

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Guadalajara



Ingeniería en Robótica y Sistemas Digitales

Clave: TE2004B.501

Parallel Programming

Matrix multiplication in open MP

Alumno:

Jorge Carrillo Castro - A01634630

Profesor:

Manuel Rodriguez Bahena

Fecha y lugar entrega:

5 de Noviembre de 2022

Zapopan Jalisco

Link github: <https://github.com/jorgais1234/parallel-programming-ITESM.git>

Para esta actividad tuvimos que hacer un programa que fuera capaz de multiplicar dos matrices y dar el resultado en otra. El programa también debía ser capaz de detectar si el tamaño de las matrices no permitía la multiplicación, es decir el número de columnas de la matriz A debe ser igual al número de filas de la matriz B. Una vez con este script, implementamos la programación en paralelo para ver cómo cambiaban los tiempos dependiendo de la cantidad de cores que utilizabamos. La parte en donde implementamos la programación paralela fue en los ciclos for que ejecutan la multiplicación.

El primer código que desarrolló necesitaba los siguientes inputs: Tamaño de la matriz A, tamaño de la matriz B y los valores de cada uno de los elementos de las matrices. Hice varias pruebas para poder demostrar el cambio en los tiempos de ejecución. Is

Prueba con 1 core

```
jorge@DESKTOP-41IP801: ~  
jorge@DESKTOP-41IP801:~$ time ./matrices_m 1  
  
Ingresar el numero de filas y columnas de la primera matriz:  
2  
2  
  
Ingresar el numero de filas y columnas de la segunda matriz:  
2  
2  
  
Introduzca los elementos de la primera matriz:  
2  
2  
2  
2  
2  
  
Introduzca los elementos de la segunda matriz:  
2  
2  
2  
2  
2  
  
OMP DEFINED, THREADCT = 1  
  
El resultado de la matriz es:  
8      8  
8      8  
  
real    0m4.284s  
user    0m0.003s  
sys     0m0.000s  
jorge@DESKTOP-41IP801:~$
```

Prueba con 2 cores

```
jorge@DESKTOP-41IP801:~$ time ./matrices_m 2
Ingresar el numero de filas y columnas de la primera matriz:
2
2
Ingresar el numero de filas y columnas de la segunda matriz:
2
2
Introduzca los elementos de la primera matriz:
2
2
22
2
Introduzca los elementos de la segunda matriz:
2
2
2
2
2
OMP DEFINED, THREADCT = 2
El resultado de la matriz es:
8      8
48     48
real    0m2.732s
user    0m0.003s
sys     0m0.000s
jorge@DESKTOP-41IP801:~$
```

Prueba con 15 cores

```
jorge@DESKTOP-41IP801:~$ time ./matrices_m 15
Ingresar el numero de filas y columnas de la primera matriz:
2
2
Ingresar el numero de filas y columnas de la segunda matriz:
2
2
Introduzca los elementos de la primera matriz:
2
2
2
2
2
Introduzca los elementos de la segunda matriz:
2
2
2
2
2
OMP DEFINED, THREADCT = 15
El resultado de la matriz es:
8      8
8      8
real    0m2.687s
user    0m0.006s
sys     0m0.000s
jorge@DESKTOP-41IP801:~$
```

Justo antes de empezar a hacer las gráficas de tiempo de todos los experimentos, me percaté que a pesar de que el tiempo de ejercicio si se reduce conforme incrementa la cantidad de threads, este no es el único factor que altera el tiempo. A pesar de que intente introducir los datos a todos los experimentos con la misma velocidad, es imposible que tarde exactamente el mismo tiempo, por lo cual decidí modificar mi código y hacer que las matrices se llenen de números aleatorios.

