baliabideak
material de aprendizaje

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Introducción a los sistemas de cómputo paralelo

Olatz Arbelaitz Gallego
José Ignacio Martín Aramburu
Javier Muguerza Rivero

Cuaderno del estudiante

IKD baliabideak 5 (2013)

INDICE

INTRODUCCIÓN

El proyecto a realizar se sitúa en el tercer tema de la asignatura Arquitectura de Computadores (2º curso del Grado en Ingeniería en Informática): introducción a los sistemas de cómputo paralelo. El proyecto abarcará un 40% del total de la asignatura, esto es 2,4 créditos ECTS. Esta dedicación supone un total de 24 horas presenciales y de 36 horas no presenciales. Dado que se formarán grupos de 3 estudiantes, dicha carga supone una dedicación de 180 horas por grupo.

PREGUNTA MOTRIZ: ¿HAY VIDA EN MARTE?

ESCENARIO PRELIMINAR

Dentro de las investigaciones que la NASA está realizando para ver si existe vida en Marte, se está preparando el lanzamiento de una sonda para fotografiar el planeta con mayor precisión y poder así analizar la posible existencia de vida en dicho planeta. Podéis encontrar más información acerca de esta misión en las siguientes referencias web:

http://www.nasa.gov/mission_pages/msl/index.html
http://en.wikipedia.org/wiki/Mars_Science_Laboratory

Debido a la gran cantidad de imágenes que se toman en este tipo de estudios, es importante obtener el menor tiempo de respuesta posible en el procesado de una imagen, de tal forma que puedan tratarse la mayor cantidad de imágenes por minuto. La NASA estima que el tiempo máximo para procesar una imagen de 10 MB debería ser de 300 milisegundos.

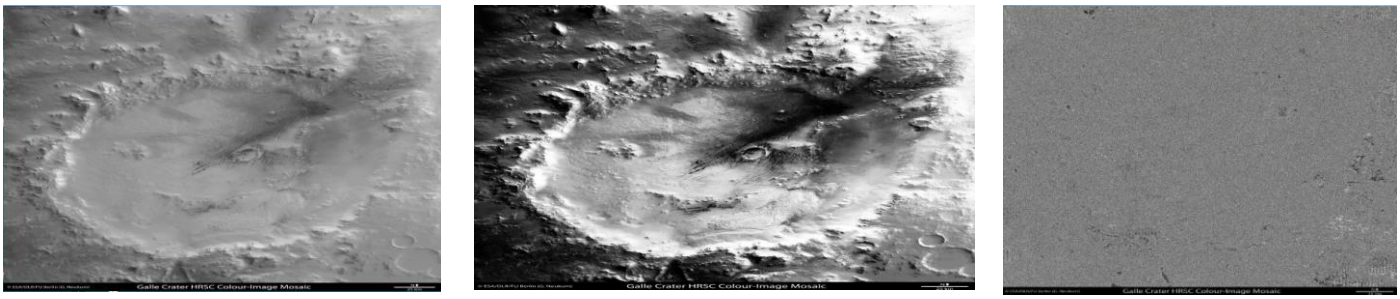
A la vista de este planteamiento, ¿qué tipo de solución consideras adecuado para hacer frente a este problema?

ESCENARIO OPERATIVO

Teniendo en cuenta el estudio previo realizado, la NASA ha decidido incluir un multiprocesador en la sonda, que será el encargado del tratamiento y envío de la imagen a su base de operaciones.

Además, nos ha pedido el diseño de un algoritmo paralelo para el tratamiento de la imagen. En primer lugar, se aplicará un filtro a la imagen original obtenida por la sonda para la mejora de su contraste, basado en el histograma de la imagen; a continuación, se encriptará la imagen para enviarla de forma secreta; y, finalmente, se preparará la imagen para su envío al centro de control.

Para cumplir el requisito de tiempo de respuesta, se quiere aprovechar la capacidad de procesado en paralelo del multiprocesador incorporado en la sonda, de forma que se ejecute en paralelo el algoritmo a desarrollar. El objetivo debe ser procesar la imagen en el menor tiempo posible para su posterior envío al centro de operaciones de la NASA, de tal forma que se traten la mayor cantidad posible de imágenes por minuto.



(a) *(b)* *(c)*

Figura 1.-Procesado de una imagen de 10 MB: (a) imagen original, (b) imagen ecualizada y (c) imagen encriptada. Imagen obtenida de ESA/DLR/FU Berlin (G. Neukum)

A modo de ejemplo, las imágenes de la Figura 1 presentan el resultado de los dos procesos a aplicar a la imagen original (Figura 1a): ecualización y encriptado (Figuras 1b y 1c respectivamente). La Tabla 1 presenta un ejemplo del rendimiento obtenido (*speed-up*) en función del número de procesadores utilizados para el proceso de las imágenes. El tamaño de la imagen original es de 10 MB.

	Tiempo de ejecución (ms)	*speed-up*
1 proceso	1542	0,9
2 procesos	773	1,8
4 procesos	397	3,4
8 procesos	208	6,5
16 procesos	116	11,7
32 procesos	72	18,8

Tabla 1.- Rendimiento obtenido en función del número de procesadores para una imagen de 10 MB.

RESULTADOS DE APRENDIZAJE

Las competencias de la asignatura relacionadas con esta parte del temario son las siguientes: "Enunciar y aplicar los conceptos de paralelismo de bajo y alto nivel" y "Programar aplicaciones sencillas de forma paralela afrontando aspectos como las dependencias de datos, la sincronización y el reparto de carga". Estas competencias se han concretado en los siguientes resultados de aprendizaje:

1. Enunciar y aplicar los conceptos de paralelismo de bajo y alto nivel
2. Analizar las dependencias de una aplicación para establecer una estrategia de paralelización de la misma
3. Identificar y aplicar los mecanismos de sincronización necesarios para el correcto funcionamiento de los programas paralelos
4. Planificar un reparto de carga equilibrado entre los procesos
5. Analizar el rendimiento de la implementación paralela de una aplicación
6. Aplicar las directivas de OpenMP para programar una solución paralela

Además, se trabajan las competencias generales C4, C8, C9 de la titulación y las competencias RI1 y RI9 de la rama común informática. El cumplimiento de estas competencias garantiza a su vez la adquisición de las competencias transversales de la titulación (CB1, CB2, CB3, CB4 y CB5). Toda la información acerca de estas competencias puede consultarse en el documento:

http://www.ehu.es/documents/340468/516505/Lista+de+competencias.pdf

LISTA DE ENTREGABLES

Entregable E1 Acta de constitución (E1.1) del grupo y documento de compromisos de los componentes del grupo, y las sucesivas actas de reuniones del grupo para el desarrollo del proyecto.

Entregable E2 Este entregable se corresponde con el póster (E2) a realizar para la discusión ABP del escenario operativo entregado al comienzo del proyecto.

Entregable E3 Bloque de entregables relacionados con el puzle a realizar al comienzo del proyecto: el informe que cada estudiante elabore del apartado que le corresponda (E3.1), su presentación (E3.2) y los ejercicios resueltos correspondientes (E3.3). En este entregable se incorporarán también aquellos materiales novedosos (no referenciados por el profesor o profesora) que el estudiante haya utilizado para el desarrollo de su parte.

Entregable E4 Informes de las evaluaciones entre pares de la presentación del puzle (E4.1) y de la presentación de la aplicación desarrollada (E4.2).

Entregable E5 Bloque de entregables relacionados con la evaluación individual de conocimientos: plantillas y ejercicios de laboratorio sobre Lenguaje C (E5.1 y E5.3, respectivamente), examen sobre el Lenguaje C (E5.2) y examen sobre paralelismo (E5.4).

Entregable E6 Desarrollo software realizado para la resolución del escenario planteado: informe técnico (E6.1) que explique la forma en la que se ha resuelto el escenario planteado y material utilizado para la presentación del desarrollo realizado (E6.2).

Entregable E7 Carpeta o portafolio (E7.1) en el que el grupo irá almacenando todo el material que vaya generando durante el desarrollo del proyecto, de acuerdo a los entregables descritos. En definitiva, todo aquel material que los miembros del grupo consideren adecuado para la mejor interpretación del desarrollo que han hecho del proyecto que se les ha planteado. Aunque se entregará la versión definitiva al finalizar el proyecto, cada grupo entregará una primera versión con el material generado hasta el momento de la realización del puzle para su corrección por parte del profesorado de cara a mejorar su contenido.

Salvo los entregables relacionados con la evaluación individual (E5.1, E5.2, E5.3 y E5.4), que tienen un carácter individual, el resto de entregables indicados en este apartado son entregables de carácter grupal.

