David Martos

Mariona Farré

Grupo: paco1208

Curso: 2022-23

PACO:

Laboratorio 4

**Índice:**

[**1. Before starting lab assignment**](#_cbwtvdpwjy44) **2**

[**2. Task decomposition analysis for Mergesort**](#_h68cod6qltnq) **2**

[2.1. Divide and conquer](#_vmcq9fpimus4) 2

[2.2. Task decomposition analysis with Tareador](#_tb9oc4tip8u0) 2

[**3. Shared-memory parallelisation with OpenMP tasks**](#_nqq27nn3b3sz) **8**

[3.1. Leaf strategy in OpenMP](#_ozsoedw803h1) 8

[3.2. Tree strategy in OpenMP](#_7ycb75fkpy9u) 10

[3.3. Optimization: Cut-off mechanism](#_8gft53syo3bm) 12

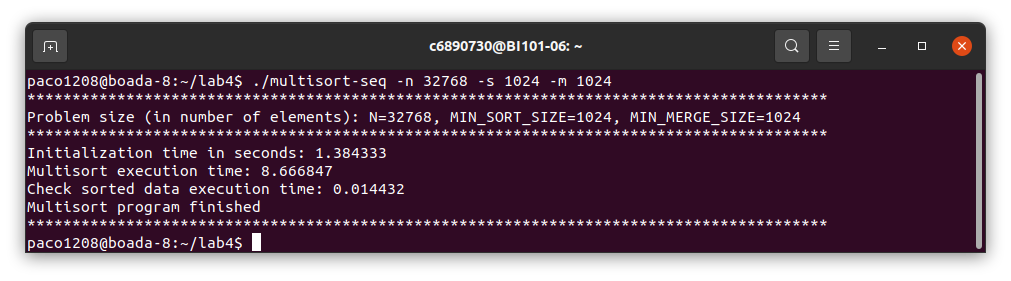
[**4. Shared-memory parallelisation with OpenMP task using dependencies**](#_o3bw6xzeysuo) **14**

# **Before starting lab assignment**

# **Task decomposition analysis for Mergesort**

## **Divide and conquer**

**For the deliverable**: Take note of the times reported for the sequential execution and use them as reference times to check the scalability of the parallel versions in OpenMP you will develop.



## **Task decomposition analysis with Tareador**

**multisort-tareador.c** is already prepared to insert Tareador instrumentation. Complete the

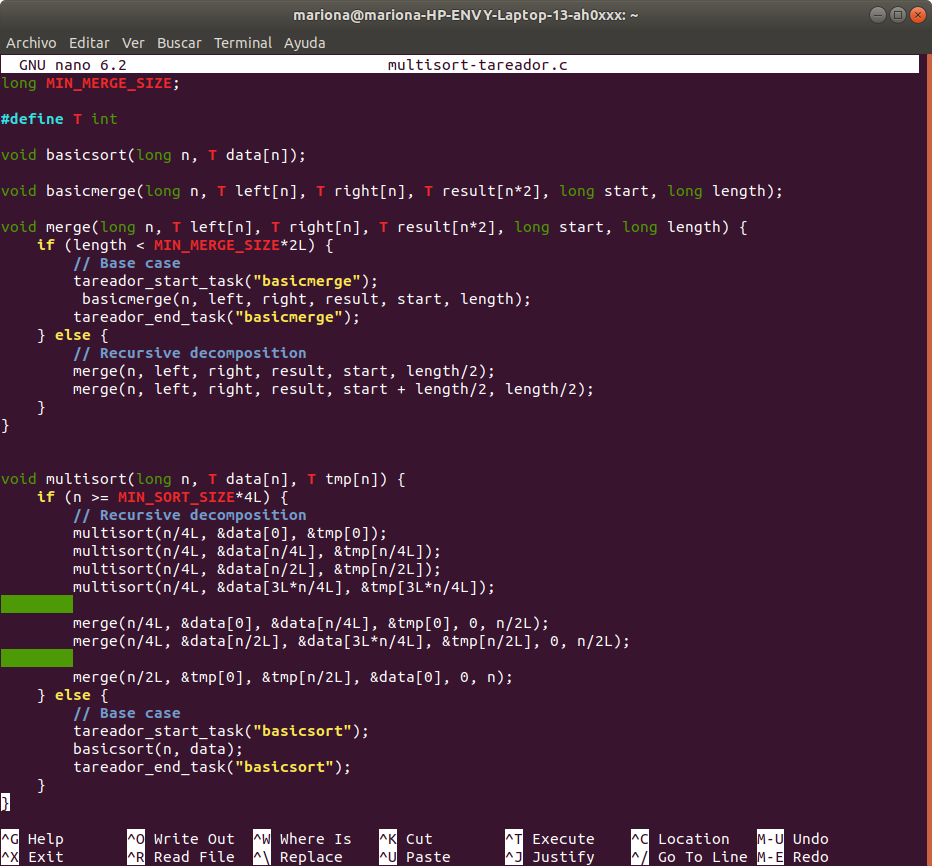
instrumentation to understand the potential parallelism that each one of the two recursive task decomposition strategies (leaf and tree) provide when applied to the sort and merge phases:

* In the leaf strategy you should define a task for the invocations of basicsort and basicmerge once the recursive divide–and–conquer decomposition stops.
* In the tree strategy you should define tasks during the recursive decomposition, i.e. when invoking multisort and merge.

**For the deliverable:** Include both tareador codes (in the .zip file) and show code excepts in the report.

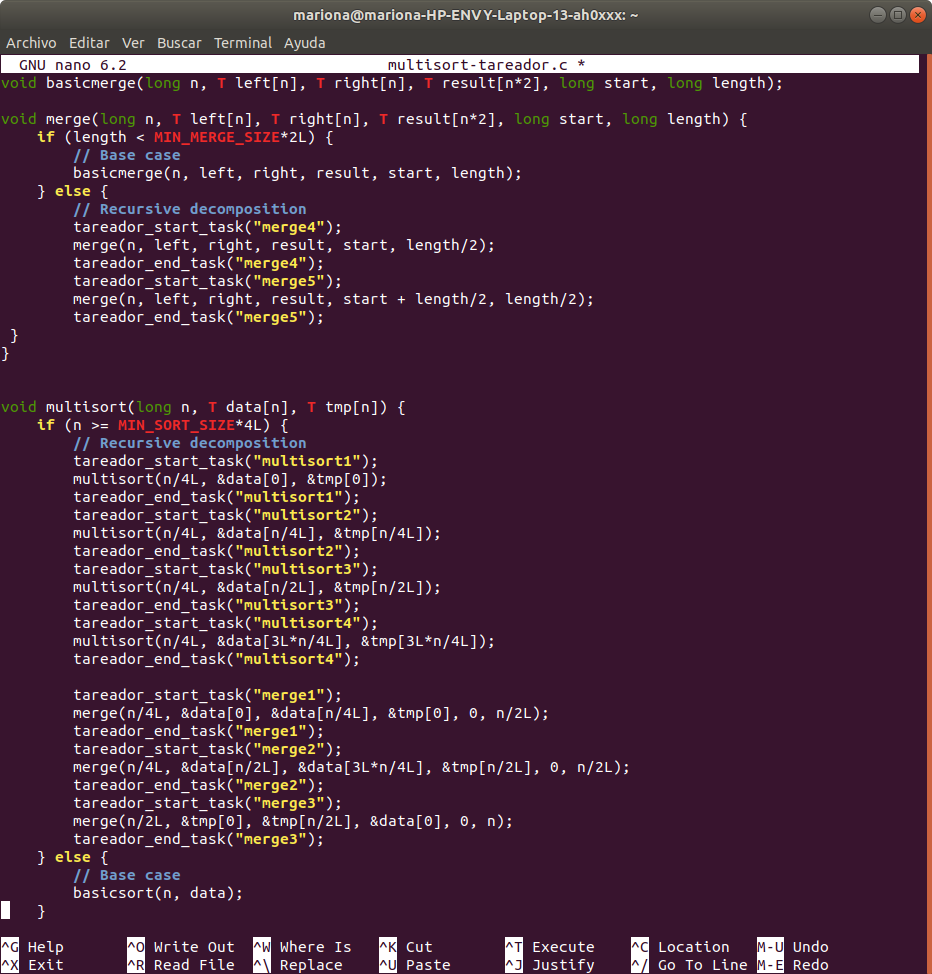
Leaf strategy:

Definir las tareas del tareador en el basicmerge y el basicsort.



