Pontificia Universidad
JAVERIANA
Cali

Res. 2333 del 2012

Ingeniería de Sistemas y Computación

Programación Paralela

Entrega 4: Reporte

Autor:
Guido Ernesto Salazar
Luis Alberto Salazar

Docente:
Eugenio Tamura

Junio, 2022

1 Introducción

La programación paralela es un paradigma de programación que busca acelerar computo sobre problemas con entradas extremadamente grandes donde se producen cálculos repetitivos o también se usa un mismo conjunto de datos con distintas operaciones. Esto, sin embargo, no quiere decir que siempre sea la mejor opción para resolver un problema, siempre y cuando lo comparemos con la solución secuencial. En este reporte se analizaran distintos programas y se hallara la complejidad de cómputo de los programas, teniendo en cuenta el tiempo de cómputo t_{comp} y el tiempo de comunicación t_comm.

Este análisis se hará sobre el producto punto entre 2 vectores.

2 Cálculo del programa Secuencial

El programa secuencial se puede hallar en [1]

```
lli calcular (int *vector1, int *vector2){
    lli suma = 0;
    int i;
    for (i = 0; i < TAM; ++i) {
        suma += (vector1[i] * vector2[i]);
    }
    return suma;
}
```

Contando suma de enteros: $(++i) \wedge (+=)$

$$t_{comp} = 2i_{add/sub} \times n$$

Contando multiplicación de enteros

$$\begin{aligned} t_{comp} &= 2i_{add/sub} \times n + i_{mul} \times n \\ t_{comp} &= (2i_{add/sub} + i_{mul}) \times n \end{aligned} \tag{1}$$

3 Cálculo del programa paralelo con broadcast a fuerza bruta

El programa en paralelo con envío de fuerza bruta se encuentra en [2]

3.1 Calculando el tiempo de computo

Contando suma de enteros:

$$t_{comp} = 3i_{add/sub} \times \frac{n}{p} + i_{add/sub} + 2i_{add/sub} \times (p-1)$$
$$t_{comp} = 3i_{add/sub} \times \frac{n}{p} + i_{add/sub} + 2i_{add/sub} \times p - 2i_{add/sub}$$
$$t_{comp} = 3i_{add/sub} \times \frac{n}{p} + 2i_{add/sub} \times p - i_{add/sub}$$

Contando multiplicación de enteros:

$$t_{comp} = 3i_{add/sub} \times \frac{n}{p} + 2i_{add/sub} \times p - i_{add/sub} + 2i_{mul} \times \frac{n}{p}$$
$$t_{comp} = (3i_{add/sub} + 2i_{mul}) \times \frac{n}{p} + 2i_{add/sub} \times p - i_{add/sub}$$
$$t_{comp} = (3i_{add/sub} + 2i_{mul}) \times \frac{n}{p} + i_{add/sub} \times (2p-1)$$

Contando división de flotantes:

$$t_{comp} = (3i_{add/sub} + 2i_{mul}) \times \frac{n}{p} + i_{add/sub} \times (2p-1) + \phi_{div} \tag{2}$$

3.2 Calculando el tiempo de comunicación

Calculando el tiempo de comunicación del envío

$$t_{comm} = (p-1)\left(\lambda + \frac{2\frac{n}{p} \times 4}{\beta}\right) + (p-1)\left(\lambda + \frac{4}{\beta}\right)$$
$$t_{comm} = (p-1)\left(2\lambda + \frac{8\frac{n}{p} + 4}{\beta}\right) \tag{3}$$

3.3 Juntando los tiempos

Uniendo las ecuaciones obtenidas en (2) y (6) obtenemos el tiempo total que demoraría el programa paralelo con comunicación de fuerza bruta, obteniendo

$$t(n,p) = t_{comp} + t_{comm}$$
$$t(n,p) = (3i_{add/sub} + 2i_{mul}) \times \frac{n}{p} + i_{add/sub} \times (2p-1) + \phi_{div} + (p-1)\left(2\lambda + \frac{8\frac{n}{p} + 4}{\beta}\right) \tag{4}$$