SISTEMA DE EVALUACIÓN

El peso del proyecto en la calificación final de la asignatura es del 40%. Se evaluará, de la forma que indica la Tabla 2, utilizando los siguientes mecanismos:

- Dos exposiciones individuales: la primera correspondiente a los distintos temas trabajados en el puzle inicial y la segunda correspondiente a la defensa final de la solución planteada en el proyecto. En la evaluación de las exposiciones se considerará la evaluación entre pares. Además, la nota que obtenga el estudiante que realice la exposición será trasladada a los componentes del grupo. Este conjunto de actividades tienen un peso conjunto del 10% de la nota.
- Dos exámenes de conocimientos mínimos: un primer examen para evaluar los conocimientos del Lenguaje C y un segundo examen para evaluar los conocimientos sobre paralelismo. En este segundo caso, el estudiante deberá obtener al menos un 30% de la valoración del examen. Estos dos exámenes tienen un peso conjunto del 15% de la nota.
- El desarrollo software realizado, junto con el informe técnico que explica la solución adoptada. Esta actividad tiene un peso del 13% de la nota.
- La carpeta o portafolio final del proyecto, cuya valoración será del 2% de la nota.

El estudiante deberá superar además las actividades de seguimiento tipo FILTRO: entrega de las plantillas de laboratorio y de los ejercicios sobre Lenguaje C, así como la entrega de los ejercicios a realizar en el puzle. Una actividad de tipo FILTRO se

calificará como APTA o NO APTA, pero no tendrá carácter sumatorio sobre la calificación final.

En caso de no superar alguna de las actividades de tipo FILTRO o no haber alcanzado como mínimo el 30% exigido en el examen de paralelismo, el estudiante no podrá continuar en esta metodología de aprendizaje.

En el sistema de evaluación se contempla un sistema de "extra bonus", esto es, independientemente de la actividad que se evalúe según el sistema de evaluación propuesto, se premiarán, con puntos adicionales sobre la calificación final, las actividades que de alguna forma sean sobresalientes.

	Evaluador				**Nota**
	Profesor		**Estudiantes**		
Individual	Plantillas laboratorio C (E5.1) Examen conocimientos C (E5.2) Ejercicios seguimiento C (E5.3) Examen paralelismo (E5.4)	FILTRO 5% FILTRO 10%			15%
Grupo	Presentación puzle (E3.2) Ejercicios puzle (E3.3) Presentación de la solución (E6.2) Informe técnico/desarrollo (E6.1) Portafolio final (E7.1)	2% FILTRO 3% 13% 2%	Presentación puzle (E4.1) Presentación solución (E4.2)	2% 3%	25%
Nota		35%		5%	40%

Tabla 2.- Sistema de evaluación del proyecto.

PLANIFICACIÓN DEL TRABAJO DEL ALUMNADO

Clase	Sesión	Actividades presenciales	Actividades no presenciales posteriores a la sesión presencial	Entregables	Dedicación	Porcentaje de evaluación (40 %)
8/10 (laboratorio)	0.1	Prácticas de laboratorio: lenguaje C	Estudio programación en C	E5.1:plantillas laboratorio	4,5 horas [NP, 4 horas]	E5.1: APTA / NO APTA
15/10 (aula)	0.2	Examen de conocimientos mínimos	Ejercicios seguimiento de C	E5.2: examen	30 minutos [NP, 2 horas]	5% (E5.2)
12/11 (aula)	1	Pregunta motriz + escenario preliminar Escenario operativo Discusión ABP: póster + debate Exposición conceptos básicos	Asimilación del planteamiento del proyecto y repaso de conceptos básicos	E1.1: acta de constitución del grupo E2: póster ABP (din-A4) E5.3: ejercicios C	20 minutos 25 minutos 25 minutos 20 minutos [NP, 1 hora]	E5.3: APTA / NO APTA
13/11 (laboratorio)	2	Prácticas de laboratorio: programación en OpenMP	Estudio de la programación en OpenMP		90 minutos [NP, 1 hora]	
16/11 (laboratorio)	3	Puzle: delimitación de tareas Puzle: estudio individual de la tarea	Estudio del problema asignado a través del puzle		30 minutos 60 minutos [NP, 4 horas]	
26/11 (aula)	4	Puzle: reunión de expertos	Revisión del puzle. Elaboración del informe escrito y de su presentación		90 minutos [NP, 2 horas]	
27/11 (aula)	5	Puzle: reunión del grupo Presentación de la aplicación	Elaboración del informe escrito y de su presentación		60 minutos 30 minutos [NP, 3 horas]	
30/11 (aula)	6	Puzle: presentación Puzle: debate aclaratorio	Implementación de la aplicación, documentación y adquisición de conocimientos	E4.1:informe de evaluación entre pares	60 minutos 30 minutos [NP, 1 hora]	2% (E4.1)
3/12 (aula)	7	Puzle: resolución de ejercicios	Implementación de la aplicación, documentación y adquisición de conocimientos	E3.1/E3.2/E3.3: informe definitivo del puzle E7.1: portafolio (revisión)	90 minutos [NP, 2 horas]	2% (E3.2) E3.3: APTA / NO APTA
4/12 (laboratorio)	8	Trabajo: desarrollo, documentación, adquisición de conocimientos	Implementación de la aplicación, documentación y adquisición de conocimientos		90 minutos [NP, 4 horas]	

Clase	Sesión	Actividades presenciales	Actividades no presenciales posteriores a la sesión presencial	Entregables	Dedicación	Porcentaje de evaluación (40 %)
10/12 (laboratorio)	9	Trabajo: desarrollo de la aplicación, documentación, adquisición de conocimientos	Implementación de la aplicación, documentación y adquisición de conocimientos		90 minutos [NP, 2 horas]	
11/12 (aula)	10	Examen de conocimientos mínimos	Implementación de la aplicación y documentación	E5.4: examen	90 minutos [NP, 2 horas]	10% (E5.4) (mínimo 30%)
14/12 (laboratorio)	11	Trabajo: desarrollo de la aplicación, documentación	Implementación de la aplicación, documentación. Preparación de la presentación.		90 minutos [NP, 5 horas]	
21/12	12		Preparación de la presentación	E6.1: informe técnico y desarrollo software	[NP, 3 horas]	13% (E6.1)
9/enero/2013	13	Defensa de la implementación realizada para resolver el escenario		E6.2: material presentación E4.2: informe de evaluación entre pares E7.1: portafolio definitivo	2,5 horas	3% (E6.2) 3% (E4.2) 2% (E7.1)

RECURSOS

Este apartado incorpora todos los recursos entregados al alumnado para la realización del proyecto: material de laboratorio, enunciados del puzle, enunciados de los ejercicios a resolver y el enunciado de la aplicación a desarrollar.

1) Material para el laboratorio inicial en OpenMP

> Máquinas y directorios

Vamos a trabajar con el siguiente multiprocesador SMP:

PowerEdge R815 (DELL)

4 procesadores AMD Opteron 6168 de 12 núcleos a 1.9 GHz

64 GB de RAM (DDR3-1333 MHz)

Conexión Gigabit Ethernet

La dirección de la máquina es `u010415.gi.ehu.es` Se puede acceder desde cualquier punto de la facultad y desde fuera de la red de la universidad (por ejemplo, desde casa) a través de VPN. En el laboratorio realizaremos la conexión remota desde una sesión local linux utilizando el comando ssh en un terminal. En concreto, realizaremos la conexión a `u010415.gi.ehu.es` utilizando el comando:

```
> ssh cuenta@u010415.gi.ehu.es
```

Los ejemplos que utilizaremos en el laboratorio se encuentran en el directorio `templates/ejemplos`. Podéis copiarlos a una nueva carpeta ejemplos en vuestro directorio utilizando los comandos:

```
> mkdir ejemplos
> cp templates/ejemplos/*.c ejemplos
```

A partir de ahora podéis trabajar en la carpeta ejemplos utilizando el comando cd para moveros entre directorios:

```
> cd ejemplos
```

> Compilación y ejecución de programas

Para compilar los programas, vamos a utilizar el compilador gcc de GNU. Para compilar un programa de nombre prog.c, que incluya directivas y funciones de OpenMP, basta con ejecutar:

```
> gcc –o prog prog.c                (versión serie)
> gcc –fopenmp –o prog prog.c       (versión paralelo)
```