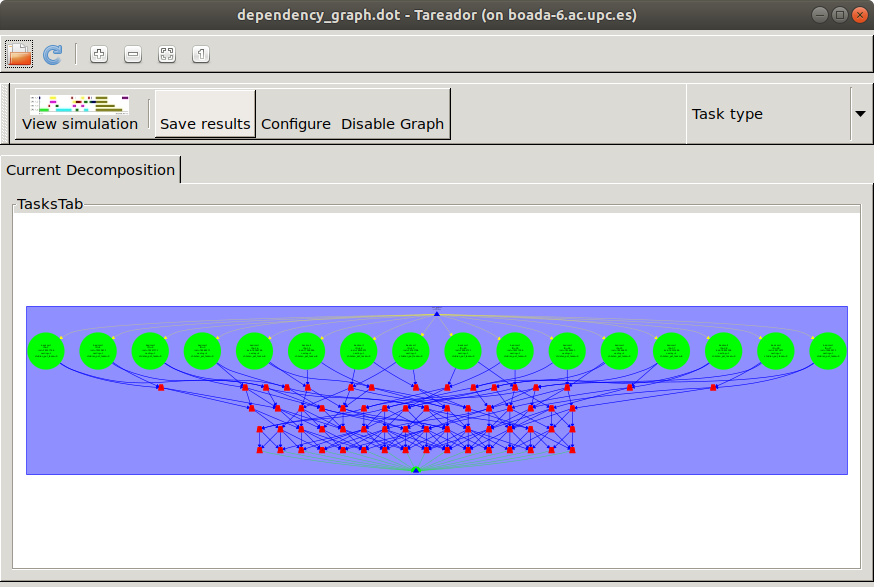
Tree strategy:

Definir las tareas del tareador en el merge y el multisort.

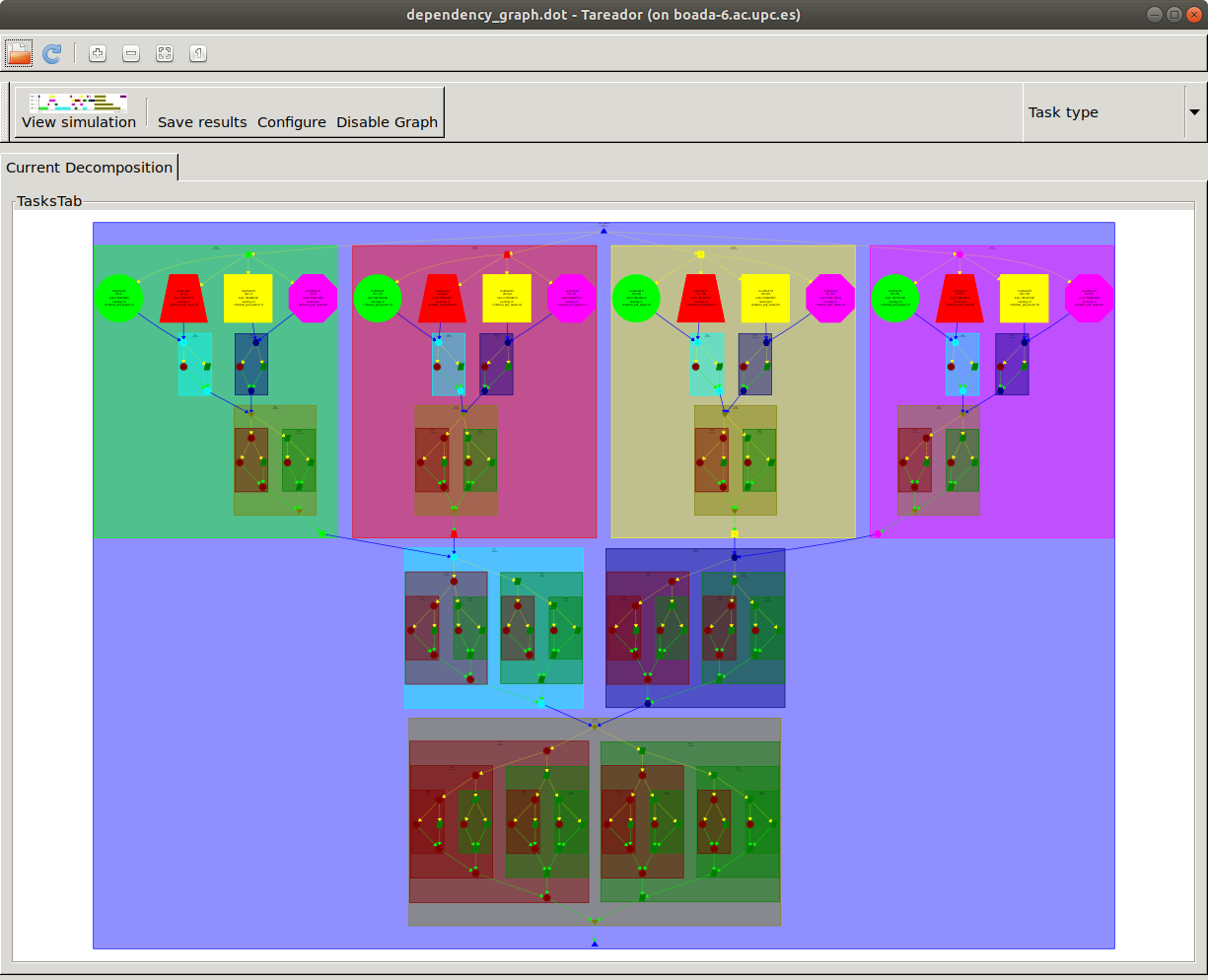
****

**For the deliverable:** From the task dependence graphs that are generated for leaf and tree, do you observe any major differences in terms of structure, types, number and granularity of tasks, ...? Save the image of the two graphs in order to include them in the report.