Mi código imprimió la matriz resultante de la multiplicación pero al momento de insertar números grandes de matrices, se aborta la ejecución y se detenía el programa.

```
jorge@DESKTOP-41IP801:~$ time ./matrices_Nimp
OMP DEFINED, THREADCT = 1

El resultado de la matriz es:
480 588 610 650 515 535 567 685 611 602 37137 6510 6715 7379 5546 5691 7269 6556 6370 4685118
37137 6510 6715 7379 5546 5691 7269 6556 6370 4685118 294463 364380 333053 250313 265618 238761 328815 296634 288310 278734769
4685118 294463 364380 333053 250313 265618 238761 328815 296634 288310 278734769 4584871 5049209 5525990 4153424 4408314 4291083 5454447 4920279 4782388 392700
278734769 4584871 5049209 5525990 4153424 4408314 4291083 5454447 4920279 4782388 392700 221961477 229430185 251888423 188722535 200219308 194990378 247846962 223
574441 217384743 -637378604 -1774981585 -1256649638 246515366 2
592700 221961477 229430185 251888423 188722535 200219308 194990378 247846962 223574441 217384743 -637378604 -1774981585 -1256649638 246515366 2
13078561 1010983214 648866253 21534838 -1663068038 -2098198861 1747633027 648066253 21534838 -1663068038 -2098198861 1747633027 -1882347329 1499474204 142
-637378604 -1774981585 -1256649638 246515366 213078561 1010983214 648066253 21534838 -1663068038 -2098198861 1747633027 -1882347329 1499474204 142
8402958 -126516801 -199447468 -498931176 -1337152828 1630043413 659108036 1229557755 1630043413 859108036 1229557755 -474380862 139101001 -10
1747633027 -1882347329 1499474204 1428402958 -126516801 -199447468 -498931176 -1337152828 1630043413 659108036 1229557755 -474380862 139101001 -10
87581478 -1802680995 -1044318094 -1070270413 -161044795 -388208116 -1166610943 1270201693 -388208116 -1166610943 1270201693 -442549656 -1967547217 508
1239557755 -474380862 139101001 -1070270413 -161044795 -388208116 -1166610943 1270201693 -442549656 -1967547217 508
328875 984373339 1223177129 -2132267772 814141190 1277145105 662199815 -489668041 1277145105 662199815 -489668041 -802371665 -36539266 898
1270201693 -442549656 -1967547217 508638975 -884373339 1233177129 -2132267772 814141190 1277145105 662199815 -489668041 -802371665 -36539266 898
026392 1141346380 -214961532 1150760623 1152639560 -1105361873 1287241909 1543511086 -1105361873 1287241909 1543511086 1569080219 -394520121 -14
-489668041 -802371665 -36539266 898026392 1141346380 -214961532 1150760623 1152639560 -1105361873 1287241909 1543511086 1569080219 -394520121 -14
22671105 -342543107 -667217024 -1628084070 -513813179 -1886472335 -831912710 -128683914 -1886472335 -831912710 -128683914 -683883242 -733572928 -42
1543511086 1569080219 -394520121 -142671165 -343543107 -667217024 -1628084070 -513813179 -1886472335 -831912710 -128683914 -683883242 -733572928 -42
930349 1017533557 -179971272 352019125 1543811388 -1370509159 -233444422 1583332651 -1370509159 -233444422 1583332651 2103445535 1506010654 -15
1270839314 -683883242 -733572928 352019125 1543811388 -1370509159 -233444422 1583332651 -1370509159 -233444422 1583332651 2103445535 1506010654 -15
17807720 -583227115 1508402777 887093585 1365670179 -763127397 -1445011066 -346401713 -763127397 -1445011066 -346401713 600156206 299182597 188
1583332651 2103445535 1506010654 -1547807720 -583227115 1508402777 887093585 1365670179 -763127397 -1445011066 -346401713 600156206 299182597 188
8356282 1956922156 1015128477 246355894 -944716414 205933840 493495788 226016114 -205933840 493495788 226016114 259907756 1944166330 -12
146401713 600156206 299182597 18883356232 -1956922156 1015128477 246355894 -944716414 205933840 493495788 226016114 259907756 1944166330 -12
27592320 -704464255 390887307 2118477675 1849136131 932322983 -1821890439 1608109199 -1821890439 1608109199 1608109199 1145897988 -366010380 -16
226016114 259907756 1944166330 -1227592230 -704464255 390887307 2118477675 1849136131 932322983 -1821890439 1608109199 1145897988 -366010380 -16
17928478 1179611735 -1883244789 1650220121 231005423 36034265 -2867674697 -377394482 36034265 -2867674697 -377394482 21009423 36034265
1608109199 1145897988 1650220121 231005423 36034265 -2867674697 -377394482 21009423 36034265 -2867674697 -377394482 21009423 36034265
457908 1956970932 1400943798 -1797295002 655484531 -1168951116 -1023855652 -444288083 -1168951116 -1023855652 -444288083 -1296062496 1980093971 213
-377394482 -966383765 1507232229 591457908 1956970932 1400943798 -1797295002 655484531 -444288083 -1168951116 -1023855652 -444288083 -1296062496 1980093971 213
3084775 125611496 -1117412884 -1164457436 718791538 1763488072 1041227747 2086012015 -1117412884 -1164457436 718791538 1763488072 1041227747 2086012015
-1023855652 -1296062496 1980093971 2130304775 125611496 -1117412884 -1164457436 718791538 1763488072 1041227747 2086012015 -1232431797 -78
5001630 -1080238279 -1851319020 224718460 1966224892 1670249460 141209543 58457205 -1851319020 224718460 1966224892 1670249460 141209543 58457205
-2086012015 -1394817138 -1232431797 -285001630 -1080238279 -1851319020 224718460 1966224892 1670249460 141209543 58457205 -1394817138 -1232431797 -78
-60203012 -1053424244 -309015005 -2052044105 -2032457233 -705107011 865238525 376263772 -705107011 865238525 376263772 -893502361 -71
58457205 -551141567 -1592638865 -1166203812 -1053424244 -309015005 -2052044105 -2032457233 -705107011 865238525 376263772 -893502361 -71
0266845 1271358837 -1789840145 -1489739859 -1096383139 1400298910 508796511 -1712234246 1400298910 508796511 -1712234246 49124909 1172753023 -60
176263772 1200673839 426690221 893502361 710266845 1271358837 -1789840145 -1489739859 1400298910 508796511 -1712234246 49124909 1172753023 -60
778820 -1688411213 426690221 893502361 710266845 1271358837 -1789840145 -1489739859 1400298910 508796511 -1712234246 49124909 1172753023 -60
*** stack smashing detected ***: terminated
Aborted

real 0m4.015s
user 0m0.003s
sys 0m0.000s
jorge@DESKTOP-41IP801:~$
```