Si no hay errores, se habrá creado el correspondiente ejecutable. Recuerda que el número de hilos a utilizar se puede controlar, entre otras maneras, mediante una variable de entorno:

```
> export OMP_NUM_THREADS=xx xx = número de threads
```

```
/******************************************************************************
  hola.c (OPENMP)
  Generacion de varios threads y modificacion del numero de threads mediante:
     export OMP_NUM_THREADS=4
******************************************************************************/

#include <stdio.h>
#include <omp.h>

#define N  24
int A[N];

main  ()
{
  int i, tid, nthr;
  for (i=0; i<N; i++) A[i]=0;

  #pragma omp parallel
  {
    nthr = omp_get_num_threads();
    tid = omp_get_thread_num();
    printf ("Thread %d de %d en marcha \n", tid, nthr);
    A[tid] = tid + 100;
    printf ("El thread %d ha terminado \n", tid);
  }
  for (i=1; i<=N; i++)
  {
   printf (" A(%2d)=%3d ", i-1, A[i-1]);
   if (!(i%8)) printf("\n");
  }
}
```

```
/*********************************************************************
  tgrano.c (SERIE)

  Ejemplo para ver el efecto del tamaino de las tareas
  (grano) en el tiempo de ejecucion
*********************************************************************/

#include <stdio.h>
#include <sys/time.h>

#define N 10000000         // tamaino maximo del vector
int A[N];

main ()
{
  struct timeval t0, t1;
  double tej;
  int  i, tam_vec;

  printf("\n Tamaino del vector a procesar --->  ");
  scanf("%d", &tam_vec);
  for (i=0; i<tam_vec; i++) A[i] = 1;

  gettimeofday(&t0, 0);
  for (i=0; i<tam_vec; i++) {
    int j;
    for (j=1; j<=1000; j++)
      A[i] = (A[i]*A[i] + 1) * (A[i]*A[i] - 1) + (j%2);
  }
  gettimeofday(&t1, 0);

  tej = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n\n Tej. (serie) = %1.3f ms\n\n", tej*1000);
}
```

```
/*****************************************************************
  speedup.c (SERIE)

 *****************************************************************/
#include <stdio.h>
#include <sys/time.h>

#define VECES 10
#define N 10000000

struct timeval      t0, t1;

int    i, veces;
int    C[N], J[N];

void TrazaTiempo(char * pTexto, struct timeval *pt0, struct timeval *pt1)
{
  double tej;
  tej = (pt1->tv_sec - pt0->tv_sec) + (pt1->tv_usec - pt0->tv_usec) / 1e6;
  printf("%s = %10.3f ms\n",pTexto, tej*1000);
}

void SacarVector(int *V, int L1, int L2)
{
  int i;
  for (i=L1; i<L2; i++)
  {
    if(i%5==0) printf("\n");
    printf("%d ",V[i]);
  }
  printf("\n");
}



// PROGRAMA PRINCIPAL
main ()
{

  for (veces=0;veces<VECES;veces++)
  {
    for(i=0; i<N; i++) {
       C[i] = i;
       J[i] = 6;
     }

    // Comienzo del programa
    gettimeofday(&t0, 0);
    for(i=0; i<N; i++)
    {
      C[i] = ((C[i]+1) / J[i])+1;
      J[i] = (C[i] * J[i] + 1) / C[i] * J[i];
    }
    gettimeofday(&t1, 0);
    TrazaTiempo("Time: ",&t0,&t1);

  }
  printf("C-> ");
  SacarVector (C, 0, 10);
  SacarVector (C, N-10,N);
  printf("J-> ");
  SacarVector (J, 0, 10);
  SacarVector (J, N-10, N);

}
```

2) Enunciados del puzle

> Planificación

Uno de los problemas a resolver al ejecutar una aplicación en paralelo es realizar un reparto equilibrado de la carga entre los procesadores para que terminen más o menos al mismo tiempo. A este aspecto se le denomina planificación o reparto (*scheduling*).

Mediante esta actividad deberás de comprender en qué consiste el problema y las diferentes alternativas para afrontarlo. Para ello, te indicamos unas referencias para poder consultar y preparar una breve exposición del problema y sus soluciones a tus compañeros de grupo.

De cara a asentar los conceptos utilizados, deberás entregar las soluciones de los ejercicios 7 y 8 de la lista de ejercicios entregada en clase. Así mismo, deberás implementar la versión paralela del programa `tarea.c`, analizar el rendimiento que se obtiene con las diferentes alternativas de planificación que proporciona OpenMP para diferente número de threads y seleccionar la mejor configuración. En un breve documento (1 o 2 caras máximo), representa mediante una tabla las pruebas realizadas, los tiempos obtenidos, el *speed-up* y la eficiencia conseguidos frente a la versión serie, razonando los resultados obtenidos.

Puedes encontrar el código del programa `tarea.c` en el directorio `templates/puzzle` de la máquina `u010415.gi.ehu.es`

Referencias generales

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartado 4 del Capítulo 2.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.

Referencias sobre planificación

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartado 5 del Capítulo 8.
- Almeida F., Giménez D., Mantas J.M., Vidal A. M.: *Introducción a la programación paralela*. Paraninfo, 2008. Apartado 3 del Capítulo 5.

```
/******************************************************************
    tarea.c (OPENMP)
      Ejemplo para ver el efecto del reparto en el tiempo de ejecución
******************************************************************/
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>

#define N 4000
#define WEIGHT 100000

int A[N],B[N],C[N];

double calculo (int veces)
{
  usleep(veces);
  return(1);
}

main ()
{
  int i, nth=-1, tid=-1;
  double total;

  struct timeval t0, t1;
  double tej;

  for (i=0; i<N; i++)
  {
   A[i] = 1;
   if ((i%8==1)&&(i<122)) A[i]=WEIGHT;
  }

  gettimeofday(&t0, 0);
  for (i=0; i<N; i++)
  {
   B[i] = calculo(A[i]);
   C[i] = tid;
  }
  gettimeofday(&t1, 0);

  tej = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;

  total = 0.0;
  for (i=0; i<N; i++) total+=B[i];

  printf("\n");
  for (i=0; i<N; i++)
   if ((i%8==1)&&(i<122)) printf ("%d= %d ",i,C[i]);
  printf("\n");

  printf("\nTej.  (%d  hilos)  =  %1.3f  ms\n\nTotal=  %.2f\n\n",  nth,
tej*1000, total);
}
```

> Sincronización

A la hora de programar una aplicación en paralelo hay que tener en cuenta la sincronización entre procesos como consecuencia del uso de variables compartidas entre los diferentes procesos.

Mediante esta actividad deberás de comprender en qué consiste el problema y las diferentes alternativas para afrontarlo. Para ello, te indicamos unas referencias para poder consultar y preparar una breve exposición del problema y sus soluciones a tus compañeros de grupo.

De cara a asentar los conceptos utilizados, deberás entregar la solución del ejercicio 9 de la lista de ejercicios entregada en clase. Así mismo, deberás implementar la versión paralela del programa `vector.c`, analizando el uso de las variables del bucle a paralelizar para determinar su correcta definición. En un breve documento (1 o 2 caras máximo), explica la solución adoptada y los resultados (*speed-up* y eficiencia) obtenidos respecto a la versión serie en función del número de threads utilizados.

Puedes encontrar el código del programa `vector.c` en el directorio `templates/puzzle` de la máquina `u010415.gi.ehu.es`

Referencias generales

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartado 4 del Capítulo 2.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.

Referencias sobre sincronización

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartados 1, 2, 3 y 4 del Capítulo 4.
- Ortega J., Anguita M., Prieto A.: *Arquitectura de Computadores*. Thomson, 2005. Apartado 3 del Capítulo 10.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001. Capítulo 5.

```
/******************************************************************************
    vector.c (SERIE)
       Calculo sobre un vector
******************************************************************************/

#include <stdio.h>
#include <sys/time.h>
#include <math.h>

#define N 10000000
#define MAX 3

float A[N];

main ()
{
  int      i;
  float media, suma;

  struct timeval  t0, t1;
  double tej;

  for(i=0; i<N; i++) A[i]=i%MAX;
  printf ("\n");

  gettimeofday(&t0, 0);

  suma = 0.0;
  for (i=0; i<N; i++) suma = suma + A[i]*A[i];
  media = suma / N;
  for (i=0; i<N; i++) A[i] = sqrt(A[i]+media)+pow(media,4);

  gettimeofday(&t1, 0);

  tej = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n\nTej. (serie) = %1.3f ms\n\n", tej*1000);

  for (i=0; i<10; i++)  printf ("A[%d]= %1.2f ", i, A[i]);
  printf ("\n\nLa media de A es: %1.2f\n\n", media);
}
```

> Rendimiento

A la hora de programar una aplicación en paralelo es importante hacerlo de forma eficiente para obtener el máximo rendimiento posible. El rendimiento de una aplicación dependerá, entre otros aspectos, de la parte de código que haya podido ser paralelizada frente a aquella parte que no haya podido paralelizarse. Es lo que se conoce como la Ley de Amdahl.