Leaf strategy:



Tree strategy:



Como podemos observar los dos gráficos son muy diferentes para cada estrategia.

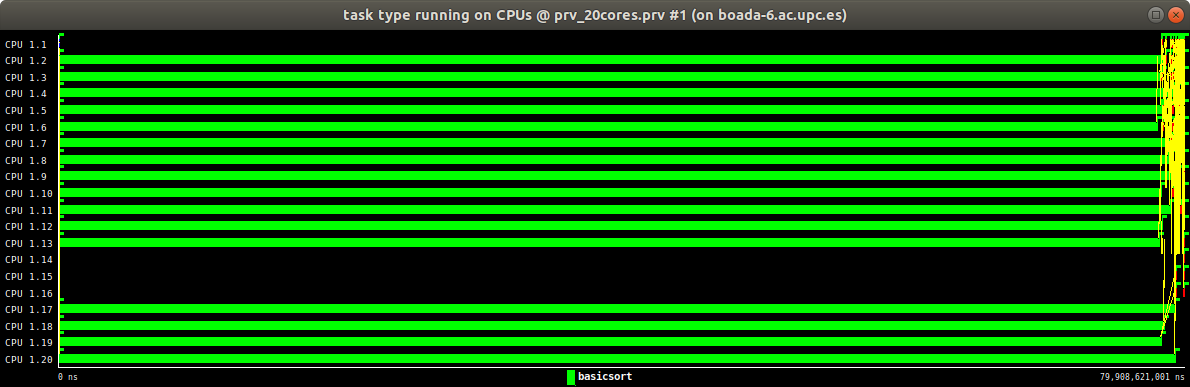
La estrategia del árbol tiene muchas mas tareas generadas porque hay muchas mas invocaciones para el basicsort y el mergesort, por lo contrario la estrategia de hoja genera solo tareas en el caso base de las funciones de merge y el multisort, no como el caso recursivo de la estrategia del árbol.

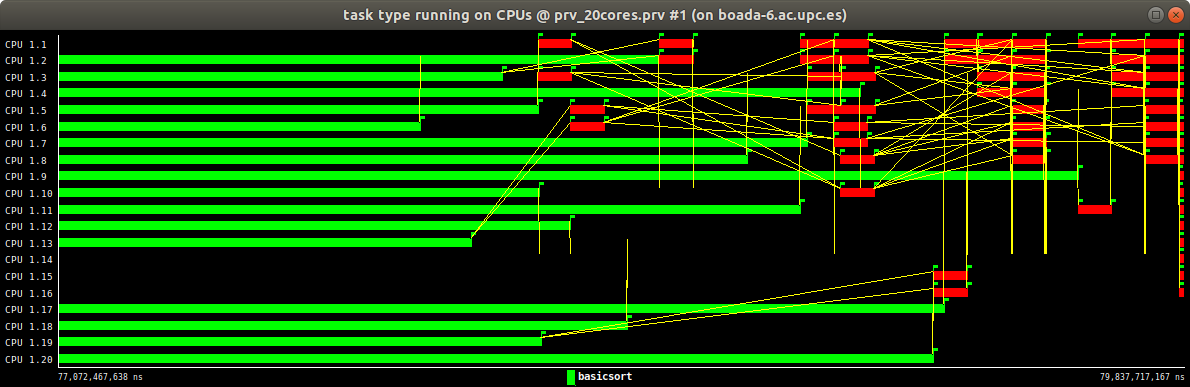
**For the deliverable:** Identify the points of the code where you should include synchronizations to guarantee the dependences

En las dos estrategias, la función merge crea dependencias porque es la función que ordena las dos partes del vector. original, mientras que en la función multisort como no depende de estas partes del vector no crea estas dependencias. Y podemos ver que la función multisort necesita mas tiempo porque hay muchas más tareas creadas.

