CS 426 Parallel Computing

Project 4 Report

Alp Ege Baştürk

21501267

02.06.2019

Parallelization Strategy

Matrix was processed in rows. Initially $\frac{Row\ Count}{Thread\ Number}$ blocks were created. In each block, specified number of threads run in parallel, each multiplying one row of the matrix with the vector. Each threads finds its position in the matrix using information of the block and thread Ids. Division to blocks was to try handling cases where row count is greater than the max number of threads that can be run on the device.

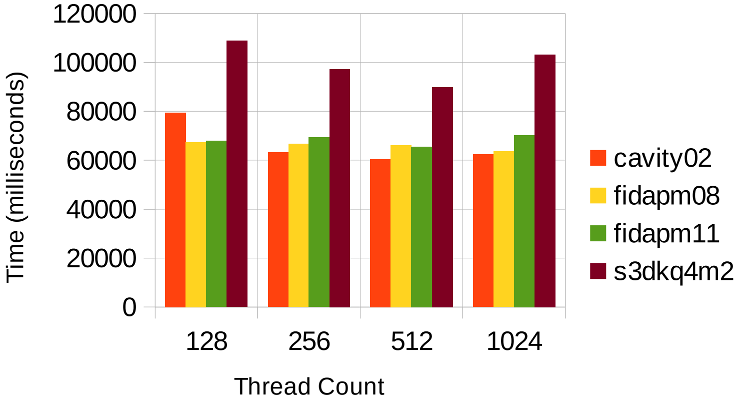
Experiments & Results

Experiments were run on a Google Cloud VM with single CPU core, 3.5 GB main memory and Tesla K80. Measurements are given in milliseconds. S3dkq4m2 matrix was found on the given matrix market [1], which is larger than the other matrices. A larger dataset was chosen since results on K80 were not separable enough to discuss clearly. Main code also runs the serial part for measurements, different parts of the code were measured separately.

Following are the results on all 4 matrices grouped by thread counts.

Parallel Time Real

Thread Count	cavity02	fidapm08	fidapm11	s3dkq4m2
128	79516	67404	67875	108833
256	63237	66665	69264	97269
512	60524	66161	65521	89890
1024	62356	63618	70283	103124

