PRÁCTICAS ARQUITECTURA DE COMPUTADORES
Curso 2008-2009. Primer cuatrimestre.
Dpto. de Electrónica y Sistemas. Facultad de Informática. A Coruña.

Parte I
PROGRAMACIÓN PARALELA MEDIANTE PASO DE MENSAJES

El objetivo de este bloque de prácticas es introducir al estudiante en el desarrollo de programas paralelos mediante librerías de paso de mensajes, poniendo de manifiesto los principales problemas que el programador debe abordar con objeto de crear un codigo paralelo a partir de un codigo secuencial. Más concretamente, se persigue intóducir al estudiante el concepto de reparto de la carga de trabajo y sus implicaciones desde el punto de vista del programador (e.g., distribuciones de datos entre los procesadores, correspondencia entre coordenadas locales y globales, sincronizacion de procesos).

Se presentan dos casos de estudio consistentes en el cálculo del número π por el método de Montecarlo y en el cálculo de una norma para cada fila de una matriz que está representada mediante un formato de almacenamiento comprimido. Las herramientas elegidas son la librería de paso de mensajes PVM (Parallel Virtual Machine) y el entorno gráfico XPVM para la monitorizacion y seguimiento de la actividad del programa paralelo.

1. Cálculo de π por el método de Montecarlo

1.1. Objetivos

Existen diversos factores que determinan la eficiencia de un programa paralelo, siendo de especial relevancia la eleccion del reparto de la carga de trabajo entre el conjunto de procesadores utilizados en el programa paralelo. Los objetivos de esta práctica son introducir al estudiante el concepto de reparto de la carga de trabajo, así como su familiarizacion con la estructura de un programa paralelo que utiliza una librería de paso de mensajes.

1.2. Metodología

Típicamente, los bucles son la parte de un programa secuencial que consumen la mayoría del tiempo de ejecucion. Así pues, el reparto de la carga de trabajo consiste en definir la forma en que se asignan las iteraciones de un bucle a un conjunto de pro-

```
for (i=1; i<=niteraciones; i++) {
    x = (double) random() / (double) RAND_MAX;
    y = (double) random() / (double) RAND_MAX;
    if ((x*x + y*y) <= 1.0) enCirculo++;
}
pi = (4.0 * enCirculo) / (double) i;
```

Figura 1: Cálculo de número π utilizando el método de Montecarlo.

cesadores. Un ejemplo clásico muy utilizado debido a su simplicidad y rendimiento es el *reparto de trabajo consecutivo*, consistente en asignar a los procesadores bloques consecutivos de iteraciones de un bucle. Tomando como ejemplo la implementacion del método de Montecarlo que se muestra en la Figura 1, el reparto consecutivo de las iteraciones del bucle entre P procesadores consistiría en ejecutar las primeras $niteraciones/P$ iteraciones en el primer procesador, las siguientes $niteraciones/P$ iteraciones en el segundo procesador, y así sucesivamente hasta ejecutar las últimas $niteraciones/P$ iteraciones en el último procesador. Por simplicidad, en esta práctica se supondrá que $niteraciones$ es divisible por P.

Se pide desarrollar dos programas paralelos del código de la Figura 1 de acuerdo con las especificaciones siguientes:

- Reparto de trabajo consecutivo implementado mediante el paradigma de programacion *master/slave* y las primitivas de comunicacion punto a punto *pvm_send* y *pvm_recv*.
- Reparto de trabajo consecutivo implementado mediante el paradigma de programacion *SPMD (Simple Program Multiple Data)* y las primitivas de comunicacion colectivas *pvm_mcast* y *pvm_reduce*.

En ambos casos utilizar la herramienta XPVM para ver el comportamiento del programa durante el proceso de desarrollo y depuracion de los programas paralelos.

Utilizando los computadores disponibles en el laboratorio 0.3, rellenar las siguientes tablas relativas a las características del hardware y a la ejecucion del programa para el cálculo del número π. El tiempo de ejecución del programa secuencial (T_{sec}) se medirá ejecutando el código secuencial en XPVM. El tiempo de ejecución del programa paralelo T_P sobre P procesadores se estimará mediante la fórmula T_{sec}/P.

Puesto de trabajo:	
Modelo de procesador:	
Velocidad del procesador:	
Memoria cache	
Memoria RAM	

`niteraciones`	`pi`	`error`	T_{sec}	T_2	T_4
10^8					
10^9					

1.3. Material complementario

- Codigo secuencial de cálculo del número π por el método de Montarlo.
- Seminario de PVM.
- Manual de usuario de XPVM.
- Ejemplo de programa paralelo que usa el paradigma de programacion SPMD.

1.4. Evaluacion

La evaluacion se realizará por el procedimiento de defensa oral. El estudiante podrá obtener una bonificacion sobre la nota final de la asignatura si entrega la práctica antes de las fechas indicadas en la siguiente tabla:

Grupo prácticas	Fecha de entrega
Martes	25 noviembre
Miércoles	26 noviembre
Jueves	27 noviembre

Los codigos desarrollados deberán cumplir las siguientes restricciones:

- El programa debe estar generalizado para cualquier número de iteraciones y para cualquier número de procesadores. Dichos parámetros serán introducidos mediante línea de comando, no mediante constantes definidas en el código fuente del programa, para evitar tener que recompilar el programa al cambiar dichos parámetros.
- El programa debe estar estructurado de forma que existan funciones específicas de manejo de las rutinas de comunicacion de PVM que encapsulen la complejidad del manejo de la librería de paso de mensajes.