Mediante esta actividad deberás de comprender los aspectos relacionados con el rendimiento en las aplicaciones paralelas, entre ellos la conocida Ley de Amdahl . Para ello, te indicamos unas referencias para poder consultar y preparar una breve exposición del problema y sus soluciones a tus compañeros de grupo.

De cara a asentar los conceptos utilizados, deberás entregar las soluciones de los ejercicios 4 y 5 de la lista de ejercicios entregada en clase. Así mismo, deberás implementar la versión paralela del programa `rendi.c`, analizando las posibilidades de paralelización o no de los distintos bucles existentes en el programa, teniendo en cuenta la correcta definición de las variables que intervienen en dichos bucles. En un breve documento (1 o 2 caras máximo), explica la solución paralela realizada y los resultados (*speed-up* y eficiencia) obtenidos respecto a la versión serie en función del número de threads utilizados.

Puedes encontrar el código del programa `rendi.c` en el directorio `templates/puzzle` de la máquina `u010415.gi.ehu.es`

Referencias generales

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartado 4 del Capítulo 2.
- Chandra R., et al.: *Parallel Programming in OpenMP*. Morgan Kaufmann, 2001.

Referencias sobre rendimiento

- http://www.sc.ehu.es/acwarfra/arpar/AP/apnagusia.htm. Apartado 5 del Capítulo 2.
- Ortega J., Anguita M., Prieto A.: *Arquitectura de Computadores*. Thomson, 2005. Apartado 5.2 del Capítulo 7.

```
/*****************************************************************
  rendi.c
*****************************************************************/

#include <stdio.h>
#include <sys/time.h>

#define VECES 5
int veces;

// Declaracion de constantes y variables

#define N1 3558000
#define N2 3980000
#define N3 3000000

struct timeval  t0, t1;

int  i, j, k;
int  sum, x;

int  A[N1], B[N1], E[N1];
int  C[N2], J[N2];
int  H[N3], N[N3];



// RUTINAS AUXILIARES
void TrazaTiempo(char * pTexto, struct timeval *pt0, struct timeval *pt1)
{
  double tej;

  tej = (pt1->tv_sec - pt0->tv_sec) + (pt1->tv_usec - pt0->tv_usec) /
1e6;
  printf("%s = %10.3f ms\n",pTexto, tej*1000);
}

void SacarVector(int *V, int L1, int L2)
{
  int i;

  for (i=L1; i<L2; i++)
  {
    if(i%5==0) printf("\n");
    printf("%d ",V[i]);
  }
  printf("\n");
}
```

```
// PROGRAMA PRINCIPAL
main ()
{
for (veces=1;veces<VECES;veces++){
  //Inicializaciones
  for(i=0; i<N1; i++)
  {
    A[i] = 0;
    B[i] = N1-i+2;
    E[i] = 0;
  }
  for(i=0; i<N2; i++){
     C[i] = (i+20) / 10;
     J[i] = 6;
   }
  for(i=0; i<N3; i++){
    H[i] = 1;
    N[i] = 3;
  }

// Comienzo del programa
  gettimeofday(&t0, 0);
  for(i=1; i<N1; i++){
    x = B[i] / (B[i] + 1);
    A[i] = (x + B[i] + 1) / 10;
    E[i] = (x * x / (x * x + 1))*(x * x / (x * x + 1));
  }
  gettimeofday(&t1, 0);
  TrazaTiempo("\nT1",&t0,&t1);

  gettimeofday(&t0, 0);
  for(i=1; i<N2; i++){
    C[i] = C[i] / (J[i] + 1);
    J[i] = C[i] * J[i] / (C[i] * J[i] + 1);
  }
  gettimeofday(&t1, 0);
  TrazaTiempo("T2",&t0,&t1);

  gettimeofday(&t0, 0);
  for(i=2; i<N3; i++){
    H[i] = 35 / (7/N[i-1] + 2/H[i]);
    N[i] = N[i] / (H[i-1]+2) + 3 / N[i];
  }
  gettimeofday(&t1, 0);
  TrazaTiempo("T3",&t0,&t1);

} //fin del for VECES

// RESULTADOS: se imprimen los primeros y los ultimos 10 elementos
  printf("\nA-> ");
  SacarVector (A, 0, 10);
  SacarVector (A, N1-10, N1);
  printf("B-> ");
  SacarVector (B, 0, 10);
  SacarVector (B, N1-10, N1);
```

```
  printf("E-> ");
  SacarVector (E, 0, 10);
  SacarVector (E, N1-10, N1);

  printf("C-> ");
  SacarVector (C, 0, 10);
  SacarVector (C, N2-10,N2);
  printf("J-> ");
  SacarVector (J, 0, 10);
  SacarVector (J, N2-10, N2);

  printf("H-> ");
  SacarVector (H, 0, 10);
  SacarVector (H, N3-10, N3);
  printf("N-> ");
  SacarVector (N, 0, 10);
  SacarVector (N, N3-10, N3);
}
```

3) Ejercicios a resolver

1.- Un multiprocesador dispone de 16 procesadores. Tras analizar un determinado programa, se han obtenido los siguientes datos: una parte del mismo, el 50%, se puede ejecutar en paralelo en los 16 procesadores; otra parte, el 25%, sólo puede utilizar 8 procesadores; el resto se ejecuta en un único procesador. Calcula el *speed-up* que se conseguirá al ejecutar el programa en el multiprocesador.

2.- Una aplicación se ejecuta en una máquina paralela de 8 procesadores. Durante 50 segundos se utilizan los 8 procesadores, durante 20 segundos se utilizan 4, y durante 30 segundos sólo 1. La carga que se reparte a los procesadores está equilibrada, aunque en el caso de la ejecución en paralelo ha habido una sobrecarga (*overhead*) del orden del 10% del tiempo de ejecución. Haz una estimación del factor de aceleración (*speed-up*) que se consigue en relación con la ejecución serie, y de la eficiencia obtenida.

3.- Una determinada aplicación paralela se comporta de acuerdo a la ley de Amdahl. Una parte de la misma puede ejecutarse en paralelo sin problemas, pero otra parte hay que ejecutarla en serie. Para una máquina con P = 64 procesadores, calcula la fracción máxima de programa que hay que ejecutar en serie si la eficiencia tiene que ser al menos de 50 %; y factor de aceleración que se conseguirá si $f = 0,9$.

4.- Se ha ejecutado una aplicación en un multiprocesador con 32 procesadores. Un 10% del código se ha ejecutado en los 32 procesadores, un 70% en 16 procesadores y el resto en un único procesador. Calcula el factor de aceleración (*speed-up*) y la eficiencia que se conseguirá (sin considerar otros factores).

5.- Se ha analizado más en profundidad la aplicación del ejercicio anterior, estudiando el coste de la comunicación entre procesos. En concreto, se ha visto que el tiempo de ejecución aumenta un 20% en la parte que se ejecuta en paralelo entre los 32 procesadores, y un 10% en la parte que se ejecuta entre 16 procesadores. Calcula el factor de aceleración (*speed-up*) y la eficiencia que se consiguen.

6.- Se ejecuta una aplicación en paralelo mediante P procesadores. Cuando se ejecuta en paralelo, por cada segundo de ejecución en serie se debe realizar una comunicación cuyo coste se puede modelar como: $40 + 16 \times P^{1/2}$ milisegundos. Sin tener en cuenta otros aspectos, calcula el factor de aceleración máximo que se puede conseguir y el número de procesadores que se debe utilizar para conseguirlo. Dibuja el tiempo de ejecución, el factor de aceleración y la eficiencia en función de P.

7.- Se ejecutan en paralelo, entre 4 procesadores, 100 iteraciones de un determinado bucle. El tiempo de ejecución de cada iteración es de 1 s, salvo una de las iteraciones que tiene un tiempo de ejecución de 20 s. Calcula el *speed-up* y la eficiencia que se consiguen en los siguientes casos:

- el reparto de iteraciones es estático consecutivo (con trozos del mismo tamaño).
- el reparto de iteraciones es estático entrelazado.
- el reparto de iteraciones es dinámico, 1 a 1 (calcula el peor y el mejor caso). En este caso el coste de una operación de asignación de tareas es de 50 ms.