Debido a esto tuve que comentar la parte en la que se imprime la matriz resultante para poder comparar los tiempos de manera correcta.

Prueba con 1 core

```
jorge@DESKTOP-41IP801:~$ time ./matrices_Nimp
Ingresar el numero de filas y columnas de la primera matriz:
80
80
Ingresar el numero de filas y columnas de la segunda matriz:
80
80
OMP DEFINED, THREADCT = 1

real 0m4.015s
user 0m0.003s
sys 0m0.000s
jorge@DESKTOP-41IP801:~$
```

```
jorge@DESKTOP-41IP801:~$ top
top - 00:39:26
  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  5 root        20   0 1744 1080 1016 S   0.0  0.0 0:00.00 init
  6 root        20   0 1744 1080 1016 S   0.0  0.0 0:00.00 init
  1 root        20   0 1744 1080 1016 S   0.0  0.0 0:00.01 init
  7 root        20   0 1764   68 S   0.0  0.0 0:00.00 init
  8 root        20   0 1764   84 S   0.0  0.0 0:00.21 init
  9 jorge        20   0 8036 4880 2224 S   0.0  0.1 0:00.09 -bash
45 jorge        20   0 8684 4236 3020 R   0.0  0.1 0:00.00 http

 1 [  0.0%] 5 [  0.0%] 9 [  0.0%] 13 [  0.0%]
 2 [  0.0%] 6 [  0.0%] 10 [  0.0%] 14 [  0.0%]
 3 [  0.0%] 7 [  0.0%] 11 [  0.0%] 15 [  0.0%]
 4 [  0.0%] 8 [  0.0%] 12 [  0.0%] 16 [  0.0%]
Mem[|||||] 110M/7.66G Tasks: 5, 2 thr; 1 running
Swap[ ] 0K/2.00G Load average: 0.00 0.00 0.00
Uptime: 00:39:26
```

Prueba con 2 core

```
jorge@DESKTOP-41IP801:~$ time ./matrices_Nimp 2

Ingresar el numero de filas y columnas de la primera matriz:
80
80

Ingresar el numero de filas y columnas de la segunda matriz:
80
80
OMP DEFINED, THREADCT = 2

real    0m2.447s
user    0m0.008s
sys     0m0.000s
jorge@DESKTOP-41IP801:~$
```

```
jorge@DESKTOP-41IP801:~$ htop

1 [ 0.0%] 5 [ 0.0%] 9 [ 0.0%] 13 [ 0.0%]
2 [ 0.0%] 6 [ 0.0%] 10 [ 0.0%] 14 [ 0.0%]
3 [ 0.0%] 7 [ 0.0%] 11 [ 0.0%] 15 [ 0.0%]
4 [ 0.0%] 8 [ 0.0%] 12 [ 0.0%] 16 [ 0.0%]
Mem[||||] 110M/7.66G Tasks: 5, 2 thr; 1 running
Swap[ ] 0K/2.00G Load average: 0.00 0.00 0.00
Uptime: 00:38:58

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
5 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.00 /init
6 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.00 /init
1 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.01 /init
7 root 20 0 1764 68 0 S 0.0 0.0 0:00.00 /init
8 root 20 0 1764 84 0 S 0.0 0.0 0:00.20 /init
9 jorge 20 0 10036 4976 3224 S 0.0 0.1 0:00.09 -bash
43 jorge 20 0 8684 4216 2996 R 0.0 0.1 0:00.00 htop
```

