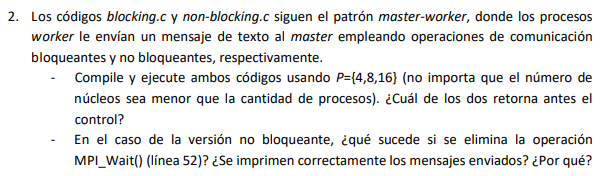
# ENTREGA 3 – SISTEMAS PARALELOS

## Ejercicio 2 – Practica 4



|  |  |  |
| --- | --- | --- |
| P = 4 | Retorno de control | |
| Blocking.c | non-blocking.c |
| Proceso 1 | 2.000097 | 0.000039 |
| Proceso 2 | 4.000079 | 2.000057 |
| Proceso 3 | 6.000039 | 4.000131 |

|  |  |  |
| --- | --- | --- |
| P = 8 | Retorno de control | |
| Blocking.c | non-blocking.c |
| Proceso 1 | 2.000099 | 0.000040 |
| Proceso 2 | 4.000245 | 2.000115 |
| Proceso 3 | 6.000305 | 4.000262 |
| Proceso 4 | 8.010901 | 6.000233 |
| Proceso 5 | 10.011212 | 8.001225 |
| Proceso 6 | 12.010685 | 10.001064 |
| Proceso 7 | 14.010673 | 12.001012 |

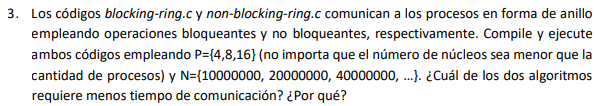
|  |  |  |
| --- | --- | --- |
| P = 16 | Retorno de control | |
| Blocking.c | non-blocking.c |
| Proceso 1 | 2.000077 | 0.000050 |
| Proceso 2 | 4.000142 | 2.000109 |
| Proceso 3 | 6.000235 | 4.000269 |
| Proceso 4 | 8.010934 | 6.000223 |
| Proceso 5 | 10.010616 | 8.001284 |
| Proceso 6 | 12.010681 | 10.001075 |
| Proceso 7 | 14.011132 | 12.001069 |
| Proceso 8 | 16.010791 | 14.001053 |
| Proceso 9 | 18.010787 | 16.001100 |
| Proceso 10 | 20.011117 | 18.001089 |
| Proceso 11 | 22.010629 | 20.001268 |
| Proceso 12 | 24.010615 | 22.001080 |
| Proceso 13 | 26.010760 | 24.001045 |
| Proceso 14 | 28.010601 | 26.000994 |
| Proceso 15 | 30.010625 | 28.001108 |

El código que usa comunicaciones no bloqueantes retorna antes el control. En este ejemplo, los workers son más lentos que el master, por lo que en la comunicación bloqueante el master se debe quedar esperando a que el worker genere el mensaje y se lo envíe, en cambio, la comunicación no bloqueante no espera a que el worker envíe el mensaje para devolver el control, por lo que el master puede continuar con su ejecución y el control se retorna mucho más rápido, aunque el tiempo final de las operaciones termine siendo similar.

Si se elimina la operación MPI\_Wait, el proceso master va a continuar ejecutando a pesar de no asegurar la recepción del mensaje del worker. Tenga o no el mensaje, va a realizar el printf que le sigue al Wait (línea 54), por lo que si la recepción todavía no se completó, es decir, no se escribió el buffer, no va a imprimirse el mensaje correcto, va a imprimir lo que haya en el buffer en ese momento.

En el código de ejemplo, al retirar la línea, se imprime "No debería estar leyendo esta frase.", que es el contenido que tenía el buffer por default, lo que indica que el pasaje de mensajes entre los procesos no se realizó.

## Ejercicio 3 – Practica 4



|  |  |  |  |
| --- | --- | --- | --- |
| P | N | Blocking-ring.c | Non-blocking-ring-c |
| P = 4 | N = 10000000 | 0.247760 | 0.221724 |
| N = 20000000 | 0.494233 | 0.440828 |
| N = 40000000 | 0.981933 | 0.882902 |
| P = 8 | N = 10000000 | 1.743007 | 0.897193 |
| N = 20000000 | 3.470623 | 1.776804 |
| N = 40000000 | 6.917952 | 3.467731 |
| P = 16 | N = 10000000 | 3.478514 | 0.815676 |
| N = 20000000 | 6.930753 | 1.687486 |
| N = 40000000 | 13.827173 | 3.439898 |

El que requiere menos tiempo de comunicación es el non-blocking-ring.

