Pràctica 1 - OpenMP

Febrer 2018

Resum

Aquesta primera pràctica se centra en fer servir OpenMP per implementar de forma concurrent el codi presentat la setmana anterior a la sessió d'introducció. Es proposen diversos experiments; alguns seran més eficients que altres. L'objectiu és veure com OpenMP reparteix la feina a realitzar entre els diversos fils que s'executen a la màquina.

1 Introducció

A la pràctica d'introducció s'ha presentat el codi a paral·lelitzar. L'objectiu en aquesta pràctica és realitzar una implementació concurrent d'aquest. Per això es proposaran diversos experiments a realitzar, alguns seran millors que altres. Es tracta de veure que OpenMP reparteix la feina entre els fils segons els atributs que nosaltres li indiquem a la construcció.

En cas que no ho hagueu fet encara, es demana realitzar abans les tasques indicades a la sessió d'introducció. En aquesta pràctica, i sense haver encara paral·lelitzat el codi, quina opció de compilació faríeu servir per obtenir una bona eficiència d'execució del codi? És convenient que compileu el codi fent servir una opció de compilació que optimitzi el codi.

2 Feina a realitzar

Es proposen a continuació un conjunt d'experiments a realitzar. La paral·lelització se centrarà en la funció `create_tree_files`, que és la funció que processa els fitxers de text. Haureu vist, a la pràctica introductòria, que aquesta funció és la que més triga en executar de totes. La idea doncs és paral·lelitzar aquesta funció fent que fils diferents processin fitxers diferents. Cal tenir en compte que alguns cops **caldrà realitzar alguna modificació del codi per tal d'adaptar-lo a la proposta realitzada**.

Tingueu en compte que l'arbre local només es fa servir temporalment per inserir-hi les paraules del fitxer que s'està processant en aquell moment. L'arbre local permet una implementació múltifil: farem que múltiples fils puguin processar múltiples fitxers a la vegada. Cada fil bolcarà la informació llegida del fitxer al seu arbre lobal. En acabar de processar un fitxer, el fil haurà de bolcar la informació de l'arbre local a l'arbre global, un element compartit entre tots els fils. Analitzeu

el codi de la funció `create_tree_files` i observeu on s'accedeix a l'arbre local i on a l'arbre global. Cal detectar doncs on, en el codi, s'han incloure les seccions crítiques necessàries. **Quina és la part de codi que cal protegir?**

Es proposa realitzar els següents experiments. Es demana provar cadascuna de les propostes (opcions 1, 2 i 3) fent servir diversos fils (per exemple, 2, 4, 8) per tal de comprovar si la paral·lelització és escalable.

1. Utilitzeu una construcció en en bucle amb planificació estàtica. Es interessant que utilitzeu el fitxer de configuració `llista_ordenada_mida.cfg` per a realitzar els experiments: en aquest fitxer els fitxers a processar s'han ordenat de forma creixent pel nombre de línies que tenen. Proveu també amb `llistat_tot.cfg`, on els fitxers estan ordenats de forma diferent.

2. Utilitzeu una construcció en en bucle amb planificació dinàmica. Proveu de fer els experiments tant amb `llista_ordenada_mida.cfg` com `llistat_tot.cfg`. S'observa alguna diferència (notable) entre els dos fitxers de configuració? Quina diferència observeu amb la planificació estàtica?

3. Utilitzeu una construcció en la tasca. Cada tasca se centrarà en llegir un fitxer, generar l'arbre local corresponent i copiar el contingut a l'arbre global. Perquè sigui més senzill programarho, es recomana modificar el codi: feu una funció que faci la tasca descrita a la frase anterior. El bucle principal de `create_tree_files` farà crides a aquesta funció.

4. Observar que la secció crítica es pot col·locar a diferents nivells: protegir tota la funció de copia de l'arbre local a global (la funció que es troba a la funció `create_tree_files` i que fa la còpia de l'arbre local al global), o bé fer una protecció més fina (protegir una alguna part interna de la funció de còpia del codi). Proveu aquesta proposta fent servir la construcció en bucle amb planificació dinàmica. Fixeu el nombre de fils al nombre de processadors del vostre ordinador i observeu si hi ha diferència en el temps d'execució.

3 Què s'ha d'entregar

El fitxer que entregueu s'ha d'anomenar `P1_Cognom1Cognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `Cognom1` és el cognom del primer component de la parella i `Cognom2` és el cognom del segon component de la parella de pràctiques. El fitxer pot estar comprimit amb qualsevol dels formats usuals (`tar.gz`, `zip`, `rar`, etc). Dintre d'aquest fitxer hi haurà d'haver dues carpetes: `src`, que contindrà el codi font, i `doc`, que contindrà la documentació addicional en PDF.

- El directori `doc` ha de contenir un document explicant
  - El nombre de processadors que té el sistema amb el qual s'han realitzat els experiments. Tots els experiments s'han de realitzar amb el mateix ordinador.
  - Els resultats obtinguts amb les diverses opcions de compilació (-g, -O, -O3) de la pràctica introductòria.

  - Quina és la part de codi que cal protegir a la implementació paral·lela? Comenteu els resultats del punt 4.
  - Les diverses implementacions concurrents realitzades (punt 1, 2 i 3). Per a cada implementació concurrent es demana incloure en el document el tros de codi paral·lelitzat, una descripció del per què s'ha paral·lelitzat d'aquesta forma i un raonament del l'acceleració obtinguda respecte les altres implementacions. Recordeu que es demana provar cadascuna de les propostes amb diferent nombre de fils. És important que tots els codis s'executin sempre a la mateixa màquina. El document haurà de tenir una llargada màxima de 10 pàgines (sense incloure la portada).

- El directori `src` ha de contenir la font completa de les diverses implementacions concurrents realitzades.

Tingueu en compte la data límit indicada a la planificació. El pes de la **documentació és un 60%**, mentre que el del codi un 40%.