Arquitectura de Computadores

Aplicación OpenMP

Como ya se ha comentado en la presentación del proyecto, queremos implementar un programa paralelo para acelerar el proceso de digitalización de los libros de la biblioteca del Fondo de los Benedictinos. En este programa debes aplicar los conceptos y las técnicas trabajadas en el puzle –sincronización, planificación y rendimiento–.

En el directorio `templates/aplicacion` tienes el material necesario para llevar a cabo esta parte del proyecto:

- `pagina_s.c:` contiene el programa principal de la aplicación a desarrollar.
- `filter.c, encript.c y preparar_subida.c`: ficheros fuente (ficheros .c) que tenéis que completar para el filtrado, cifrado o encriptado y "subida" a la plataforma on-line de las páginas a tratar. Además, el módulo `pixmap` contiene las funciones de gestión de imágenes que son necesarias para completar la aplicación. Os entregamos los ficheros de cabecera (ficheros .h) y módulos objeto (ficheros .o) necesarios.
- `Libro1.pgm` (2.758 x 3.986 píxeles), `Libro2.pgm` (4.096 x 4.980 píxeles) y `Libro3.pgm` (6.520 x 8.080 píxeles): ficheros que contienen las páginas a procesar. Son páginas de diferentes tamaños, con lo que el tiempo de procesado también será distinto para cada caso. Utilizad la página `Libro1.pgm` (la más pequeña) para desarrollar la aplicación y, posteriormente, analizar los resultados obtenidos también para las otras dos páginas.

NOTA. Para visualizar las páginas, desde la sesión en linux podéis utilizar el comando `display` que recibe como parámetro la imagen a visualizar. Para ello, la conexión a la máquina remota debe realizarse utilizando opción –X del comando `ssh`.

```
$ ssh -X cuenta@u010415.gi.ehu.eus
$ display imagen.pgm &
```

El trabajo a realizar es el siguiente:

A. Versión serie de la aplicación

- Leer la documentación relativa al filtrado paso-bajo de imágenes (página eGela de la asignatura): apartado 6.2.2 del doucmento `ILWIS.pdf` y el apartado 3.4.1 del documento `Filtros.pdf`. Completa el código de las funciones para tratar las páginas del fichero `filter.c`

  Las imágenes de las páginas están en blanco y negro y cada pixel puede tomar un nivel de gris que varía desde 0 (negro) a 255 (blanco)

- Completa el código de las funciones de encriptado de los ficheros `encript.c`. Para implementar el algoritmo de encriptado consulta el documento `Hill.pdf` (página eGela de la asignatura).

- Completar el código de la función de "subida" de la página en el fichero `preparar_subida.c.` Esta función procesa las filas de la imagen de la página encriptada para generar un código de error. El código objeto para generar este código de error está en el fichero `upload.o`. El tiempo para generar este código es variable y depende de la posición y del valor del pixel.

- Genera el comando de compilación correspondiente a la versión serie de la aplicación. Para ello debes editar un fichero con el comando y cambiarle los permisos para convertirlo en ejecutable: `chmod 700 fichero_comando.` Ten en cuenta que se utilizan algunas funciones matemáticas, por lo que debes añadir la opción `–lm` en el comando de compilación.

- Prueba el correcto funcionamiento del programa serie utilizando las páginas proporcionadas. En el directorio de trabajo podéis encontrar el ejecutable **`recuperar`** que permite obtener la imagen de la página filtrada a partir de la cifrada para comprobar el correcto funcionamiento del proceso de encriptado. Como parámetros, este programa recibe las páginas cifrada y filtrada, y los valores necesarios para el correcto descifrado de la página cifrada.

B. Versión paralela de la aplicación

- Implementa una versión paralela eficiente del programa utilizando OpenMP. Comprueba su correcto funcionamiento. Para ello puedes imprimir los resultados en un fichero y con el comando `diff` compararlos con los resultados serie.

- Estudiar el rendimiento obtenido en la implementación paralela, en función del número de threads, frente a la versión serie del programa. Ejecuta el programa paralelo con 2, 4, 8, 16, 24 y 32 threads.

C. Escribe el informe técnico de la aplicación.

- Realiza una memoria que explique claramente la implementación realizada y los resultados obtenidos. Este informe debe contener los programas propuestos (serie y paralelo), los resultados obtenidos, tiempos y gráficas que representen el comportamiento de los programas.

El esquema de las funciones que forman parte de la aplicación es el siguiente:

pagina_s.c (programa principal)

filter (.c, .h)
copiar_pagina
filtrar_pagina
generar_pagina_filtrada

encript (.c, .h)
encriptar_pixeles
generar_pagina_encriptada

preparar_subida (.c, .h) [upload.o (disponible)]
preparar_subida

El programa principal recibe dos parámetros: la página (en formato pgm) y el límite de filtrado. El filtro se aplica a la imagen de la página mientras que el valor medio de los píxeles de la página menos el mínimo supere ese límite de filtrado.

```
$ pagina_s Libro1.pgm 177
```

El límite de filtrado es distinto para cada página:

- `Libro1.pgm`: 177
- `Libro2.pgm`: 177,5
- `Libro3.pgm`: 125

Para comprobar la corrección del programa hay que tener en cuenta que para procesar `Libro1.pgm` se necesitan 10 pasos y el límite de finalizado es 176,723.

pixmap (.c, .h, **.o**)

los ficheros pixmap contienen definiciones y funciones auxiliares para tratar las imágenes, entre ellas:

estructura de datos *imagen* (*struct*) de tamaño (h × w)

constantes NEGRO (0), BLANCO (255), y MAX_VAL (256)

load_pixmap (…) leer una imagen y cargar la estructura correspondiente.

store_pixmap (…) almacenar los datos de una imagen en el disco.

generar_pagina (…) generar una estructura de datos para una página e inicializar los pixeles de la imagen de la página con un valor

borrar_pagina (…) libera la memoria dinámica asignada a una imagen de la página

```
/*****************************************************************************
  pagina_s.c: programa principal
*****************************************************************************/

void main (int argc, char **argv)
{
  char name[100];
  int i;

  pagina page_ori,page_fil,page_cif;    // Paginas: original, filtrada y cifrada

  struct timeval t0,t1;                 // Calculo de tiempos
  double tej1, tej2, tej3,tej;

  if (argc != 3) { printf ("\nUSO: programa pagina (.pgm) limite\n"); exit (0);}

  if (load_pixmap(argv[1], &page_ori) == 0) {
    printf ("\nError en lectura del fichero de entrada: %s\n\n",argv[1]);
    exit (0);
  }

  printf("\n --> Procesando la pagina: %s (%d x %d). Limite: %.1f\n",
                argv[1],page_ori.h,page_ori.w,atof(argv[2])\n");

  gettimeofday(&t0, 0);
  generar_pagina_filtrada(&page_ori,&page_fil,atof(argv[2]));
  gettimeofday(&t1, 0);
  tej1 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Filtrar la pagina: Tej. serie = %1.1f ms", tej1*1000);

  gettimeofday(&t0, 0);
  generar_pagina_cifrada(page_fil,&page_cif);
  gettimeofday(&t1, 0);
  tej2 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Cifrar la página: Tej. serie = %1.1f ms", tej2*1000);

  gettimeofday(&t0, 0);
  preparar_subida(page_cif);
  gettimeofday(&t1, 0);
  tej3 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Preparar la subida: Tej. serie = %1.1f ms", tej3*1000);

  tej=tej1+tej2+tej3;
  printf("\nTOTAL: Tej. serie = %1.1f ms\n\n", tej*1000);

  strcpy(name,argv[1]);
  name[strlen(name)-4]='\0';
  strcat(name,"_fil_ser.pgm");
  store_pixmap(name,page_fil);

  strcpy(name,argv[1]);
  name[strlen(name)-4]='\0';
  strcat(name,"_cif_ser.pgm");
  store_pixmap(name,page_cif);

  borrar_pagina(page_ori);
  borrar_pagina(page_fil);
  borrar_pagina(page_cif);
}


/*****************************************************************************
  preparar_subida.c: funciones a implementar
*****************************************************************************/

void preparar_subida(pagina in_page)
{

   // código para preparar la subida de la página
   // tened en cuenta la cabecera de la función upload (upload.h)
   //
   // extern void upload (unsigned char vp1, int i, int j, int h, int w);
   // pixel de la página, su posición (i,j) y dimensiones de la página (h,w)
   // se trata de aplicar la función upload a cada pixel
}
```

```
/*****************************************************************************
  filter.c: funciones a implementar
*****************************************************************************/

#define NUM_IT 100


/* Copia la página in_page en la página out_page */
/*************************************************/
void copiar_pagina(pagina *in_page, pagina *out_page)
{
 int i,j;
 for (i=0;i<in_page->h;i++)
   for (j=0;j<in_page->w;j++) out_page->im[i][j]=in_page->im[i][j];
}


/* Aplicar el filtro                                    */
/********************************************************/
double filtrar_pagina(pagina *in_page, pagina *out_page)
{
  // código para aplicar el filtro de suavizado a la página
  // se aplica a la página in_page, dejando el resultado en out_page
  // la función calcula dos valores:
  // el valor medio y mínimo de todos los píxeles de la página filtrada
  // la función devuelve el valor medio menos el mínimo
}


/* Genera la página filtrada               */
/*******************************************/
void generar_pagina_filtrada(pagina *in_page, pagina *out_page, float limite)
{
  generar_pagina(out_page,in_page->h,in_page->w,NEGRO);

  // codigo para generar la pagina filtrada a partir de la pagina original
  // condicion final de filtrado:
  //    (a) superar el número máximo de iteraciones
  //    (b) valor medio menos mínimo inferior al limite
  // asegurar que la página final queda en out_page
}
```

```
/*****************************************************************************
  encript.c: funciones a implementar
*****************************************************************************/

#include "pixmap.h"

void encriptar_pixeles (unsigned char *v1, unsigned char *v2)
{
      // POR HACER: codigo para el encriptado de 2 pixeles de la imagen
}

void generar_pagina_encriptada (pagina in_page, pagina *out_page)
{
      generar_pagina(out_page, in_page.h, in_page.w, NEGRO);

      // POR HACER: codigo para generar la página encriptada
}
```