4 Cálculo del programa paralelo con broadcast con el árbol binario

El programa en paralelo con envío de fuerza bruta se encuentra en [3]

4.1 Calculando el tiempo de computo

Contando suma de enteros:

$$t_{comp} = 5i_{add/sub} \times \lceil log_2(p) \rceil + 5i_{add/sub} + 2i_{add/sub} \times \frac{n}{p}$$
$$t_{comp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + 2i_{add/sub} \times \frac{n}{p}$$

Contando multiplicación de enteros:

$$t_{comp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + 2i_{add/sub} \times \frac{n}{p} + i_{mul} \times \frac{n}{p}$$
$$t_{comp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{n}{p}(2i_{add/sub} + i_{mul})$$

Contando división de enteros:

$$t_{comp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{n}{p}(2i_{add/sub} + i_{mul}) + i_{div} \times \lceil log_2(p) \rceil$$

Contando división de flotantes:

$$t_{comp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{n}{p}(2i_{add/sub} + i_{mul}) + i_{div} \times \lceil log_2(p) \rceil + \phi_{div} \quad (5)$$

4.2 Calculando el tiempo de comunicación

Calculando el tiempo de comunicación del envío

$$t_{comm} = \sum_{i=0}^{\lceil log_2(p) \rceil} \sum_{j=\lceil log_2(i) \rceil}^{\lceil log_2(p) \rceil} \left(\lambda + \frac{8\frac{n}{2^{(j+1)}} + 4}{\beta} \right) \quad (6)$$

4.3 Juntando los tiempos

Uniendo las ecuaciones obtenidas en (2) y (6) obtenemos el tiempo total que demoraría el programa paralelo con comunicación de fuerza bruta, obteniendo

$$t(n,p) = t_{comp} + t_{comm}$$
$$t(n,p) = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{n}{p}(2i_{add/sub} + i_{mul}) + i_{div} \times \lceil log_2(p) \rceil + \phi_{div} + \sum_{i=0}^{\lceil log_2(p) \rceil} \sum_{j=\lceil log_2(i) \rceil}^{\lceil log_2(p) \rceil} \left(\lambda + \frac{8\frac{n}{2^{(j+1)}} + 4}{\beta} \right) \quad (7)$$

5 Usando OpenMp

El uso de OpenMp sobre los programas paralelos agregan la característica de que se puede trabajar con paralelización dentro de la máquina, por lo que se acelera la capacidad de cómputo al poder realizar más operaciones en distintas partes. Esta mejora se hará sobre las ecuaciones (2) y (5) obteniendo

$$t_{omp} = (3i_{add/sub} + 2i_{mul}) \times \frac{\frac{n}{p}}{c} + i_{add/sub} \times (2p - 1) + \phi_{div} \quad (8)$$
$$t_{omp} = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{\frac{n}{p}}{c}(2i_{add/sub} + i_{mul}) + i_{div} \times \lceil log_2(p) \rceil + \phi_{div} \quad (9)$$

6 Mezclando Mpi + OpenMp

Lo que permite hacer la programación paralela sobre ciertos programas es el uso de paralelismo fuera de la máquina al añadir capacidad de cómputo como más computadores o dentro de la máquina, añadiendo más unidades de procesamiento. Por lo que el analisis de los tiempos ahora se reflejarían de la siguiente manera

$$t(n,p,c) = t_{omp} + t_{comm}$$
$$t_{bf}(n,p,c) = 3i_{add/sub} + 2i_{mul}) \times \frac{\frac{n}{p}}{c} + i_{add/sub} \times (2p - 1) + \phi_{div} + (p - 1)\left(2\lambda + \frac{8\frac{n}{p} + 4}{\beta}\right) \quad (10)$$

$$t_{bt}(n,p,c) = 5i_{add/sub}(\lceil log_2(p) \rceil + 1) + \frac{\frac{n}{p}}{c}(2i_{add/sub} + i_{mul}) + i_{div} \times \lceil log_2(p) \rceil + \phi_{div} + \sum_{i=0}^{\lceil log_2(p) \rceil} \sum_{j=\lceil log_2(i) \rceil}^{\lceil log_2(p) \rceil} \left(\lambda + \frac{8\frac{n}{2^{(j+1)}} + 4}{\beta} \right) \quad (11)$$