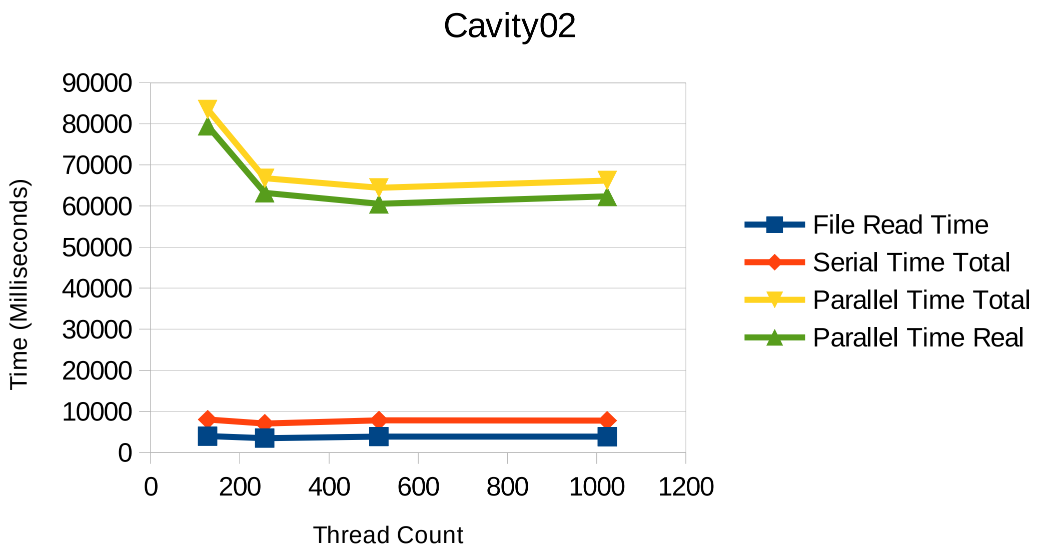


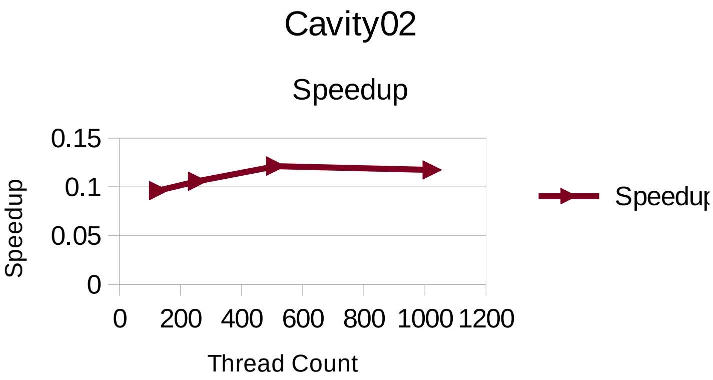
Following are the tables derived from the raw data from shell script.

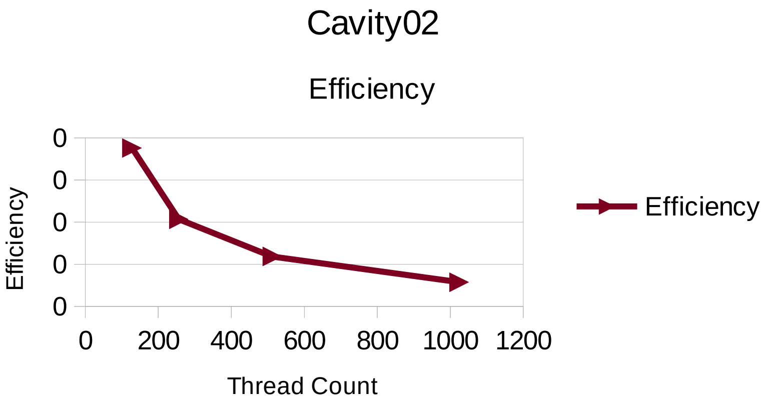
Cavity02

cavity02

Thread Count	File Read Time	Serial Time Total	Parallel Time Total	Parallel Time Real	Speedup	Efficiency
128	3982	8031	83498	79516	0.0961819445	0.000751421441232
256	3502	7055	66739	63237	0.1057103043	0.000412930876249
512	3878	7805	64402	60524	0.1211918885	0.00023670290713
1024	3859	7768	66215	62356	0.1173148078	0.000114565242015



