c=0
do i=1,n
c=c+1
a(i)=b(i)+1.
d(c)=a(i+1)
end do
```

Q2. Parallelize the following loop using `PARALLEL DO` loops (auxiliary storage is allowed) **(4 marks)**:

```
do i=1,n-1
a(i)=a(i+1)
end do
```

Q3. The recursive loop

```
do i=2,n
a(i)=a(i)+a(i-1)
end do
```

is typically considered a barrier to parallelization. However, this operation can be parallelized to a certain degree by using partial sums (i.e. you sum isolated pieces of the array first and then bring them together in a specific order). Write an algorithm for the above operation that while taking O(n log n) operations can still be run in parallel using the `PARALLEL DO` in OpenMP. Auxiliary storage will be necessary, and for simplicity you may assume that n and the number of threads are both powers of 2. **(6 marks)**

Q4. The sieve of Eratosthenes is a well known method for determining the number of prime numbers below a given number. The algorithm works by eliminating all the multiples of all primes less than or equal to the square root of the given number. The remaining numbers are then primes. Pseudo-code for the algorithm:

```
Eratosthenes(n){
a[1] := 0
for i := 2 to n do a[i] := 1
p := 2
while (p^2 ≤ n) do {
j := p^2
while (j ≤ n) do {
a[j] := 0
j := j+p}
repeat p := p+1 until a[p] = 1
}
return(a)}
```

Implement this algorithm in a code that uses a task queue within an OpenMP parallel region to achieve parallel execution. Compare speed up for n=1000 on 1,2,4 processors. Do the same for n=1000000. Explain the scaling results you achieve. **(8 marks)**