CONCLUSIÓN
PARALELISMO
PARALELISMO
ESPEJO
GRISES
TÉCNICAS

Multiprocesadores

Segundo Examen Parcial

Luis Fernando A01730944
Saúl Román A01730641

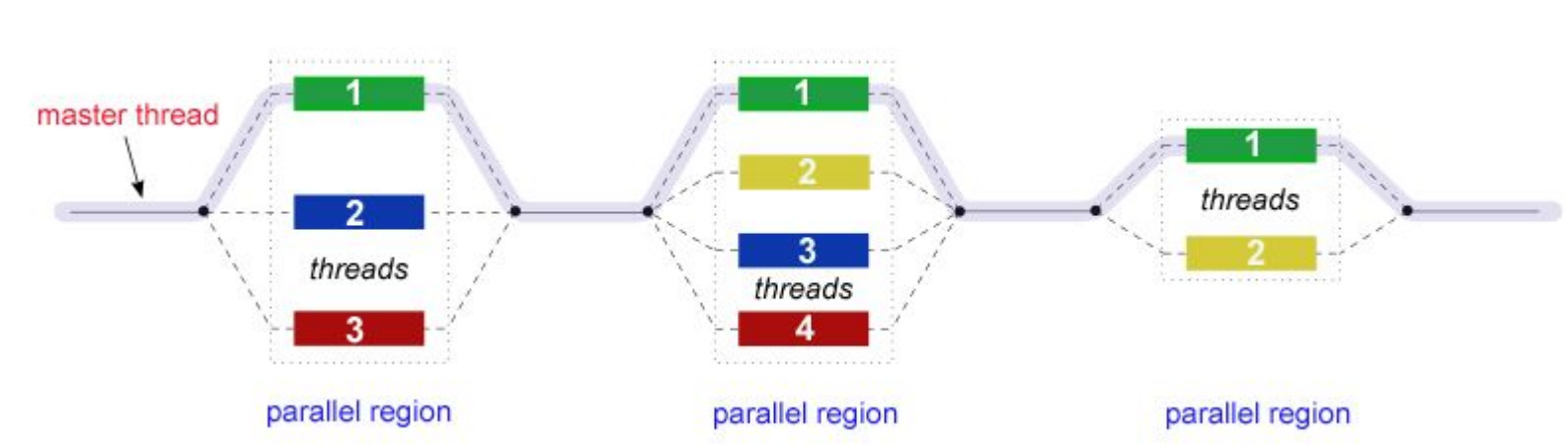
Técnicas de optimaizaciónde recursos.

- Código original:

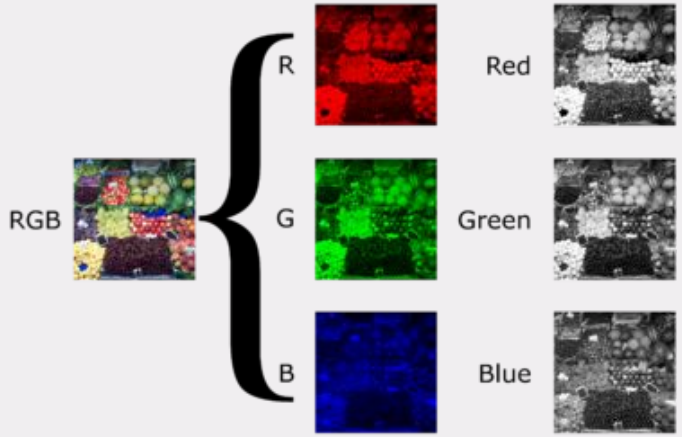
```
for(int i = 0; i < 100; i++)
  doStuff(i);
```

- Código modificado:

```
for(int i = 0; i < 100; ){
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
  doStuff(i); i++;
}
```



Existen Diversas técnicas que nos ayudan a mejorar el tiempo de procesamiento de algún programa computacional como lo pueden ser **técnicas de acceso a memoria** y la **paralelización**.



RGB a Escala de grises.

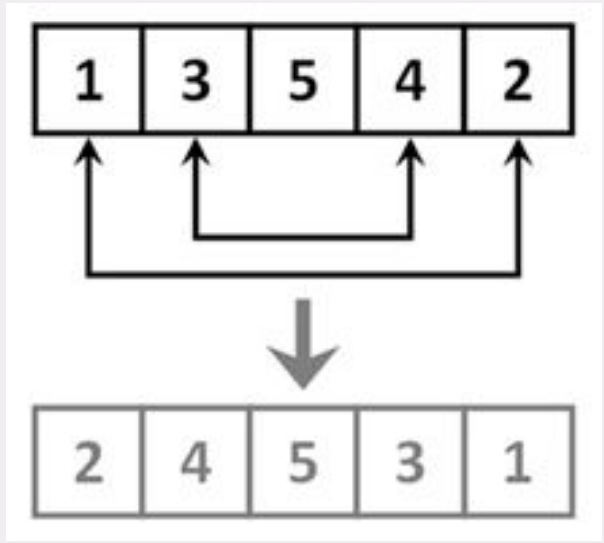
```
b = *(arr_in + i);
g = *(arr_in + i + 1);
r = *(arr_in + i + 2);
unsigned char pixel = 0.21 * r + 0.72 * g + 0.07 * b;
my_arr[index].b = pixel;
my_arr[index].g = pixel;
my_arr[index].r = pixel;
```

Fragmento de código.

