**1.Heu obtingut alguna millora en l’execució del codi respecte a fer servir només 1ﬁl? Podeu comparar-ho amb el que heu fet a la pràctica 4? Hi ha alguna diferència en el temps d’execució del codi? Comenteu les vostres respostes.**

Per poder fer comparacions de temps amb més marge hem decidit fer les comparacions amb la llista més gran de fitxers a llegir, així seria més fàcil veure la diferència de temps d'execució dels programes.

|  |  |  |
| --- | --- | --- |
| Practica 5 | 1 fil | 2 fils |
| Llista.cfg | 58.60 seg | 35.57 seg |

Analitzant els resultats entre utilitzar 1 fil o utilitzar 2 podem veure que la diferencia de temps es casi el doble per la qual cosa es molt optim la utilització d’almenys dos fils per llegir els fitxers.

Per l’altre costat vam realitzar proves amb més fils i els resultats no van ser gaire diferents, en el cas d’utilitzar 4 fils el temps que ens va sortir va ser de 29 segons, sent una diferencia de 6 segons amb el de dos fills, per la qual cosa a partir dels dos fils ja no es troba una gran diferencia entre dos i més fills.

|  |  |  |
| --- | --- | --- |
| Temps | Practica 4 | Practica 5 |
| Llista.cfg | 1 min 09 seg | 33 seg |

**2. Els ﬁls secundaris, a l’hora d’actualitzar l’arbre compartit a partir de la seva pròpia local, pot utilitzar diverses estratègies:**

* **a) fer una secció crítica que bloquegi tot l’arbre mentre un ﬁl fa l’actualització a partir de la seva pròpia local,**
* **b)fer una secció crítica que bloquegi cada node de forma independent cada cop que calgui actualitzar el comptador.**
* **Aquesta darrera opció permet que múltiples ﬁls actualitzin al mateix temps l’arbre compartit a partir de la seva còpia local. Creieu que té alguna avantatge utilitzar l’opció b) respecte l’a)? Heu pogut fer proves per comparar les dues propostes? Sou lliures de mostrar altres solucions implementades.**

En el nostre cas, tal com vam fer a la practica anterior, realitzem la opció b, on bloquegem cada node de manera independent quan hem d’actualitzar el comptador de nodes, en aquest cas aquest bloqueig el trobem a la funció de index\_words\_line.

Igualment vam agafar les idees proposades en aquesta pregunta i vam decidir també implementar la estratègia a). En el nostre programa tenim la funció localToGlobal que el que realitza es el pas entre l’arbre local al global. Dins d’aquesta funció cridem al copyRecursive que s’encarregara de iterar per tot l’arbre fent aquestes copies de l’arbre local al global bloquejant abans de fer l'actualització i desbloquejant al acabar-hi. Posteriorment es crida discursivament per buscar els fills esquerres i drets.

|  |  |  |
| --- | --- | --- |
|  | Implementació a | Implementació b |
| Temps | 52,126671 s | 56,660690 s |
| Paraula que més apareix | 4633807 | 4638196 |

Desprès de realitzar les proves per comparar les dues opcions, veiem que aplicant únicament la implementació a, el temps d'execució es lleugerament menor, però a l’hora de mostrar-nos la paraula que més apareix es incorrecte, ja que el que es fa a la opció a, es bloquejar l’arbre al fer el canvi entre el local al global, però l’increment del comptador de paraules no es realitza per la qual cosa la sortida de la paraula que més apareix es incorrecte.

Per l’altre costat al aplicar la opció b, la sortida de la paraula que més apareix es la correcta encara que, amb un temps major.