Scan Sky: OpenMP

García Prado, Sergio Calvo Rojo, Adrián

11 de marzo de 2017

Resumen

Abstract.

I. Introducción

En este documento se exponen el conjunto de mejoras realizadas sobre "un código secuencial para contar el número de objetos diferentes que se ven en una imagen o fotografía celeste, en general de espacio profundo, obtenida por un radiotelescopio"[1]

"Las imágenes ya han sido procesadas, discretizando la luminosidad observada en cada punto en 16 diferentes niveles de grises o colores. Los pixels del fondo del espacio, una vez eliminado el ruido, tienen el índice de color 0. Los pixels de la imagen con una luminosidad o color suficientemente parecidos se representan con un mismo valor entre 1 y 15."[1]

"La imagen se carga en una matriz desde un fichero de texto plano. El fichero contiene un número entero en cada línea. Las dos primeras líneas contienen el número de filas y columnas de la imagen. El resto son números entre 0 y 15 con los valores de cada pixel, ordenados por filas."[1]

"Los pixels del mismo índice de color que están juntos, en horizontal o vertical (no en diagonal), se considera que son del mismo objeto. El programa etiqueta cada objeto de la imagen con una etiqueta diferente. Todos los pixels del mismo objeto tendrán la misma etiqueta. Para determinar el número de objetos, al final se cuentan el número de etiquetas diferentes. Los píxeles de índice 0 no se etiquetan."[1]

II. Optimización

[TODO]

```
#pragma omp parallel \
default(none), \
shared(k_indexer, k_max, matrixData, matrixResult, \
    matrixResultCopy,columns,rows, flagCambio,numBlocks)
{
    // ...
}
```

Figura 1: *Inicialización de la región paralela*

[TODO]

[TODO]

[TODO]

```
/* 4.2.2 Computo y detecto si ha habido cambios */
#pragma omp for \
schedule(static), \
reduction(||:flagCambio),\
private(k)
for(k=0;k<k_max;k++){
    if((matrixData[k_indexer[k]-columns] == matrixData[k_indexer[k]]) &&
        (matrixResult[k_indexer[k]] > matrixResultCopy[k_indexer[k]-columns]))
    {
        matrixResult[k_indexer[k]] = matrixResultCopy[k_indexer[k]-columns];
        flagCambio = 1;
    }
    // ...
}
```

Figura 2: *Inicialización de la región paralela*

[TODO]

[TODO]

```
#pragma omp for \
nowait,\
schedule(dynamic, ((rows-1)*(columns-1))/omp_get_num_threads()), \
private(i,j,k)
for(i = 1; i < rows-1; i++){
    for(j = 1; j < columns-1; j++){
        // Si es 0 se trata del fondo y no lo computamos
        if(matrixData[i*(columns)+j]!=0){
            matrixResult[i*(columns)+j]=i*(columns)+j;
            #pragma omp atomic capture
            {
                k = k_max; k_max++;
            }
            k_indexer[k] = i*(columns)+j;
        } else {
            matrixResult[i*(columns)+j]=-1;
        }
    }
    matrixResult[i*(columns)]=-1;
    matrixResult[i*(columns)+columns-1]=-1;
}

#pragma omp for \
nowait,\
schedule(static), \
private(j)
for(j=1;j< columns-1; j++){
    matrixResult[j]=-1;
    matrixResult[(rows-1)*(columns)+j]=-1;
}
```

Figura 3: *Inicialización de la región paralela*

```
/* 4.2.1 Actualizacion copia */
#pragma omp single
{
    flagCambio=0;
    temp = matrixResultCopy;
    matrixResultCopy = matrixResult;
    matrixResult = temp;
}
```

Figura 4: *Inicialización de la región paralela*

```
/* 4.3 Inicio cuenta del numero de bloques */
#pragma omp for \
schedule(dynamic, k_max/omp_get_num_threads()), \
private(k),\
reduction(+:numBlocks)
for(k=0;k<k_max;k++){
    if(matrixResult[k_indexer[k]] == k_indexer[k]) numBlocks++;
}
```

Figura 5: *Inicialización de la región paralela*

III. Conclusiones

[TODO]

Referencias

[1] Computación Paralela, 2016/17.