Report Lab1

4. Optimization:

From the situation, we know that the heat calculation will repeat m times for a m iteration work from the edge of (n-1)*(n-1) matrix to the inside, and the center area(n-1-m)*(n-1-m) would remain to be zero and no need to repeat calculation. Thus, under such principle, to use the memory effectively, I optimized my program by avoid repeat calculating the center zero area based on the matrix multiplication.

6. Execute programs (iterations=50)

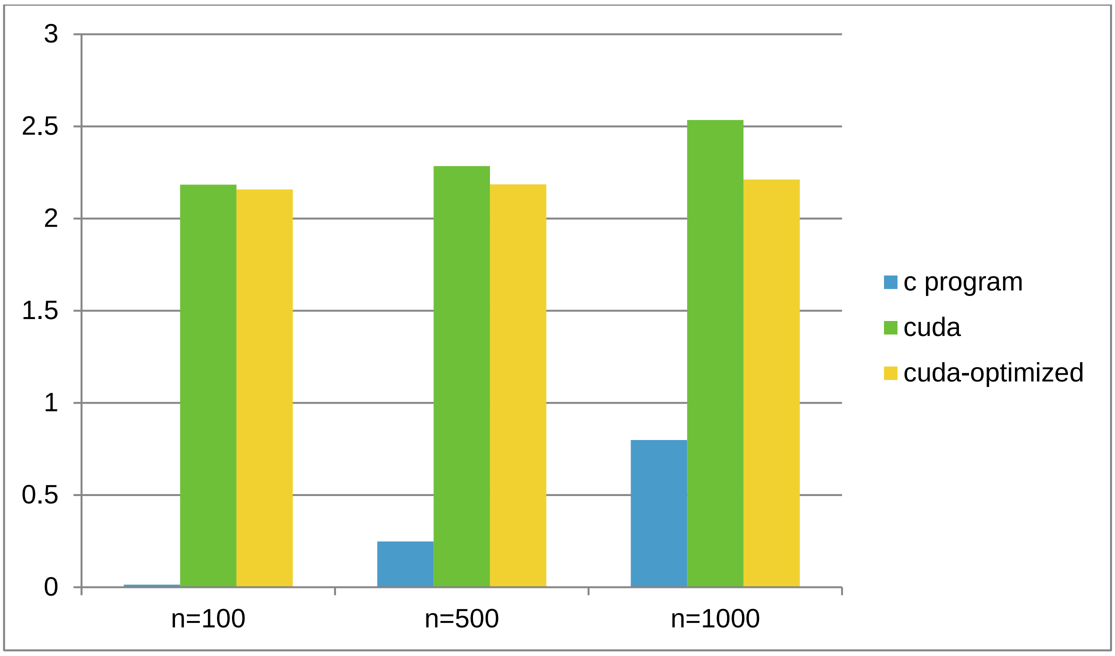
c program	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	0.016	0.014	0.014	0.013	0.013	0.014
n=500	0.250	0.248	0.248	0.247	0.247	0.248
n=1000	0.878	0.909	0.763	0.713	0.725	0.7976
Segmentation Fault						
n=10,000						
n=100,000						
n=1,000,000						

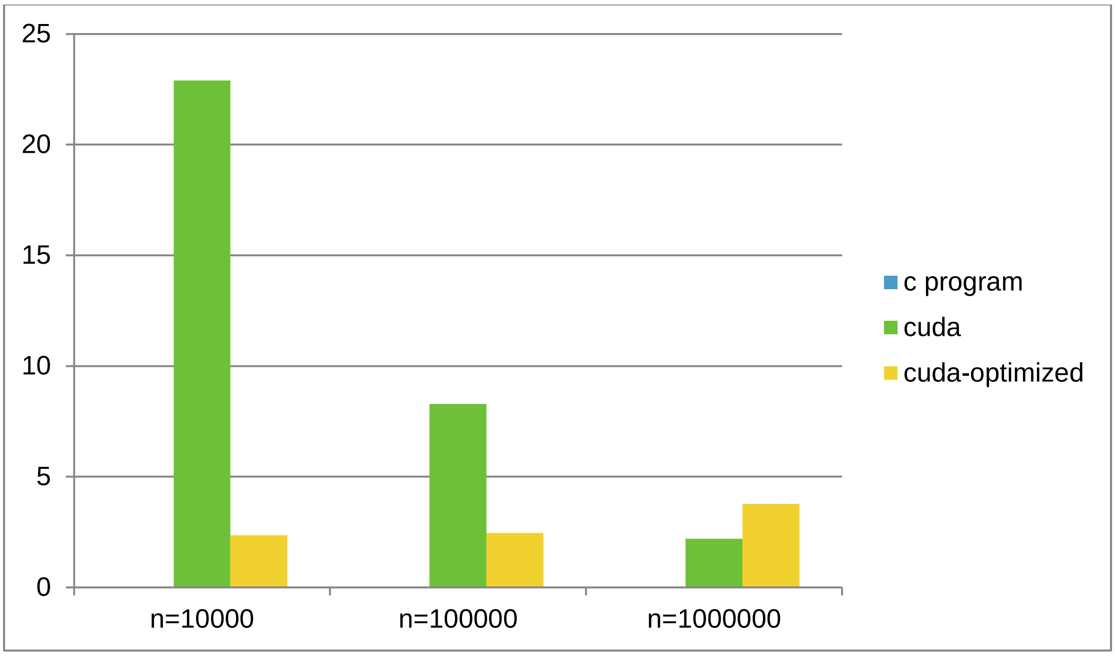
cuda	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	2.211	2.191	2.165	2.172	2.177	2.1832
n=500	2.292	2.286	2.276	2.270	2.295	2.2838
n=1,000	2.609	2.521	2.471	2.586	2.487	2.5348
n=10,000	24.916	22.717	22.178	22.354	22.357	22.9044
Error Value						
n=100,000	8.430	8.091	7.778	8.052	8.142	8.2986(??)
n=1,000,000	2.194	2.231	2.199	2.178	2.171	2.1946(??)

