Tema 2

COMPUTACIÓN Y PROGRAMACIÓN PARALELA MEDIANTE PASO DE MENSAJES

- Fundamentos de Programación mediante Paso de Mensajes. Primitivas y herramientas
- Ejemplos básicos de Algoritmos Paralelos con Paso de Mensajes
- Esquemas y costes de comunicación en redes

Tema 2

COMPUTACIÓN Y PROGRAMACIÓN PARALELA MEDIANTE PASO DE MENSAJES

WILKINSON, B.; ALLEN. M. Prentice Hall, 1999
Parallel Programming. Techniques and Applications Using Networked Workstations and Parallel Computers.

GEIST, A.; BEGUELIN, A; DONGARRA, J. et alt MIT Press,1994
PVM Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.

KUMAR, V.; GRAMA, A.; GUPTA, A.; KARYPIS, G.
The Benjamin/Cummings Publishing Company, 1994
Introduction to Parallel Computing.

Conocimientos previos (en Tema 1)

- Modelo de Prog.Paralela con Paso de Mensajes
  - procesos que colaboran para resolver un problema
  - espacios de memoria independientes
  - coordinación y comunicación por Paso de Mensajes
- Factible y apropiado
  - en multiprocesadores
  - en multicomputadores
  - en redes de estaciones de trabajo

Estructura genérica de un ALPPM

- Estructura Básica de un Algoritmo Paralelo PM
  - 1º Reparto de tareas y datos
  - 2º Procesamiento Paralelo
  - 3º Reunión y Combinación ⇒ Resultados
- Organización del Algoritmo Paralelo
  - organización maestro-esclavos
  - organización nodos iguales [con líder]
- Modelos de escritura de los programas
  - modelo MPMD
  - modelo SPMD [con líder]

Opciones de Programación

1. diseñar y emplear un lenguaje de programación paralelo 'ad hoc';
2. extender la sintaxis y las palabras reservadas de un lenguaje secuencial de alto nivel;
3. recurrir a un lenguaje secuencial de alto nivel proporcionando una librería de funciones para paso de mensajes.

Nos centramos en la OPCIÓN 3ª

Herramientas

Sistemas de programación PPM (daemon + librería)
uso en redes de estaciones de trabajo (sockets)

PVM (Parallel Virtual Machine)

primer estándar 'de facto', funciones para C y Fortran

para redes homogéneas y heterogéneas

MPI (Message Passing Interface)

intento de estandarización académica e industrial

orientado preferentemente a redes homogéneas

nuevas versiones nativas para supercomputadores

Creación de Procesos Paralelos

- modos de creación
  - estática: el programa paralelo se inicia con la creación del conjunto de las tareas paralelas (sin especificar ubicación) desde la línea de órdenes

    $> mpirun -np <num> <nombre_prog>
  - dinámica: el programa paralelo se inicia con una tarea primaria que se lanza desde la línea de órdenes y que puede lanzar otras tareas paralelas (y éstas pueden crear otras), incluso especificando su ubicación, mediante una función de la forma

    ```
    crear(nombre_prog, num, &tids)
    ```

    ☞ ≠ fork

Creación de Procesos Paralelos

- Métodos de identificación de procesos
  - identificación correlativa — de 0 a P-1
  - identificación no correlativa — tid's

  - autoidentificación — `mitid( )`
  - identificación del padre — `padre( )`

Creación de Procesos Paralelos

PVM	MPI
creación dinámica	**creación estática**
modelos SPMD o MPMD	**modelo SPMD**
identificadores no correlativos	**identificadores correlativos 0 a P-1**

Funciones para Paso de Mensajes

- Primitivas Básicas
  - `enviar (id_tarea, &datos, num)`
  - `difundir (vec_tareas, numt, &datos, num)`
  - `recibir (id_tarea, &datos, num)`
    - `recibir (-1, &datos , num )` recibir de cualquiera
  - `recibirN (id_tarea, &datos, num)`
- Esquemas de comunicación ↳
  - difusión
  - distribución
  - reunión

Ejemplo 1

Cálculo del mínimo de (1-sen(x)) para N naturales

- planteamiento secuencial 
- opciones:
  - creación dinámica sólo inicial
  - modelo SPMD con líder
  - únicamente comunicación entre padre e hijos
- estructura
  0. Creación de procesos paralelos
  1. Reparto de trabajo
  2. Cálculo paralelo
  3. Reunión de resultados parciales y resultado

Ejemplo 1

Cálculo del mínimo de (1-sen(x)) para N naturales

1. Reparto de trabajo

   cada tarea debe saber qué trabajo va a realizar

2. Cálculo paralelo

   cada tarea calcula el mínimo en su intervalo

3. Reunión de resultados parciales y resultado

   se recogen resultados parciales y se combinan
   = Resultado Global

   (secuencialmente o, si procede, en paralelo)

Cálculo del mínimo de (1-sen(x)) para N naturales

```
programa min1senx

dato de entrada N
dato de entrada P {número de procesos)

int tids[P], i, ancho, lim[2]
real min, aux
```