7 Tiempos teóricos

En los tiempos teóricos se usarán como parámetros fijos los valores $i_{sum/res} = 1 \wedge i_{mul} = 3 \wedge i_{div} = 90 \wedge \phi_{div} = 27$ para las ecuaciones (7), (4), (11) y (10). También se tomará los valores dé $\lambda = 6\mu s \wedge \beta = 125MB/s$.

Entonces, se elaboró un gráfico y una tabla por cada versión paralela, en los cuales se compara el tiempo secuencial, los valores ingresados y el speedup entre cada uno. En la primera tabla se va a mostrar los resultados teóricos de la versión de fuerza bruta. Cabe añadir, que el tiempo obtenido es en nanosegundos.

Matrix Size	ts	2 Processors		4 Processors		8 Processors	
		t_bf	speedup	t_bf	speedup	t_bf	speedup
512	2560	2347,368	1,091	1224,056	2,091	704,408	3,634
1024	5120	4652,733	1,1	2378,104	2,153	1282,797	3,991
2048	10240	9263,464	1,105	3686,200	2,185	2439,576	4,197
4096	20480	18484,925	1,108	9302,392	2,202	4753,133	4,309
8192	40960	36927,848	1,109	18534,776	2,210	9380,248	4,367
16384	81920	73813,693	1,110	36999,544	2,214	18634,477	4,396

Figure 1: Brute Force Table.

En donde, `t_bF` significa el tiempo de fuerza bruta. A continuación se va a mostrar el gráfico de dicha tabla.

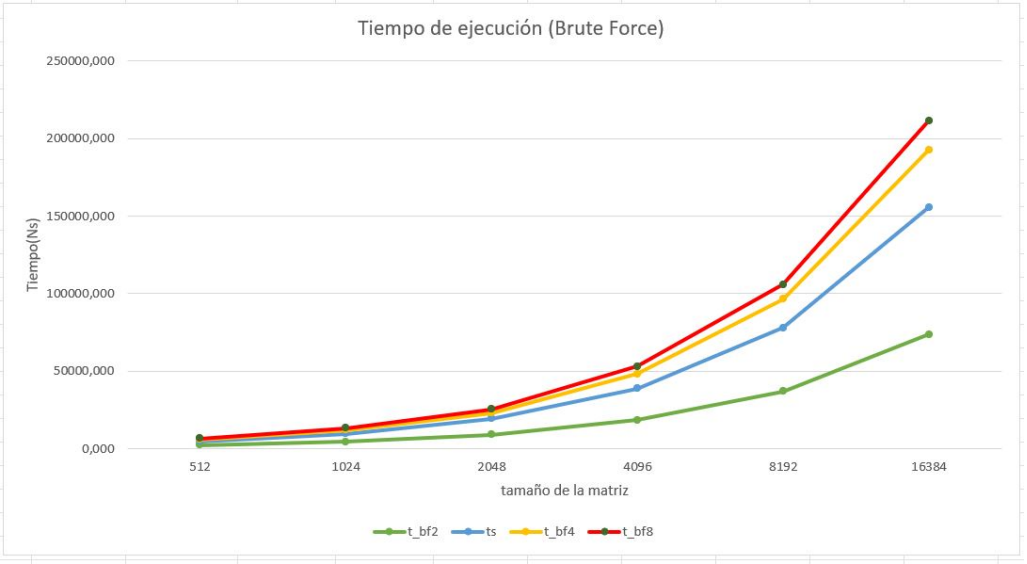

Figure 2: Gráfico del tiempo de ejecución de 1

La siguiente tabla que se va a mostrar, es la tabla de los resultados obtenidos por el algoritmo del árbol binomial.