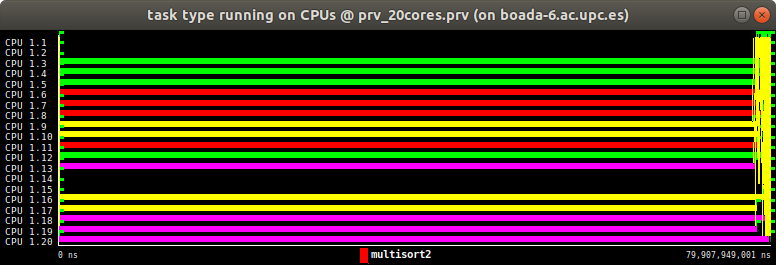
**For the deliverable:** Capture the necessary Paraver windows in order to support your explanations in the deliverable.

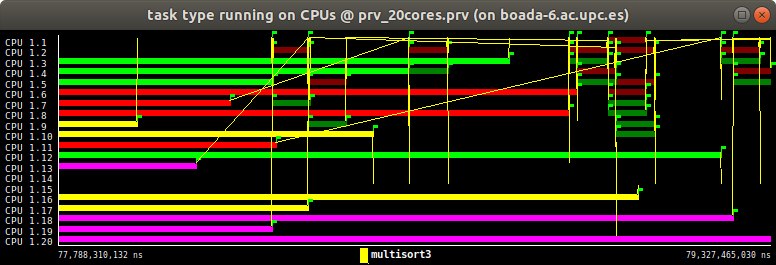
Leaf strategy:





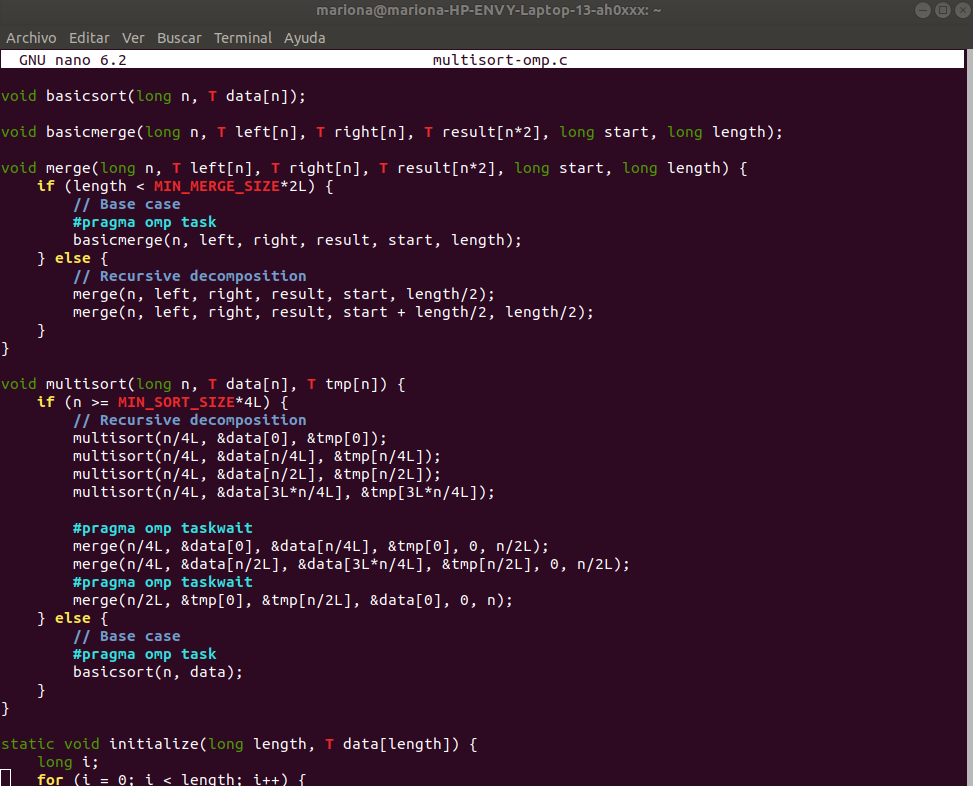
Tree strategy:

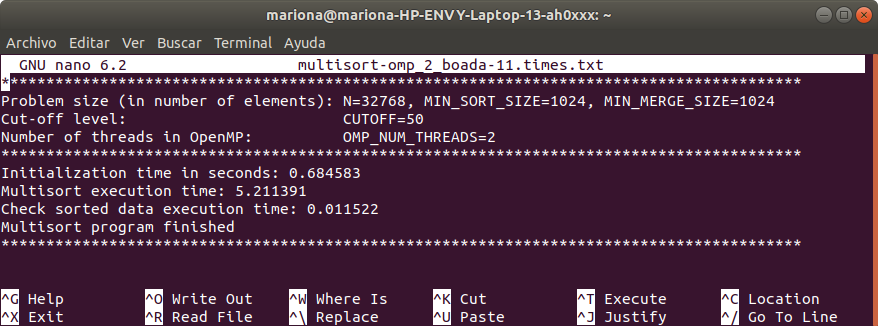




# **Shared-memory parallelisation with OpenMP tasks**

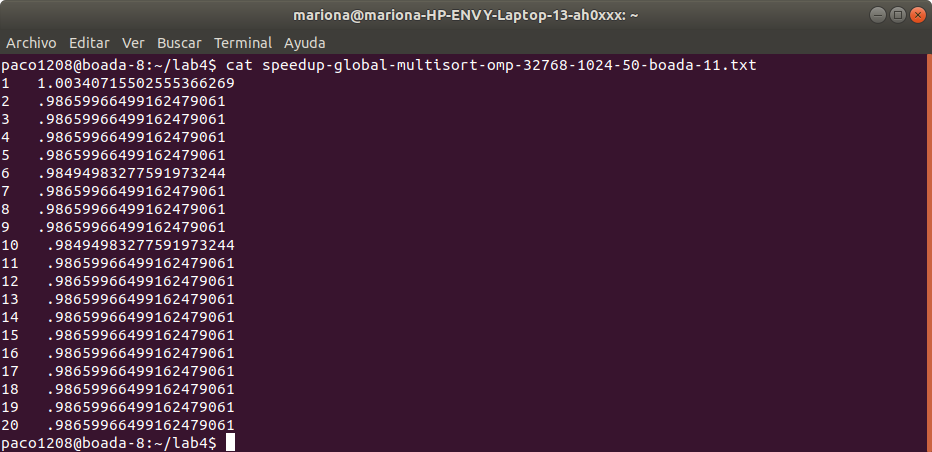
## Leaf strategy in OpenMP



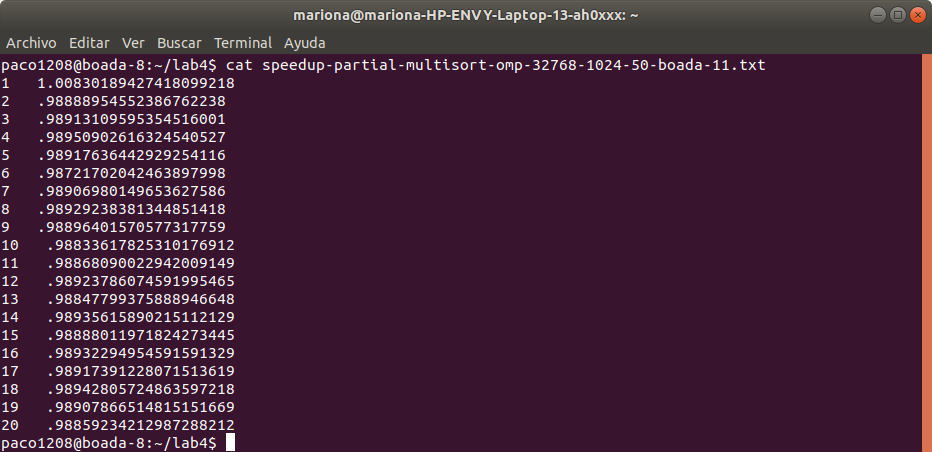


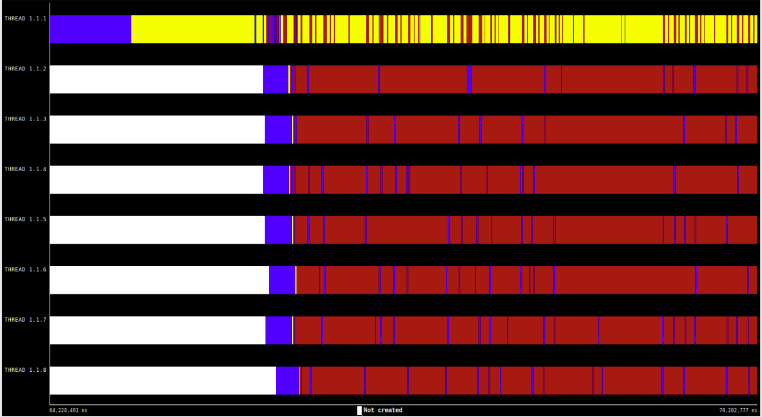
Text files:

speedup-global:

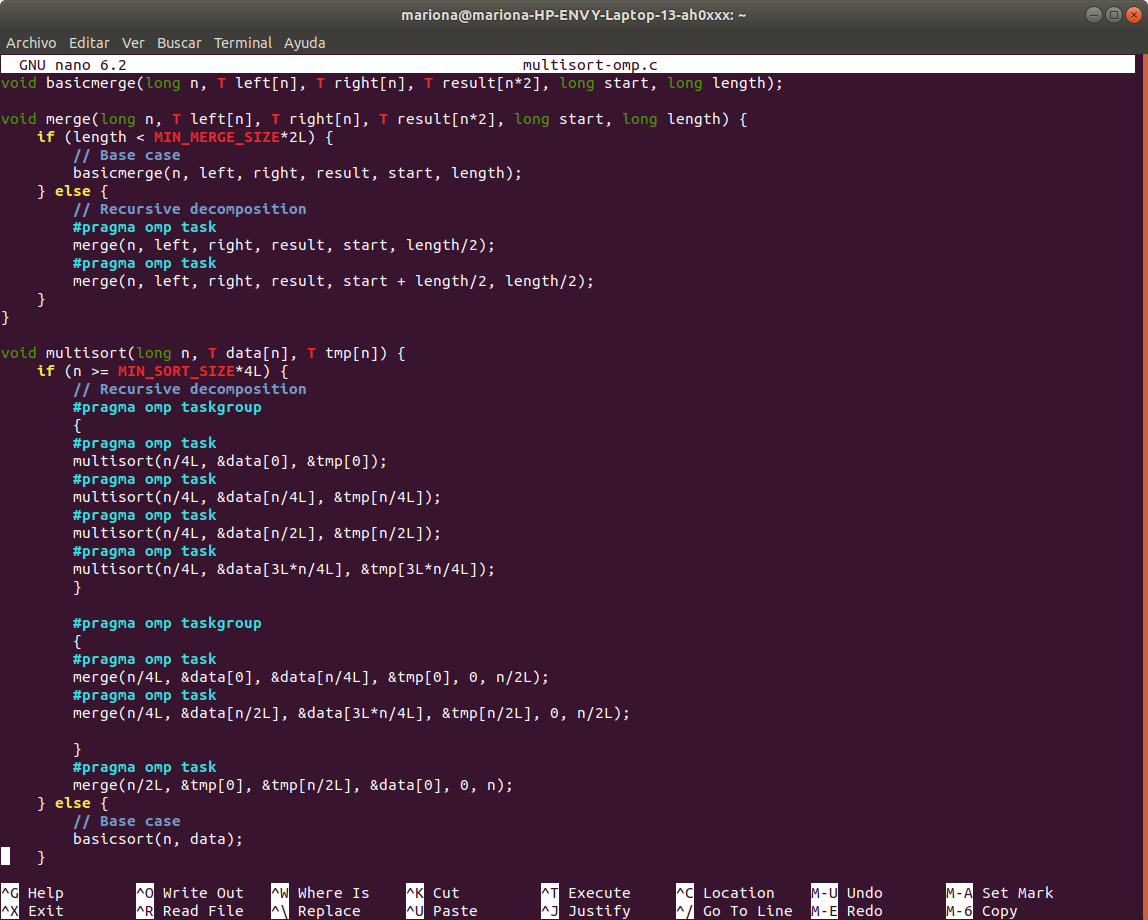


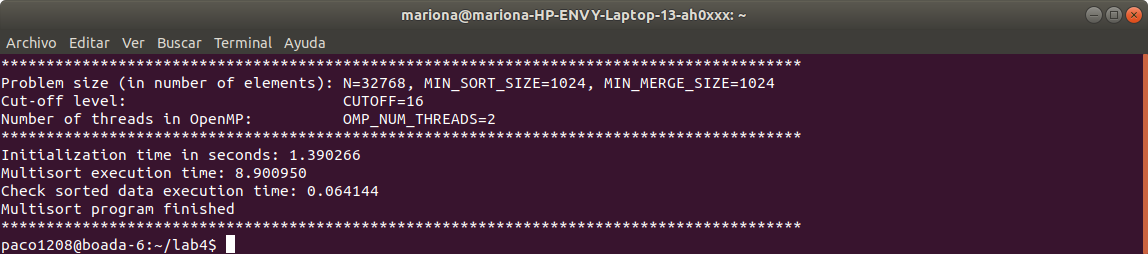
speedup partial:



****

## Tree strategy in OpenMP





# 

Mientras la versión leaf es implementada creando tascas en los casos base, y la versión

del árbol solo las crea antes de las llamadas del multisort y merge functions.