Cálculo del mínimo de (1-sen(x)) para N naturales

1. Reparto de trabajo *(con N múltiplo de P)*:

```
si lider
  ancho = N/P
  desde i=1 hasta P-1
    lim[0] = i*ancho
    lim[1] = lim[0] + ancho-1
    enviar( tids[i], lim, 2)
  fdesde
  lim[0] = 0
  lim[1] = ancho-1
si_no
  recibir( padre(), lim, 2)
fsi
```

Cálculo del mínimo de (1-sen(x)) para N naturales

2. Cálculo paralelo

```
{cada uno calcula el mínimo del intervalo
que se le haya asignado}

min = 1-sen(lim[0])
desde i=lim[0]+1 hasta lim[1]
  aux = 1-sen(i)
  si(aux<min)
    min = aux
  fsi
fdesde
```

Cálculo del mínimo de (1-sen(x)) para N naturales

3. Reunión de resultados parciales y resultado

```
si lider
  desde i=1 hasta P-1
    recibir( -1, &aux, 1)
    si aux<min
      min = aux
    fsi
  fdesde
  escribir( min )
  { 'min' es el resultado final}
si_no
  enviar( padre(), &min, 1)
fsi
terminar
```

Cálculo del mínimo de (1-sen(x)) para N naturales

Beneficios del uso de la computación paralela

repartir entre los P procesos el número de cálculos que haríamos secuencialmente

$\Rightarrow$ igual resultado // menor tiempo de ejecución

hacer en cada proceso el mismo número de cálculos que haríamos secuencialmente con uno solo

$\Rightarrow$ igual* tiempo de ejecución // más cálculo

Cálculo del mínimo de (1-sen(x)) para N naturales

1g. Reparto de trabajo *(si N no es múltiplo de P)*:

```
si lider
  ancho = (N+P-1)/P {N/P por exceso}
  resto = N - (P-1)*ancho {de 0 a ancho}
  desde i=1 hasta P-1
    lim[0] = resto + (i-1)*ancho
    lim[1] = lim[0] + ancho-1
    enviar( tids[i], lim, 2)
  fdesde
  lim[0] = 0
  lim[1] = resto-1
si_no
  recibir( padre(), lim, 2)
fsi
```

☠ *NO hacer*

(P-1)[N/P] para los hijos*

N-(P-1)[N/P] para el padre*

Ejemplo 2

Suma de un vector de N números con P procesos

1. Reparto de trabajo

   la tarea primaria lee el vector completo del fichero

   

   cada tarea debe recibir una porción del vector

2. Cálculo paralelo

   cada tarea calcula la suma de su porción

3. Reunión de resultados parciales y resultado

   la tarea primaria recoge los resultados y los suma
   = Resultado Global

Suma de un vector de N números con P procesos

```
programa sumvecp

datos de entrada N, P
datos de entrada reales v[N]

int tids[P], i, tam
real suma, aux, *vec
```

Suma de un vector de N números con P procesos

1. Reparto de trabajo

```
si lider
  vec = reservar(N)
  leer(vec, N)
  tam = N/P
  difundir(&(tids[1]), P-1, &tam, 1)
  desde i=1 hasta P-1
    enviar( tids[i], &vec[i*tam], tam)
  fdesde
si_no
  recibir( padre(), &tam, 1)
  vec = reservar(tam)
  recibir( padre(), &vec, tam)
fsi
```

Suma de un vector de N números con P procesos

2. Cálculo paralelo

```
{cada uno calcula la suma
de los tam elementos de su porción en vec
y la pone en su variable local suma}

suma = vec[0]
desde i=1 hasta tam-1
  suma += vec[i]
fdesde
```

Suma de un vector de N números con P procesos

3. Reunión de resultados parciales y resultado

```
si lider
  desde i=1 hasta P-1
    recibir( -1, &aux, 1)
    suma += aux
  fdesde
  escribir( suma )
  { 'suma' es el resultado final}
si_no
  enviar( padre(), &suma, 1)
fsi
terminar
```

Propuestas de ejercicios

- Solución única de una ecuación f(x)=0 con un margen de error E en la x
- Producto escalar de vectores de talla N
- Normalización de un vector de talla N

Ejemplo 3: raíz única (1)

```
programa raizunica

datos de entrada real fx(), error

int tids[P+1], i, tam
real min, max, lim[2]

{este es un ejemplo con P+1 donde líder no trabaja}
```

Ejemplo 3: raíz única (2)

```
mientras (max-min) >= 2*error
  si lider
    desde i=1 hasta P
      lim[0] = min + i*(max-min)/P
      lim[1] = lim[0] + (max-min)/P
      enviar(tids[i], lim, 2)
    fdesde
    recibir(-1, lim, 2)
    min = lim[0]
    max = lim[1]
  si_no
    recibir(lider, lim, 2)
    si( fx(lim[0])*fx(lim[1]) < 0 )
      enviar(lider, lim, 2)
    fsi
  fsi
fmientras
```

Ejemplo 3: raíz única (3)

```
si lider
   solucion = (min+max)/2
fsi
terminar
```

Propuestas de ejercicios

- Producto escalar de vectores de talla N
- Normalización de un vector de talla N