5. Actividad: Una aplicación simple distribuida

Objectivos: Desarrollar una simple aplicación distribuido de modo cliente servidor basandose en patrones de diseño concurrente y distribuidos y aprovechando de paquetes disponibles en Java.

Metodología: Esta práctica se realizará en horas de prácticas presenciales y en horas no presenciales. El tiempo de dedicación estimada es de 3 semanas (lo que queda del curso).

5.1. Repaso de las prácticas anteriores

1. Sabemos como lanzar cierto número de hilos basado en parámetros de la línea de comando.
2. Sabemos sincronizar los hilos al final del programa/hilo principal.
3. Sabemos medir el tiempo tanto de la parte concurrente del programa como de la parte secuencial del programa.
4. Sabemos distribuir una carga de trabajo a varios hilos. Los hilos realizan su trabajo de forma cuasi-paralelo e independiente y se sincronizan cuando todas las tareas hayan terminado.
5. Sabemos usar variables simples compartidos.
6. Sabemos usar las posibilidades de Java para sincronizar hilos de forma simple.
7. Sabemos sincronizar los hilos de forma directa, es decir, que un hilo despierta de forma directa a otro hilo (usando solamente las posibilidades intrínsicas del lenguaje Java).

5.2. Contexto

1. Queremos implementar un sistema de cálculo simple en una *nube* de ordenadores organizados como una arquitectura cliente-servidor. Como tarea de cálculo calculamos una parte del conjunto Mandelbrot.

   `(http://es.wikipedia.org/wiki/Conjunto_de_Mandelbrot)`

   Como formato simple para almacenar una imagen se puede aprovechar del formato de ficheros `PGM`.

   `(http://en.wikipedia.org/wiki/Netpbm_format)`

   Al principio bastaría generar ficheros en blanco y negro, donde por ejemplo un píxel negro indica que las coordenadas correspondientes pertencen al conjunto, un píxel blanco que no.

5.3. Una sistema de cálculo remoto

1. Elabora un paquete de servidor de tal manera que el servidor esté a la espera de recibir peticiones de trabajo de clientes.

2. Cuando un cliente pide trabajo al servidor, el servidor asigna a tal cliente un bloque de la imagen de tamaño adecuado para que el cliente calcule la parte correspondiente de la imagen, es decir, el cliente devuelve los píxeles calculados según la especificación del servidor (coordinadas de la ventana y parámetros necesarios).

3. El servidor colecciona todos los datos devueltos por los clientes hasta que la imagen esté completo, en cual momento escribe el fichero resultante.

4. Considera en el uso del patrón de diseño productor/consumidor para la implementación del control del flujo de datos.

5. Al lanzar el servidor se debe especificar por lo menos la región del conjunto Mandelbrot por calcular, la resolución de la misma, el número máximo de iteraciones, y el nombre de fichero de salida.

6. Al lanzar el cliente se debe especificar por lo menos los datos necesarios para contactar al servidor.

7. (mejora) Una vez funcionando el sistema de cálculo en sí, intenta modificar el sistema para que sea robusto a fallos en un cliente, es decir, si un cliente todavía no ha devuelto el resultado y llega una nueva petición de trabajo, el cálculo faltante se puede asignar otra vez a tal nueva petición.