Imágenes originales

La técnica que permite convertir a una imagen en rgb a escala de grises, consiste en tomar los colores rgb, pasar dichos colores por una **fórmula** que hace un balance para obtener imágenes en escala de grises. Dicha fórmula es: $pixel = 0.21 * r + 0.72 * g + 0.007 * b$

Original Invertida



Algoritmo

```
for (int i = 0; i < alto - 1; i++)
{
    for(int j = 0; j < ancho-1; j++){
        out_arr[(i*ancho)+j].b = my_arr[(i*ancho)+ancho-j].b;
        out_arr[(i*ancho)+j].g = my_arr[(i*ancho)+ancho-j].g;
        out_arr[(i*ancho)+j].r = my_arr[(i*ancho)+ancho-j].r;

    }
}
```

Fragmento de código

Se establece una relación de **pixel a pixel** en la que se **invierte** la orientación con respecto al plano vertical, ya que es un arreglo de una dimensión, para el cual se consigue tener un efecto espejo de imagen.

OMP Schedule

```
#pragma omp parallel
{
  #pragma omp for ordered schedule(static,1)
  for (int i = 0; i < ((ancho * 3) + comp) * alto; i += 3){
    b = *(arr_in + i);
    g = *(arr_in + i + 1);
    r = *(arr_in + i + 2);
    unsigned char pixel = 0.21 * r + 0.72 * g + 0.07 * b;
    my_arr[index].b = pixel;
    my_arr[index].g = pixel;
    my_arr[index].r = pixel;
    index++;

    count += 3;
    if (count == ancho * 3){
      i += comp;
      count = 0;
    }
  }

  //Efecto Espejo
  #pragma omp for ordered schedule(static,1)
```

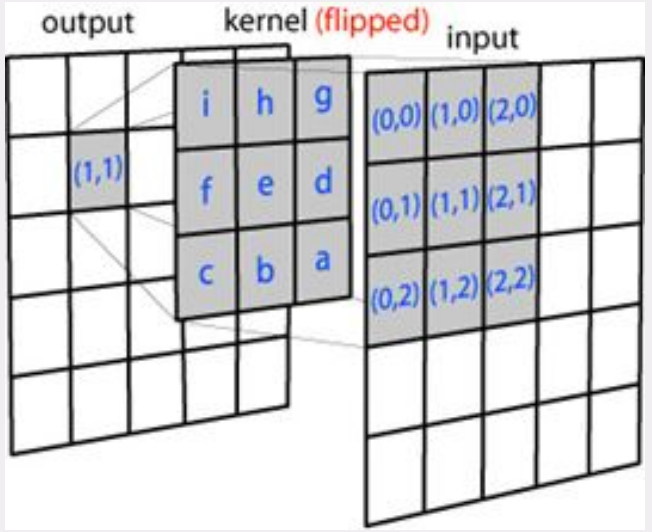
0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Zero Padding

Dividido

```
for (int i = 0; i < (((ancho * 3) + comp) * alto) / 4; i++)
{
  *(arr_in1 + i) = fgetc(image);
  *(arr_out1 + i) = *(arr_in1 + i);
}

for (int i = 0; i < (((ancho * 3) + comp) * alto) / 4; i++)
{
  *(arr_in2 + i) = fgetc(image);
  *(arr_out2 + i) = *(arr_in2 + i);
}
```

```
arr_out1 = (unsigned char *)divisionImg(arr_in1,

arr_out2 = (unsigned char *)divisionImg(arr_in2,

arr_out3 = (unsigned char *)divisionImg(arr_in3,

arr_out4 = (unsigned char *)divisionImg(arr_in4,
```

