**Reporte procesando imágenes con el Fork-Join framework**

Melannie Torres (A01361808)

Tecnológico de Monterrey, Campus Querétaro

*A013618086@itesm.mx*

23 de septiembre de 2018

**Resumen**

Este documento muestra la comparación tanto de la implementación como de los

resultados de procesar una imagen para convertirla a escala de grises de manera secuencial y concurrente utilizando el framework *Fork-Join*.

**1. Introducción**

*Fork-join* es un framework de java que permite aprovechar multiples procesadores. *Fork-Join* utiliza un algoritmo llamado *work-stealing* en el cual las threads que terminan su trabajo pueden “robar” tareas de aquellas que aún no han terminado. Se llama *fork* a la fase en la que las tareas se parten en tareas más simples y *join* a la fase que se unen todas las tareas para entregar el resultado final.

A fin de poder mostrar el incremento de la velocidad de ejecución utilizando *Fork-Join*, se utilizará una métrica que muestra la razón entre el tiempo de ejecución de manera secuencial y de manera concurrente llamada *speedup*.

**2. Implementación**

El transformador de imágnes a escala de grises se implementó en java de manera secuencial y de manera concurrente utilizando *Fork-Join*.

Equipo utilizado:

o Mac OS Sierra 10.12.6

o Procesador: 1.3 GHz Intel Core i5

o Memoria: 4 DB 1600 MHz DDR3

Para calcular qué tono de gris un pixel después de la transformación se utilizó la siguiente fórmula:

gray = (r\*factor\_r)+(g\*factor\_g)+(b\*factor\_b);

Donde r, g y b son los valores del color actual y factor\_r, factor\_g y factor\_b son los factores con los que se hizo la transformación.

Una vez obtenido este valor se fue asigno r, g y b para que la imagen fuera transformada.

La transformación de manera secuencial se realizó recorriendo pixel por pixel modificando uno a la vez. Transformar la imagen de christian-joudrey-96208-unsplash.jpg tomó en promedio segundos y la de saketh-garuda-391215-unsplash.jpg segundos.

A fin de que la transformación fuera realizada utilizando *Fork-Join* se creó una pool de la cuál las threads tomaban las tareas.

pool = new ForkJoinPool(MAXTHREADS);

pool.invoke(new LenaForkJoin(src, dest, w, h, 0, src.length));

Transformar la imagen de christian-joudrey-96208-unsplash.jpg tomó en promedio segundos y transformar la imagen de saketh-garuda-391215-unsplash.jpg segundos.

**3. Resultados.**

El *speedup* es la métrica que muestra cuánto se mejoró la velocidad de un programa al paralelizarlo.

Con la imagen: christian-joudrey-96208-unsplash.jpg

Con la imagen : saketh-garuda-391215-unsplash.jpg

En promedio se obtuvo un *speedup* de 1.895, es decir que el programa corre aproximadamente 1.895 veces más rápido con el uso de *Fork-Join*.

Este trabajo fue relativamente sencillo ya que se utilizó como base los códigos vistos en la clase de Multiprocesadores con Pedro Pérez.

**3. Notas adicionales.**

Dentro de los archivos .java subidos, se encuentran programas de Pedro Pérez, ya que son dependencias de algunos archivos entregados.

**3. Referencias**

ForkJoinPool (Java Platform SE 8 ). (2018). Recuperado de https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html

Java - Fork/join, the work-stealing thread pool. (2017). Retrieved from https://www.logicbig.com/tutorials/core-java-tutorial/java-multi-threading/fork-and-join.html