Parallel Computing
Degree in Computer Science Engineering (ETSINF)
Year 2022-23 ◇ Partial exam 6/11/23 ◇ Block OpenMP ◇ Duration: 1h 30m

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Question 1 (1.1 points)
Given the following function:

```
void f(double A[N][N], double B[N][N], double v[N]){
  double s;
  int i, j, k;
  for(k=0; k<N; k++){
    s=0.0;
    for(i=0; i<N; i++)
      for(j=0; j<N; j++)
        s+=A[i][j]*B[i][j];
    v[k]=v[k]/s;
    for(i=0; i<N; i++)
      for(j=0; j<i; j++)
        A[i][j]+=B[i][j]/s;
  }
}
```

0.2 p. (a) If possible, parallelize the outermost loop of the previous code, justifying the answer.

Solution: It is not possible to parallelize the outer loop, since there are dependencies between the iterations of the k loop, since at each iteration of the loop the matrix A is modified.

0.6 p. (b) Parallelize the two inner loops as efficiently as possible (loops with i), using a single parallel region.

Solution:

```
void f(double A[N][N], double B[N][N], double v[N]){
  double s;
  int i, j, k;
  for(k=0; k<N; k++){
    s=0.0;
    #pragma omp parallel
      {
      #pragma omp for private(j) reduction(+:s)
      for(i=0; i<N; i++)
       for(j=0; j<N; j++)
          s+=A[i][j]*B[i][j];
      #pragma omp single nowait
      v[k]=v[k]/s;
      #pragma omp for private(j)
      for(i=0; i<N; i++)
        for(j=0; j<i; j++)
          A[i][j]+=B[i][j]/s;
      }
  }
}
```

0.3 p. (c) Calculate the sequential time, parallel time, speedup and efficiency corresponding to a single iteration of loop k, assuming that only the first inner loop has been parallelized (first loop with i).

Solution:

$$t(N) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 2 + 1 + \sum_{i=0}^{N-1}\sum_{j=0}^{i-1} 2 \approx \sum_{i=0}^{N-1} 2N + 2\sum_{i=0}^{N-1} i \approx 2N^2 + 2\frac{N^2}{2} = 3N^2\text{flops.}$$

$$t(N,p) = \sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 + \sum_{i=0}^{N-1}\sum_{j=0}^{i-1} 2 \approx \sum_{i=0}^{\frac{N}{p}-1} 2N + N^2 = \frac{2N^2}{p} + N^2 = \left(\frac{2}{p}+1\right)N^2\text{flops.}$$

$$S(N,p) = \frac{t(N)}{t(N,p)} = \frac{3N^2}{\left(\frac{2}{p}+1\right)N^2} = \frac{3}{\frac{2}{p}+1}.$$

$$E(N,p) = \frac{S(N,p)}{p} = \frac{3}{2+p}.$$

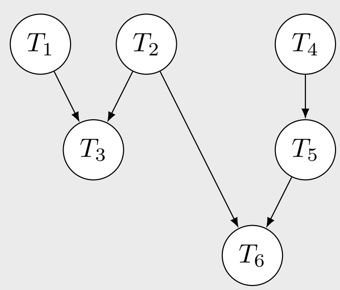
Question 2 (1.2 points)

Given the following function, where each of the invoked functions modifies only the vector that it receives as first argument:

```
void func(double a[], double b[], double c[], double d[], int n) {
  double x;
  task1(b,a,n);      /* Cost: 4n flops */
  x=task2(c,a,n);    /* Cost: 3n flops */
  task3(b,c,n);      /* Cost: 2n flops */
  task4(d,a,n);      /* Cost:  n flops */
  task5(d,a,n);      /* Cost:  n flops */
  task6(d,x,n);      /* Cost: 3n flops */
}
```

0.4 p. (a) Draw the task dependency graph. Mark the critical path indicating its length and obtain the average degree of concurrency.

Solution:



Critical path: T1→T3 or also T2→T6.

Length of critical path: $L = 4n + 2n = 6n$ flops

Average degree of concurrency:

$$M = \frac{4n + 3n + 2n + n + n + 3n}{6n} = \frac{14n}{6n} = 2.33$$

0.6 p. (b) Write a parallel version using OpenMP. The execution time must be minimized.