En este ejemplo de programa, los datos a enviar de un proceso a su próximo no dependen de los datos a recibir de su proceso anterior. Cuando se hace el receive y posteriormente el send, pasa lo siguiente:

Con comunicaciones bloqueantes, primero se espera a que la transferencia del receive se termine, es decir, cada proceso antes de enviar sus datos tiene que esperar a recibir los datos del proceso anterior, por lo que esa espera es una pérdida de tiempo en comunicación.

Con comunicaciones no bloqueantes, cuando se hace el receive, se prepara el buffer para recibir los datos, y se retorna el control al proceso antes de que la transferencia sea efectiva, por lo que el proceso comienza a realizar el send. Esto hace que cada proceso no tenga que esperar a recibir la información de su proceso anterior antes de comenzar a enviar la propia.

## Algoritmo secuencial

## Algoritmo paralelo empleado MPI

Tiempos de ejecución:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | P = 8 (1 nodo) | | P = 16 (2 nodos) | | P = 32 (4 nodos) | |
| Tiempo de ejecucion | Tiempo de comunicación | Tiempo de ejecucion | Tiempo de comunicación | Tiempo de ejecucion | Tiempo de comunicación |
| 512 | 0.102289 | 0.030445 | 0.110702 | 0.078513 | 0.119856 | 0.113892 |
| 1024 | 0.646052 | 0.085031 | 0.600887 | 0.332304 | 0.482665 | 0.373728 |
| 2048 | 4.789172 | 0.400957 | 3.371191 | 1.190826 | 2.525798 | 1.462144 |
| 4096 | 36.636862 | 1.433410 | 22.251770 | 4.632734 | 14.724527 | 5.950580 |

\*nota: el bs es el mismo que en las demás entregas, 64, sin embargo en algunos casos se adaptó por la cantidad de procesos: cuando P = 32 y n = 512, bs = 16, cuando P = 32 y n = 1024, bs = 32 y cuando P = 16 y n = 512, bs = 32.

SpeedUp:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **8** | **16** | **32** |
| **512** | 5.49321041 | 5.07574389 | 4.68808403 |
| **1024** | 6.73693139 | 7.24330531 | 9.01745103 |
| **2048** | 7.27379702 | 10.3332813 | 13.791865 |
| **4096** | 7.34493126 | 12.0932058 | 18.2753058 |

Eficiencia:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **8** | **16** | **32** |
| **512** | 0.6866513 | 0.31723399 | 0.14650263 |
| **1024** | 0.84211642 | 0.45270658 | 0.28179534 |
| **2048** | 0.90922463 | 0.64583008 | 0.43099578 |
| **4096** | 0.91811641 | 0.75582536 | 0.5711033 |

## Algoritmo paralelo empleando MPI+OpenMP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | P = 16 (2 nodo) | | P = 32 (4 nodos) | |
| Tiempo de ejecucion | Tiempo de comunicación | Tiempo de ejecucion | Tiempo de comunicación |
| 512 | 0.115139 | 0.074477 | 0.115868 | 0.095737 |
| 1024 | 0.577998 | 0.285436 | 0.500231 | 0.351015 |
| 2048 | 3.370430 | 1.117912 | 2.488862 | 1.387795 |
| 4096 | 21.894036 | 4.470355 | 14.136399 | 5.454058 |

\*nota: el bs es el mismo que en las demás entregas, 64, sin embargo en algunos casos se adaptó por la cantidad de procesos: cuando P = 32 y n = 512, bs = 16, cuando P = 32 y n = 1024, bs = 32 y cuando P = 16 y n = 512, bs = 32.

Speedup:

|  |  |  |
| --- | --- | --- |
| **n** | **16** | **32** |
| **512** | 4.88014487 | 4.84944074 |
| **1024** | 7.5301437 | 8.70079623 |
| **2048** | 10.3356144 | 13.9965434 |
| **4096** | 12.2908007 | 19.035628 |

Eficiencia:

|  |  |  |
| --- | --- | --- |
| **n** | **16** | **32** |
| **512** | 0.30500905 | 0.15154502 |
| **1024** | 0.47063398 | 0.27189988 |
| **2048** | 0.6459759 | 0.43739198 |
| **4096** | 0.76817504 | 0.59486338 |