Programación Paralela

Seminario 2. Message Passing Interface

Juan Francisco Díaz Moreno

7 de abril de 2020

Ejercicio 1. Send Receive

En esta versión del programa Send Receive, el proceso 0 difunde su identificador al resto de procesos con identificadores pares y el proceso 1 hace lo mismo con los impares.

Una primera aproximación era el proceso 0 y 1 mandasen su identificador al siguiente del grupo correspondiente (rank + 2) y que el resto esperasen un mensaje del proceso (rank – 2). Además, antes de cada envío se comprobaba que el proceso al que tocaría enviar el identificador existe comprobando (rank + 2 < size). Este ejecutable es el que el Makefile llama p1_1.

La aproximación final crea un comunicador *comm* separando los procesos con identificador par e impar (color = rank % 2) y gracias a un if else que comprueba el color, el proceso 0 de cada grupo del comunicador realiza un *broadcast* al resto de procesos del grupo.

Ejercicio 2. Cálculo de PI

Esta versión del programa que calcula una aproximación de PI difiere de la original en que cada proceso calcula un bloque de iteraciones, en lugar de calcularlas cíclicamente.

Para ello, el proceso 0 calcula la primera iteración que debe calcular cada proceso y cuántas debe realizar. Posteriormente, envía al resto de procesos ambos valores y cada uno calcula su última iteración y las realiza.

Finalmente, en esta versión todos los procesos conocen el valor final de pi gracias a un MPI_Allreduce() que sustituye al MPI_Reduce de la versión original.

Ejercicio 3. Producto Escalar

Esta versión del programa que calcula el producto escalar de dos vectores reduce la carga de trabajo del proceso 0 en cuanto al vector B.

En la versión original, el proceso 0 inicializaba el todo el vector B y lo distribuía entre todos los procesos con un MPI_Scatter.

En esta versión, el proceso 0 sólo inicializa el vector A. Cada proceso inicializa su parte del vector B calculando el segmento que le corresponde de la siguiente manera:

```
istart = rank * ( tam / size );
iend = istart + ( tam / size );
int j = 0;
for( long i = istart; i < iend; i++ ) {
        VectorBLocal[j] = ( i + 1 ) * 10;
        j++;
}
```

Ejercicio 4. Comunicadores

Esta versión del ejercicio sobre comunicadores añade un dato con el que trabajar que se reparte entre los procesos con identificador impar.

Originalmente, se creaba un comunicador *comm* que separaba los procesos con identificador par de los impares y otro comunicador *comm_inverso* que añadía a todos los procesos, pero en orden inverso. Un valor *a* se distribuía entre los procesos pares de *comm* y otro valor *b* entre todos los de *comm_inverso*.

En esta versión, además, el proceso 0 del grupo de *comm* con identificadores impares, es decir, el proceso 1 de MPI_COMM_WORLD, inicializa un vector con tantos valores como procesos impares en *comm*, y lo reparte entre estos últimos. Para filtrar el grupo de procesos impares, el MPI_Scatter se realiza dentro de un if que comprueba que lo ejecuten los procesos impares: *if((rank % 2) != 0)*.