Prueba con 10 cores

```
jorge@DESKTOP-41IP801:~$ time ./matrices_Nimp 10

Ingresar el numero de filas y columnas de la primera matriz:
80
80

Ingresar el numero de filas y columnas de la segunda matriz:
80
80
OMP DEFINED, THREADCT = 10

real    0m1.967s
user    0m0.005s
sys     0m0.000s
jorge@DESKTOP-41IP801:~$
```

```
jorge@DESKTOP-41IP801: ~
1 [ 0.0%] 5 [ 0.0%] 9 [ 0.0%] 13 [ 0.0%]
2 [ 0.0%] 6 [ 0.0%] 10 [ 0.0%] 14 [ 0.0%]
3 [ 0.0%] 7 [ 0.0%] 11 [ 0.0%] 15 [ 0.0%]
4 [ 0.0%] 8 [ 0.0%] 12 [ 0.0%] 16 [ 0.0%]
Mem[ ]
Swap[ ]
112M/7.66G Tasks: 5, 2 thr; 1 running
0K/2.00G Load average: 0.00 0.00 0.00
Uptime: 00:40:22

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
5 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.00 /init
6 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.00 /init
1 root 20 0 1744 1080 1016 S 0.0 0.0 0:00.01 /init
7 root 20 0 1764 68 0 S 0.0 0.0 0:00.00 /init
8 root 20 0 1764 84 0 S 0.0 0.0 0:00.22 /init
9 jorge 20 0 10036 4980 3224 S 0.0 0.1 0:00.10 -bash
66 jorge 20 0 8684 4236 3020 R 0.0 0.1 0:00.00 htop
```

