SISTEMAS CONCURRENTES Y PARALELOS
PRÀCTICA 3

Presentación

Objetivos

Esta práctica tiene como objetivo la sincronización de una aplicación concurrente multi-hilo utilizando los mecanismos de sincronización de phtreads. Para su realización se pondrán en práctica los conceptos de sincronización presentados en la asignatura.

Presentación de la práctica

Hay que presentar los archivos con el código fuente realizado. Toda la codificación se hará exclusivamente en lenguaje C/C++.

Enunciado de la práctica

Esta práctica tiene como base la aplicación de mapreduce concurrente realizada en la primera práctica de la asignatura. En esta segunda práctica nos vamos a centrar en dotar a la aplicación de los mecanismos de sincronización oportunos que permita una ejecución correcta, determinista y eficiente de la misma.

Sincronización

Se deberá implementar las siguientes sincronizaciones:

- **Evitar las condiciones de carrera**. Se tendrá que analizar todas las operaciones que pueden generar condiciones de carrera y implementar las secciones críticas que se consideren oportunas para garantizar un funcionamiento correcto y determinista de la aplicación. El objetivo final, es que todas las clases (Map, Reduce) implementadas en la aplicación sean thread-safe.

- **Sincronización Condicional.** En esta versión no se permite la utilizar los join para sincronizar las distintas etapas del MapReduce (básicamente entre la etapa de Suffle y la etapa de Reduce). En vez de los joins utilizaremos sincronización condicional para garantizar que los hilos de ejecución de los Reduce no empiecen hasta que todos los hilos de las etapas anteriores se hayan completado.

- **Estadísticas**. La aplicación calculará una serie de estadísticas locales a cada objeto/hilo y las globales durante la ejecución de cada una de las etapas:

    - Split: Número de ficheros leídos, número total de bytes leídos, número de líneas leídas y número de tuplas de entrada generadas.

    - Map: Número de Tuplas de entrada procesadas, número de bytes procesados, número de tuplas de salida generadas.

    - Suffle: Número de tuplas de salida procesadas, número de claves procesadas.

    - Reduce: Número de claves diferentes procesadas, número de ocurrencias procesadas, valor medio ocurrencias/clave, número bytes escritos de salida.

- **Progreso**. La aplicación mostrará por pantalla las estadísticas parciales al inicio y al final cada etapa (Split, Map, Suffle y Reduce). Mientras se muestran las estadísticas por pantalla, el resto de hilos no podrán continuar con su trabajo.

Al implementar las sincronizaciones se tienen que utilizar todos los mecanismos de sincronización que hemos visto en clase. En la versión pthread hay que utilizar: *Mutex*, Barreras y variables de condición.

Funciones involucradas.

En la práctica podéis utilizar las siguientes funciones de c:

- `int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr);`
- `int pthread_mutex_lock(pthread_mutex_t *mlock)`
- `int pthread_mutex_unlock(pthread_mutex_t *mlock)`
- `pthread_mutex_destroy(&lock);`
- `int pthread_barrier_init(pthread_barrier_t *restrict barrier, const pthread_barrierattr_t *restrict attr, unsigned count);`
- `int pthread_barrier_wait(pthread_barrier_t *barrier);`
- `int pthread_barrier_destroy(pthread_barrier_t *barrier);`
- `int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *attr);`
- `int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mlock);`
- `int pthread_cond_signal(pthread_cond_t *cond);`
- `int pthread_cond_broadcast(pthread_cond_t *cond);`
- `int pthread_cond_destroy(pthread_cond_t *cond);`

Análisis prestaciones

Calcular el tiempo de ejecución de la versión concurrente sincronizada y compararlo con el de la versión concurrente sin sincronizar y con la versión secuencial. Discutir los resultados obtenidos.

Analizar también, cómo evoluciona el tiempo de ejecución de la nueva versión en función del número de hilos utilizados en la aplicación concurrente. Entregar un pequeño informe en donde se muestren (preferiblemente de forma gráfica) y expliquen los resultados obtenidos.