8.- Se quieren ejecutar 8 tareas independientes en una máquina SMP con 4 procesadores. El tiempo de ejecución (segundos) de cada una de las tareas es el siguiente: 5, 10, 6, 15, 8, 3, 2, 1. Calcula el factor de aceleración (*speed-up)* que se consigue respecto al tiempo total de ejecución serie para estos dos tipos de planificación: estática entrelazada con tamaño 2 (“static,2”) y dinámica (“dynamic,1”).

9.- Se quiere ejecutar el siguiente programa en un multiprocesador. Escribe la versión paralela del código utilizando OpenMP.

```
min = INT_MAX;
for (i=0; i<N; i++) if (A[i] < min) min = A[i];
for (i=0; i<N; i++) A[i]=A[i]-min;
```

10.- Se desea ejecutar en paralelo, en un sistema de memoria compartida de tipo SMP, el siguiente código:

```
....
printf(“Introduce el valor de X”);
scanf(“%d”,&x);

for (i=0; i<N; i++) A[i] = A[i] * X;
for (i=0; i<N; i++) B[i] = B[i] * A[i] + X;

min = INT_MAX;
for (i=0; i<N; i++) if (B[i] < min) min = B[i];
for (i=0; i<N; i++) B[i] = B[i] – min;

for (i=0; i< N; i++) printf (“%d\n”,B[i]);
....
```

Escribe el código paralelo utilizando OpenMP, indicando el ámbito de cada variable, privada o compartida. En el caso de ejecutar bucles en paralelo, la planificación debe ser dinámica.

4) Enunciado de la aplicación

Como ya se ha comentado en la presentación del proyecto, queremos implementar un programa paralelo para el procesado de las imágenes captadas por la sonda que recorre Marte. En el directorio `templates/aplicacion` tienes el material necesario para llevar a cabo esta parte del proyecto:

- El fichero `imagen.c` contiene el programa principal de la aplicación a desarrollar. Para completar la aplicación son necesarias algunas funciones de gestión de imágenes y de transmisión de las mismas que ya se entregan implementadas en los módulos `pixmap` y `transmit_aux`. Por ello, os entregamos los correspondientes módulos objetos (ficheros `.o`) y ficheros de cabecera (ficheros `.h`).
- Las funciones básicas de filtrado, encriptado y preparación de la transmisión que debéis implementar se encuentran en los módulos `ecualizacion`, `encript`, `encript_aux` y `transmit`. Os proporcionamos los correspondientes ficheros fuente (ficheros `.c`) y ficheros de cabecera (ficheros `.h`).
- Los ficheros `Marte1.pgm`, `Marte2.pgm` y `Marte3.pgm` contienen las imágenes a procesar. Son imágenes de diferentes tamaños, con lo que el tiempo de procesado también será mayor. Utilizad la imagen `Marte1.pgm` para desarrollar la aplicación y, posteriormente, analizar también los resultados obtenidos para las otras dos imágenes.

El trabajo a realizar es el siguiente:

- Completar el código de las funciones de tratamiento de imágenes en el fichero `ecualizacion.c`. Para implementar el filtro de ecualización de la imagen podéis consultar la página http://en.wikipedia.org/wiki/Histogram_equalization y el Apartado 6.2.1 del documento `ILWIS.pdf` (página Moodle de la asignatura).
- Completar el código de las funciones de encriptado de los ficheros `encript.c` y `encript_aux.c`. Para implementar el algoritmo de encriptado podéis consultar el correspondiente apartado del documento `Hill.pdf` (página Moodle de la asignatura).
- Completar el código de la función de preparación de la transmisión en el fichero `transmit.c`.
- Generar los comandos de compilación correspondientes a la versión serie y paralela de la aplicación.
- Probar el correcto funcionamiento del programa utilizando las imágenes proporcionadas. En el directorio de trabajo podéis encontrar el ejecutable `recuperar` que permite obtener la imagen ecualizada a partir de la cifrada para comprobar el correcto funcionamiento del proceso. Como parámetros, este programa recibe, además de la correspondiente imagen cifrada, los valores necesarios para el correcto descifrado de la misma.
- Implementar una versión paralela eficiente del programa utilizando OpenMP y comprobar su correcto funcionamiento. Estudiar el rendimiento obtenido en la implementación paralela, en función del número de threads, frente a la versión serie del programa.
- Realizar una memoria que explique claramente la implementación realizada y los resultados obtenidos.

NOTA. Para visualizar las imágenes, 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`.

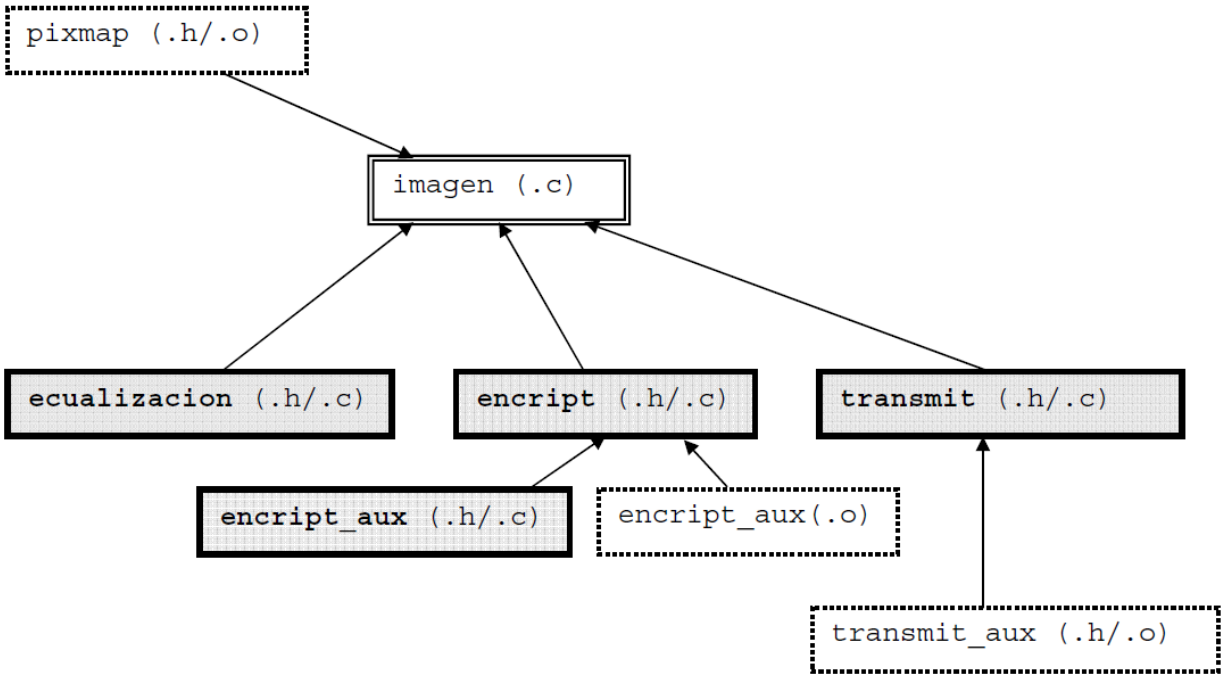
```
$ display imagen.pgm &
```

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

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

  imagen imagen_ori,imagen_ecu,imagen_cif;
  int histo[MAX_VAL], vmin_ha;
  struct timeval t0,t1;
  double tej1, tej2, tej3, tej4, tej5, tej6, tej;