Prueba con 15 cores

```
jorge@DESKTOP-41IP801:~$ time ./matrices_Nimp 15

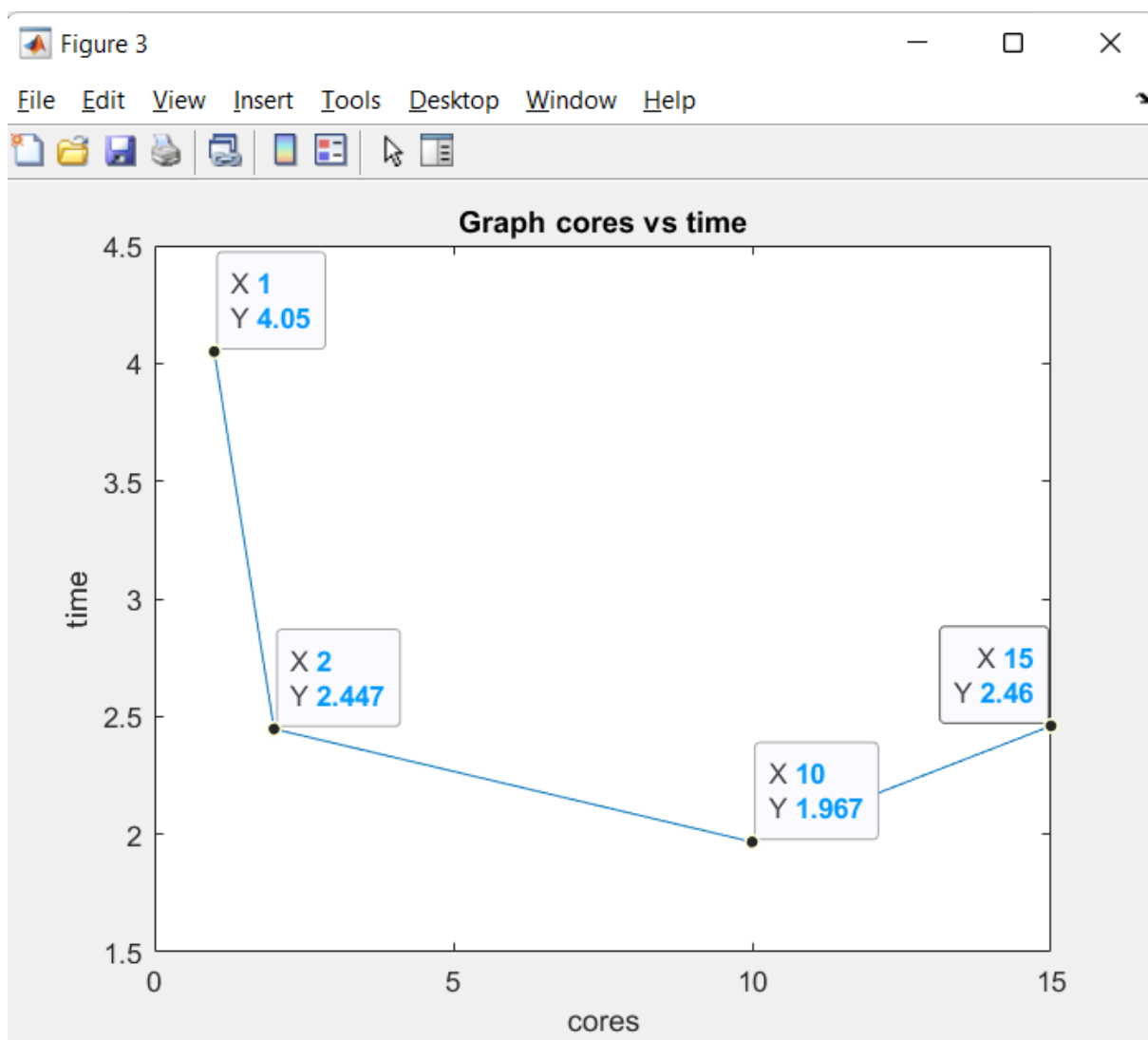
Ingresar el numero de filas y columnas de la primera matriz:
80
80

Ingresar el numero de filas y columnas de la segunda matriz:
80
80
OMP DEFINED, THREADCT = 15

real    0m2.460s
user    0m0.005s
sys     0m0.000s
jorge@DESKTOP-41IP801:~$
```

```
jorge@DESKTOP-41IP801:~$ top
top - 00:41:33
  PID USER      PRI  NI  VIRT   RES   SHR  S CPU% MEM%   TIME+  Command
  5 root        20    0 1744  1080  1016  S   0.0   0.0   0:00.00 /init
  6 root        20    0 1744  1080  1016  S   0.0   0.0   0:00.00 /init
  1 root        20    0 1744  1080  1016  S   0.0   0.0   0:00.01 /init
  7 root        20    0 1764    68    0  S   0.0   0.0   0:00.00 /init
  8 root        20    0 1764    84    0  S   0.0   0.0   0:00.23 /init
  9 jorge       20    0 10036 4980  3224  S   0.0   0.1   0:00.10 -bash
 82 jorge       20    0 8684  4312  3096  R   0.0   0.1   0:00.00 htop

Mem[ 112M/7.66G] Tasks: 5, 2 thr; 1 running
Swap[ 0K/2.00G]   Load average: 0.00 0.00 0.00
                  Uptime: 00:41:33
```



Conclusión

En esta actividad aprendí a implementar la programación en paralelo en mis códigos. En los primeros experimentos que hice los resultados no eran confiables porque dependían mucho de la velocidad con la que ingresaba los datos. La segunda parte (donde solo inserto el tamaño de la matriz) es la más confiable por lo cual la utilice para mi análisis. En la gráfica se puede ver que al inicio se sigue la regla que entre más cores pongamos, menor es el

tiempo de ejecución. En la última prueba donde puse la mayor cantidad de colores, el tiempo subió. Esto puede ser por dos razones:

- Tarde más en introducir el tamaño de las matrices
- Llega un punto en el que seguir agregando cores perjudica el rendimiento ya que es complicado hacer la coordinación de esa cantidad de cores

Esta actividad me fue de mucha utilidad ya que en clase todo es sencillo por que vamos de la mano del profesor, pero el verdadero reto empieza cuando es momento de hacerlo solo. Tardé un poco en definir la línea de `#Pragma` por que no sabia que variables debían de ser privadas, hasta que recordé que en clase mencionaron que los contadores siempre son los privados.