Actividad de programación: Procesamiento de imágenes con forkjoin

Esteban Pérez Herrera (A01204739)
Tecnológico de Monterrey, Campus Querétaro
A01204739@itesm.mx

19 de septiembre de 2018

Resumen

Durante la realización de esta práctica se realizó el mismo proceso, en este caso el procesamiento de una imagen de color a escala de grises, tanto de manera secuencial como paralela, donde la última fue capaz de lograr un mejor desempeño.

1. Introducción

Hoy en día el equipo de cómputo cuenta con la capacidad de ejecutar múltiples tareas a de manera paralela gracias a que posee varios procesadores que físicamente se encargan de ello, esto con el fin de acelerar la tarea previamente ejecutada de manera secuencial. La tarea seleccionada fue procesar una imagen a color para posteriormente pasarla a escala de grises

2. Implementación

- Equipo utilizado.
  - CPU: I7 4710HQ, 4C,8TH
  - Memoria: 16GB RAM
  - OS: Ubuntu 16.04 LTS

La primera función en ser definida fue la implementación de la función que transforma cada pixel de su valor rgb a escala de grises, esto se lleva a cabo tomando los valores de r,g y b de manera individual para posteriormente multiplicarlos por su factor ya definido previamente, se asignan los valores al pixel correspondiente en el destino, esta función es igual en ambos casos.

```
private void gray_pixel(int ren, int col) {

          int i, j;

          int pixel, dpixel;

          float r=0, g=0, b=0;
```

```
    pixel = src[(ren*width)+col];
    r = (float) ((pixel & 0x00ff0000) >> 16);
    g = (float) ((pixel & 0x0000ff00) >> 8);
    b = (float) ((pixel & 0x000000ff) >> 0);

    r*=.299;
    g*=.587;
    b*=.114;

          dpixel = (0xff000000)
                   | (((int) (r + g + b)) << 16)
                   | (((int) (r + g + b)) << 8)
                   | (((int) (r + g + b)) << 0);
          dest[(ren * width) + col] = dpixel;
     }
```

En este función se lleva a cabo la transformación de la imagen, durante la misma se recorre por completo la imagen, hay una función llamada `computeDirectly()` en el fork-join, que es prácticamente igual excepto que en lugar de recorrer la imagen completo definiendo `size` sólo recorrer el segmento que se le asigna.

```
void doMagic() {
    int index, size;
    int ren, col;

    size = width * height;
    for (index = 0; index < size; index++) {
      ren = index / width;
      col = index % width;
      gray_pixel(ren, col);
    }
  }
```

Finalmente la siguiente función solamente incluida en el forkjoin ejecuta el uso de threads, donde consiste en primero verificar el tamaño de la carga de trabajo, para saber si proceder con el procesamiento de su segmento asignado o proceder a dividir el tamaño de la tarea en dos y asignarlos.

```
@Override
protected void compute() {
     if((this.end - this.start) <= GrayScale_ForkJoin.MIN ) {
          computeDirectly();
     }else{
          int middle = (end + start) / 2;
          invokeAll(new GrayScale_ForkJoin(src, dest, width, height, start, middle),
          new GrayScale_ForkJoin(src, dest, width, height, middle, end));
     }
}
```

3. Resultados

Las pruebas de los programas fueron realizadas tanto de manera loca, se demostró una mejora en la implementación paralela sobre la secuencial.

Serial	ForkJoin	Speedup
218.2	61.8	3.53

3. Referencias

Pérez, P. O. (n.d.). Clase ITESM.