

Computación Paralela y Distribuida

2022-II

José Fiestas

16/09/22

Universidad de Ingeniería y Tecnología
jfiestas@utec.edu.pe

Práctica Grupal Dirigida 03:

Unidad 4: MPI

**PD03: 2 pts.(dos codigos
funcionables)**

**Tarea 3: 4 pts (completa, con
análisis y comentarios)**

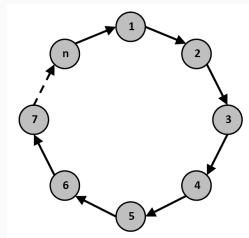
Ejercicio 1: Bloqueo en Anillo (1 pt)

En clase vimos el problema de **bloqueo mutuo** en el paso de mensajes en un anillo (*ejemplo01_bloqueada.cpp*)

Solucione el problema usando **comunicación bloqueada**, pero e.g. reordenando el envío/recibo por proceso o restringiendo el acceso a algunos procesos para que no aparezca el bloqueo mutuo.

Utilice la mayor cantidad de procesos posible y mida tiempos de ejecución significativos, para lo cual puede hacer que el mensaje sea pesado (e.g. un array). Genere una gráfica **tiempo vs. procesos**.

Para tener un resultado más preciso, realice varios experimentos y promedie resultados.

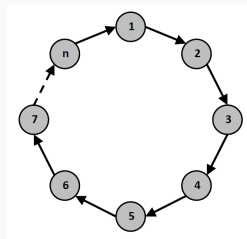


Ejercicio 2: Latencia en Anillo (1 pt)

Desarrolle un código que permita, con la solución del ejercicio 1, medir la **latencia** de un anillo de p procesos, entendiendo por **latencia** el tiempo que tarda un mensaje de tamaño 0 en circular entre todos los procesos.

Mida ahora la **latencia** en el código con directivas de comunicación no-bloqueada (*ejemplo01_nobloqueada.cpp*)

Puede aproximar el tiempo de latencia comparando el tiempo de comunicación de data de distinto tamaño, de mayor a menor ¿Cuál resultado considera representa mejor el tiempo de latencia? Para tener un resultado más preciso, promedie varios experimentos. Indique el **error** en el cálculo de latencia.








Ejercicio 2: Multiplicación matriz-vector (2 pts)

El siguiente fragmento de código secuencial calcula el **producto** de una **matriz** cuadrada **A** por un **vector** **v**, ambos de la misma dimensión **N**, almacenando el resultado en el **vector** **x**

```
int i, j;
    int A[N][N], v[N], x[N];
    leer(A,v);
    for (i=0;i<N;i++) {
        x[i]=0;
        for (j=0;j<N;j++) x[i] += A[i][j]*v[j];
    }
    escribir(x);
```

Ejercicio 3: (cont.)

- a) Programe en MPI el producto en paralelo, teniendo en cuenta que el proceso P_0 obtiene inicialmente la matriz \mathbf{A} y el vector \mathbf{v} , y realiza una distribución de \mathbf{A} por bloques de filas consecutivas sobre todos los procesos, enviando asimismo \mathbf{v} a todos. P_0 debe obtener el resultado.
- b) Obtenga el **costo computacional** y de **comunicación** del algoritmo paralelo. Utilice un tamaño adecuado para la matriz y vector (realice pruebas para determinarlo), y el número de procesos 2,4,6,...,p (hasta donde sea posible), manteniendo el **tamaño del problema constante**. Grafique T_{ej} vs p .
- c) Realice el experimento con un tamaño variable de N, y grafique speedup **S vs p**, así como **E vs p**, para los casos usados.
- d) Discuta la escalabilidad débil y fuerte con los resultados obtenidos

-  David B. Kirk and Wen-mei W. Hwu *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. isbn: 978-0-12-415992-1.
-  Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014.
-  Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011. isbn: 978-0-12-374260- 5.
-  Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003. isbn: 0071232656.
-  Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Program- ming*. 1st. Addison-Wesley Professional, 2010. isbn: 0131387685, 9780131387683.