Serie 1

This serie can be made only with a sheet of paper and a pen. The objective is to exercise the theoretical analysis of a code and understand if it is worth or not to parallelize the sequential code and what could be its theoretical parallel performance.

The answer of the first five exercices are to be found in the five proposed videos.

Exercise 1 (Big O notation)**.**

(a) Define in one sentence the time and space complexities for an algorithm.

(b) Assume you have two algorithms A_1 (with time complexity $\mathcal{O}_t(n\ log\ n)$ and space complexity $\mathcal{O}_s(n)$) and A_2 (with time complexity $\mathcal{O}_t(n^3)$ and space complexity $\mathcal{O}_s(n^3)$). Which one is worth to be parallelized ?

Exercise 2 (Vocabulary)**.**

(a) Why a DAG (Direct Acyclic Graph) is useful for a theoretical analysis of a parallelisation strategy for a particular algorithm ?

(b) What is the critical path in a DAG of tasks with their dependencies ?

(c) What are the two main kind of parallelism ?

(d) What is one of the main solutions to solve the problem of "irregular parallelism" ?

Exercise 3 (Machine model)**.**

(a) What ILP, TLP and DLP stands for ?

(b) What is a vector intrinsic ?

(c) Classify the memories according to their speed of access : DRAM, L2,L1,L3,SSD ?

(d) Give the 4 types of parallel architectures according to Flynn's taxonomy and an example of each.

Exercise 4 (Threads, processes and the OS)**.**

(a) What are the main differences between a thread and a process ?

Exercise 5 (Performance aspects)**.**

(a) What is the definition of the speedup ?

(b) What is the definition of the parallel efficiency ?

(c) What is the definition of the Amdahl's law (with the serial part f and the serial execution time T_1) and what does this law measures ?

(d) What is the definition of the Gustafsson's law (with the non-parallelizable part α and the portion of parallelized part P) and what does this law measures ?

We work on a Poisson equation solver in 2D. Here follows the Poisson problem :

The Poisson's equation is:

$$\nabla^2 \varphi = f \tag{1}$$

that is in 2D

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \varphi(x, y) = f(x, y) \tag{2}$$

with $\partial\Omega = 0$

The equation is discretized on a 2D rectangular domain of size NxN. We use centered finite differences to approximate (2). This will lead to a linear system $Ax = b$. Using a Jacobi iterative method to solve it and expressing φ^{k+1} as a function of φ^k , we obtain

$$\varphi_{i,j}^{k+1} = \frac{1}{4} \left(\varphi_{i+1,j}^k + \varphi_{i-1,j}^k + \varphi_{i,j+1}^k + \varphi_{i,j-1}^k - f_{i,j} h^2 \right) \tag{3}$$

where $h = \frac{1}{N}$

For our preliminary results, we set $f_{i,j} = -2\pi^2 sin(\pi ih) * sin(\pi jh)$

Here follows a bulk of a sequential code to solve the 2D poisson equation using Jacobi's method. We set `eps = .005` the minimal error that the l2-norm must reach.

```
int main() {
  int     i, j, k;
  float **u, **uo, **f;
  float   h, l2;

  h = 1. / (double)N;

  // Allocation of the arrays of pointers
  malloc ...

  // initialization of u0 and f
  for(i = 0; i < N; ...

  k=0;
  start = second();
  do {
    l2 = 0.;
    for(i = 1; i < N-1; i++) {
      for(j = 1; j < N-1 ;j++) {
        // computation of the new step
        u[i][j] = 0.25 * ( uo[i-1][j] + uo[i+1][j] + uo[i][j-1] + uo[i][j+1] - f[
    i][j]*h*h);

        // L2 norm
        l2 += (uo[i][j] - u[i][j])*(uo[i][j] - u[i][j]);
      }
    }

    // copy new grid in old grid
    for(i = 0; i < N; i++){
      for(j = 0; j < N; j++){
        uo[i][j] = u[i][j];
```

```
32        }
33      }
34
35      k++;
36    } while(l2 > eps);
37    end = second();
38
39    double t = end - start;
40    printf("T=%.5f s for %d steps (%.5f s/step)\n", (end - start), k, (end - start)
        /k);
41
42
43    // deallocation of the rows
44    free ...
45
46    return 0;
47  }
```

Exercise 6 (Computational complexities)**.**

(a) compute the computational complexity ("big O" notation) of the algorithm (in **time**)

(b) compute the computational complexity ("big O" notation) of the algorithm (in **space**)

(c) is this problem worth to be parallelized ?

Exercise 7 (Theoretical analysis : Amdhal's law)**.**

We assume a very simple parallelization strategy : each processor p gets $\frac{N}{size}$ number of lines of the grid. Where $size$ is the total number of processor. We consider sd_p the subdomain of size $N \times \frac{N}{size}$ belonging to processor p.

At each time step the algorithm is :

1. compute a step of the Jacobi solution on sd_p
2. send the last line of sd_p to processor $p+1$ and the first line of sd_p to processor $p-1$
3. receive the last line of sd_{p-1} from processor $p-1$ and the first line of sd_{p+1} from processor $p+1$
4. compute the local l2 norm
5. send the local l2 norm to processor 0.
6. (a) if $p == 0$ compute the global l2 norm by suming all the local l2 norms
   (b) if $p == 0$ send the global l2 norm to all processors
7. receive the global l2 norm from process 0.
8. Compare with espilon. Exit if reached.

Please answer the following questions (use Table 1) :

(a) give an estimation of f (the sequential part of the code that can not be parallelized).

(b) What is the upper bound of the speed up in the case of Amdahl's law ?

Exercise 8 (Theoretical analysis : Gustafsson's law)**.**
We assume the same parallelization strategy as above.

(a) What would be the maximum efficiency of this parallel 2D poisson solver at 128 processors ?

N	t_{total} in [s]	t_{init} in [s]	n_{steps}	t_{step} in [s/step]
128	0.003	0.000	109	0.00003
256	0.496	0.004	8194	0.00006
384	1.083	0.011	7445	0.00015
512	2.242	0.021	8238	0.00027
640	4.215	0.043	8234	0.00051
768	5.500	0.051	6088	0.00090
896	7.736	0.078	5655	0.00137
1024	15.246	0.153	7395	0.00206
1152	23.317	0.234	7450	0.00313
1280	34.051	0.341	7724	0.00441
1408	40.636	0.407	8371	0.00485
1536	75.720	0.758	13049	0.00580
1664	87.591	0.874	13222	0.00662
1792	85.202	0.851	13255	0.00643
1920	107.046	1.071	13652	0.00784
2048	126.070	1.261	13964	0.00903
2176	132.905	1.329	13521	0.00983
2304	236.203	2.361	18680	0.01264
2432	246.091	2.461	18814	0.01308
2560	258.232	2.582	17515	0.01474

Table 1: Measurements of a 2D poisson solver. N is the size of the problem (grid size = $N \times N$), Execution time is in seconds. Third column is the number of iterations steps to reach an espilon of 0.005. Last column shows the time per iteration