  if (argc != 2) { printf ("\nUSO: programa imagen\n"); exit (0);}

  // Lectura de la imagen de entrada: solo imagenes graylevel en formato .pgm
  if (load_pixmap(argv[1], &imagen_ori) == 0) {
    printf ("\nError en lectura del fichero de entrada: %s\n\n",argv[1]);
    exit (0);
  }
  printf("\n --> Procesando la imagen. Tamano: %d -- %d\n",imagen_ori.h,imagen_ori.w);
  for (i=0;i<MAX_VAL;i++) histo[i]=0;
  gettimeofday(&t0, 0);
  histograma(imagen_ori,histo);
  gettimeofday(&t1, 0);
  tej1 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Calcular histograma: Tej. serie = %1.1f ms", tej1*1000);
  gettimeofday(&t0, 0);
  histograma_acumulado(histo);
  gettimeofday(&t1, 0);
  tej2 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Acumular histograma: Tej. serie = %1.1f ms", tej2*1000);
  gettimeofday(&t0, 0);
  vmin_histograma_acumulado(histo,&vmin_ha);
  gettimeofday(&t1, 0);
  tej3 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Calcular valor min histograma: Tej. serie = %1.1f ms", tej3*1000);
  gettimeofday(&t0, 0);
  generar_imagen_ecualizada(imagen_ori,&imagen_ecu,histo,vmin_ha);
  gettimeofday(&t1, 0);
  tej4 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Ecualizar la imagen: Tej. serie = %1.1f ms", tej4*1000);
  gettimeofday(&t0, 0);
  generar_imagen_cifrada(imagen_ecu,&imagen_cif);
  gettimeofday(&t1, 0);
  tej5 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("\n     Cifrar la imagen: Tej. serie = %1.1f ms\n", tej5*1000);
  gettimeofday(&t0, 0);
  preparar_transmision(imagen_cif);
  gettimeofday(&t1, 0);
  tej6 = (t1.tv_sec - t0.tv_sec) + (t1.tv_usec - t0.tv_usec) / 1e6;
  printf("     Preparar la transmision: Tej. serie = %1.1f ms\n\n", tej6*1000);
  tej=tej1+tej2+tej3+tej4+tej5+tej6;
  printf("\n\n TOTAL: Tej. serie = %1.1f ms\n\n", tej*1000);
  strcpy(name,argv[1]);
  name[strlen(name)-4]='\0';
  strcat(name,"_ecu.pgm");
  store_pixmap(name,imagen_ecu);
  strcpy(name,argv[1]);
  name[strlen(name)-4]='\0';
  strcat(name,"_cif.pgm");
  store_pixmap(name,imagen_cif);
  liberar_imagen(imagen_ori);
  liberar_imagen(imagen_ecu);
  liberar_imagen(imagen_cif);
}
```

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

/* Calcula el histograma de niveles de gris en la imagen                    */
/****************************************************************************/

void histograma(imagen in_imagen, int *histo)
{
       // codigo para el calculo del histograma de la imagen
}


/* Calcula el histograma acumulado                                          */
/****************************************************************************/

void histograma_acumulado(int *histo)
{
       // codigo para el calculo del histograma acumulado
}


/* Calcula el valor minimo del histograma acumulado                          */
/****************************************************************************/

void vmin_histograma_acumulado(int *histoa, int *vmin_ha)
{
       // codigo para el calculo del minimo del histograma acumulado
}


/* Genera la imagen ecualizada utilizando el histograma ecualizado          */
/****************************************************************************/

void generar_imagen_ecualizada(imagen in_imagen, imagen *out_imagen,
                               int *histoa, int vmin_ha)
{
       generar_imagen(out_imagen,in_imagen.h,in_imagen.w,NEGRO);

       // codigo para generar la imagen ecualizada
}



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

void encript (unsigned char *v1, unsigned char *v2)
{
       // codigo para el calculo del encriptado de 2 pixeles de la imagen
}



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

void generar_imagen_cifrada(imagen in_imagen, imagen *out_imagen)
{
       generar_imagen(out_imagen,in_imagen.h,in_imagen.w,NEGRO);

       // codigo para generar la imagen encriptada
}
```