Matrix Size	ts	2 Processors		4 Processors		8 Processors	
		t_bt	speedup	t_bt	speedup	t_bt	speedup
512	2560	1414,368	1,810	890,107	2,876	690,824	3,706
1024	5120	2695,733	1,899	1534,203	3,337	1016,627	5,036
2048	10240	5258,464	1,947	2822,395	3,628	1668,232	6,138
4096	20480	10383,925	1,972	5398,779	3,793	2971,443	6,892
8192	40960	20634,848	1,985	10551,547	3,882	5577,864	7,343
16384	81920	41136,693	1,991	20857,083	3,928	10790,707	7,592

Figure 3: Binary Tree Table.

En donde `t_bt` es el tiempo teórico obtenido para el árbol binomial. Cuyo gráfico es el siguiente.

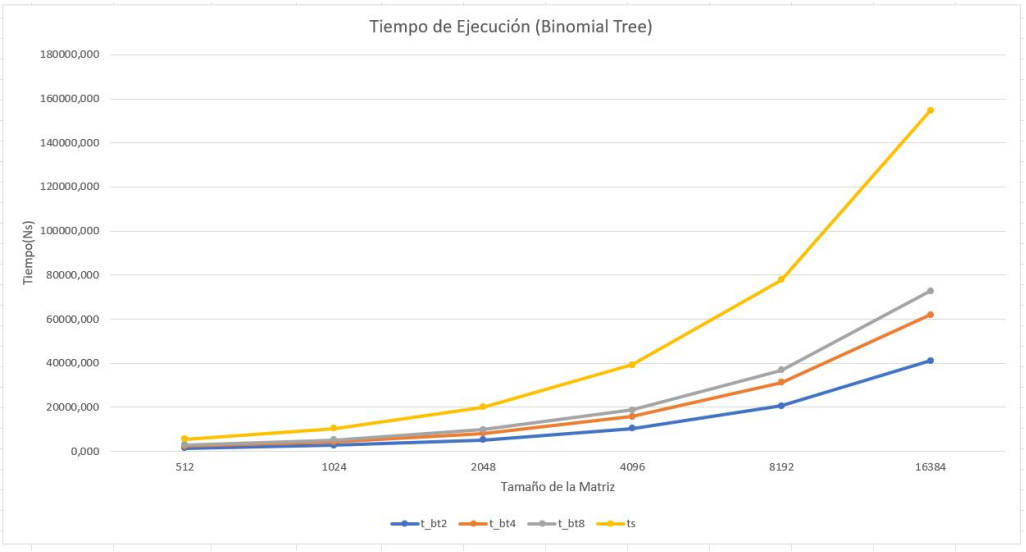

Figure 4: Gráfico del tiempo de ejecución de 3

Ahora bien, por último, se muestra la tabla de `OpenMPI` y `OpenMP`, en donde se usaron dos procesos y 8 procesos, repitiendo en cada uno 2, 4 y 8 cores.

Matrix Size	ts	2 Processors - 2 Cores				2 Processors - 4 Cores				2 Processors - 8 Cores			
		t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup	t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup	t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup
512	2560	1195,368	2,142	774,368	3,306	619,368	4,133	454,368	5,634	331,368	7,726	294,368	8,697
1024	5120	2348,733	2,180	1415,733	3,617	1196,733	4,278	775,733	6,600	620,733	8,248	455,733	11,235
2048	10240	4655,464	2,199	2698,464	3,794	2351,464	4,354	1418,464	7,219	1199,464	8,537	778,464	13,154
4096	20480	9268,925	2,209	5263,925	3,891	4660,925	4,394	2703,925	7,574	2356,925	8,689	1423,925	14,383
8192	40960	18495,848	2,215	10394,848	3,940	9279,848	4,414	5274,848	7,765	4671,848	8,767	2714,848	15,087
16384	81920	36949,693	2,217	20656,693	3,966	18517,693	4,424	10416,693	4,424	9301,693	8,807	5296,693	15,466
Matrix Size	ts	8 Processors - 2 Cores				8 Processors - 4 Cores				8 Processors - 8 Cores			
		t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup	t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup	t_mpi_mp_bf	speedup	t_mpi_mp_bt	speedup
512	2560	416,408	6,148	530,824	4,823	272,408	9,398	450,824	5,679	200,408	12,774	410,824	6,231
1024	5120	706,797	7,244	696,627	7,350	418,797	12,225	536,627	9,541	274,797	18,632	456,627	11,213
2048	10240	1287,576	7,953	1028,232	9,959	711,576	14,391	708,232	14,459	423,576	24,175	548,232	18,678
4096	15360	2449,133	8,362	1691,443	12,108	1297,133	15,789	10251,443	19,478	723,133	28,399	731,443	27,999
8192	20480	4772,248	8,583	3017,864	13,573	2468,248	16,595	1737,864	23,569	1316,248	31,118	1097,864	37,308
16384	25600	9418,477	8,698	5670,707	14,446	4810,477	17,029	3110,707	26,335	2506,477	32,683	1830,707	44,748

Figure 5: `OpenMPI` and `OpenMP` table.

En dónde `t_mpi_mp_bf` es el tiempo teórico calculado para el algoritmo de fuerza bruta para `OpenMP` y `OpenMPI`, `t_mpi_mp_bf` es el tiempo calculado para el algoritmo del árbol binomial para `OpenMP` y `OpenMPI`. Para el gráfico, se dividió en dos, la figura 6 representa el tiempo de ejecución para dos procesos y la figura 7 representa el tiempo de ejecución para ocho procesos.

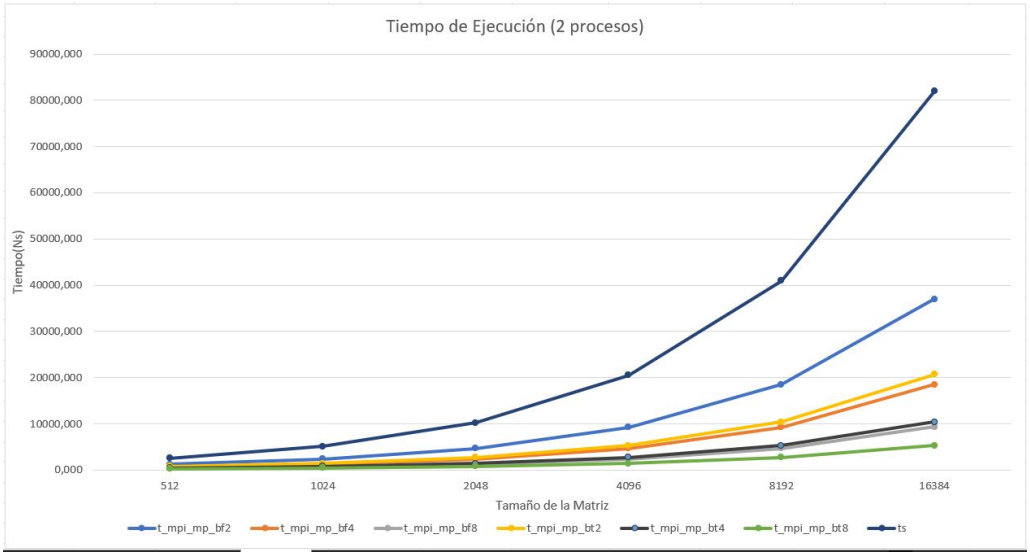

Figure 6: Gráfico del tiempo de ejecución de 5 para dos procesos.

