FUNCIONES BLOQUEANTES

Ejercicio MPI

Computación Grid y Supercomputación

Curso 2019-2020

Ander José Fernández Vega

Índice

Introducción y enunciado

El objetivo de este ejercicio es realizar programa para el uso y manejo de diferentes funciones bloqueantes con MPI. Las indicaciones a seguir expuestas en el enunciado son las siguientes:

1. Trabaja con 2, 4 y 5 procesos.
2. El maestro tiene una matriz 10000x10000 con valores aleatorios entre 0 y 100.
3. El maestro distribuye la matriz a partes iguales entre todos los procesos, incluido él mismo.
4. Todos calculan el máximo de todos los elementos que reciben.
5. Todos envían el máximo que han calculado, de forma que todos los procesos conocen el máximo de la matriz (se calcula el máximo global).
6. Con el valor correspondiente al máximo global, todos los procesos actualizan sus elementos de la matriz (los que recibieron en el paso 3), de forma que pasan a valer lo que valían antes entre el valor del máximo para así normalizarlos entre 0 y 1.
7. Todos los procesos envían su parte de matriz actualizada al maestro.
8. Calcula el tiempo desde justo antes de distribuir la matriz (paso 3) hasta justo después de que la reciba el maestro (paso 7).
9. Compara los tiempos obtenidos para 5 ejecuciones con 2, 4 y 5 procesos.

Implementación

Para la implementación del código, como se requiere en el enunciado, se ha utilizado MPI. El código en cuestión se encuentra en el archivo *bloqueantes.c* del repositorio de GitHub del siguiente enlace:

https://github.com/ULE-Informatica/mpi-funciones-bloqueantes-2020-afernv10

De manera resumida, a continuación se expone la estructura del código:

A. Inicialización de variables.
B. SIncronización de procesos e inicio de contador de tiempo con **MPI_Barrier** y **MPI_Wtime**.
C. Uso de **MPI_Scatter** para repartir la matriz del maestro entre los procesos.
D. Cálculo de los máximos parciales (1 por cada proceso)
E. Uso de **MPI_Allreduce** para obtener un máximo global a todos los procesos.
F. Normalización de los valores.
G. Uso de **MPI_Gather** para enviar los datos al maestro.
H. Sincronización de los procesos y fin del contador de tiempo con **MPI_Barrier** y **MPI_Wtime**.
I. Liberación de memoria, finalización del programa y MPI.

A mayores, en el código se observa una función para la impresión de los valores en forma de matriz. Se codificó para la comprobación del funcionamiento correcto del programa. En la versión entrega la llamada a esta función se encuentra comentada, ya que no es necesaria y entorpece en la visualización por terminal.

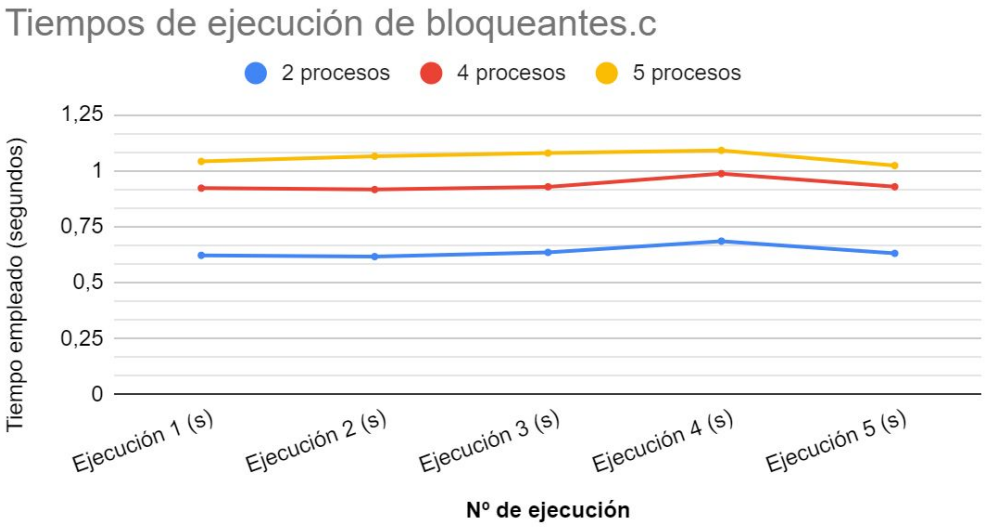
Comandos para la compilación y ejecución del código utilizados.

