Análisis

Jesús Rodríguez Heras

12 de noviembre de 2018

Resumen

Desarrollo de la tabla con las conclusiones de usar varios hilos en la resolución de los problemas 1 y 2 de la práctica 4.

Índice

1. `matVector.java`

Para las pruebas realizadas, hemos considerado una matriz cuadrada con el mismo número de columnas que elementos tiene el vector por el cual se multiplica dicha matriz.

Elementos	% CPU	Tiempo (s)
500	13	0.014
1000	17	0.011
1500	19	0.013
2000	21	0.015
2500	31	0.016
3000	32	0.021
3500	38	0.024
4000	47	0.026
4500	58	0.034
5000	64	0.04

Tabla 1: Valores de matVector.

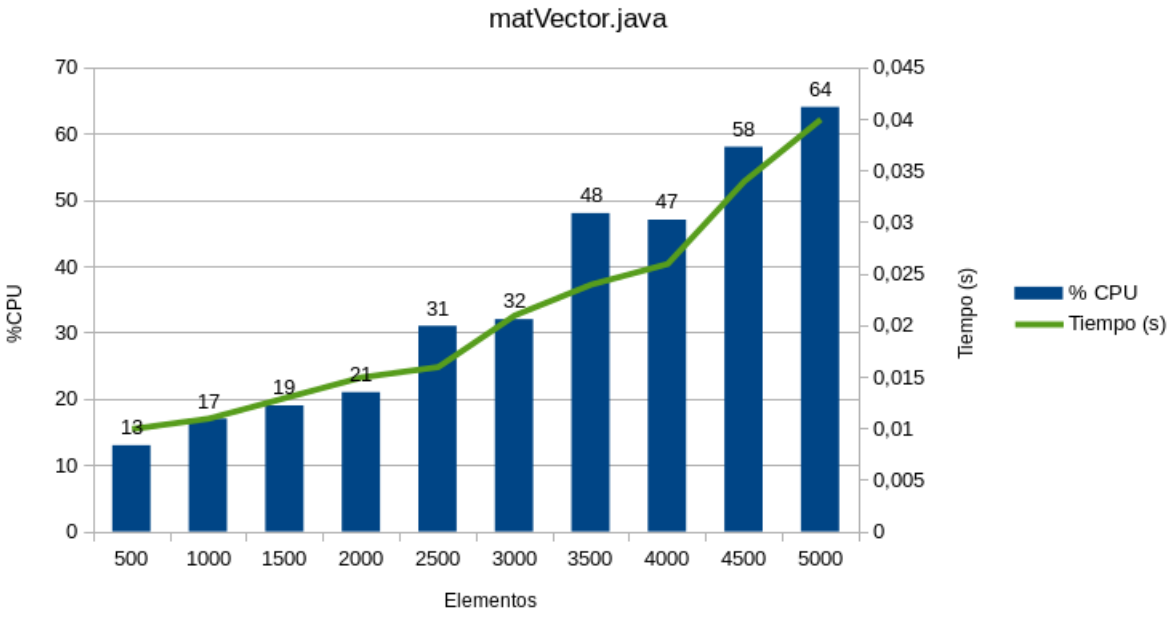

Figura 1: Gráfica de matVector.

2. `matVectorConcurrente.java`

Para las pruebas realizadas, hemos utilizado una matriz cuadrada con el mismo número de columnas que elementos tiene el vector por el cual se multiplica dicha matriz. Este algoritmo crea un hilo por cada elemento del vector resultante.

Elementos	% CPU	Tiempo (s)
500	18	0.093
1000	20	0.096
1500	24	0.133
2000	23	0.186
2500	22	0.216
3000	36	0.232
3500	46	0.276
4000	47	0.269
4500	49	0.32
5000	43	0.356

Tabla 2: Valores de matVectorConcurrente.

