Paradigmas de Computação Paralela

(UCE Computação paralela de Distribuída)
18-February-2014

Program parallelization with OpenMP

These exercises aim to introduce the basic concepts of program parallelization for shared memory platforms with OpenMP

1) Compile the following program (with GNU gcc use the –fopenmp flag). Execute the program multiple times and explain the results.

```
...
int main() {
    #pragma omp parallel
    for(int i=0;i<100;i++)  {
            int id = omp_get_thread_num();
            printf("%d:i%d ", id, i);
    }
}
```

2) Introduce the following directives between the *#pragma omp parallel* and the "for". Explain the results.

    2.1. *#pragma omp master*
    2.2. *#pragma omp single*
    2.3. *#pragma omp critical*
    2.4. *#pragma omp for*

3) Introduce the following changes to the program in 1):

    3.1. Include a barrier inside the loop, after the printf statement (*#pragma omp barrier*).
    3.2. Include the directive *#pragma omp ordered* inside the loop in the program developed in 2.4, also adding *ordered* to *#pragma omp for*

4) Exploit the impact of the following of the loop scheduling options in program 2.4:

    4.1) *schedule(static)* and *schedule(static,10)*
    4.2) *schedule(dynamic)* and *schedule(dynamic,10)*
    4.3) *schedule(guided)*

5) Compile and execute the following program several times and explain the results

```
...
int main() {
    int w=10;
    #pragma omp parallel
     #pragma omp for
    for(int i=0;i<100;i++)  {
        int id = omp_get_thread_num();
        printf("%d:ai%d w=%d\n", id, i,w++);

     }
     printf("W=%d\n",w);
}
```

6) Include the following declarations in the *for*

- 6.1) *private(w)*
- 6.2) *firstprivate(w)*
- 6.3) *firstprivate(w) lastprivate(w)*
- 6.4) *reduction(+:w)*

7) Develop a parallel version of the following programs

```
...
for (i = 0; i < length; i++)
   sum = sum + data[i];

...
```

```
double f( double a ) {
              return (4.0 / (1.0 + a*a));
}

double pi = 3.141592653589793238462643;

int main() {
              double mypi = 0;
              int n = 1000000000; // number of points to compute
              float h = 1.0 / n;

              for(int i=0; i<n; i++) {
                            mypi = mypi + f(i*h);
              }
              mypi = mypi * h;
              printf("Aproximacao de pi = %.10f \n", (pi - mypi));
}
```