Màquina multiprocessador amb 4 processadors unicore 4x Opteron 844 a 1.8Ghz, que es comuniquen entre ells mitjançant links directes i accedeixen a una memòria de 4GB mitjançant.

Màquina multithread amb 2 processadors, 8 cores per processador i 8 threads per core, en total pot executar 128 threads simultàniament. La màquina es tracta d’un Sun Enterprise T5240 a 1.4 Ghz.

El codi donat, un cop paral·lelitzat, l’executarem amb x nombre de threads i sobre un vector que pot tenir x elements. Aquest codi serà una versió més senzilla del quickSort.

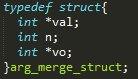
Mesurarem el temps d’execució i el SpeedUp (Temps d’execució seqüencial / Temps d’execució paral·lelitzat).

Algorisme:

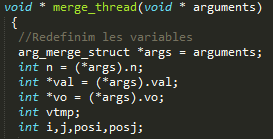
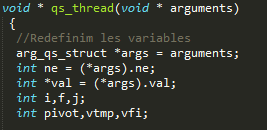
* Es dividirà el vector en parts i és farà una ordenació local d’aquests segments del vector.
* Després és cridarà al merge, aquest el que farà és anar comparant 2 a 2 els elements més petits de cada segment i els anirà ordenant, de forma que s’haurà d’executar log2(Parts en les que s’ha dividit el vector).

Per separar la fase de sort i la de merge necessitem una barrera, de la mateixa forma que en les diverses etapes de merge 🡪 mutex

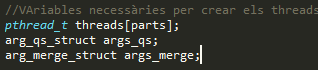
Hem agut d’implementar dues estructures addicionals per poder passar els arguments necessaris als fils d’execució de la funció *quickSort* i *merge*:



A més, hem hagut de implementar una nova funció per al quickSort i *merge* perquè es pugin executar des de diversos fils (modificació de la capçalera i declaració de les variables passades a través de les estructures implementades), tot i que mantindrem la funció *quickSort* ja donada per les crides recursives dins del mateix fil:



En el *main* hem de instanciar les dues estructures implementades i també una variable que ens apunti a l’identificador de cada fil:



A més, enlloc de realitzar diverses crides seqüencials a la funció *qs* i *merge2* crearem diversos fils que s’executaran paral·lelament, per evitar que no comenci el *merge* abans no acabi el *sort* i entre els diferents nivells de *merge*, farem ús de la funció pthread\_join(), que s’espera fins que acabi l’execució dels fils.

