Entrega 4

Villalba, Vizcaino

June 4, 2014

Contents

1 El problema

Dado el problema que dada una matriz A de NxN obtenga el valor máximo, el valor mínimo y valor promedio de A, luego debe armar una matriz B de la siguiente forma:

- Si el elemento a i,j $<$ promedio(A) entonces b i,j = min(A).
- Si el elemento a i,j $>$ promedio(A) entonces b i,j = max(A).
- Si el elemento a i,j = promedio(A) entonces b i,j = promedio(A).

Teniendo en cuenta que el problema requiere analizar todos los elementos de la matriz, optamos por procesar la misma elemento a elemento utilizando un unico for.

2 Secuencial

El algoritmo secuencial en primera instancia recorre toda la matriz A con el objetivo de calcular el mínimo, máximo y la suma de todos los elementos, para que luego con estos datos, y utilizando el promedio de la suma, sea posible completar la matriz B.

3 MPI

El algoritmo MPI respeta la forma de procesar las matrices con la diferencia de que cada proceso se encargará de analizar sólo una porcióon de la matriz. Para ello hemos utilizando comunicaciones colectivas, lo que nos permite la transferencia de datos entre todos los procesos que pertenecen a un grupo específico, así como un diseño modular, son ejemplos, funciones tales como MPIScatter, MPIGather y MPIAllreduce. La sentencia MPIScatter que nos permite dividir los elementos a procesar entre todos los procesos. Respetando el cálculo que se realiza en el algoritmo secuencial, cada proceso buscara un mínimo, un máximo y calculará la suma de todos los elementos que le fueron asignados por la libreria MPI. Para poder determinar los valores finales de cada uno de estos cálculos realizados se utiliza MPIAllreduce, que se encargará de comunicar y dejar los mismos resultados en todos los procesos. Luego de encontrar mínimos, máximos y promedios, se realiza la comunicación nuevamente de datos para completar la matriz B. Como cada proceso genera una parte de la solución se realiza la recolección de datos mediante MPIGather.

4 Resultados

4.1 Tablas

4.1.1 MPI 4px1M

	Tiempo Secuencial	Tiempo MPI	Speedup	Efficiency
2048	0.052820	0.032138	1.6435372	0.4108843
4096	0.238790	0.126734	1.8841826	0.47104565
8192	1.027831	0.506247	2.0302955	0.50757388
16384	4.179818	2.036680	2.0522704	0.5130676

4.1.2 MPI 4px2M

	Tiempo Secuencial	Tiempo MPI	Speedup	Efficiency
2048	0.052820	4.946691	0.010677845	2.6694613 (-3)
4096	0.238790	11.659558	0.020480193	5.1200483 (-3)
8192	1.027831	45.985576	0.022351161	5.5877903 (-3)
16384	4.179818	183.695907	0.022754007	5.6885018 (-3)

4.1.3 MPI 4px4M

	Tiempo Secuencial	Tiempo MPI	Speedup	Efficiency
2048	0.052820	1.593860	0.033139673	8.2849183 (-3)
4096	0.238790	6.328705	0.037731258	9.4328145 (-3)
8192	1.027831	25.303319	0.040620402	0.010155101
16384	4.179818	101.674330	0.041109865	0.010277466

4.2 Conclusiones

Los resultados muestran que el tiempo de ejecucióon del algoritmo y el volumen de datos procesados en una sola computadora, fué mejor que realizando el cálculo distribuido. Esto significa que intercambiar muchos datos no es aconsejable.

4.3 Maquinas utilizadas

Los cálculos se realizaron utilizando las máquinas de la sala de postgrado.