Solution:

```
void func(double a[], double b[], double c[], double d[], int n) {
  double x;
  #pragma omp parallel
  {
    #pragma omp sections
    {
      #pragma omp section
         task1(b,a,n);
      #pragma omp section
         x=task2(c,a,n);
      #pragma omp section
      {
         task4(d,a,n);
         task5(d,a,n);
      }
    }
    #pragma omp sections
    {
      #pragma omp section
         task3(b,c,n);
      #pragma omp section
         task6(d,x,n);
    }
  }
}
```

0.2 p. (c) Obtain the speedup of the parallel version when run with 3 processors.

Solution: Sequential execution time:

$$t(n) = 4n + 3n + 2n + n + n + 3n = 14n \text{ flops}$$

The parallel execution time is $4n$ flops for the first `sections`, plus $3n$ for the second `sections`:

$$t(n, p) = 4n + 3n = 7n \text{ flops}$$

Speedup:

$$S(n, p) = \frac{14n}{7n} = 2$$

Question 3 (1.2 points)

Function `pluviometry` computes different statistical values related to water precipitation occurred during a set of `ND` days in a territory for a total of `NM` months. In turn, function `month_day` provides the identifier of the month to which a certain day belongs. Parallelize the function by means of OpenMP employing just one parallel region. The ordering of days in vector `risk_days` is not relevant for the parallel version.

```
void pluviometry(float precipitation[ND],int risk_litres,float average_rain_month[NM]) {
   int i, month, nrainy_days=0, max_day, nrainy_days_month[NM];
   int nrisk_days=0, risk_days[ND];
   float sum_rain=0, precipitation_max=0, average_rain;
   float sum_rain_month[NM];
```

```
/* Initialize the vectors entries to 0 */
...

for (i=0;i<ND;i++) {
  if (precipitation[i]>0) {
     sum_rain+=precipitation[i];
     nrainy_days++;
     if (precipitation[i]>risk_litres) {
        risk_days[nrisk_days]=i+1;
        nrisk_days++;
     }
     if (precipitation[i]>precipitation_max) {
        precipitation_max=precipitation[i];
        max_day=i;
     }
     month=month_day(i);
     nrainy_days_month[month]++;
     sum_rain_month[month]+=precipitation[i];
  }
}
average_rain=sum_rain/nrainy_days;
for (i=0;i<NM;i++)
   average_rain_month[i]=sum_rain_month[i]/nrainy_days_month[i];


 /* More code */
...
}
```

Solution:

```
void pluviometry(float precipitation[ND],int risk_litres,float average_rain_month[NM]) {
   int i, month, nrainy_days=0, max_day, nrainy_days_month[NM];
   int nrisk_days=0, risk_days[ND];
   float sum_rain=0, precipitation_max=0, average_rain;
   float sum_rain_month[NM] ;

   /* Initialize the vectors entries to 0 */
   ...

   #pragma omp parallel
   {
      #pragma omp for private(month) reduction(+:sum_rain,nrainy_days)
      for (i=0;i<ND;i++) {
         if (precipitation[i]>0) {
            sum_rain+=precipitation[i];
            nrainy_days++;
            if (precipitation[i]>risk_litres) {
               #pragma omp critical (risk)
               {
```

```
                    risk_days[nrisk_days]=i+1;
                    nrisk_days++;
                 }
              }
              if (precipitation[i]>precipitation_max) {
                 #pragma omp critical (maximum)
                 {
                    if (precipitation[i]>precipitation_max) {
                       precipitation_max=precipitation[i];
                       max_day=i;
                    }
                 }
              }
              month=month_day(i);
              #pragma omp atomic
              nrainy_days_month[month]++;
              #pragma omp atomic
              sum_rain_month[month]+=precipitation[i];
           }
        }
        average_rain=sum_rain/nrainy_days;
        #pragma omp for
        for (i=0;i<NM;i++)
           average_rain_month[i]=sum_rain_month[i]/nrainy_days_month[i];

     /* More code */
     ...
  }
```