Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final de Prácticas
(Llamamiento Especial)
3 de Febrero de 2022

1. Notas de Procedimiento

1. Dispone de 180 minutos para completar el ejercicio.
2. Puede utilizar el material bibliográfico (libros) y copia de API que estime convenientes.
3. Entregue sus productos, utilizando la tarea de entrega disponible en el Campus Virtual, en un fichero (`.rar, .zip`) de nombre

   `Apellido1_Apellido_2_Nombre`

   que contendrá una subcarpeta por cada enunciado del examen, la cual contendrá a su vez el conjunto de ficheros que den solución a ese enunciado en particular. Cada fichero de código debe incluir un comentario con su nombre, apellidos y D.N.I.
4. Los productos de examen deben entregarse **única y exclusivamente** mediante la tarea de entrega habilitada para ello en el Campus Virtual, y dentro del plazo de tiempo establecido. Entregas efectuadas incumpliendo lo anterior no serán consideradas a efectos de corrección.

2. Criterios Generales de Corrección

Los códigos entregados deben satisfacer los siguientes criterios[1]:

- compilar correctamente sin errores ($-n$);
- ejecutarse sin lanzamiento de excepciones u otras anomalías graves de ejecución ($-n$);
- poseer una semántica que da soporte a la solución pedida en términos de entrada/salida, seguridad, vivacidad, rendimiento, etc. ($-n$);

[1] Entre paréntesis, se indica la puntuación a descontar derivada del incumplimiento del criterio, para un enunciado valorado con n puntos.

- recoger íntegramente todas las especificaciones establecidas en el documento de examen e implementarlas tal y como se especifican ($-1{,}5$/especificación).
- estar contenidos en ficheros conformes en número, formato y nomenclatura de nombres con lo explicitado en este documento de examen ($-n$).

3. Enunciados de Examen

1. (3.0 puntos) Considere el espacio vectorial sobre el cuerpo de los números reales dado por $\mathbb{R}^4 = \{(x_1, x_2, x_3, x_4) : x_i \in \mathbb{R}, i = 1, 2, 3, 4\}$, Definimos arbitrariamente a continuación las siguientes operaciones sobre vectores:

   - suma: si $q_1 = (x_1, x_2, x_3, x_4)$ y $q_2 = (y_1, y_2, y_3, y_4)$, entonces $q_1 + q_2 = (x_1 + y_1, x_2 + y_2, x_3 + y_3, x_4 + y_4)$
   - producto escalar: si $q_1 = (x_1, x_2, x_3, x_4)$ y $q_2 = (y_1, y_2, y_3, y_4)$, entonces $q_1 \cdot q_2 = \sum_{i=1}^{4} x_i \cdot y_i$
   - escalado de un vector: si $q = (x_1, x_2, x_3, x_4)$ y $p \in \mathbb{R}$, entonces $p \cdot q = (p \cdot x_1, p \cdot x_2, p \cdot x_3, p \cdot x_4)$
   - traza: $q = (x_1, x_2, x_3, x_4)$, entonces $traza(q) = \sum_{i=1}^{4} x_i$

   Se pide escribir una aplicación distribuida utilizando el *framework* Java-RMI que implemente las operaciones sobre vectores indicadas. Para ello, debe utilizar como elemento básico la siguiente interfaz (que estará en un fichero `IVector.java`):

```
import java.rmi.*;
public interface IVector extends Remote{
  public float[] sumVector(float[] q1, float[] q2)
          throws RemoteException;
  public float   pEscalVector (float[] q1, float[] q2)
          throws RemoteException;
  public float[] xVector(float[] q, float p)
          throws RemoteException;
  public float   trazaVector(float[] q)
          throws RemoteException;
}
```

Especificaciones adicionales:

- No utilice paquetes (`package`);
- Servidor: código en `sVector.java`; ejecución sobre el puerto por defecto; debe exportar cuatro objetos remotos distintos, que deben poseer diferentes nombres de servicio.
- Cliente: código en `cVector.java`; comienza por imprimir la lista de servicios remotos, previa consulta al servidor; cada operación se realizará sobre uno de los diferentes objetos remotos; primero lee por teclados dos vectores, pide al primer objeto servidor que los sume y muestra en pantalla los dos vectores leídos y el vector suma; a continuación, pide a un segundo objeto servidor multiplicar escalarmente los dos vectores ya leídos y muestra el resultado en pantalla; ahora, lee un número real, y escala el primer vector con él utilizando un tercer objeto servidor, mostrando en pantalla el vector escalado; finalmente, se imprime la traza del primer vector, calculada por el cuarto objeto servidor. Las lecturas de datos deben ir acompañadas de mensajes impresos indicando claramente los datos que se piden.