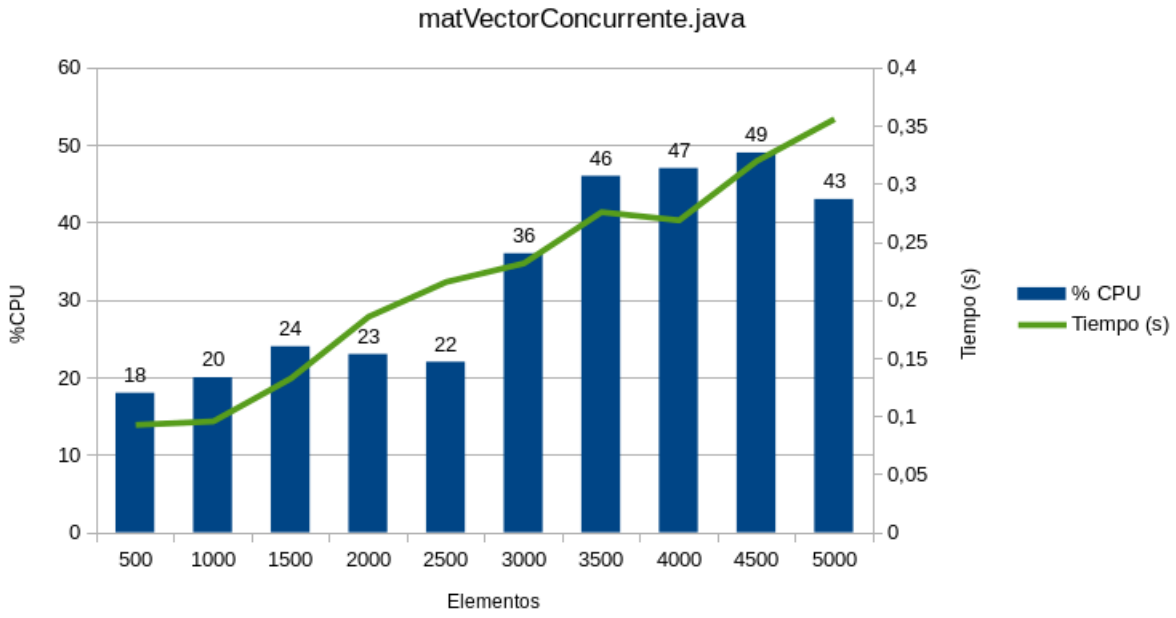

Figura 2: Gráfica de matVectorConcurrente.

3. `prodMat.java`

Para las pruebas realizadas hemos usado matrices cuadradas.

Elementos	% CPU	Tiempo (s)
500	36	0.395
1000	100	15.982
1500	100	63.443
2000	100	159.822
2500	100	323.924

Tabla 3: Valores de prodMat.

Viendo como va evolucionando la tabla, poner más valores es innecesario, ya que sería inviable para valores más altos.

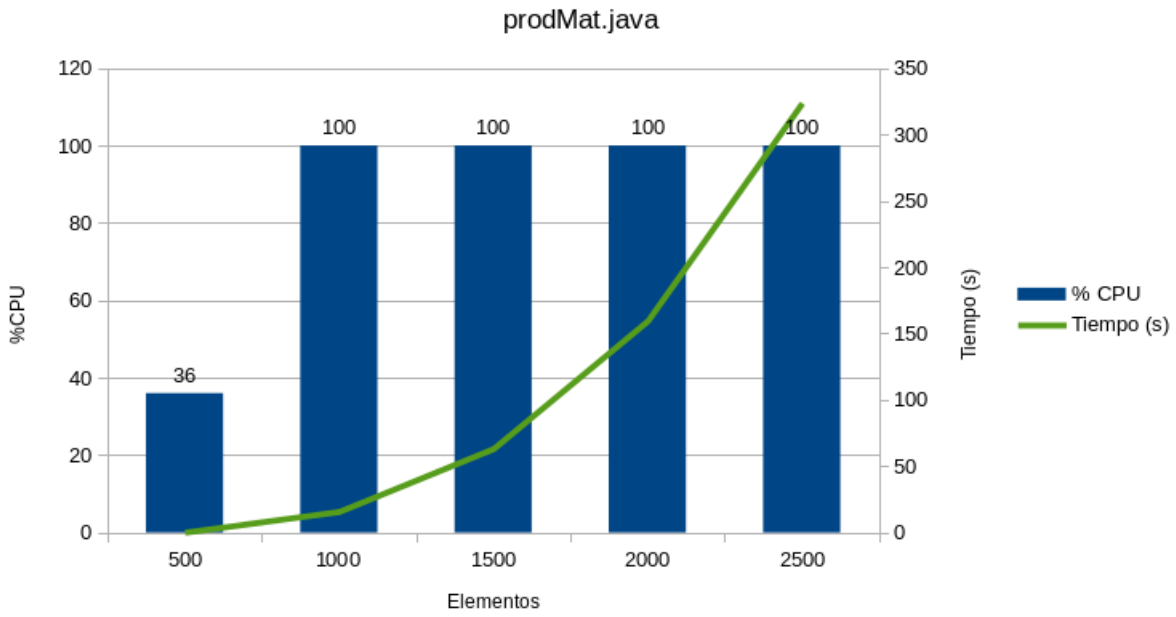

Figura 3: Gráfica de prodMat.