cuda-optimized	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	2.272	2.196	2.129	2.016	2.173	2.1572
n=500	2.287	2.121	2.174	2.150	2.195	2.1854
n=1000	2.223	2.183	2.280	2.187	2.182	2.211

n=10,000	2.476	2.353	2.331	2.309	2.333	2.3604
n=100,000	2.426	2.635	2.348	2.413	2.459	2.4562
n=1,000,000	3.950	3.743	3.811	3.764	3.621	3.7778

7.





8. a) GPU is more beneficial when n>=1000. Because GPU's multi-core architecture is very suitable for doing the same operation on large number of data sets.

b) CPU processes data linearly, so the speedup at its lowest for CPU version should be when n is the largest number. The speedup at its lowest for GPU version is when n is smallest version. Because GPU is not for complex operation on small number of data, it performs well when n is as large as possible.

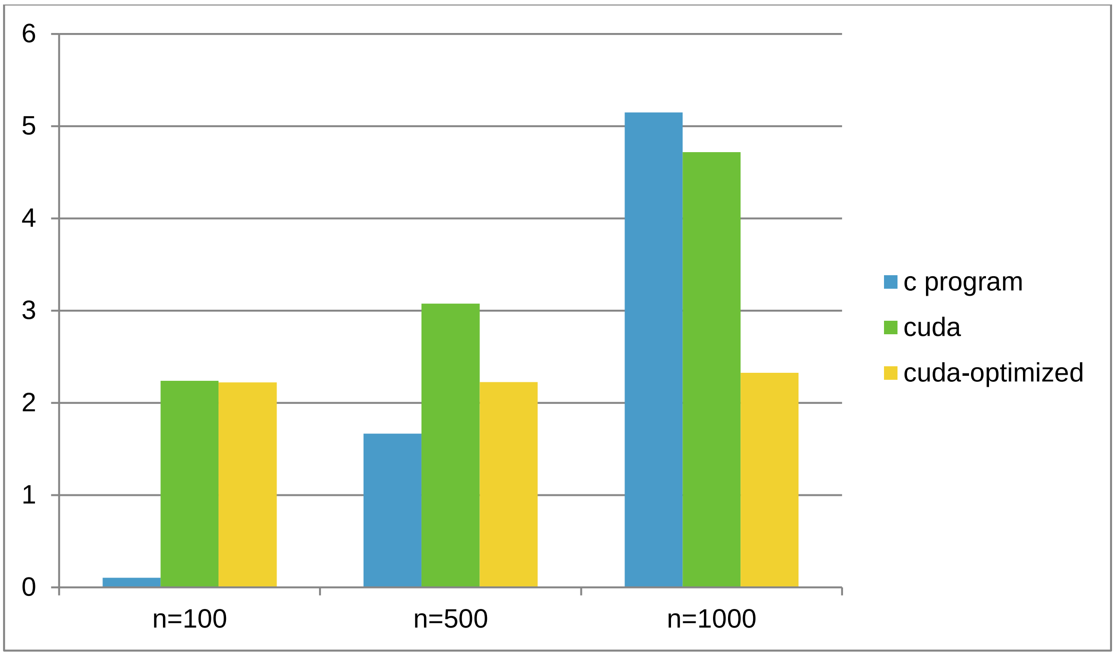
c) Due to the same reason mentioned in b), for CPU version, the speedup at its highest is when n is the smallest number. The speedup at its highest for GPU version is on the opposite side, when n is a large number.