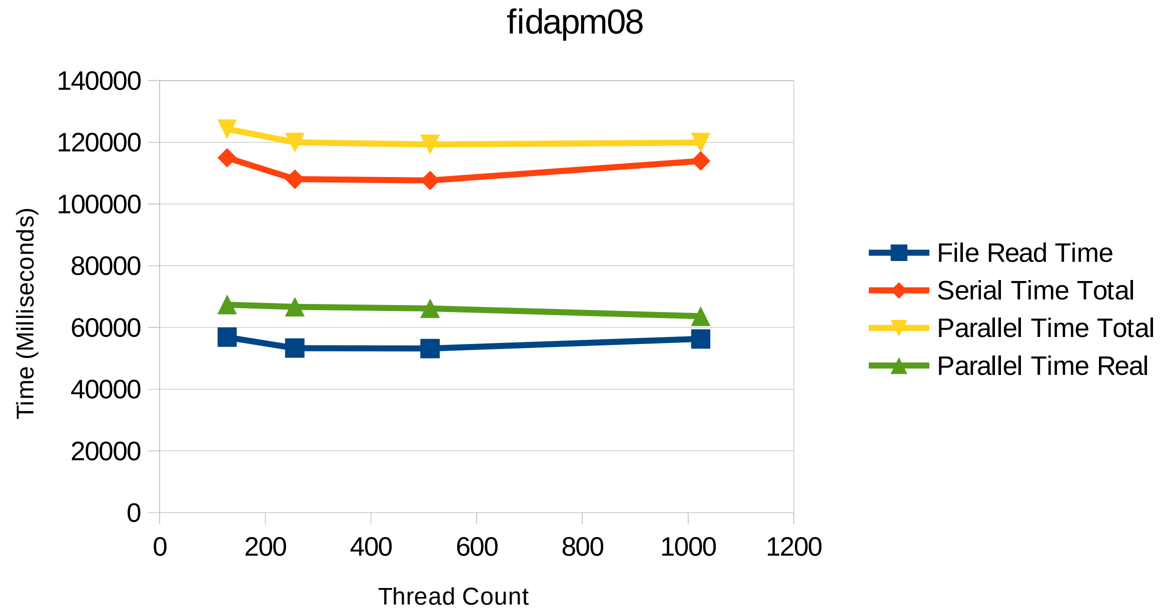


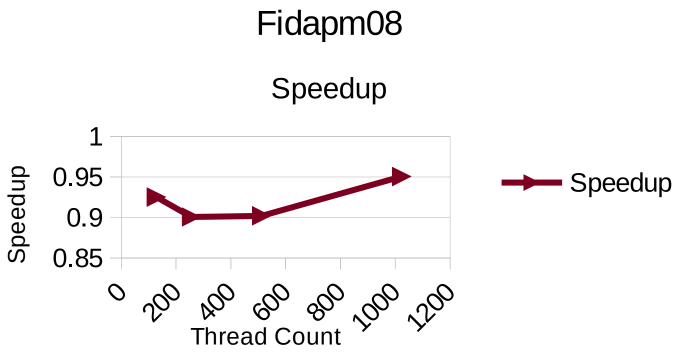


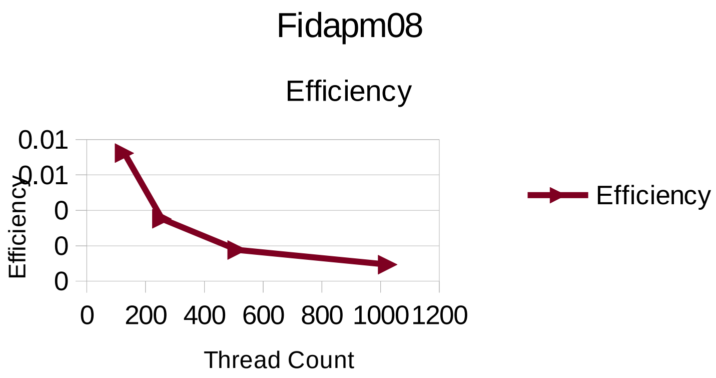
Fidapm08

fidapm08

Thread Count	File Read Time	Serial Time Total	Parallel Time Total	Parallel Time Real	Speedup	Efficiency
128	56861	114961	124265	67404	0.9251277512	0.00722756055607
256	53304	108035	119969	66665	0.9005243021	0.003517673055123
512	53178	107629	119339	66161	0.9018761679	0.001761476890413
1024	56286	113973	119904	63618	0.9505354283	0.000928257254241



