Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 10

Se le plantean a continuación algunos ejercicios sencillos de programación distribuida/paralela con paso de mensajes mediante MPJ-Express. Debe desarrollar los programas que se piden. **Documente todo su código con etiquetas (será sometido a análisis con `javadoc`). Todos los ficheros de código deben incluir las órdenes de compilación y ejecución que propone como comentarios.**

1. Enunciados

1. Escriba un programa donde el proceso master/root (`id=0`) envíe dos vectores enteros de cuatro componentes a un proceso slave (`id=1`). El proceso slave efectuará el producto interno (producto escalar) de ambos vectores, y devolverá el resultado al proceso master, que lo imprimirá en pantalla. Fije los valores de ambos vectores en el código, y utilice los métodos `Send/Recv` para comunicación de datos entre procesos. Incluya un comentario al principio del código, con las órdenes de compilación (`javac`) y ejecución (`mpjrun.bat`) que propone para gestionar su código Guarde su trabajo en `prodInterno.java`.

2. Escriba un programa donde el proceso master/root (`id=0`) envíe utilizando `Bcast` un array de diez componentes enteros a cuatro procesos slaves (`id=1, 2, 3, 4`). Fije los valores del vector en el código, Cada slave efectúa un escalado del vector según su identificador (por ejemplo, el proceso slave con `id=3`, escala el vector multiplicando por tres todas sus componentes) y lo imprime en pantalla.Incluya un comentario al principio del código, con las órdenes de compilación (`javac`) y ejecución (`mpjrun.bat`) que propone para gestionar su código Guarde su trabajo en `escalMultiple.java`.

3. Escriba con MPJ-Express un programa que efectúe de manera distribuida la búsqueda de números primos en el rango $[0 - 10^7]$. El proceso 0 enviará a todos los procesos del comunicador el tamaño del rango de análisis que les corresponde mediante `Bcast`; cada proceso efectuará la búsqueda de números primos en el rango que le corresponde, y volcarán resultados al proceso 0 mediante `Reduce`. Para su comodidad, asuma un tamaño del

comunicador igual 10, Guarde su trabajo en `distributedIntegers.java`.

4. Una forma elegante de obtener aproximaciones al número π es mediante la obtención de aproximaciones numéricas a la siguiente integral; conforme se mejora la aproximación a la integral, la aproximación a π mejora:

$$\pi \approx \int_0^1 \frac{4}{1+x^2} dx$$

   Se pide paralelizar el cálculo de la aproximación a π, utilizando MPJ-Express, considerando lo siguiente:

   - utilizar sumas de *Riemann* para aproximar la integral.
   - si no recuerda el concepto de suma de *Riemann*, puede utilizar ChatGPT como fuente.
   - utilizar un parámetro de precisión n que indicará en cuántos subintervalos se divide el área bajo la curva.
   - el proceso 0 actuará como *master* y enviará una fracción del número de subintervalos de trabajo a 4 procesos *slaves*, que calcularán la subsuperficie asignada, y la enviarán de vuelta al *master*, que las sumará y presentará los resultados.
   - se proporciona en la carpeta de la práctica una posible solución secuencial al problema, que debe paralelizar.
   - guarde su trabajo en `piAproxMPJ.java`.

5. Escriba un programa de diseño libre que haga uso de los métodos de MPJ-Express `Scatter/Gather`. No vaya a lo mínimo; sea creativo. Guarde su trabajo en `scatterGather.java`.

6. Pida a ChatPGT la escritura de un programa en Java, que utilizando el API de MPJ-Express, efectúe el producto distribuido de matrices de 4×4 números enteros.