2. (3.5 puntos) Utilizando la biblioteca de paso de mensajes MPJ-Express, escriba un programa en Java que efectúe de manera distribuida el producto de dos matrices cuadradas de tipo `float` de acuerdo a las especificaciones que se indican, para las matrices A y B siguientes:

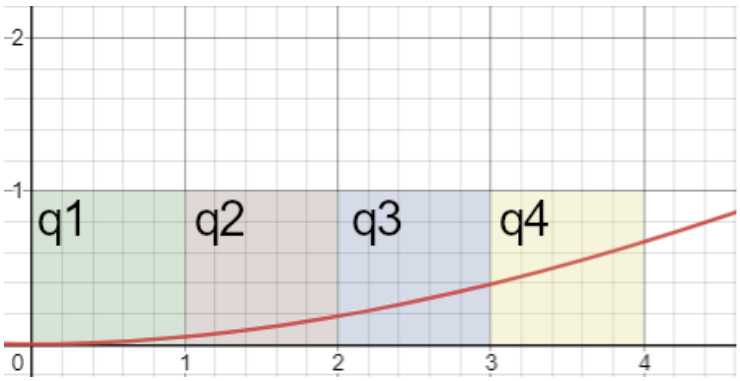
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \; B = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

   - No utilice paquetes (`package`);
   - El programa *master* (`rank=0`) definirá en su código las matrices indicadas en este documento obligatoriamente, y enviará a tres procesos *slaves* (`rank=1, 2 ,3`) la primera/segunda/tercera fila de la matriz A respectivamente, y la matriz B completa, utilizando obligatoriamente los métodos `Send/Recv`. **Se prohíbe el envío de la matriz A íntegra desde el *master* a los *slaves*; el incumplimiento de esta especificación invalida por completo el ejercicio, con independencia de los criterios generales de corrección**.
   - Los procesos slaves efectuarán respectivamente las siguientes operaciones sobre las dos matrices recibidas:
     - *Slave* con `rank=1`: computa la primera fila de la matriz producto y la envía al master.
     - *Slave* con `rank=2`: computa la segunda fila de la matriz producto y la envía al master.
     - *Slave* con `rank=3`: computa la tercera fila de la matriz producto y la envía al master.
     - El proceso master recibe las filas y las integra en la matriz producto resultante imprimiéndola.

- Las impresiones de los resultados deben in acompañadas de texto que ilustre claramente el contenido que se imprime. Las matrices deben imprimirse obligatoriamente en formato de matriz (cuadrado de números).
- Incluya en su programa como comentarios las órdenes de línea de comando necesarias para compilar y ejecutar su programa. Esta última debe incluir el valor del *flag* –`np` para crear el árbol de procesos que propone. **La ausencia de estas órdenes invalida por completo el ejercicio, con independencia de los criterios generales de corrección**.
- Guarde el código en `prodMat.java`.

3. (3.5 puntos) Se desea disponer de un programa que calcule, aplicando el método de Monte-Carlo, la siguiente integral definida (el desarrollo de la función en el intervalo de integración $[0, 4]$ se ilustra en la Figura):

$$\int_0^4 \frac{x^2}{(x+20)} dx$$



Desarrolle utilizando el lenguaje Java un programa que permita calcular la integral de forma paralela de acuerdo al siguiente conjunto de especificaciones:

- No utilice paquetes (`package`);
- Guarde el código en `intDefinida.java`.
- Tareas paralelas modeladas mediante implementación de la interfaz `Callable`.
- Resultados parciales recolectados utilizando la interfaz `Future`.
- Habrá exactamente cuatro tareas paralelas. Cada una efectuará la integral de la función en un subsegmento (q1, q2, q3 y q4) del intervalo total, aplicando en el mismo el método de Monte-Carlo. AYUDA: la función a integrar está acotada superiormente en todo el intervalo de integración por uno (ver Figura).
- Cada tarea debe disponer de su propio generador de números aleatorios, independiente de los demás; los generadores estarán incializados con diferentes semillas obligatoriamente.

- Cada tarea utilizará un total de 10^8 puntos aleatorios.
- Las tareas se procesarán a través de un ejecutor de *pool* de hebras de clase `ThreadPoolExecutor`.
- El programa calculará la integral de forma secuencial y paralela, y terminará imprimiendo los tiempos en nanosegundos de cálculo secuencial y paralelo, los valores de la integral calculada secuencial y paralelamente, y finalmente el *speedup*, con un ***output*** como el siguiente:

```
Tiempo secuencial: n nanosegundos
Tiempo paralelo: m nanosegundos
Integracion secuencial: p (p es el valor calculado)
Integración paralela: q (que es el valor calculado)
Speedup: s
```