Computação Paralela

November-2020

Task Parallelism

These exercises aim to introduce tasks parallelism. Compile the programs with `gcc -O3 -std=c99 -fopenmp -lm <<file>>`. Set the number of threads, before running the program, with `export OMP_NUM_THREADS=<<NT>>`

Develop an OpenMP code to implement the parallel execution of the Fibonacci function, using the `task` and `taskwait` pragmas.

```c
#include <stdio.h>
#include <omp.h>

double fib(double n) {
    double i, j;
    if (n<2) return(n);
    i = fib(n-1);
    j = fib(n-2);
    return(i+j);
}
int main() {
    double r;
    double time = omp_get_wtime();

    r = fib(40);
    printf("F=%.0f Time=%f\n", r, omp_get_wtime()-time);
    return(1);
}
```

1) Execute the program with 1 and 2 threads and explain the results.

2) **Tuning**: change the Fibonacci to compute the fib(45) and change the parallelisation to execute sequentially when n<45 (parallelism cut-off). Execute the program with 1, 2 and 4 threads and explain the results. What is the best parallelism cut-off on this machine?

3) **Scalability:** Study the scalability of the Fibonacci parallelisation by running the program with different numbers of threads and compute the gain by dividing the sequential execution time by the parallel execution time. How does the algorithm scale with the number of threads? What would be the ideal gain?