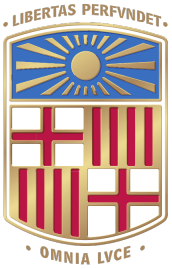

UNIVERSITAT DE BARCELONA

Pràctica 5. Programació multifil: Paradigma productor-consumidor

Javier González Béjar
Alejandro Guzmán Requena

5 desembre 2022

ÍNDEX

1 INTRODUCCIÓ

La cinquena pràctica de L'assignatura es centra en el processament de l'arxiu fent servir múltiples fils. En la darrera pràctica ens centràvem en els semàfors dintre de la programació multifil, ara modifiquem el mateix codi per implementar monitors centrant-nos en el paradigma productor-consumidor. Amb aquest paradigma aconseguirem separar, en fils diferents, la tasca de la lectura de dades de disc de l'extracció de la informació de les dades.

En aquest cas es tindran diversos fils secundaris i un sol de principal. El fil principal llegirà de disc un bloc de N línies seguides del fitxer per posteriorment que els fils secundaris puguin extreure a informació i així actualitzar la matriu num_flights. El fil principal i secundaris es comunicaràn amb un buffer creat previament pel principal, així com els fils secundaris necessaris.

2 TREBALL REALITZAT AL LABORATORI

1. Qüestions plantejades

A continuació es respondràn diverses qüestions que s'han plantejat en l'enunciat de la pràctica. S'acompanyarà la resposta a dites preguntes per imatges si es creu convenient.

Pregunta: El productor, el fil primari, és l'únic fil encarregat de llegir el fitxer de dades. Llegirà el fitxer de dades en blocs de N línies i "transferirà" cada bloc al buffer que comparteixen productor i consumidors. El buffer tindrà una mida per poder emmagatzemar B blocs, i es recomana que B sigui igual o superior al nombre F de fils secundaris. Teniu alguna noció de perquè ha de ser així? Podeu fer algun experiment per veure què passa si agafeu, per exemple, B = 1 amb F = 2 fils?

Per a major rendiment computacional, es recomana que el nombre de blocs del buffer sigui superior o igual al nombre de fils consumidors degut al fet que si tenim un nombre de blocs inferiors, els fils consumidors romandràn en espera fins que el fil productor generi nova informació.

En realitzar les proves per a verificar la explicació prèvia, observem efectivament que en executar el codi amb un nombre de blocs inferior al nombre de fils es nota una lleugera diferència en el temps d'execució, si ho comparem amb utilitzar un nombre de blocs igual o superior té un major temps computacional el primer cas.

Pregunta: Els fils secundaris accedeixen a recursos (variables) compartits entre ells. Quines seran les seccions crítiques? Quines parts del codi són les que s'han de protegir? Cal protegir la lectura del fitxer? Cal protegir l'extracció de dades del bloc? Cal protegir l'actualització de la variable num_flights? Comenteu la vostra resposta?

Les seccions crítiques seràn dues situades en parts diferents del codi. La primera es troba en el moment que s'accedeix al buffer de dades compartit entre tots els fils creats, la segona secció crítica es troba en modificar la matriu compartida de nombre de vols.

La lectura de fitxer no s'ha de protegir, degut a que només accedeix a aquest fitxer un sol fil (el productor) i la pràctica està plantejada per tenir només un productor i diversos consumidors. En cas de tenir-ne més d'un si que s'hauria de protegir per evitar l'accés a la vegada entre di-

versos productors.

En canvi, l'extracció de dades si que cal protegir-la per a que no se sobreescrigui la informació.

2. Experiments amb diferents L, B i F

B = 10 i F = 2

L	1	25	50	100
Temps 2007.csv	58.549908 s	20.039146 s	19.213037 s	19.133585 s
Temps 2008.csv	54.919078 s	19.098459 s	18.184519 s	18.108176 s

Table 2.1: Execució amb diferent nombre de línies per bloc

Primer variem el nombre de línies de bloc, i és senzill comprovar que a un major nombre de línies, els temps d'execució disminueixen de forma dràstica. Això es pot relacionar amb els continguts estudiats a teoria, ja que a menor línies passades per bloc, més cops hem d'utilitzar mecanismes de bloqueig, el que resulta en una ralentització de l'execució.

L = 100 i F = 2

B	1	2	5	10
Temps 2007.csv	19.433731 s	19.062896 s	19.816329 s	19.426910 s
Temps 2008.csv	18.653615 s	18.167807 s	18.121688 s	18.353221 s

Table 2.2: Execució amb diferent nombre de blocs

Quan variem el nombre de blocs no s'aprecia un canvi significant en quant a temps d'execució. No obstant, es pot apreciar que amb només 1 bloc l'algorisme és mínimament més lent. Podem deduir que això succeeix perquè aquest bloc és processat per 1 fil i l'altre es queda pausat durant el seu processament.

L = 100 i B = 10

F	1	2	3	4
Temps 2007.csv	24.267787 s	19.426910 s	11.993590 s	11.133585 s
Temps 2008.csv	23.939178 s	18.353221 s	12.307394 s	11.048157 s

Table 2.3: Execució amb diferent nombre de fils

Finalment, comprovem els resultats de l'execució amb diferent nombre de fils. Determinem que a major nombre de fils emprats, el temps d'execució és menor, degut a que és possible paralel·Litzar el processament de tasques. Això a l'entorn específic de proves del nostre ordinador. És possible que a un ordinador amb menors prestacions no tingués lloc aquesta millora. Especificacions com el nombre de CPUs són factors determinants.

3 CONCLUSIONS

Ens agradaria concloure que aquesta pràctica ens ha resultat de nou principalment útil per aprendre la importància del control de les seccions crítiques dels programes, en un paradigma de programació multifil amb monitors, ja que quan emprem més d'un fil d'execució tenim el risc de patir solapaments que derivin en resultats erronis.

A nivell personal pensem que la dificultat ha estat elevada en comparació a l'anterior, però l'hem aconseguit assolir sense gaires problemes. També agraïm haver fet la tercera pràctica com a preàmbul d'aquesta ja que això ens ha facilitat la comprensió del codi. Estem per tant satisfets per tant amb el nostre treball.