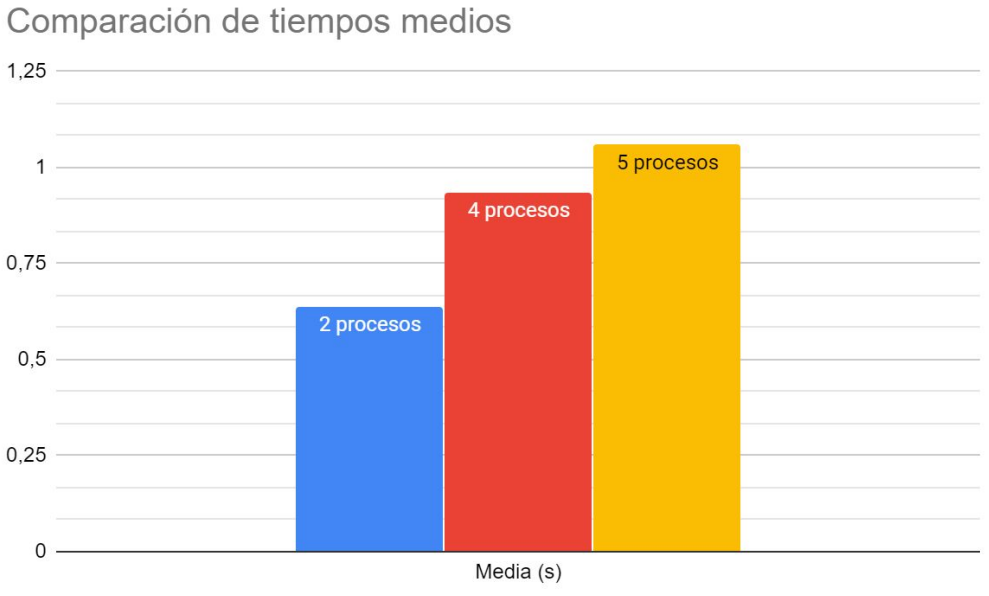
- *mpicc bloqueantes.c -o exec*
- *mpirun -np [2, 4 o 5] ./exec*

Pruebas y resultados obtenidos

A continuación se exponen las pruebas realizadas para el código con 2, 4 y 5 procesos. Se realizan 5 ejecuciones de cada uno, obteniendo el cálculo de la media del tiempo que tardan.

	2 procesos	4 procesos	5 procesos
Ejecución 1 (s)	0,621859	0,921795	1,04092
Ejecución 2 (s)	0,616202	0,915356	1,064059
Ejecución 3 (s)	0,63495	0,927752	1,078259
Ejecución 4 (s)	0,685249	0,986482	1,090247
Ejecución 5 (s)	0,63078	0,927932	1,022373
Media (s)	0,637808	0,9358634	1,0591716





Conclusiones

En las pruebas realizadas hemos podido ver que con la ejecución con mayor número de procesos el tiempo obtenido ha sido mayor en todas las ocasiones. Por lo tanto concluimos con que a mayor número de procesos con el uso de funciones bloqueantes, mayor tiempo tardará el procesador en ejecutar el programa. Esto podemos razonarlo de la siguiente manera: si las funciones bloqueantes trabajan de manera que cada vez que un proceso entra en ellas, no se sigue con el código, es decir, hasta que la función de MPI no se ejecute completamente no continuará; el incremento de procesos que usan estas funciones afecta directamente al tiempo total utilizado.