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

void preparar_transmision(imagen in_imagen)
{

       //* codigo para preparar la transmision de la imagen
}
```

ESQUEMA DE LA APLICACIÓN



ANEXO

Este Anexo incorpora todos los documentos que deben ser cumplimentados por el alumnado en el desarrollo del proyecto: actas de constitución y documento de compromisos del grupo, actas de las reuniones de trabajo, plantilla de dedicación no presencial al proyecto y encuestas de satisfacción. Así mismo, se han incluido las rúbricas de evaluación utilizadas para realizar la co-evaluación de las presentaciones, y las guías para la elaboración de informes y mantenimiento del portafolio.

1) Acta de constitución del grupo

ACTA DE CONSTITUCIÓN DE GRUPO Y DOCUMENTO DE COMPROMISOS

En Donostia, a …… de ..... de 2012

Los abajo firmantes acuerdan constituir un grupo de trabajo para desarrollar el proyecto asociado al tema "Introducción a los Sistemas de Cómputo Paralelo" en la asignatura Arquitectura de Computadores.

Para ello, se comprometen a lo siguiente:

- Asistir a las reuniones de grupo que se realicen, tanto en clase, en sesiones presenciales, como fuera de ella, en actividades no presenciales.
- Realizar el trabajo asignado dentro del grupo en los plazos fijados.
- Llevar preparados a las reuniones los trabajos que se hayan comprometido a realizar.
- Asegurarse de que todos los miembros del grupo entienden todo el trabajo desarrollado.
- Hacer todo lo posible por conseguir un buen funcionamiento del grupo.
- Si surgen conflictos, comentarlos con franqueza, pero con respeto, con el objetivo de resolverlos.
- En caso de no cumplir con las obligaciones acordadas para el buen funcionamiento del grupo, asumir las posibles consecuencias: cambio de grupo, expulsión del grupo…
- ……..
- ……..

Nombre y apellidos	Firma
Nombre y apellidos	Firma
Nombre y apellidos	Firma

VºBº del profesor o de la profesora

2) Acta de sesión de trabajo del grupo

ACTA DE SESIÓN DE TRABAJO DE GRUPO

En Donostia, a …… de ..... de 2012

A las …. horas, se reúne el grupo de trabajo formado para trabajar el proyecto asociado al tema "Introducción a los Sistemas de Cómputo Paralelo" en la asignatura Arquitectura de Computadores.

COMPONENTES DEL GRUPO QUE ASISTEN A LA REUNIÓN:

1) ……..
2) ……..
3) ……..

TEMAS TRATADOS Y DECISIONES TOMADAS:

1) ……..
2) ……..
3) ……..
4) ……..
5) ……..

TEMAS PENDIENTES Y DISTRIBUCIÓN DE TAREAS PARA LA SIGUIENTE REUNIÓN:

1) ……..
2) ……..
3) ……..
4) ……..
5) ……..

Y para que quede constancia de todo lo tratado y acordado en la reunión, finalizada la misma a las …… horas, todos/as los/las asistentes firman esta acta en señal de conformidad.

……	……...	……...
Fdo.:……	Fdo.:……	Fdo.:……

3) Plantilla de dedicación no presencial al proyecto por parte del estudiante

Asignatura: **Arquitectura de Computadores** (2012/2013)

Horas de trabajo NO PRESENCIAL asociadas al Tema 3: introducción al paralelismo (metodología Aprendizaje Basado en Problemas)

Actividad	Tiempo (horas/min.)
Tareas preliminares	
estudio del lenguaje de programación C	
contextualización del proyecto y repaso de conceptos básicos	
estudio de OpenMP	
Total tareas preliminares	
Puzzle	
lectura y búsqueda de información	
resolución de los ejercicios	
resolución del programa	
puesta en común en el grupo	
preparación de la presentación	
Total puzzle	
Aplicación	
lectura y búsqueda de información	
desarrollo: diseño y programación, análisis de opciones/tiempos	
escritura de la memoria	
preparación de la presentación	
Total aplicación	
Examen Paralelismo	
Total preparación de examen	
Total tema paralelismo (suma de totales)	

4) Encuesta de satisfacción sobre la asignatura

Dpto. ATC - FI - UPV **ARQUITECTURA DE COMPUTADORES**

Para poder retocar las asignaturas, repensarlas, en una palabra, para hacer las cosas mejor, tu opinión es imprescindible. Por eso te pedimos un pequeño esfuerzo para rellenar este cuestionario. Por favor, responde a las cuestiones con total libertad, pues es tu verdadera opinión lo que nos interesa. Muchas gracias.

>> La asignatura

- ¿Qué opinas de la asignatura?

no me ha gustado, no me interesa	☐
interesante, pero mal explicada	☐
una más	☐
es interesante y merece la pena	☐
imprescindible en esta carrera	☐
(otras, indicar) →	

- Comparada con otras que estás cursando, ésta asignatura es

mucho menos interesante	☐
menos interesante	☐
similar	☐
más interesante	☐
mucho más interesante	☐

- ¿Tienes alguna sugerencia para mejorar el material de la asignatura (apuntes, ejercicios, transparencias, página web...?

- Califica la asignatura en su globalidad (calidad, interés, actualidad, utilidad, ...), **de 0 a 10**

>> El profesor

- Con intención de mejorar su trabajo, ¿cuál es la **crítica principal** que harías a tu(s) profesor(es)? ¿Cuál es el **aspecto más positivo** que destacarías de tu(s) profesor(es)?

- En conjunto, califica el trabajo de tu(s) profesor(es), **de 0 a 10**

>> Evaluación continua

- En cuanto a la distribución de la carga de trabajo de la asignatura a lo largo del cuatrimestre, ¿ha estado bien distribuido? ¿Las horas invertidas en la asignatura son acordes a los créditos ECTS de la misma (1 crédito ECTS = 10 horas presenciales + 15 horas de trabajo no presencial: estudio, elaboración de prácticas, ejercicios, etc.)?

- En cuanto a la coordinación con otras asignaturas, ¿se ha repartido la carga de manera que no se te juntaran excesivas horas en cortos periodos de tiempo? Si ha habido saturación en algún momento, ¿cuándo ha sido? ¿Qué asignaturas crees que han sido las responsables de ello?

5) Encuesta de satisfacción ERAGIN sobre el proyecto

CUESTIONARIO DE OPINIÓN SOBRE LA METODOLOGÍA ABP (Tema Paralelismo)				
Te pedimos que nos des tu opinión sobre varios aspectos de la metodología que se ha seguido en el aula. Tus respuestas serán analizadas, y nos permitirán mejorar nuestras propuestas en el futuro. Por eso, te pedimos que le dediques el tiempo necesario, y contestes con sinceridad. Muchas gracias.				
Teniendo en cuenta todos los aspectos de la metodología que hemos trabajado, tu **valoración global** del planteamiento y desarrollo de la experiencia es: ☐ nada satisfactoria ☐ poco satisfactoria ☐ bastante satisfactoria ☐ muy satisfactoria				
Justifica tu valoración:				
Valora el grado en que consideras que la metodología seguida **te ha ayudado a aprender**, en **comparación** con planteamientos metodológicos más tradicionales: ☐ me ha ayudado menos ☐ me ha ayudado igual ☐ me ha ayudado más ☐ me ha ayudado mucho más				
Valora el grado en que consideras que el **uso de esta metodología te ha ayudado** a: ("1" muy poco, "2" poco, "3" bastante, "4" mucho)				
Comprender contenidos teóricos	1	2	3	4
Establecer relaciones entre teoría y práctica	1	2	3	4
Relacionar los contenidos de la asignatura y obtener una visión integrada	1	2	3	4
Aumentar el interés y la motivación por la asignatura	1	2	3	4
Analizar situaciones de la práctica profesional	1	2	3	4
Indagar por tu cuenta en torno al trabajo planteado	1	2	3	4
Tomar decisiones en torno a una situación real	1	2	3	4
Resolver problemas o ofrecer soluciones a situaciones reales	1	2	3	4
Desarrollar tus habilidades de comunicación (oral o escrita)	1	2	3	4
Desarrollar tu autonomía para aprender	1	2	3	4
Tomar una actitud participativa respecto a tu aprendizaje	1	2	3	4
Mejorar tus capacidades de trabajo en grupo	1	2	3	4
Desarrollar competencias necesarias en la práctica profesional	1	2	3	4
El sistema de evaluación seguido ha sido adecuado a la metodología	1	2	3	4
La orientación proporcionada por el/la profesor/a durante el proceso, ¿ha satisfecho tus necesidades? ☐ Poco ☐ Suficiente ☐ Bastante ☐ Mucho				
¿Cambiarías algo? ¿Se te ocurre alguna propuesta de mejora?				
Si el próximo curso/módulo/cuatrimestre pudieras elegir, ¿optarías por esta metodología? ☐ Sí ☐ No				

6) Guía para la estructura y contenido del portafolio

Arquitectura de Computadores

Algunas **sugerencias** generales sobre la estructura y el contenido del **portafolio**

Como idea general, el portafolio **no debería ser una yuxtaposición de material** generado durante el desarrollo del proyecto, sino que, debería contener la información de manera estructurada, con un formato determinado (con cierta homogeneidad), y contextualizar la información contenida (por ejemplo, no incluir una tabla sin su correspondiente párrafo explicativo o sin un análisis de la misma, aunque sea breve).

Por otro lado, es importante que no os toméis el portafolio como algo estático e invariable, sino como **algo dinámico y mejorable**. Por ejemplo, incluir una solución corregida de los ejercicios del puzzle tras la puesta en común realizada en clase, podría ser una mejora a incluir en ese momento. De esta manera, la información que quede al final del proyecto en el portafolio será completa y correcta. Por ello, es conveniente que en cada entrega del portafolio indiquéis en una hoja aparte las mejoras, añadidos, etc. que hayáis realizado desde la última revisión.

Finalmente, os proponemos una **estructura mínima** que el portafolio debería tener, aunque queda a vuestro criterio añadir todo aquello que consideréis adecuado. Tomadlo como una recomendación. Algunos de los contenidos (no todos) tendrán su correspondencia en la versión magnética del portafolio. Por ejemplo, los códigos fuente, transparencias utilizadas en las presentaciones, memoria de la aplicación desarrollada, etc. deberán estar también en la memoria magnética. Sin embargo, por ejemplo, las actas asociadas a las reuniones, no.

La última versión del portafolio habrá que entregarla el miércoles **9 de enero** de 2013 (día de la presentación del proyecto). Por una parte, se entregará la versión en papel, y, por otra parte, se subirá a través de Moodle la versión digital.

Estructura mínima propuesta

>> Título del proyecto e identificación del grupo y de sus componentes

>> Índice con los contenidos del portafolio

>> Enunciado del proyecto y fuentes consultadas

Con el enunciado del proyecto, fuentes de información y bibliografía utilizadas (pueden ser las referencias o la información directamente)

>> Actividades durante el proyecto:

- Póster

- Puzle

Presentación de los problemas planteados y resumen de las soluciones encontradas en la bibliografía
Ejercicios resueltos
Versión paralela de los programas planteados para cada problema: código fuente comentado y análisis de los resultados obtenidos.
Por ejemplo:

- Para el caso de *rendimiento* podríamos incluir un resumen y análisis de los tiempos obtenidos (por ejemplo, estimación del tiempo mediante Amdahl).
- Para el caso del *reparto o planificación* un análisis del tiempo de ejecución según las estrategias de reparto analizadas.
- Para el caso de la *sincronización*, un análisis de la validación de la solución adoptada (comprobación de que la solución planteada es correcta).

- Aplicación desarrollada

Memoria de la aplicación desarrollada (ver documento entregado a parte).

>> Apéndice 1 - Actas

Acta de constitución del grupo y actas de las reuniones realizadas

