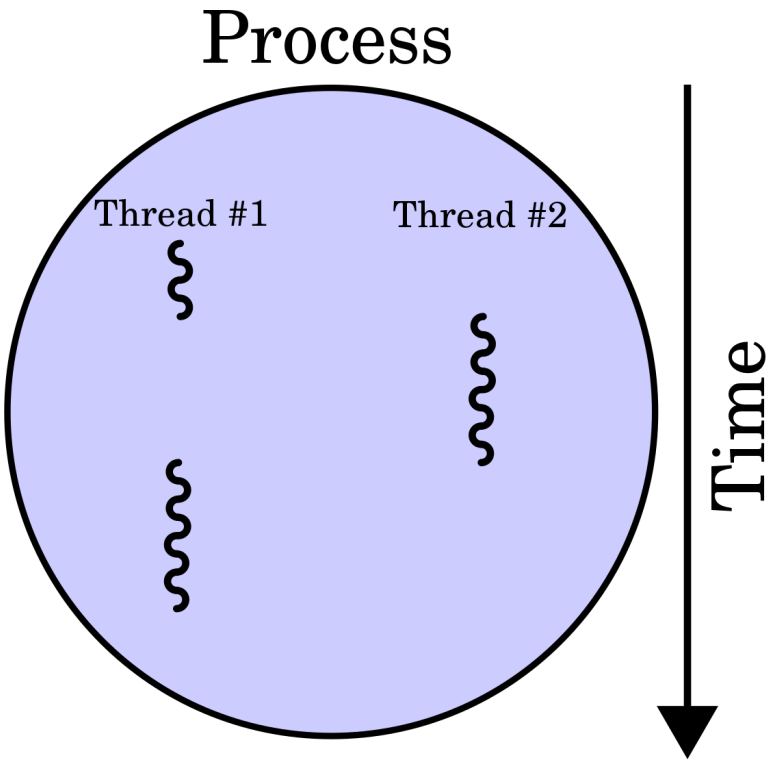
SISTEMES OPERATIUS 2:
Programació multifil:
paradigma productor-consumidor



Lin Zhipeng
Oscar de Caralt

Curs 2022-2023

Introducció:

La pràctica es centra en el processament de l'arxiu fent servir múltiples fils. Cada fil secundari llegeix de disc un bloc de N línies seguides del fitxer per extreure'n després la informació i així actualitzar la matriu num_flights. En aquesta nova pràctica es realitzarà una implementació multifil fent servir l'algorisme 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. La implementació de l'algorisme es faran servir les variables condicionals dels monitors.

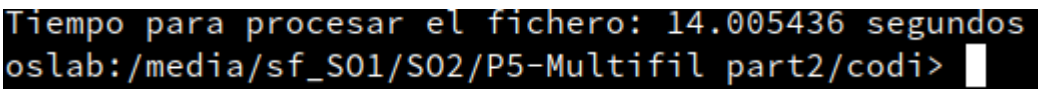
Preguntes a respondre:

1- 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?

Si B és igual o superior al nombre F de fils secundaris (consumidors), ens assegurem que el buffer sempre tingui suficient espai per emmagatzemar els blocs de dades que genera el productor. Això evita que el productor hagi d'esperar fins que hi hagi espai disponible al buffer, la qual alentiria l'execució del programa. D'aquesta manera, el productor pot continuar llegint el fitxer de dades i transferint els blocs al buffer de manera constant i eficient, mentre que els fils secundaris poden consumir els blocs del buffer sense interrupcions. Així també evitem que els fils consumidors no estiguin constantment esperant que el fil productor generi informació

Experiments:

- si B > F (B=3 i F=2):



- si B < F (B=1 i F= 2):

```
Tiempo para procesar el fichero: 14.124115 segundos
oslab:/media/sf_SO1/SO2/P5-Multifil part2/codi>
```

Com es pot apreciar quan el nombre de blocs a emmagatzemar al buffer és més gran que el nombre de fils, es redueix el temps de processament perquè els fils estan ocupats processant dades en lloc d'esperar que hi hagi més dades disponibles per processar. Això es deu al fet que els fils poden ser assignats a diferents blocs de dades en paral·lel i, per tant, poden processar més dades en menys temps.

A més, en aquesta situació, és possible que els fils no hagin de competir tant per accedir a les dades, ja que hi ha més dades disponibles per processar. Això pot reduir el temps que es triga a bloquejar i desbloquejar les dades compartides i, per tant, millorar l'eficiència del programa.