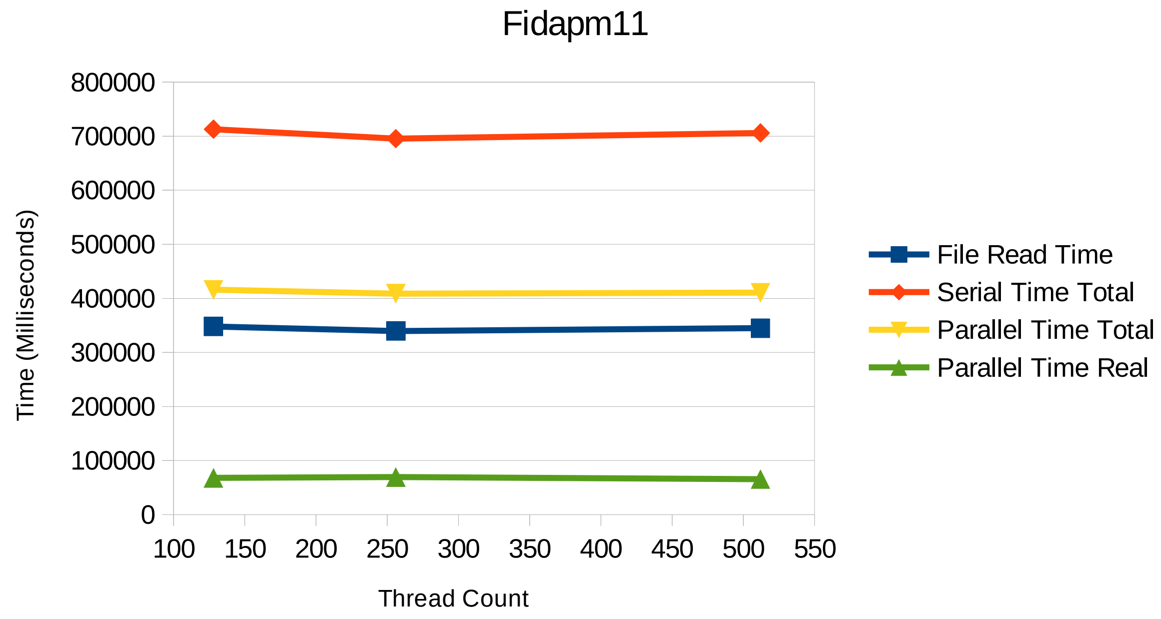


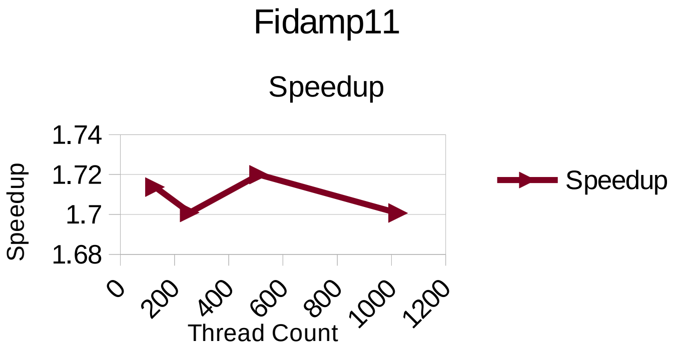


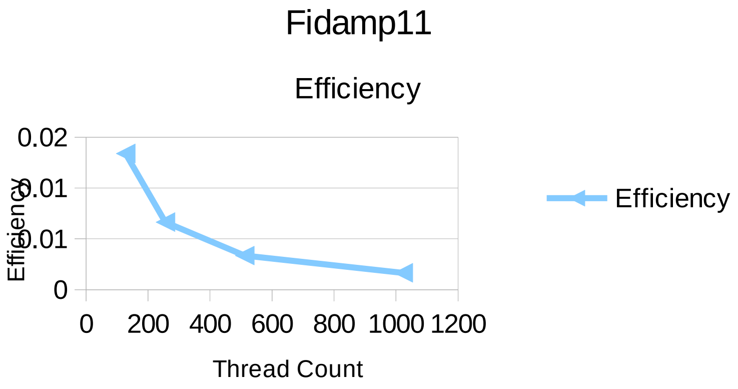
Fidapm11

fidapm11

Thread Count	File Read Time	Serial Time Total	Parallel Time Total	Parallel Time Real	Speedup	Efficiency
128	348101	712885	415976	67875	1.7137647364	0.013388787003337
256	339568	695421	408832	69264	1.7009945405	0.006644509924003
512	344925	705940	410446	65521	1.7199339255	0.00335924594831
1024	344197	704933	414480	70283	1.7007648137	0.00166090313842