4. `prodMatConcurrente.java`

Para las pruebas realizadas, hemos usado matrices cuadradas. Este algoritmo crea un hilo por cada fila de la matriz resultante.

Elementos	% CPU	Tiempo (s)
25	24	0.083
50	17	0.211
100	34	0.738
200	53	3.609
300	66	13.348
400	69	37.195
500	74	114.118

Tabla 4: Valores de prodMatConcurrente.

Debido al uso de excesivos hilos, si aumentamos el numero de filas/columnas a más de 500, el programa tarda mucho en ejecutarse y con la batería de pruebas mostrada se ve el crecimiento que tiene.

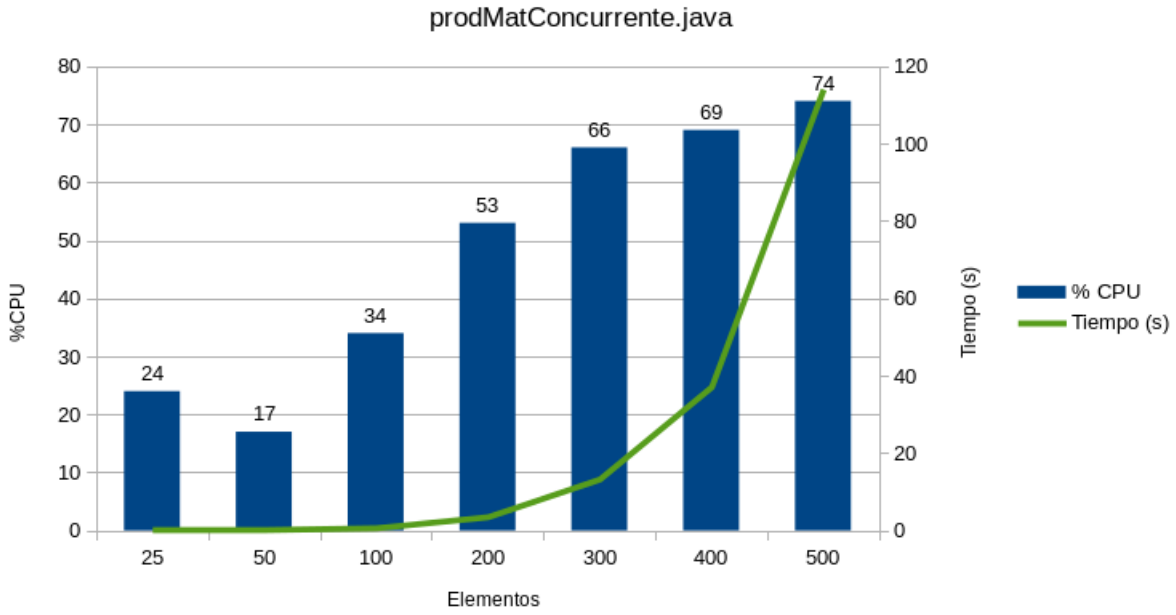

Figura 4: Gráfica de prodMatConcurrente.

5. Impresiones recabadas

Viendo los resultado obtenidos en las pruebas de los diversos algoritmos, podemos deducir que el hecho de crear más hilos, no nos determina la velocidad con la que se ejecutará un programa.

Esto es así, debido a que la creación de hilos, nos genera un coste de tiempo y de memoria que puede ser crucial en otros aspectos.

El ejemplo más claro lo obtenemos entre los algoritmos `prodMat.java` y `prodMatConcurrente.java` donde, con dos matrices de $500x500$ elementos, en el primero de ellos (secuencial) tardamos 0.395 segundos, mientras que en el segundo (concurrente) tardamos 114.118 segundos.