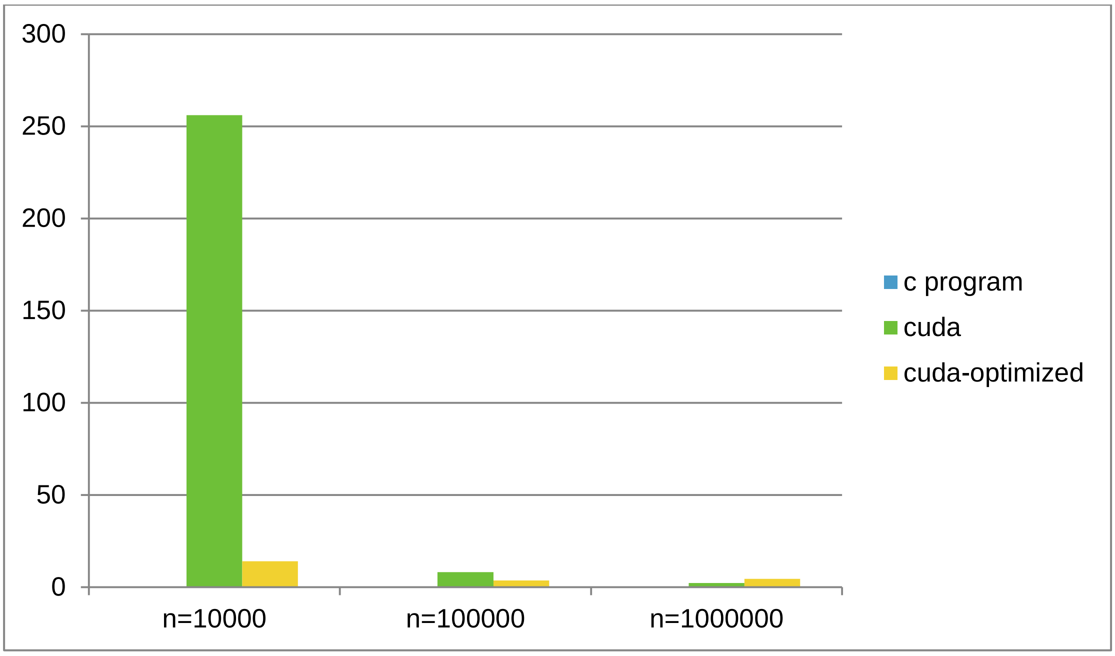
9. Execute programs (iterations=500)

c program	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	0.113	0.102	0.103	0.102	0.094	0.1028
n=500	1.539	1.952	1.625	1.645	1.561	1.6644
n=1000	5.258	5.120	5.035	5.101	5.234	5.1496
Segmentation Fault						
n=10,000						
n=100,000						
n=1,000,000						

cuda	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	2.287	2.219	2.239	2.241	2.211	2.2394
n=500	2.950	3.318	3.031	3.110	2.971	3.076
n=1,000	4.342	5.574	4.745	4.232	4.695	4.7176
n=10,000	241.632	195.033	284.626	293.782	265.414	256.0974
Error Value						
n=100,000	8.167	7.995	8.236	8.145	8.128	8.1342
n=1,000,000	2.270	2.183	2.186	2.206	2.162	2.2014

cuda-optimized	1st time(s)	2nd time(s)	3rd time(s)	4th time(s)	5th time(s)	average(s)
n=100	2.318	2.194	2.200	2.199	2.196	2.2214
n=500	2.272	2.230	2.201	2.210	2.219	2.2264
n=1000	2.448	2.300	2.316	2.324	2.236	2.3248
n=10,000	14.042	14.316	13.951	13.572	14.306	14.0374
Error Value						
n=100,000	3.754	3.565	3.570	3.755	3.475	3.6238
n=1,000,000	4.534	3.973	4.150	4.200	3.817	4.5348





a) GPU is more beneficial when n>=1000. Because GPU’s multi-core architecture is very suitable for doing the same operation on large number of data sets.

b) CPU processes data linearly, so the speedup at its lowest for CPU version should be when n is the largest number. The speedup at its lowest for GPU version is when n is smallest version. Because GPU is not for complex operation on small number of data, it performs well when n is as large as possible.

c) Due to the same reason mentioned in b), for CPU version, the speedup at its highest is when n is the smallest number. The speedup at its highest for GPU version is on the opposite side, when n is a large number.

10. When increasing the number of iterations, the computation time costs more, but GPU does better than CPU when the iteration time is large enough. Because GPU’s multi-core architecture is very suitable for the same instruction stream sent to multi-core parallelly.