Imagen que contiene Patrón de fondo

Descripción generada automáticamente

Práctica 3

Asignatura: *Sistemas Concurrentes y Paralelos*

Alumno: *Sergio Beltrán Guerrero (49259651E)*

*Martí Serratosa Sedó (48139505E)*

1. **Decisiones relacionadas con la implementación**
   1. **Uso de mutex**

En esta practica, hemos usado los mutex en varios sitios de la implementación para poder controlar que no se produzcan condiciones de carrera.

En el *MapReduce* hemos usado los mutex para poder comprobar que es el primer hilo que entra en la etapa para así poder imprimir las estadísticas iniciales. También los usamos en la etapa del Split, para poder guardar los mapas en el vector Mappers.

En la etapa de Reduce, usamos los mutex para guardar en el multimap Input las cuentas de las palabras (palabra, número de veces que aparece)

* 1. **Uso de barreras**

Las barreras las usamos para suplir al join que teníamos entre las 3 primeras etapas (Split, Map y Suffle) y el Reducer.

Por lo tanto, una vez el padre crea los hilos pasa a estado “wait” hasta que todos los hilos acaben. Lo mismo pasa dentro de los hilos, una vez acaba el hilo de hacer las tres primeras etapas pasa a estado “wait” hasta que todos acaben.

* 1. **Uso de variables de condición**

Las variables de condición las hemos usado para imprimir los resultados finales de las estadísticas.

Al final de cada etapa, los hilos envían una señal al proceso padre. Una vez el padre ha recibido tantas señales como hilos tiene, procede a imprimir por pantalla los resultados de la etapa.

1. **Estadísticas y Log**

Para la implementación de las estadísticas, hemos creado dos clases. La clase *Statistics* y la clase *Logger*. La funcionalidad de la clase *Statistics* es ir almacenando todas las estadísticas que nos pide el enunciado de cada etapa.

La clase *Logger* sirve para imprimir los resultados por pantalla con la fecha. Además, estos resultados se guardan en la carpeta /log que crea la propia clase.

1. **Análisis de resultados**

Para realizar el análisis de resultados hemos hecho las pruebas en tres máquinas distintas. Las máquinas en cuestión son:





**3.1 Resultados de Prácticas comparados entre los ordenadores**

A continuación, hacemos una breve recopilación de los datos recogidos en la primera práctica.



Podemos observar que el ordenador Msi con Linux mejora los tiempos notablemente. Aún así, los tiempos siguientes ejecutados con código concurrente son mucho mejores.

 

Podemos observar que los tiempos de la práctica 3 son todos mayores que la práctica 1 al tener que hacer más cálculos. Aún así, queremos comentar que **a medida que aumentamos el número de reducers los tiempos empeoran comparativamente.** Con dos reducers la diferencia entre la práctica 1 y 3 de media en los tres ordenadores es de 1-2 segundos mientras que con 64 reducers la diferencia ya se ha elevado a 4 segundos aproximadamente.

Después de debatirlo creemos que **la ejecución de la práctica 3 con 8 reducers es la mejor** opción en los tres ordenadores al optimizar el tiempo de la mejor manera sin usar demasiados recursos

**3.2 Resultados de Ordenadores comparados entre las prácticas**

En esta sección compararemos los resultados de las prácticas ejecutadas en cada ordenador.



Podemos observar que en el ordenador de Linux la diferencia entre las prácticas 1 i 3 con 2 reducers es la menor de todos los ordenadores, adjuntos a continuación.



Vemos que a medida que aumentamos los reducers la diferencia entre prácticas también aumenta, siendo inicialmente muy pequeña. Estos son debidos a que a más reducers hay que hacer más cálculos estadísticos. El ordenador 2 tiene el mayor tiempo de ejecución de forma secuencial con una diferencia abismal con el código concurrente.



El ordenador tres es el que presenta peores tiempos de forma concurrente, pero sigue habiendo el patrón de una diferencia pequeña entre prácticas 1 y 3 al inicio que se va aumentado a medida que aumentan los reducers. En este caso al tener menos núcleos de alto rendimiento observamos como con 2 reducers la diferencia no es muy grande en comparación con los otros ordenadores.

1. **Bibliografía**

* <https://www.geeksforgeeks.org/multithreading-in-cpp/>
* <https://stackoverflow.com/questions/17419893/c11-thread-multiple-threads-waiting-on-a-condition-variable>
* <https://cppcodetips.wordpress.com/2014/01/02/a-simple-logger-class-in-c/>
* <https://www.delftstack.com/es/howto/cpp/cpp-create-directory/>

1. **Annexos**

En la página siguiente hay la captura de los cálculos y gráficos hechos en Excel.