Indicar en el informe las características del hardware en donde habéis obtenido los resultados (especialmente el número de procesadores/núcleos).

Evaluación

La evaluación de la práctica se realizará en función del grado de concurrencia logrado. Podemos definir 5 grados de concurrencia:

- Lectura fichero concurrente. El objetivo es que todos los splits de entrada se lean de forma concurrente.
- Maps Concurrentes. Cada una de las tareas de Map de la aplicación se deben ejecutar de forma concurrente entre si. Tenemos tantas tareas de Map como splits.
- Reducers concurrentes. El usuario puede especificar el número de reducers que se tienen que crear/ejecutar en la aplicación. Estos reducers se tienen que ejecutar de forma concurrente entre si.
- Suffle concurrente: Si la estructura de control utilizada lo permite, la ordenación, unión y partición también se debe realizar de forma concurrente.
- Procesamiento datos en pipeline o flujo de datos: Siempre que las sincronizaciones no lo impidan se debería permitir tener diferentes datos procesándose en diferentes etapas. Es decir, no se tiene que ejecutar todas las etapas secuencialmente entre si (excepto la etapa de reducción). Por lo tanto, mientras la etapa de Split lee un trozo del fichero, el map puede estar procesado una línea anterior y el suffle estar ordenando los resultados parciales generados previamente.

Para que la aplicación se puede ejecutar concurrentemente, os debéis asegurar que las estructuras de datos utilizadas son thread-safe y que no generarán condiciones de carrera.

Para obtener la máxima nota de esta práctica, se tiene que lograr una aplicación determinista que implemente todas las sincronizaciones mencionadas de forma eficiente y que el tiempo de ejecución de la aplicación concurrente sea mejor que el tiempo de ejecución de la versión secuencial.

En caso de cualquier error, la aplicación deberá eliminar mediante pthread_cancel.

[1]Bibliografía: En el contexto de un trabajo académico, la bibliografía es una lista de referencias que aparece al final de nuestro trabajo y que recoge, ordenada alfabéticamente, la información completa tanto de todas las fuentes citadas en el mismo, como de las consultadas durante su preparación.

Formato de entrega

MUY IMPORTANTE: La entrega de código que no compile correctamente, implicará suspender TODA la práctica.

No se aceptarán prácticas entregadas fuera de plazo (salvo por razones muy justificadas).

[1]https://biblioguias.biblioteca.deusto.es/c.php?g=213251&p=1406847#:~:text=Qu%C3%A9%20es%20una%20bibliograf%C3%ADa,las%20consultadas%20durante%20su%20preparaci%C3%B3n.

La entrega presencial de esta práctica es obligatoria para todos los miembros del grupo.

Comenzar vuestros programas con los comentarios:

```
/* ----------------------------------------------------------------
Práctica 3.
Código fuente: WordCount.c
Grau Informàtica
NIF i Nombre completo autor1.
NIF i Nombre completo autor2.
---------------------------------------------------------------- */
```

Para presentar la práctica dirigiros al apartado de Actividades del Campus Virtual de la asignatura de Sistemas Concurrentes y Paralelos, ir a la actividad "Práctica 1" y seguid las instrucciones allí indicadas.

Se creará un fichero tar/zip con todos los ficheros fuente de la práctica, con el siguiente comando:

```
$ tar -zcvf prac1.tgz fichero1 fichero2 ...
```

se creará el fichero "prac1.tgz" en donde se habrán empaquetado y comprimido los ficheros "fichero1", fichero2, y ...

Para extraer la información de un fichero tar se puede utilizar el comando:

```
$ tar -zxvf tot.tgz
```

El nombre del fichero tar tendrá el siguiente formato: "Apellido1Apellido2PRA1.tgz". Los apellidos se escribirán sin acentos. Si hay dos autores, indicar los dos apellidos de cada uno de los autores separados por "_". Por ejemplo, el estudiante "Perico Pirulo Palotes" utilizará el nombre de fichero: `PiruloPalotesPRA1.tgz`

Fecha de entrega

Entrega a través de Sakai el 21 de diciembre, entrega presencial en la clase de grupo de laboratorio el mismo día