**Entrega 3**

Alumnos: Franco Montanari y Ramiro Antonio

Estrategia de paralelización:

Pensamos un algoritmo en el que cada proceso realice la búsqueda del máximo, mínimo y suma de todos los elementos de 1 / (cantidad de procesos-1) de cada una de las matrices a multiplicar y una vez que ese proceso saco estos números buscados, actualiza los máximo, mínimo y suma totales al mismo tiempo que realiza la multiplicación de todas las matrices (solo de la parte que le tocó). El proceso maestro coordina las actividades de los demás y ante los pedidos de datos para ser procesados, éste se los envía bajo demanda. Cada proceso, una vez finalizados los cálculos, le envía el máximo, mínimo y suma locales de la sección que le tocó, y luego el maestro calcula el escalar pedido y se lo envía a cada proceso para que realice la multiplicación de su parte de la matriz resultado por el escalar. Por lo tanto la estrategia utilizada es una paralelización de datos de entrada y salida ya que se toman en cuenta las matrices de entrada para dividir las tareas para cada uno de los procesos y al finalizar cada uno procesa su parte de la matriz resultado y se lo envía al maestro que acumula el resultado final.

Algoritmo secuencial:

|  |  |  |
| --- | --- | --- |
| M / N | 512 | 1024 |
| 16 | 11,872 s | 101,284 s |
| 32 | 25,944 s | 209.765 s |

Algoritmo paralelo en OpenMPI:

Con 4 procesos

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 4,29 s | 33,60 s |
| 32 | 8,83 s | 69,29 s |

Speedup:

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 2,7673 | 3,014 |
| 32 | 2,9381 | 3,027 |

Eficiencia:

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 0,691825 | 0,7535 |
| 32 | 0,734525 | 0,75675 |

Con 8 procesos

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 2,21 s | 16,07 s |
| 32 | 4,50 s | 33,90 s |

Speedup:

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 5,3719 | 6,3026 |
| 32 | 5,7653 | 6,1877 |

Eficiencia:

|  |  |  |
| --- | --- | --- |
| M / N | 504 | 1008 |
| 16 | 0,6714875 | 0,787825 |
| 32 | 0,7206625 | 0,7734625 |

Los datos anteriores fueron tomadas en la sala de PC de Posgrado (máquinas provistas por la cátedra).

Conclusiones:

Con MPI se logra una mejor performance que se ve afectada por el pasaje de mensajes pero que logra una mayor paralelizacion ya que se aprovecha la mayor cantidad de procesadores disponibles en más de una máquina.

Con estos algoritmos se pudo obtener un speedup cercano a 3 para cuatro procesadores y cercano a 6 para ocho procesadores, y una eficiencia de 0,67 a 0,77 por lo que llegamos a la conclusión de que el problema provisto es altamente paralelizable con respecto a los datos.