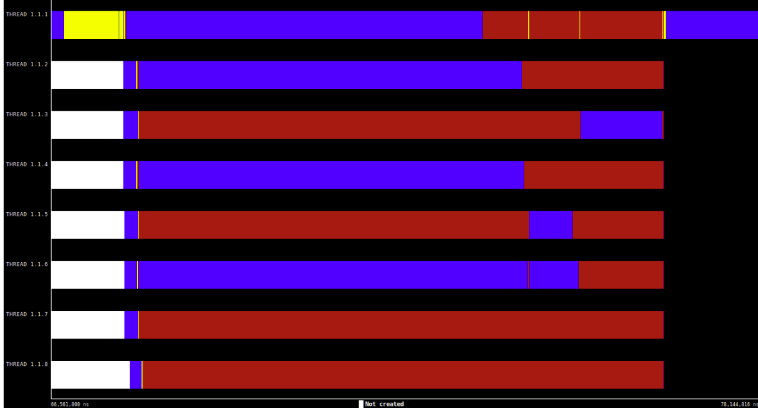
Mientras que en el tiempo de ejecución no hay mucha diferencia, pero la versión árbol es más eficiente.

Podemos observar que las dos versiones tienen problemas de escalabilidad, però la versión árbol mejora considerablemente bien el speedup con la función multisort que puede seguir más las escalabilidad.

## Optimization: Cut-off mechanism

## 

sbatch -p exectuion submit-extrae.sh multisort-omp 8 0



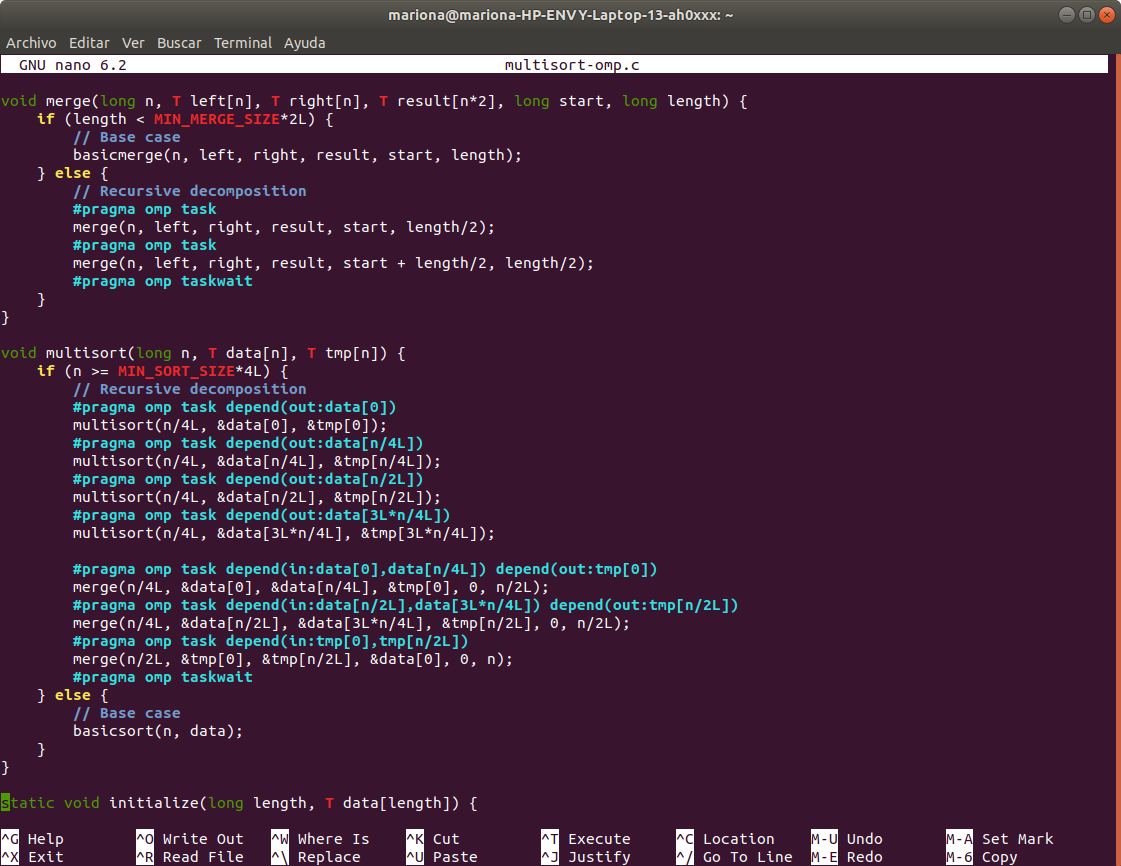
sbatch -p exectuion submit-extrae.sh multisort-omp 8 1

## 

## 

# **Shared-memory parallelisation with OpenMP task using dependencies**

**For the deliverable**: Include the tables of modelfactors, scalability plots and Paraver windows to support the comparison.



OMP\_NUM\_THREADS=8 CUTOFF=16 ./multisort-omp -n 32768 -s 1024 -m 1024

