Lista Enlazada:

Resultados de acuerdo a la tabla 4.3:

	1	2	4	8
Read-Write Locks	0,01211906667	0,0763754	0,413084	0,9542759
One mutex for the entire list	0,0114671	0,1200681	0,7968294	2,2586454
One mutex per node	0,0462275	0,3488624		

En esta prueba se realizaron 99900 operaciones Member, 50 operaciones Insert y 50 operaciones Delete. Se puede observar que el mejor desempeño lo obtiene Read-Write Locks excepto en la ejecución secuencial donde el mejor desempeño lo obtiene One mutex for the entire list.

Resultados de acuerdo a la tabla 4.4:

	1	2	4	8
Read-Write Locks	2,548376267	9,0727157	20,5475632	42,9285364
One mutex for the entire list	2,2214123	7,5118578	15,8351603	22,2687824
One mutex per node	4,3756151	19,7101552	53,8014024	113,6549608

Sección 4.10:

La sección 4.10 nos habla principalmente de la coherencia de cache y el gran impacto que tiene al momento de tomar el tiempo de ejecución de algún algoritmo paralelo, esto debido a que cuando por ejemplo, tenemos una variable compartida y dos hilos, cada hilo tendrá una variable local en caché, antes de que cualquier hilo pueda modificar su valor local de acuerdo al valor global, debe realizar una verificación para asegurar la coherencia de caché. ESta operación consume tiempo de ejecución que se ve reflejado en el tiempo total de la ejecución del programa paralelo.

Resultados de acuerdo a la tabla 4.5:

	8000000 x 8		8000 x 8000		8 x 8000000	
1	0,1700814	1	0,1628228	1	0,1863375	1
2	0,3413422	0,2491362041	0,1661206	0,4900740787	0,1908823	0,4880952818
4	0,7780797	0,05464780793	0,1735537	0,2345423924	0,2060917	0,2260371233

De acuerdo a los resultados podemos concluir que el tiempo de ejecución aumenta cada vez que se incrementa el número de hilos utilizados. la eficiencia máxima se logra al utilizar 2 hilos en una matriz de 8000 x 8000.

Función strtok:

La función strtok nos permite separar un string en partes, para ello recibe el string que queremos separar y a continuación debemos indicar los separadores que tendrá en cuenta.

Al intentar utilizar esta función en un ambiente paralelo, nos enfrentamos al problema en el que un hilo no sabe dónde se quedó el anterior hilo y por consiguiente separar una parte del string que otro hilo ya separó o, de lo contrario, no separar una parte del string que debía ser separado. Éste problema se soluciona utilizando la función strtok_r la cual es thread-safe, debido a que tiene un parámetro más el cual es un puntero que indica a cada hilo en qué parte del string se quedó el hilo que se ejecutó antes.

Ejercicio 4.4:

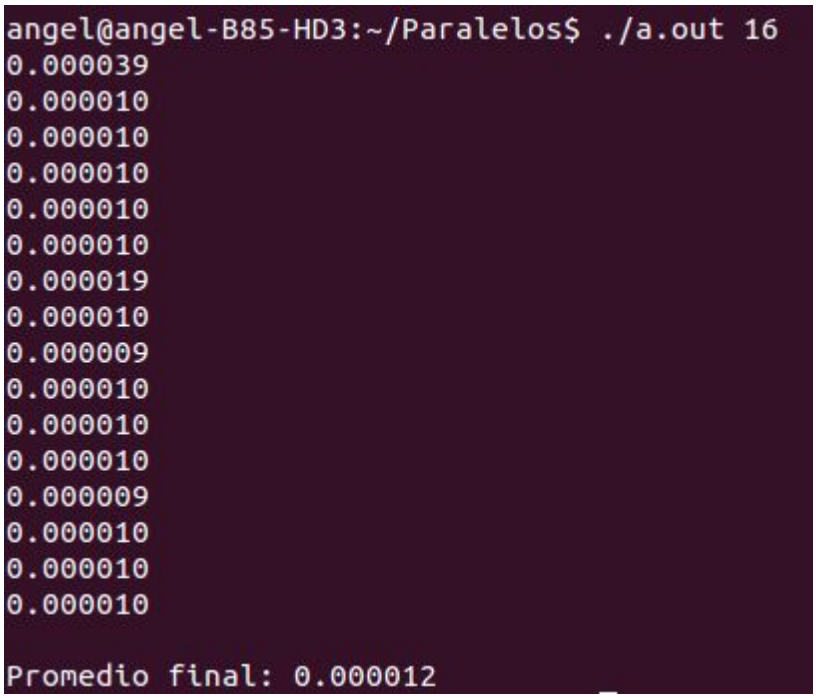
Ejecución con 1 hilo:

```
angel@angel-B85-HD3:~/Paralelos$ g++ -std=c++11 -pthread hilos.c
angel@angel-B85-HD3:~/Paralelos$ ./a.out 1
0.000033

Promedio final: 0.000033
```

Ejecución con 4 hilos:

```
angel@angel-B85-HD3:~/Paralelos$ ./a.out 4
0.000033
0.000010
0.000010
0.000010

Promedio final: 0.000016
```

Ejecución con 16 hilos:



De los resultados, podemos concluir que efectivamente existe la tendencia de, entre más número de hilos, menor es el promedio de tiempo de ejecución utilizado las funciones que permiten crear y esperar a un hilo hasta que termine.

Pruebas realizadas:

Pruebas de acuerdo a la tabla 4.3:

	1	2	4	8
Read-Write Locks	0,012769	0,075884	0,409138	0,945298
	0,012658	0,074821	0,413698	0,935312
	0,012598	0,07695	0,41375	0,960589
	0,012556	0,075546	0,421913	0,936434
	0,012185	0,075782	0,403849	0,981984
	0,012246	0,075976	0,395012	0,93031
	0,012242	0,07763	0,427831	0,955663
	0,012122	0,078989	0,411407	0,946284
	0,012482	0,07513	0,427448	0,993889
	0,012104	0,077046	0,406794	0,956996
One mutex for the entire list	0,012085	0,123292	0,783148	2,249697
	0,011329	0,11511	0,897249	2,323088
	0,011506	0,125796	0,837015	2,152748
	0,011704	0,1156	0,8519	2,201745

	0,0112	0,105876	0,762051	2,331888
	0,011249	0,120529	0,754024	2,243003
	0,011685	0,123428	0,800309	2,259144
	0,01113	0,126678	0,802823	2,346379
	0,011175	0,121351	0,752286	2,205307
	0,011608	0,123021	0,727489	2,273455
One mutex per node	0,045612	0,265005		
	0,045849	0,359121		
	0,04793	0,363341		
	0,04559	0,386406		
	0,045791	0,360655		
	0,045384	0,29816		
	0,045953	0,363061		
	0,046836	0,371301		
	0,047035	0,374169		
	0,046295	0,347405		

Pruebas de acuerdo a la tabla 4.4:

	1	2	4	8
Read-Write Locks	2,797594	9,182442	20,604063	42,894141
	2,705669	8,963139	20,553049	43,136955
	2,675873	9,13061	20,491347	42,894226
	2,703471	9,100777	20,5646	42,927264
	2,749076	8,889627	20,478327	42,859301
	2,713765	9,028656	20,48108	43,026332
	2,797787	9,065751	20,489076	42,690328
	2,724545	9,216134	20,437129	43,344634
	2,704797	9,07765	20,841112	42,583885
	2,696994	9,072371	20,535849	42,928298
One mutex for the entire list	2,151842	7,306481	15,642034	22,235554
	2,181739	7,449228	15,799578	23,935985
	2,197178	7,86068	15,682649	21,24461
	2,280415	7,540033	16,224392	22,269573
	2,144899	7,62008	15,566672	22,309807
	2,247747	7,419666	16,001194	23,534889

	2,208295	7,627509	15,746129	22,361795
	2,247619	7,379029	16,681101	21,594955
	2,165058	7,576441	15,456437	21,710043
	2,389331	7,339431	15,551417	21,490613
One mutex per node	4,321041	19,638049	53,588269	115,383942
	4,304934	19,484876	53,629228	115,045867
	4,298725	19,502307	53,756798	115,027746
	4,388936	20,365425	53,903326	113,723653
	4,39753	19,50722	53,990717	112,785956
	4,414432	19,91678	53,82807	113,374777
	4,400385	20,070339	53,82746	113,297172
	4,410367	19,485273	53,981363	112,034261
	4,437	19,556632	53,771686	113,149035
	4,382801	19,574651	53,737107	112,727199

Pruebas de acuerdo a la tabla 4.5:

	8000000 x 8	8000 x 8000	8 x 8000000
1	0,169338	0,163112	0,187449
	0,169536	0,162471	0,187588
	0,168846	0,162723	0,185653
	0,171797	0,162536	0,185972
	0,16928	0,162299	0,18608
	0,169377	0,163756	0,184839
	0,169757	0,163719	0,186992
	0,171162	0,162947	0,187253
	0,171323	0,162338	0,186401
	0,170398	0,162327	0,185148
2	0,32163	0,163927	0,188427
	0,363552	0,164218	0,18848
	0,334762	0,163784	0,192097
	0,387133	0,163756	0,195362
	0,341599	0,167472	0,192418
	0,320769	0,167249	0,193484
	0,319951	0,168071	0,192205
	0,321551	0,167493	0,189033
	0,384811	0,167634	0,189307

	0,317664	0,167602	0,18801
	0,732284	0,173851	0,202174
	0,724074	0,173828	0,202229
	0,751762	0,17381	0,215018
	0,864284	0,173736	0,214352
	0,861713	0,173997	0,21699
	0,737793	0,173917	0,202328
	0,733084	0,173672	0,199665
	0,861845	0,171063	0,202731
	0,762257	0,173738	0,202867
4	0,751701	0,173925	0,202563