>> Apéndice 2 – Horas dedicadas

Resumen de horas no presenciales dedicadas al proyecto. Esta información proporcionarla desglosada por alumno y actividad (tareas preliminares, puzzle: consulta bibliográfica, resolución de los ejercicios/programas, preparación de la presentación...; aplicación: lectura y comprensión, programación, análisis de tiempos, escritura de la memoria, preparación de la presentación...). Para ello, se os entregará una plantilla a rellenar que deberéis entregar también en la versión digital del portafolio.

7) Guía para la elaboración de informes escritos

Arquitectura de Computadores

Algunas **sugerencias** generales para la **redacción** de **informes de resultados**

Una parte importante de la evaluación de esta parte de la asignatura se realiza en base al trabajo realizado para la resolución del proyecto planteado. La resolución de dicho proyecto se plasma finalmente en un **informe de resultados**, en el que se debe explicar el problema, la solución propuesta, los resultados obtenidos, la justificación y consecuencias de los mismos, etc. El informe de resultados es el producto final del trabajo realizado, por lo que conviene dedicarle un mínimo de atención, tanto al fondo como a la forma.

En cuanto a la forma, y a modo orientativo, dicho informe podría estar estructurado de la siguiente manera:

- *Índice*

- *Breve introducción*
  Presentando el problema y planteando el enfoque de la resolución que se va a explorar.

- *Desarrollo del proyecto*
  En este apartado se presentará el análisis realizado, las decisiones tomadas, las soluciones propuestas y descartadas (razonándolo), la evaluación de las mismas, etc. También se deben mostrar y explicar las partes de código más importantes (más abajo se sugiere el formato).

- *Conclusiones*
  Tanto desde el punto de vista técnico, como desde la experiencia durante el desarrollo del proyecto.

- *Bibliografía*

- *Apéndices*
  Se pueden añadir los apéndices que se consideren adecuados. Como mínimo, uno con todo el código fuente generado.

Hemos recogido en esta hoja unas cuantas **sugerencias** que os pueden ayudar a evitar ciertas deficiencias habituales en este tipo de informes; tened en cuenta que un buen trabajo (contenido) necesita también ser bien explicado (forma):

1. Las tablas con resultados deben indicar siempre las unidades correspondientes (ms, byte, MB/s...). En caso de que realicéis varios experimentos y obtengáis resultados diferentes (por ejemplo, de tiempo de ejecución), además del valor medio, también es útil indicar la desviación típica, o los valores máximo y mínimo obtenidos. Estos cuatro valores —máximo, mínimo, media y desviación típica— ofrecen información diferente sobre el resultado obtenido.

2. En muchos casos, una tabla con resultados es más que suficiente para expresar un determinado comportamiento, y no es necesario ningún tipo de representación gráfica. Utilizad los gráficos cuando éstos impliquen un cierto valor añadido.

3. La representación gráfica de los datos es útil en tanto en cuanto permite visualizar globalmente el comportamiento del sistema, o permite que nos centremos en partes concretas del mismo. Por ello, las escalas de los ejes X e Y deben ser las adecuadas para que se observe bien la función que se está representando en todo su rango de valores. En general, la escala suele ser lineal o logarítmica. Los ejes deben ir etiquetados, incluyendo las unidades de las medidas que representan.

4. En la mayoría de los casos, el eje X debe seguir una determinada escala para que la función dibujada pueda ser interpretada correctamente. Por ejemplo, no se pueden representar igualmente espaciados en el eje X tiempos de ejecución de un programa correspondientes a tamaños de un vector de 100, 200, 500, 600 y 2000 enteros.

5. Conviene añadir un pie a las principales figuras y tablas del informe, del estilo de:
"Figura/Tabla 3. Tiempo de ejecución (ms) del programa pr1 en función del tamaño de los vectores (double)".

6. En muchos casos es necesario incluir en un informe trozos de código. Si es así, incluid sólo aquello que es estrictamente necesario para interpretar lo realizado, y comentadlo adecuadamente. Para el código, utilizad un tipo de letra de paso constante y un tamaño reducido (por ejemplo, courier 8) y espaciado simple. Indexad el código para que sea fácilmente legible.

7. La explicación de los resultados obtenidos debe ser clara y concisa; no basta con indicar qué ocurre ("la curva primero sube y luego baja"), sino por qué ocurre. Si no se tiene explicación para un determinado comportamiento hay que dejar constancia de ello o aventurar una hipótesis.
En muchos casos, una determinada hipótesis lleva a hacer algún otro experimento y a verificar los resultados obtenidos con los esperados. Efectuar hipótesis y pruebas más allá de lo inicialmente previsto —ser creativo— es una práctica adecuada y bien valorada, aunque también es necesario considerar el tiempo disponible para ello.

8. Dado que el contenido de un informe técnico/científico suele ser de por sí complejo, la redacción del mismo debe ser lo más sencilla posible —frases cortas, sencillas, legibles...— sin renunciar por ello a explicar cuestiones complejas. Un texto farragoso e ininteligible hace imposible analizar su contenido.
Si es posible, dejad el texto un cierto tiempo y volved a releerlo; vosotros mismos os daréis cuenta de los posibles fallos del texto. También es útil que una persona diferente a la que ha escrito el texto lo lea, para ver si entiende lo que está escrito. Recordad que un informe de este tipo va dirigido a terceras personas, no al que lo escribe.

9. Las herramientas actuales de edición permiten generar textos con un mínimo nivel de "calidad" formal: una maquetación adecuada, gráficos de calidad, sin faltas de ortografía, etc. El "texto con fallos 0" es difícil de conseguir, pero no debemos estar muy lejos del mismo.
Las páginas deben ir numeradas, y, si el documento es extenso, puede ser útil incluir un pequeño índice. No por tener muchas hojas es mejor un documento de este tipo, aunque tampoco hay que pasarse en sentido contrario.
Utilizad el formato que más os guste; si queréis, ésta es una sugerencia de formato simple: tipo de letra times, 11 ptos; espaciado sencillo; márgenes de 2,5/3 cm; impresión a doble cara. Si hacéis referencia a "colores" (de un gráfico, por ejemplo), recordad imprimir al menos ese trozo en color.
Finalmente, unas grapas o un encuadernado sencillo es siempre preferible a unas hojas sueltas.

En resumen, el informe de resultados debería ser conciso, claro y legible, como para que pudiera ser entendido por cualquier otro compañero con unos mínimos conocimientos sobre el tema.

8) Rúbrica para las presentaciones orales

Rúbrica para la co-evaluación de presentaciones orales

Aspecto a evaluar	Excelente	Satisfactorio	Mejorable	Deficiente
Organización. Claridad, lógica, estructuración, razonamiento.	La presentación es clara, lógica y está bien estructurada. Los/las oyentes pueden seguir la línea de razonamiento.	En general es clara, lógica y está bien estructurada. Algunos aspectos pueden resultar confusos.	Algunas ideas no están claras. Saltan de unas ideas a otras, sin orden. Cuesta seguir la lógica del discurso.	No es nada clara ni lógica. No tiene estructura. Es imposible entender nada.
Estilo. Exposición de ideas, ritmo, postura, volumen, tono, pausas.	El nivel de la presentación es adecuado para la audiencia. Expone las ideas a un ritmo adecuado. No lee de un papel. Se le ve cómodo/a delante del grupo y puede ser escuchado/a por todos/as. Realiza pausas en los momentos oportunos.	El nivel de la presentación es, en general, adecuado para la audiencia. El ritmo es variable. A veces lee de un papel. Se le ve un poco incómodo/a y la audiencia tiene algunos problemas para escuchar. Realiza pocas pausas.	Algunos aspectos de la presentación son demasiado elementales o demasiado sofisticados para la audiencia. El ritmo a veces es demasiado rápido o demasiado lento. Lee bastante de un papel. Se le ve incómodo/a y la audiencia tiene bastantes problemas para escuchar. Realiza muy pocas pausas.	El nivel de la presentación es totalmente inadecuado para la audiencia. El ritmo es inexistente. Lee todo de un papel. Se le ve muy incómodo/a y la audiencia tiene que estar muy atenta para poder escuchar. No realiza pausas.
Lenguaje, vocabulario. Científico, correcto.	Emplea un vocabulario adecuado..	El vocabulario es bastante adecuado.	El vocabulario es poco adecuado.	El vocabulario es completamente inadecuado.

Recursos audiovisuales de apoyo. Homogeneidad, tamaño de letra, claridad.	Los recursos audiovisuales contribuyen a la calidad de la presentación. El tamaño de letra es muy adecuado y puede ser visto por toda la audiencia. La información está bien organizada y facilita la comprensión del tema. Recalca los aspectos principales del trabajo.	Los recursos audiovisuales contribuyen a la calidad de la presentación. El tamaño de letra es adecuado y puede ser visto por toda la audiencia. La información está bien organizada pero faltan algunos de los aspectos principales del trabajo.	Los recursos son de escasa calidad o se utilizan de forma inapropiada. El tamaño de letra es pequeño, lo que dificulta la lectura. Se ha incluido excesiva información. Se da excesiva importancia a información secundaria. Es una presentación confusa para la audiencia.	No han preparado recursos audiovisuales de apoyo a la presentación.
Contenidos. Correctos, adecuados, suficientes.	Cumple todo lo indicado. Demuestra un completo entendimiento del tema	No son correctos, o son poco adecuados, o insuficientes. Demuestra buen entendimiento del tema	No son ni correctos ni adecuados, o son insuficientes. Entiende algunas partes del tema.	Incorrectos, inadecuados e insuficientes. No parece entender el tema.
Límite de tiempo. Se ajusta al tiempo máximo asignado; capacidad de síntesis.	Distribución temporal equilibrada, adecuada a los contenidos. Se ajusta bien al tiempo disponible.	Distribución temporal equilibrada. Se ajusta bastante al tiempo disponible, aunque necesitaría un poco más, pero es capaz de sintetizar.	Distribución temporal algo descompensada y desajustada, pero es capaz de sintetizar.	Distribución temporal muy descompensada (por exceso o por defecto) con respecto al tiempo disponible. En caso de exceso, no es capaz de sintetizar.

9) Acta para la co-evaluación de las presentaciones orales

ACTA DE COEVALUACION DE UNA PRESENTACIÓN ORAL

En Donostia, a …… de ..... de 2011

Aspecto	Peso(%)	Valoración(1)	Calificación (peso*valoración)	Observaciones
Organización	25			
Estilo	15			
Lenguaje	10			
Recursos audiovisuales	15			
Contenidos	25			
Límite de tiempo	10			
Total	100	------		

(1) Valoración: Excelente [1] Satisfactorio [0.7 – 0.9] Mejorable [0.4 – 0.6] Deficiente [0 – 0.3]

Nombre y apellidos del evaluador: ……
Grupo evaluado:……

Fdo.:……..

Arbelaitz, O., Martín J.I., Muguerza J. (2013). Introducción a los sistemas de cómputo paralelo
http://cvb.ehu.es/ikd-baliabideak/ik/arbelaitz-5-2013-ik.pdf