

Computación de alto rendimiento

Año: 2021

TP4 “Uso básico de OpenMP”

Salim Taleb, Nasim A.

Docente: Garelli, Luciano

Carrera: Lic. en Bioinformática

SEMINARIO DE CÁLCULO PARALELO

GUIA DE TRABAJOS PRÁCTICOS Nº 4

USO BÁSICO DE OpenMP

1) Implementar un código con OpenMP que realice la suma de todos los elementos

de una matriz en paralelo.

1. Probar con diferentes formas de acceso a las componentes de la matriz (por

columna y por fila).

2. Comparar tiempos y speedup en ambos casos.

3. Discutir los resultados obtenidos.

2) Implementar una versión paralela del Teorema de los Números Primos para

arquitecturas de memoria compartida empleando OpenMP.

1. Describir cuales variables deben ser compartidas y cuales privadas. ¿Por qué?

2. Determinar el speedup para diferentes números de threads.

3. Emplear diferentes esquemas de distribución de carga. Comparar rendimiento

y escalabilidad.

3) Implementar un código utilizando OpenMP que efectúe el producto de dos

matrices densas en paralelo.

1. Describir cuales variables deben ser compartidas y cuales privadas. ¿Por qué?

2. Determinar el speedup para diferentes números de threads.

Desarrollo

1) Se ejecutó el código en el clúster de la universidad en un nodo con 12 procesadores. Dando como resultado los siguientes datos:

|  |  |  |  |
| --- | --- | --- | --- |
| Acceso por fila | |  |  |
| N hilos | Bytes de matriz | Tiempo | Speedup |
| 1 | 16 | 0,00002 | 1 |
| 2 | 16 | 0,000097 | 0,206186 |
| 4 | 16 | 0,000092 | 0,217391 |
| 8 | 16 | 0,000356 | 0,05618 |
| 12 | 16 | 0,003146 | 0,006357 |
| 1 | 1024 | 0,000004 | 1 |
| 2 | 1024 | 0,000004 | 1 |
| 4 | 1024 | 0,000275 | 0,014545 |
| 8 | 1024 | 0,000071 | 0,056338 |
| 12 | 1024 | 0,006156 | 0,00065 |
| 1 | 1048576 | 0,001295 | 1 |
| 2 | 1048576 | 0,000747 | 1,733601 |
| 4 | 1048576 | 0,00038 | 3,407895 |
| 8 | 1048576 | 0,000284 | 4,559859 |
| 12 | 1048576 | 0,005918 | 0,218824 |
| 1 | 1073741824 | 0,831862 | 1 |
| 2 | 1073741824 | 0,427217 | 1,947165 |
| 4 | 1073741824 | 0,236425 | 3,518503 |
| 8 | 1073741824 | 0,139908 | 5,945779 |
| 12 | 1073741824 | 0,120914 | 6,879782 |
|  |  |  |  |
| Acceso por columna | |  |  |
| N hilos | Bytes de matriz | Tiempo |  |
| 1 | 16 | 0,000006 | 1 |
| 2 | 16 | 0,000056 | 0,107143 |
| 4 | 16 | 0,000204 | 0,029412 |
| 8 | 16 | 0,000084 | 0,071429 |
| 12 | 16 | 0,003803 | 0,001578 |
| 1 | 1024 | 0,000004 | 1 |
| 2 | 1024 | 0,001857 | 0,002154 |
| 4 | 1024 | 0,000144 | 0,027778 |
| 8 | 1024 | 0,000086 | 0,046512 |
| 12 | 1024 | 0,003425 | 0,001168 |
| 1 | 1048576 | 0,001536 | 1 |
| 2 | 1048576 | 0,000797 | 1,927227 |
| 4 | 1048576 | 0,000414 | 3,710145 |
| 8 | 1048576 | 0,000296 | 5,189189 |
| 12 | 1048576 | 0,003505 | 0,438231 |
| 1 | 1073741824 | 0,838737 | 1 |
| 2 | 1073741824 | 0,462329 | 1,814156 |
| 4 | 1073741824 | 0,255122 | 3,287592 |
| 8 | 1073741824 | 0,148792 | 5,636976 |
| 12 | 1073741824 | 0,112081 | 7,483311 |

1. Comparando los tiempos para 1,4 y 12 hilos se obtiene la siguiente gráfica:

2. Comparando los speedup del último tamaño de matriz (1gB):

3. En principio, se esperaría que el acceso por fila sea más rápido que el acceso por columna, sin embargo, no queda del todo claro a través de los datos obtenidos. Para el caso del análisis de los speedups en la matriz mas grande se puede observar cierta mejoría por parte del acceso por fila, excepto en el último dato.

2) El código se ejecutó en un nodo de 12 procesadores del clúster de la universidad, obteniendo los siguientes datos:

|  |  |  |
| --- | --- | --- |
| static |  |  |
| N hilos | Chunk | Tiempo |
| 1 | 1000 | 6,444677 |
| 1 | 10000 | 6,212629 |
| 1 | 100000 | 6,215325 |
| 1 | 1000000 | 6,215406 |
| 2 | 1000 | 3,911614 |
| 2 | 10000 | 3,869734 |
| 2 | 100000 | 3,869521 |
| 2 | 1000000 | 3,870542 |
| 4 | 1000 | 2,112175 |
| 4 | 10000 | 2,069712 |
| 4 | 100000 | 2,069701 |
| 4 | 1000000 | 2,06969 |
| 8 | 1000 | 1,106222 |
| 8 | 10000 | 1,130889 |
| 8 | 100000 | 1,118404 |
| 8 | 1000000 | 1,119204 |
| 12 | 1000 | 0,778317 |
| 12 | 10000 | 0,770142 |
| 12 | 100000 | 0,837083 |
| 12 | 1000000 | 0,846729 |
|  |  |  |
| dinamic |  |  |
| N hilos | Chunk | Tiempo |
| 1 | 1000 | 6,254533 |
| 1 | 10000 | 6,210692 |
| 1 | 100000 | 6,21096 |
| 1 | 1000000 | 6,210841 |
| 2 | 1000 | 3,127266 |
| 2 | 10000 | 3,105522 |
| 2 | 100000 | 3,124295 |
| 2 | 1000000 | 3,327899 |
| 4 | 1000 | 1,585714 |
| 4 | 10000 | 1,55511 |
| 4 | 100000 | 1,580116 |
| 4 | 1000000 | 1,9408 |
| 8 | 1000 | 0,813747 |
| 8 | 10000 | 0,779892 |
| 8 | 100000 | 0,815476 |
| 8 | 1000000 | 1,302456 |
| 12 | 1000 | 0,566797 |
| 12 | 10000 | 0,520776 |
| 12 | 100000 | 0,561385 |
| 12 | 1000000 | 0,861164 |
|  |  |  |
| guided |  |  |
| N hilos | Chunk | Tiempo |
| 1 | 1000 | 6,208998 |
| 1 | 10000 | 6,229528 |
| 1 | 100000 | 6,229782 |
| 1 | 1000000 | 6,130581 |
| 2 | 1000 | 3,128575 |
| 2 | 10000 | 3,110888 |
| 2 | 100000 | 3,120122 |
| 2 | 1000000 | 3,199117 |
| 4 | 1000 | 1,589657 |
| 4 | 10000 | 1,558324 |
| 4 | 100000 | 1,580273 |
| 4 | 1000000 | 1,789642 |
| 8 | 1000 | 0,815846 |
| 8 | 10000 | 0,780499 |
| 8 | 100000 | 0,824928 |
| 8 | 1000000 | 1,173568 |
| 12 | 1000 | 0,562774 |
| 12 | 10000 | 0,522198 |
| 12 | 100000 | 0,528641 |
| 12 | 1000000 | 0,856153 |

1. Para este caso no es necesario que ninguna variable sea privada, n es una variable privada por defecto ya que esta usada en la iteración, también se debe tener en cuenta las condiciones de carrera que pueden generarse con la variable primes ya que está siendo actualizada constantemente.

2. Speedup para los distintos números de threads en cada distribución:

Para este caso, no se observa una diferencia al usar una cantidad de 4 o menos hilos, pero si se puede evidenciar que a medida que la cantidad de hilos aumenta se hace mas notable que a menor número de chunks, mayor speedup, lo cuál tiene sentido ya que al haber una mayor cantidad de hilos y menor tamaño de chunks, habrá una mayor cantidad de particiones que podrán ser distribuidas entre los hilos y permite un mayor aprovechamiento de los mismos.

Lo anterior mencionado se evidencia aún mas si se utiliza una distribución de carga dinámica, cabe recalcar que la diferencia entre un chunk de 1000 y 100000 es casi ninguna.

Nuevamente sucede algo similar a los anteriores resultados, también aumenta el speedup del chunk de 100000 y 12 hilos con respecto al dinámico.

3. Para los diferentes esquemas de carga se realizó una comparación utilizando chunk de 10000 que resulta ser de los más óptimos.

Se observa una gran mejoría en el speedup a mayor cantidad de hilos en las distribuciones dinámica y guiada con respecto a la estática, en cuanto a la distribución dinámica y guiada no se observan diferencias significativas.

3) Se ejecutó el código en un nodo de 12 procesadores del clúster de la universidad y se obtuvieron los siguientes datos:

|  |  |  |
| --- | --- | --- |
| Tam de A | Hilos | Tiempo |
| 16 | 1 | 0,00002 |
| 1024 | 1 | 0,000057 |
| 10000 | 1 | 0,001709 |
| 102400 | 1 | 0,047687 |
| 1048576 | 1 | 1,143652 |
| 16 | 2 | 0,000115 |
| 1024 | 2 | 0,000064 |
| 10000 | 2 | 0,001043 |
| 102400 | 2 | 0,029529 |
| 1048576 | 2 | 0,522216 |
| 16 | 4 | 0,000154 |
| 1024 | 4 | 0,000055 |
| 10000 | 4 | 0,00055 |
| 102400 | 4 | 0,015561 |
| 1048576 | 4 | 0,282352 |
| 16 | 8 | 0,000138 |
| 1024 | 8 | 0,000082 |
| 10000 | 8 | 0,000422 |
| 102400 | 8 | 0,008139 |
| 1048576 | 8 | 0,195987 |
| 16 | 12 | 0,000915 |
| 1024 | 12 | 0,000032 |
| 10000 | 12 | 0,000389 |
| 102400 | 12 | 0,005906 |
| 1048576 | 12 | 0,122002 |

1. Para este caso, las variables privadas deben ser las que se encargan de recorrer los bucles e iterar, i, j y k que es privada por estar usada en el primer bucle. Por otro lado, las matrices A y B no necesitan ser privadas ya que no se modifican sus valores y son solamente leídos, y tampoco para la matriz C ya que cada hilo se encarga del calculo se cada elemento por separado.

2. Analizando para número de threads, se obtienen las siguientes gráficas:

Se puede observar que a medida que el problema crece, también crece el speedup para ejecuciones con una gran cantidad de hilos, sin embargo, en problemas pequeños es menos eficiente utilizar una gran cantidad de hilos para resolverlos.

Códigos

Ejercicio1:



Ejercicio2:



Ejercicio3:

