CS 6848: Lab 1 Report.

Mahesh Gupta (CS12M018)

15th Feb 2013

Report:-

For matrix multiplication, 6 different versions were written. The approach used in each of these is explained as:

1. Simple Matrix multiplication with optimization by taking transpose of matrix 2 and executing 3 for loops.
2. Simple Block matrix multiplication with block size 16 as base case.
3. Parallel matrix multiplication, same as case 1 but out of 3 loops, outer loop made concurrent using **cilk_for** construct.
4. Parallel matrix multiplication, same as case 1 but out of 3 loops, inner loop made concurrent using **cilk_for** construct.
5. Parallel matrix multiplication, same as case 1 but out of 3 loops, but both outer and inner loop made concurrent using **cilk_for** construct.
6. Parallel matrix multiplication, sameas case 2 but for each method invocation a new thread is spawned using **cilk_spawn** and to synchronize between the operations, using **cilk_sync** construct and block size 16 as the base case.

The 6 different approaches were tried for the matrices of different order from 64*64 to 4096*4096 (8192*8192 matrix was taking too long to execute that it was not possible to use it.).

Evaluation Process:

For each approach, we execute the algorithm for different input size, and for each input size 5 different inputs and average of those 5 is taken. Then for each algorithm it is evaluated how it performs as input size increases and how different algorithms grow as input size increases.

Runtime:

a. For Block multiplication, as the input size increases the run time increases by the factor of 8-10

b. For Simple multiplication, as the input increases the run time increases by factor of around 10-12.

c. For Parallel inner cilk version, as input size increases runtime increases by factor of around 5.

d. For Parallel outer cilk version, as input increases the runtime increases by factor of 10.

e. For inner outer cilk, as input increases the runtime increases by factor of 10.

f. For parallel block cilk, as input increases the runtime increase by factor of 4.

g. As number of processors increases, there is no significant difference in runtime except in case of outer for loop approach.

Effect of Various Factors: According to our observation, following factors have mainly influenced the performance of different program.

1. **Effect of block size for Serial and Parallel Versions:** The experiment was performed for different input sizes for different block size. It has been observed that.

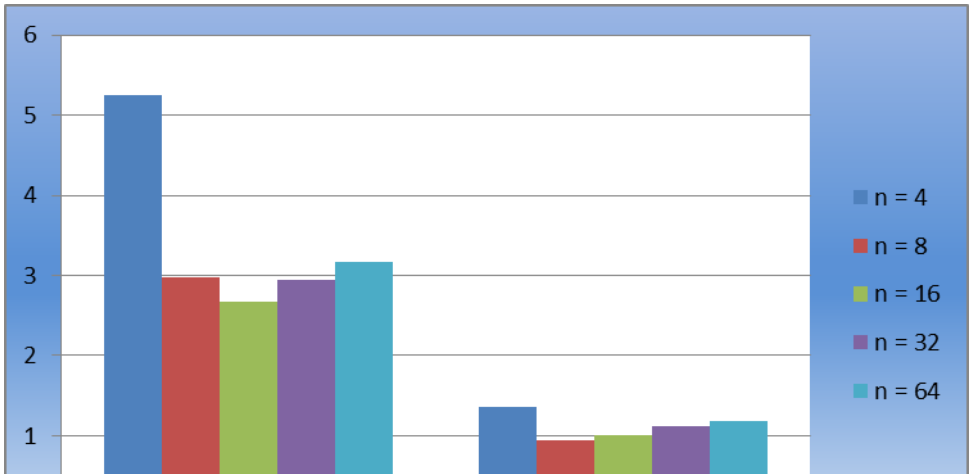
   a. For serial block multiplication, when the block size is around 8-16, performance is optimal.

   b. For parallel block multiplication, when the block size is around 16-32, performance is optimal.

   c. For the both the cases, it was observed that as the input increases to twice the value, the total time taken also increases by 8-10 times.

   d. For the same input matrix, parallel multiplication is about 2.5-3 times faster than serial multiplication.

	Block size	run 1	run 2	run 3	run 4	run 5	Average time
Serial	4	5.772	5.101	5.148	5.101	5.148	5.254
	8	3.76	2.839	2.808	2.715	2.776	2.9796
	16	3.229	2.512	2.511	2.543	2.527	2.6644
	32	3.774	2.762	2.73	2.776	2.714	2.9512
	64	3.916	2.995	2.979	2.996	2.964	3.17
parallel	4	1.357	1.342	1.357	1.342	1.357	1.351
	8	0.936	0.936	0.936	0.952	0.951	0.9422
	16	0.998	1.0014	0.999	0.998	0.999	0.99908
	32	1.124	1.123	1.139	1.123	1.107	1.1232
	64	1.17	1.18	1.17	1.185	1.17	1.175



2. **Effect of 'cilk_for' at various places for Simple Parallel algorithm:** The experiment was performed by using cilk_for at different places and by observing the efficiency obtained for the different values. It was observed that:

   a. For outer and inner-outer cilk_for versions, as the input size increases by 2, the time factor increases by almost 10 times.

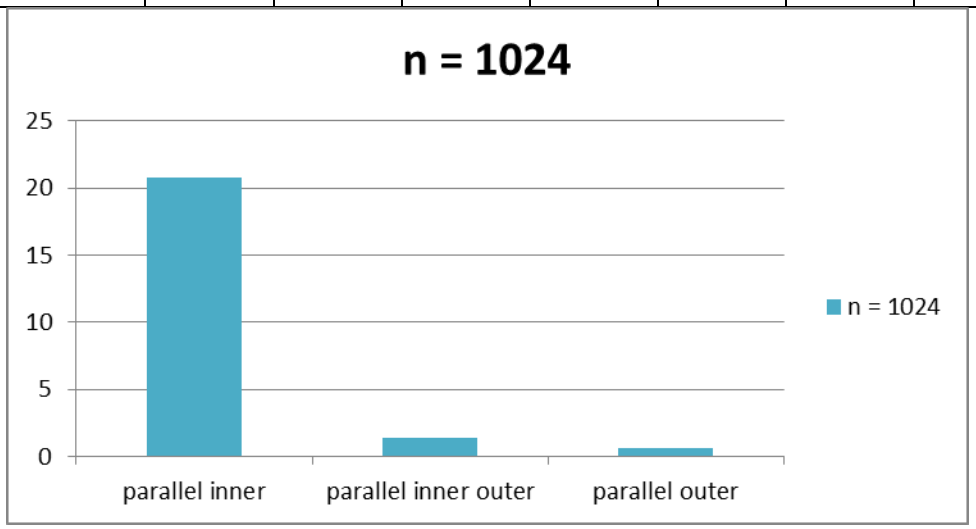
   b. For inner cilk_for versions, as the input size increases by 2, the time factor increases by 8-10 times.

   c. Outer cilk_for version has been found more efficient than inner-outer cilk_for version which has been found more efficient than inner cilk_for version.

   d. For the same input size outer cilk_for versions is almost 3-5 times higher than inner-outer cilk_for versions and inner-outer cilk_for version is almost 2-3 times fater than inner cilk_for mode.

prg type	block size	run 1	run 2	run 3	run 4	run 5	avg time
parallel inner for cilk	64	0.035	0.034	0.034	0.034	0.033	0.034
	128	0.245	0.188	0.158	0.161	0.211	0.1926
	256	0.492	0.536	0.539	0.549	0.539	0.531
	512	2.658	2.64	2.631	2.657	2.651	2.6474
	1024	31.831	28.58	14.404	14.503	14.427	20.749
	2048	196.036	206.199	185.986	188.211	176.886	190.6636
	4096	1238.8	1019.08	832.621	1105.45	1134.21	1066.032
parallel outer for cilk	64	0.001	0.001	0.001	0.001	0.001	0.001
	128	0.007	0.007	0.006	0.005	0.006	0.0062
	256	0.027	0.027	0.026	0.025	0.025	0.026
	512	0.178	0.184	0.176	0.184	0.185	0.1814
	1024	1.392	1.401	1.413	1.438	1.397	1.4082
	2048	11.129	11.21	11.15	12.661	11.821	11.5942
	4096	88.701	149.76	181.21	153.45	155.65	145.7542

parallel both for cilk	64	0.005	0.004	0.004	0.005	0.004	0.0044
	128	0.022	0.022	0.027	0.022	0.028	0.0242
	256	0.098	0.108	0.105	0.102	0.099	0.1024
	512	0.617	0.625	0.635	0.623	0.623	0.6246
	1024	4.444	4.617	4.629	4.841	4.95	4.6962
	2048	36.078	35.364	36.369	71.169	51.464	46.0888
	4096	380.797	509.469	567.896	397.38	421.45	455.3984



Exercise Questions:

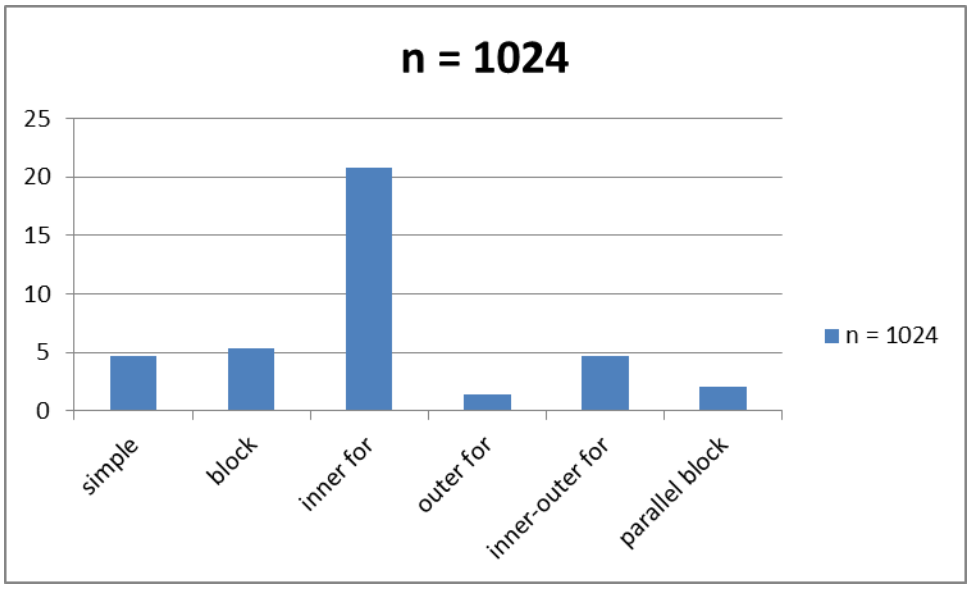
1. To compare the runtime of simple, block and cilk_for implementations the statistics obtained for n = 1024 is given the figure. It can be observed that.

   a. For inner for version, since the number of spawns are too high i.e. close to 1 million or even higher so it takes too large time to execute as compare to to inner-outer for version where not all the outer for spawns are executed together but then also they are sufficiently high to keep to introduce sufficient delay.

   b. For simple block version, since the number of operations is comparatively less than simple for loop version hence the time taken is less.

   c. For parallel block version, as the division with block size 16 causes not very high number of spawns and also reduces the work and hence the time required is less.

   d. Thus overall observation can be made as simple multiplication requires more time than block multiplication which requires more time cilk_for version which requires more time than cilkified block multiplication.



2. The runtime was calculated for 1, 2 and 4 processors. It can be observed that:

Ideally, as the number of processors are increased since parallel execution of program is done, the run time should also decrease. But as can be observed the run time value is not affected to a large extent by the no of processors.

Infact expect for outer loop case in none of cases any noticable improvement can be seen.

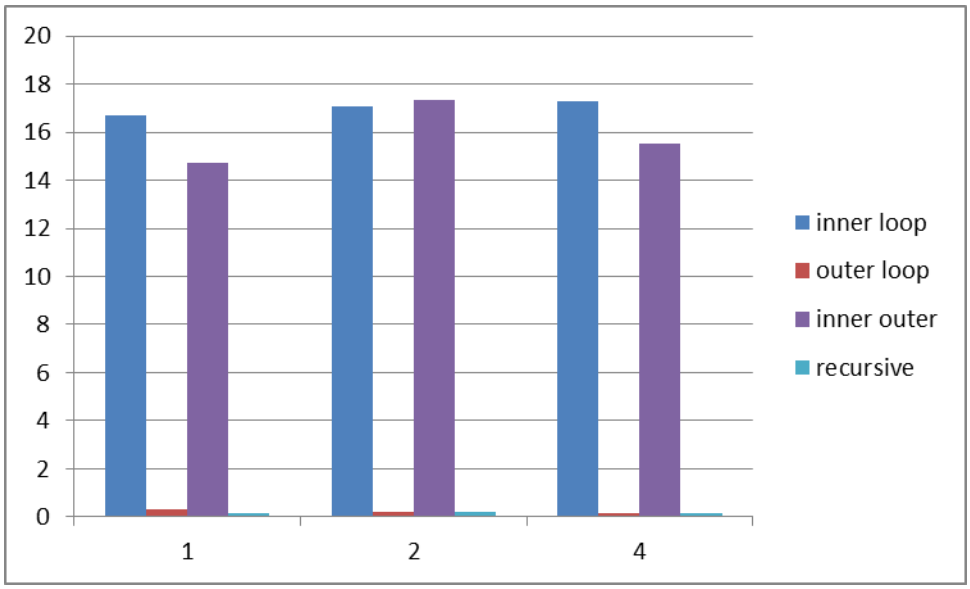
The cause of not significant improvement can be:

a. As more processors are introduced each spwan does not do significant amount of task and hence creation and deletion of each spawn itself takes sufficient time and we do not see any major improvement.

b. As more processors are working they can do the execution in parallel but the memory which they access becomes the bottleneck as it can allow serial execution and hence other processors keep on waiting for the one process.

Size = 512	No of processors	Run 1	Run 2	Run 3	Run4	Run 5	Avg time
Inner loop	1	21.637	15.974	9.235	16.583	19.937	16.6732
	2	20.685	18.548	17.254	15.475	13.432	17.0788
	4	20.592	18.876	19.094	15.959	11.856	17.2754
outer loop	1	0.327	0.281	0.281	0.281	0.266	0.2872
	2	0.234	0.219	0.218	0.218	0.188	0.2154
	4	0.187	0.171	0.172	0.172	0.156	0.1716
Inner outer	1	13.993	15.647	16.427	14.976	12.574	14.7234
	2	17.207	18.299	16.801	19.125	15.248	17.336
	4	14.633	16.895	16.364	19.843	9.922	15.5314

recursive	1	0.203	0.171	0.172	0.156	0.156	0.1716
	2	0.203	0.187	0.172	0.172	0.187	0.1842
	4	0.187	0.172	0.156	0.156	0.187	0.1716



3. All Programs were checked against Cilkscreen. No races were obtained. No Further reports were generated.

4. For each of the program, cilkview was executed and the parallelism values were obtained as

   a. Inner outer for expected = 13.93, obtained = 1293

   b. Outer for expected = 10 obtained = 78

   c. Inner for expected = 0.70 obtained = 90

   d. Block version expected = 11.74 obtained = 186

   Here, expected parallelism is obtained by taking ratio of speedup estimated for 32 processors and 2 processors.

   As per the theoretical calculation, parallelism order should be as:

   Inner-outer should be almost same as block version and block version should be greater than outer version. Inner version should have very less value.

   Practically, as for inner for version number of spawns are too high and each spawn does very little work and hence most of time should go for creation and termination and hence the parallelism should not be high. But as the system does not support large number of spawns to execute and hence this becomes equivalent to serial execution and hence parallelism obtained for outer and inner for is almost the same.

   For Block version as the block size is 16, hence each spawn has significant read operation to perform. Thus, it is more efficient.

Execution Results are:-

Method	**N**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Avg time**
Block Multiplication	64	0.016	0.001	0.001	0.001	0.001	0.004
	128	0.016	0.001	0.001	0.001	0.001	0.004
	256	0.062	0.078	0.078	0.062	0.078	0.0716
	512	0.531	0.483	0.484	0.452	0.406	0.4712
	1024	3.26	4.758	5.21	5.086	5.164	4.6956
	2048	39.7	42.978	42.698	42.712	42.978	42.2132
	4096	403.869	402.044	404.39	417.66	398.721	405.3368

Simple Multiplication	64	0.001	0.001	0.002	0.002	0.002	0.0016
	128	0.01	0.009	0.009	0.009	0.009	0.0092
	256	0.04	0.046	0.042	0.042	0.045	0.043
	512	0.308	0.309	0.314	0.309	0.312	0.3104
	1024	5.375	5.383	5.368	5.429	5.372	5.3854
	2048	46.275	47.889	45.223	46.798	47.657	46.7684
Parallel Inner Cilk	64	0.035	0.034	0.034	0.034	0.033	0.034
	128	0.245	0.188	0.158	0.161	0.211	0.1926
	256	0.492	0.536	0.539	0.549	0.539	0.531
	512	2.658	2.64	2.631	2.657	2.651	2.6474
	1024	31.831	28.58	14.404	14.503	14.427	20.749
	2048	196.036	206.199	185.986	188.211	176.886	190.6636
	4096	1238.8	1019.08	832.621	1105.45	1134.21	1066.032
Parallel Outer Cilk	64	0.001	0.001	0.001	0.001	0.001	0.001
	128	0.007	0.007	0.006	0.005	0.006	0.0062
	256	0.027	0.027	0.026	0.025	0.025	0.026
	512	0.178	0.184	0.176	0.184	0.185	0.1814
	1024	1.392	1.401	1.413	1.438	1.397	1.4082
	2048	11.129	11.21	11.15	12.661	11.821	11.5942
	4096	88.701	149.76	181.21	153.45	155.65	145.7542
Parallel Inner outer Cilk	64	0.005	0.004	0.004	0.005	0.004	0.0044
	128	0.022	0.022	0.027	0.022	0.028	0.0242
	256	0.098	0.108	0.105	0.102	0.099	0.1024
	512	0.617	0.625	0.635	0.623	0.623	0.6246
	1024	4.444	4.617	4.629	4.841	4.95	4.6962
	2048	36.078	35.364	36.369	71.169	51.464	46.0888
	4096	380.797	509.469	567.896	397.38	421.45	455.3984
Parallel Block Cilk	64	0.001	0.001	0.001	0.001	0.015	0.0038
	128	0.001	0.001	0.001	0.001	0.015	0.0038
	256	0.032	0.031	0.047	0.031	0.031	0.0344
	512	0.156	0.172	0.156	0.156	0.156	0.1592
	1024	2.044	2.043	2.06	2.043	2.044	2.0468
	2048	36.078	35.364	36.369	71.169	51.464	46.0888
	4096	74.61	74.30	74.89	83.21	85.46	78.494