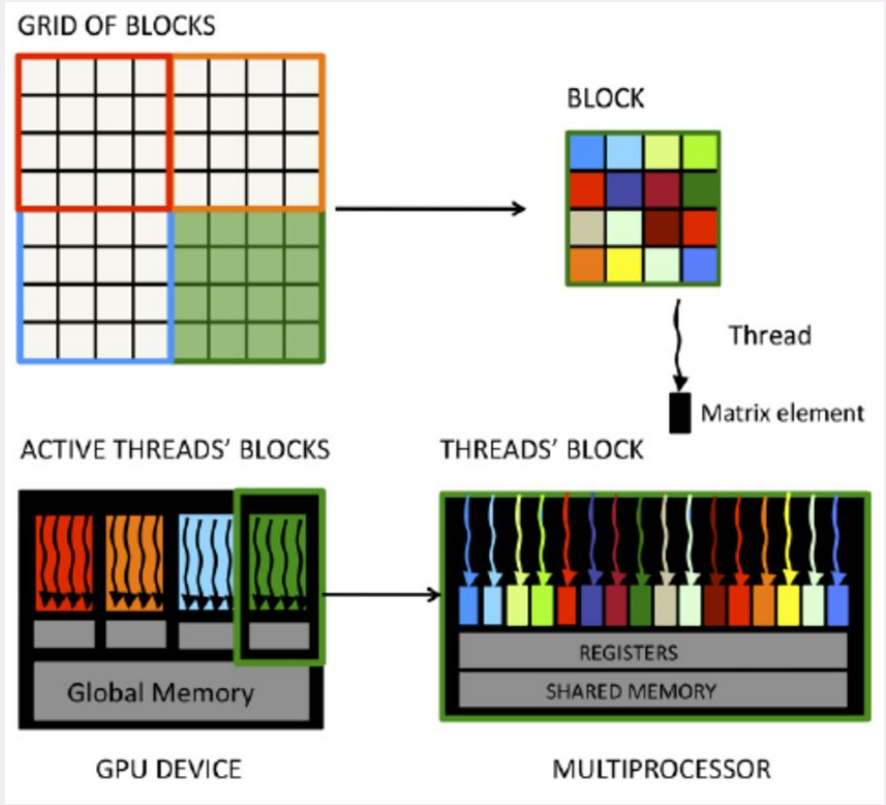


Convolución

```
#define NUM_THREADS 4

void main(){
  omp_set_num_threads(NUM_THREADS);
  #pragma omp parallel
  {
    #pragma omp sections
    {
      #pragma omp section
      section1();
      #pragma omp section
      section2();
    }
  }
}
```

OMP Sections

La técnica usada en este caso fue **dividir en cuatro** secciones iguales la imagen de entrada y asignar un **hilo de ejecución** a cada una de dichas secciones con el objetivo de **mejorar los tiempos** de ejecución. En este caso como se ve en la imagen se dividió en cuatro parte por igual y se asigna cada bloque a un hilo distinto.



Reparto del proceso

El **efecto blur** se logra haciendo una **convolución,** es decir una multiplicación de matrices, entre la matriz de entrada como es una imagen y una matriz de valores numéricos que se considera como **kernel**.

Recursos de nuestros sistemas

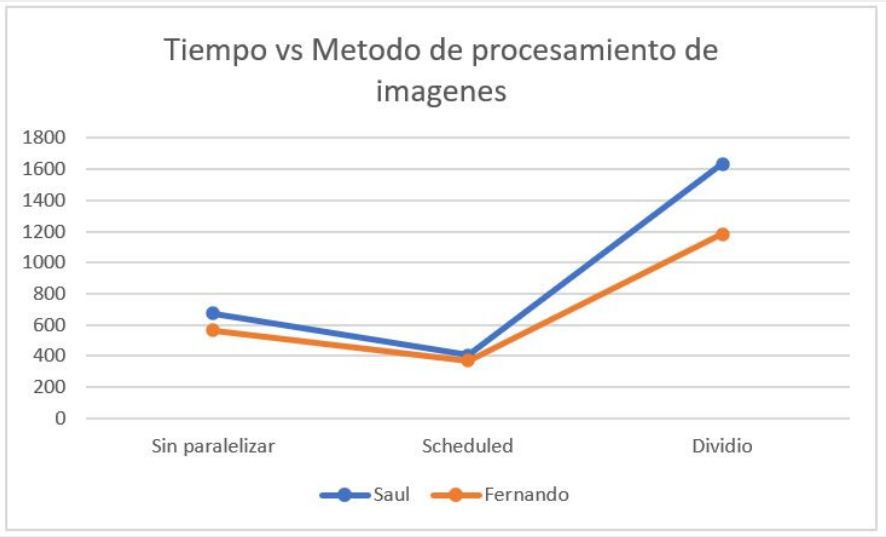
Pc	Sin paralelizar	Scheduled	Dividido
Saul (ubuntu)	674.141	407.065	1634.256
Fernando (Windows)	565.379	370.284	1184.1

Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.60 GHz
AM): 8.00 GB (7.89 GB utilizable)
Sistema operativo de 64 bits, procesador x64

Fig. 9 Recursos del sistema Saul

50
ositivo
DESKTOP-MOPQC99
Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz
16.0 GB

Fig. 10 Recursos del sistema Luis Fernando



CONCLUSIÓN
PARALELISMO
PARALELISMO
ESPEJO
GRISES
TÉCNICAS