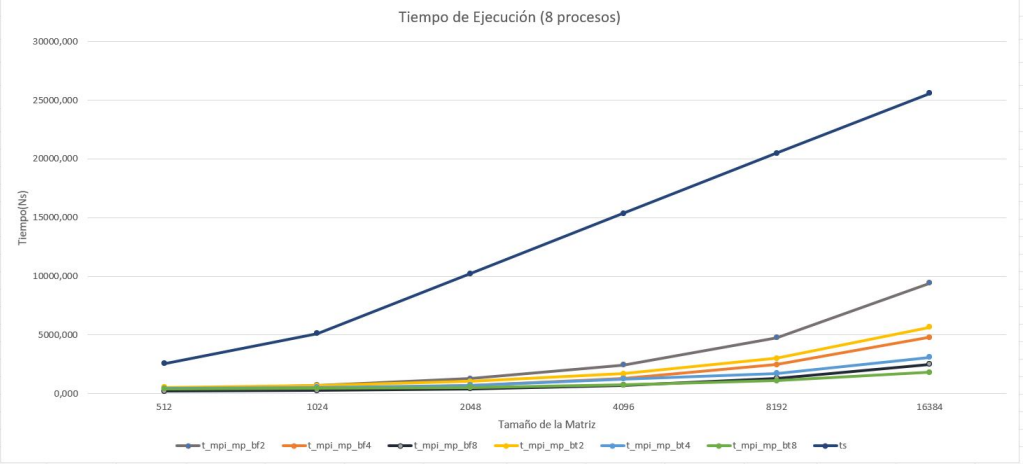

Figure 7: Gráfico del tiempo de ejecución de 5 para ocho procesos.

8 Conclusiones

Después de haber construido las fórmulas para la estimación de tiempo, se puede observar la complejidad que lleva calcular la estimación de tiempo de los programas paralelos, pues no solo toman tiempo de programar, ya que hay que tener en consideración muchos detalles de optimización en temporalidad y espacialidad, sino de construir el tiempo en el que se ejecutarían teóricamente. El tiempo calculado por medio de las fórmulas, es distinto al tiempo que realmente toma en la máquina, pues hay más variables y factores atípicos que no se tienen en cuenta la hora de calcular el tiempo teórico.

Al obtener los datos, es posible ver como van cambiando respecto a la cantidad de procesos y cores usados, de hecho, es posible identificar que tipo de aproximación paralela comienza a volverse eficiente en el tiempo según la cantidad de procesos, cores y memoria. Pero, sobre todo, que la versión secuencia en cuestión de tiempo se vuelve más grande en el tiempo que las versiones paralelas, esto demuestra una vez más que la programación paralela permite optimizar el tiempo en el cálculo y el procesamiento de datos.

Por último, solo queda por añadir que probando los programas en paralela y calculando los tiempos, es posible identificar para qué usos realmente optimizan tiempo y se vuelven eficientes, ya que a pesar de que los algoritmos secuenciales toman más tiempo calculando y procesando lo mismo que las aproximaciones paralelas, ellos pueden ser más sencillos de programar y de esta manera obtener los mismos datos con una diferencia de tiempo pequeña a nivel de razón humana, pero grande a nivel computacional. Lo importante es identificar los usos y si realmente vale la pena

hacer una aproximación secuencial o una paralela para optimizar tiempo.

9 Referencias

[1]https://github.com/luisalazago/Entregas-Programacion-Paralela/blob/master/Entrega%201/Entrega1_Secuencial.c

[2]https://github.com/luisalazago/Entregas-Programacion-Paralela/blob/master/Entrega%201/Entrega1.2_Paralela.c

[3] https://github.com/luisalazago/Entregas-Programacion-Paralela/blob/master/Entrega%201/Entrega1_BinomialTree.c