Cristian Dávila
Enrique Gonzalez

1. Which is the overhead associated with the activation of a parallel region in OpenMP? Is it constant? Reason the answer based on the results reported by the ompparallel code and the the trace visualized with Paraver.

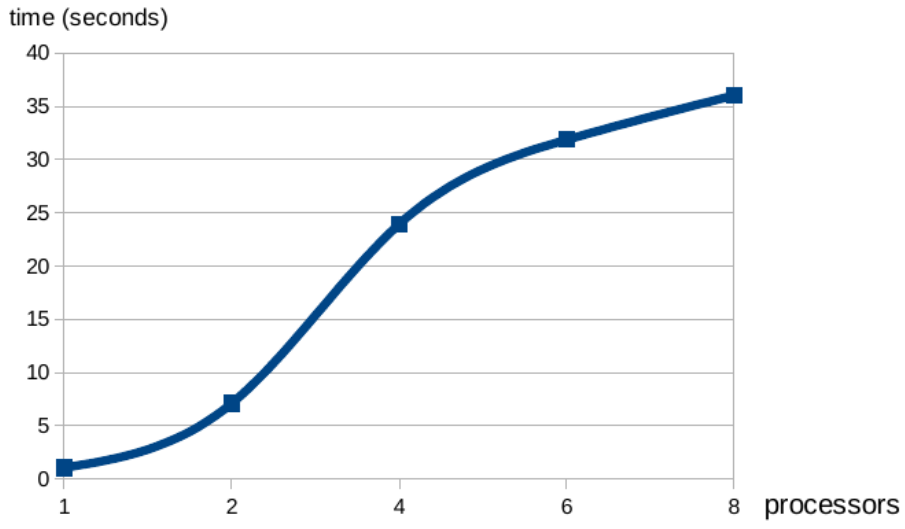
El temps de la creació de threads no es constant, el temps de creació disminueix a mesura que anem executant més threads. Arribat a un punt podríem dir que es torna “constant” i l’overhead per la creació de cada un és de 0.2 microsegons. (aprox.)

2. Which is the minimum overhead associated with the execution of critical regions in OpenMP?

És el temps que triga quan el nombre de threads és 1, així veiem el temps mínim de les regions critiques ja que la part variable és per l’espera dels locks de la variable als altres threads quan fem l’execució per més d’un thread.
Agafem l’execució de dos threads i restem al total de l’execució el temps de creació d’aquests dos threads. Aquest overhead és de 1.5 microsegons. (aprox.)

3. In the presence of lock conflicts and true sharing (as it happens in dotprod mutex everytime), how the overhead associated with critical increases with the number of processors? How this overhead is decomposed? Reason the answer based on the results visualized with Paraver.

En el codi tenim la regió crítica dins del for, això provoca que tinguem locks conflicts, per tant si augmentem el nombre de processadors, tenim més processadors esperant a que el recurs quedi lliure. La gràfica representa el temps que triga la execució en funció dels processadors. Podem comprobar com el temps d’execució s’incrementa amb els processadors afegits, per tant a més processadors, tenim més temps d’overhead, a causa de la sincronització entre ells.



4. In the presence of false sharing (as it happens in dotprod vectorsum), which is the additional average access latency that you observe to memory? Which causes this increase in the memory access time? Reason the answer based on the results visualized with Paraver.

És a causa de que tenim false sharing degut al data race quan tots els threads executen aquests bucles for:

```
for (i=start; i<end ; i++)
        vectorsum[myid] += (x[i] * y[i]);
}
for (i=0; i<nthreads; i++)
        dotstr.sum += vectorsum[i];
}
```

El temps adicional degut a la latència de memòria es el temps total que tarda cada thread entre el nombre d'accessos crítics (on els altres threads li poden causar false sharing): 0.031041625.

5. Complete the following table with the execution times of the different versions of dotprod that
we provide to you. The speed{up has to be computed with respect to the execution of the serial version. For each version and number of threads, how many executions have you performed?

version	1 thread	8 threads	speedUp
dotprod serial	0.314371	-	1
dotprod mutex everytime	1.07622	36.045	0.03
dotprod mutex	0.360863	0.06648	5.43
dotprod vectorsum	0.382826	0.248333	1.54
dotprod vectorsum padding	0.385043	0.069879	5.51