

Computação Paralela

October-2018

Program parallelization with OpenMP (part2)

These exercises aim to introduce concepts of program parallelization for shared memory platforms with OpenMP. Compile the program with `gcc -O3 -std=c99 -fopenmp -lm <<file>>.` Set the number of threads before running the program with `export OMP_NUM_THREADS=<<NT>>`

- 1) Consider the following OpenMP program to compute the dot product of two vectors

```
#define size 100000
double a[size], b[size];

void main() {
    // init
    for(int i=0; i<size; i++) {
        a[i] = 1.0/((double) (size-i));
        b[i] = a[i] * a[i];
    }

    // computedot
    double dot = 0;

    for(int i=0; i<size; i++) {
        dot += a[i]*b[i];
    }

    printf("Dot is %18.16f\n", dot);
}
```

- a) Try to parallelize the code by using the *omp parallel for* pragma. Are you able to get same results with different number of threads and in different runs? Explain why the program does not work correctly in parallel. What is needed for correct computation?
- b) Use the *reduction* clause to compute the dot correctly. Compile and run your code also without OpenMP. Do you get exactly same results in all cases?
- 2) Develop a parallel version of the following program and measure its scalability.

```
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));
}
```