Programación Concurrente

Trabajo Práctico

Se desea implementar en Java un programa concurrente que encuentre números *perfectos*. Un número es perfecto si es igual a la suma de sus divisores propios positivos. Por ejemplo, 6 es un número perfecto ya que sus divisores (1, 2, y 3) suman 6.

El programa debe tomar como parámetro un entero n que indica la cantidad de números perfectos a buscar, y otro t que indica la cantidad de threads a utilizar. El programa debe imprimir los números perfectos a medida que los encuentra, y terminar su ejecución una vez que encuentra n números perfectos.

Elementos a entregar

Programa Java

Se deberán entregar los fuentes de la resolución del TP en un archivo `.zip`. El programa debe respetar la siguiente estructura:

1. Una clase `Main` con el punto de entrada del programa, tiene la responsabilidad de inicializar las estructuras mencionadas a continuación y producir números enteros grandes (`BigInteger`) consecutivos para que múltiples threads verifiquen si son perfectos o no. El programa debe terminar únicamente cuando la cantidad de números perfectos deseada haya sido alcanzada, y finalmente informar el tiempo transcurrido desde el inicio de la ejecución.

2. Una clase `Buffer` (implementada como un monitor utilizando métodos synchronized) que actúa como una cola FIFO concurrente de capacidad acotada. Es decir, bloquea a un lector intentando sacar un elemento cuando está vacía y bloquea a un productor intentando agregar un elemento cuando está llena. La capacidad del Buffer también debe ser un parámetro configurable desde la clase `Main`.

3. Una clase `PerfectWorker` que extiende de `Thread` y realiza la verificación de si un número es perfecto o no. Un `PerfectWorker` debe tomar los números a verificar de un `Buffer` conocido al momento de su creación. Si el número a verificar es negativo el `Thread` debe preparase para finalizar su ejecución sincronizándose con los otros threads utilizando la clase `Barrier`.

4. Una clase `ThreadPool`, que se encarga de instanciar e iniciar la cantidad de `PerfectWorkers` pedida por un usuario.

5. Cualquier otra clase que considere necesaria.

Al iniciar el programa la clase `Main` debe delegar la iniciación de los threads necesarios en la clase `ThreadPool` y luego introducir de a uno los números a verificar en el `Buffer`. Cada `PerfectWorker` en funcionamiento debe tomar números de a uno del `Buffer` y verificar si son o no perfectos. Cabe destacar que es inadmisible utilizar una cantidad de threads menor a la solicitada por el usuario.

Informe

Además del código, se requiere un informe corto en formato `pdf`, respetando el modelo presentado a continuación, que incluya los tiempos de ejecución del programa para encontrar los primeros 6 números perfectos con los siguientes parámetros:

- **Threads:** 1, 2, 4, 8 y 16
- **Tamaño del Buffer:** 1, 2, 4, 8 y 16

El informe, que es **condición necesaria para la aprobación del TP**, debe respetar el siguiente formato:

1. **Autoría:** Nombres, e-mail y número de legajo de quienes hicieron el trabajo (máximo 2 personas salvo excepciones acordadas con los docentes).
2. **Introducción:** Sección explicando el dominio del TP, el diseño del código y cualquier consideración adicional que considere relevante para la correcta interpretación de los resultados.
3. **Evaluación:** Sección explicando el proceso de evaluación, incluyendo los detalles del hardware donde se ejecutan las pruebas (modelo de microprocesador, memoria disponible y versión del sistema operativo). En esta sección deben incluirse una tabla reportando los tiempos en los que se logró encontrar la totalidad de las soluciones para cada cantidad de threads configurada, y el gráfico de la curva mostrando la progresión (superponiendo los tiempos para las distintas cantidades de threads).
4. **Análisis:** Sección en la que debe hacerse un análisis de los datos obtenidos en la evaluación, en particular destacando la combinación de parámetros con la que se obtuvo el tiempo de ejecución mínimo y la combinación que obtuvo el tiempo máximo.

Forma de Entrega

Se deben enviar el archivo `.zip` con el código del programa y el `.pdf` con el informe por email a las casillas de correo electrónico de *ambos* profesores con el subject:
`Entrega TP PCONC 1S2022 - (apellido1) - (apellido2)`
(donde `apellido1` y `apellido2` son los apellidos de las personas que constituyen el grupo). El mensaje debe ir *con copia* a la otra persona que integre el grupo, de forma tal que se pueda hacer la devolución respondiendo ese correo.

Es posible trabajar en un repositorio *git* compartido por las personas que compongan el grupo. En este caso, el repositorio debe ser **privado**. Al entregar, se deberá agregar a los profesores al repositorio. De todos modos se solicita que envíen el mail con las condiciones especificadas, aunque en vez de tener los archivos adjuntos incluirán el link al respositorio.

Fecha de Entrega

El TP debe entregarse antes del **Sábado 18 de Junio** a las **23:59hs**. En caso de ser necesario, se cuenta con una fecha de reentrega el **Sábado 9 de Julio**.