2- 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 són aquelles parts del codi en les quals diferents fils accedeixen i modifiquen recursos compartits. En aquest cas, les seccions crítiques seran aquelles en les quals els fils secundaris accedeixen al buffer per extreure els blocs de dades, així com aquelles en les quals actualitzen la variable num_flights que conté el nombre total de vols processats.

Per protegir aquestes seccions crítiques, és necessari utilitzar mecanismes de sincronització, com ara semàfors o variables de condició, per garantir que només un fil a la vegada pugui accedir a aquestes seccions i evitar conflictes de concurrència.

No seria necessari protegir la lectura del fitxer, ja que el productor és l'únic fil encarregat d'aquesta tasca i, per tant, no hi haurà conflictes de concurrència en aquesta secció del codi.

En resum, és necessari protegir les seccions crítiques en les quals els fils secundaris accedeixen al buffer i actualitzen la variable num_flights per evitar conflictes de concurrència.

Exercicis a fer:

Temps d'execució codi base de la pràctica 3 (sense fils) (fitxer 2007.csv)

```
Tiempo para procesar el fichero: 29.488943 segundos
oslab:/media/sf_SO1/SO2/P3-Anàlisi de codi/codi>
```

Temps d'execució de la pràctica 5 (fitxer 2007.csv):

```
Tiempo para procesar el fichero: 29.050784 segundos
oslab:/media/sf_SO1/SO2/P4-Multifil part1/src>
```

Temps d'execució de la pràctica 6 (fitxer 2007.csv):

```
Tiempo para procesar el fichero: 13.956844 segundos
oslab:/media/sf_SO1/SO2/P5-Multifil part2/codi>
```

Temps d'execucio per diferent nombre de linies per bloc.

Nombre de blocs B=10, nombre de fils secundaris F=2 i farem servir el fitxer 2007.csv.

Nombre de línies per Bloc	1	10	100	10000
temps (s)	45,82	17,52	14,23	14,06

Amb una línia per bloc, el temps d'execució és molt més gran a causa del repetit ús de mecanismes de bloqueig. Amb un major nombre de línies no s'utilitzen tant aquests mecanismes de bloqueig i, per tant, el temps d'execució és menor

Temps d'execució per diferent nombre de blocs.

Fixarem com a nombre de línies per bloc a 100, nombre de fils secundaris F=2 i farem servir el fitxer 2007.csv.

Nombre de blocs	1	2	5	10
temps (s)	15,31	15,19	14,87	14,16

En general, quan s'executa en múltiples fils, es divideix el treball en diferents blocs que es processen en paral·lel pels fils. Per tant, una quantitat més gran de blocs pot permetre al programa processar més dades en paral·lel, el que pot millorar el rendiment. No obstant això, és important tenir en compte que una quantitat molt gran de blocs també pot consumir més recursos del sistema i pot no ser tan eficient com una quantitat més petita en alguns casos. És necessari trobar un equilibri adequat entre la quantitat de blocs i el rendiment del programa.

Conclusions:

Un cop finalitzada aquesta pràctica hem pogut treure diverses conclusions sobre el temps d'execució del nostre programa multifil.

La mida del buffer afecta el temps d'execució del programa. Una mida de buffer més gran pot reduir la quantitat de vegades que el programa ha de fer servir el disc per llegir o escriure dades, el que pot millorar el rendiment. Malgrat això, una mida de buffer molt gran també pot consumir més memòria i pot no ser tan eficient com un buffer més petit en alguns casos.
La quantitat de línies per bloc també pot afectar el temps d'execució del programa. Una quantitat més gran de línies per bloc pot reduir la quantitat de vegades que el programa ha de fer servir el disc per llegir o escriure dades, el que pot millorar el rendiment. Malgrat això, una quantitat molt gran de línies per bloc també pot consumir més memòria i pot no ser tan eficient com una quantitat més petita en alguns casos.
La quantitat de blocs també pot afectar el temps d'execució del programa. Una quantitat més gran de blocs pot permetre al programa processar més dades en paral·lel, el que pot millorar el rendiment. Malgrat això, una quantitat molt gran de blocs també pot consumir més recursos del sistema i pot no ser tan eficient com una quantitat més petita en alguns casos.

És necessari trobar un equilibri adequat d'aquestes 3 variables per a obtenir un rendiment òptim per al nostre programa.