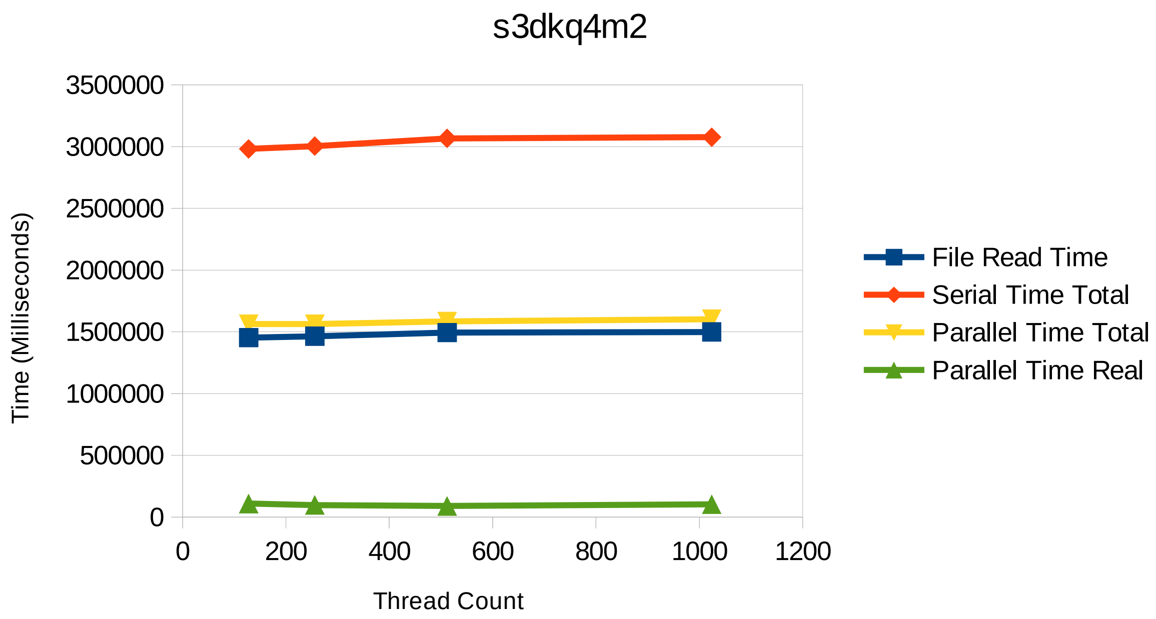


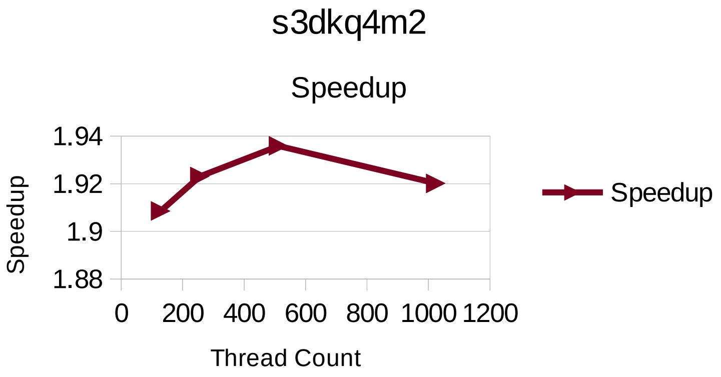


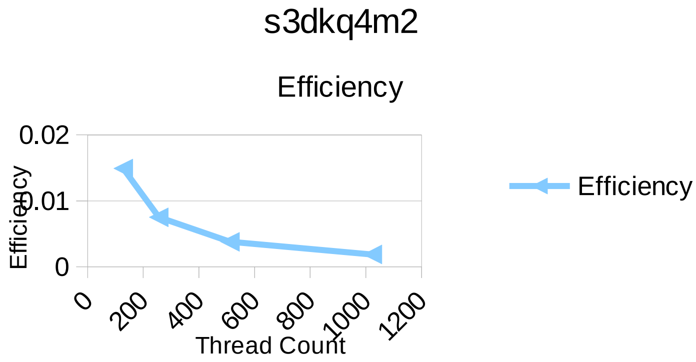


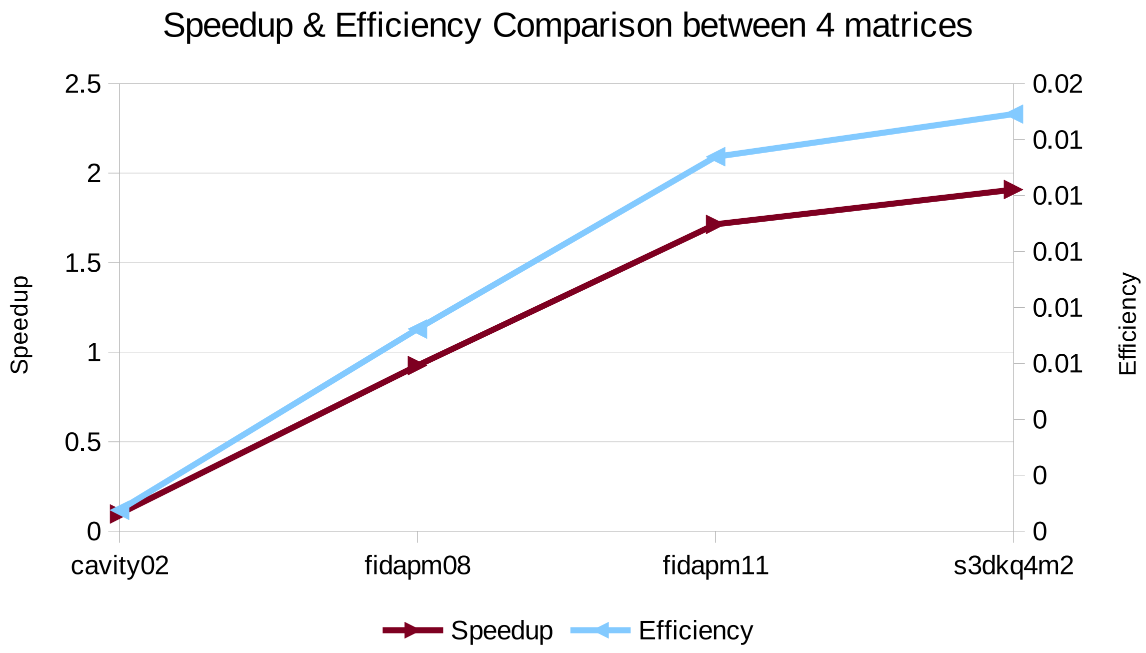
S3dkq3m2

Thread Count	File Read Time	Serial Time Total	Parallel Time Total	Parallel Time Real	Speedup	Efficiency
128	1453232	2981266	1562065	108833	1.908541578	0.014910481077932
256	1464769	3003940	1562038	97269	1.9230902193	0.007512071169203
512	1494053	3066340	1583943	89890	1.9358903698	0.003781035878501
1024	1499078	3076474	1602202	103124	1.9201536386	0.001875150037651









Discussion

As it can be seen on the column chart on the first page, real parallel time decreases when more threads are used. However it starts to increase for 1024 threads. This might be due to low data amount compared to number of threads. In this case more threads doesn't provide benefit the overcome additional costs.

Considering the serial-parallel comparison, it can be seen that parallel performance is better for matrices other than the cavity02. This might be due to matrix sizes. In the 2nd and 3rd matrices' parallel time improves slightly with the number of threads, the small change might be due to matrix size. Results of the 4th matrix shows a better improvement up to 1024 threads.

Speedup and Efficiency graphs show similar patterns to the time graphs. Efficiency drops with more threads used on same matrix, but efficiency among different matrices increase as data size increases as it can be seen from the last graph names Speedup & Efficiency Comparison between 4 matrices. This shows that using GPU is efficient for large data. Speedup also shows a similar pattern.

References

[1] “S3DKQ4M2: Finite element analysis of cylindrical shells Cylindrical shell, uniform 150x100 quadrilateral mesh, R/t=1000”. https://math.nist.gov/MatrixMarket/data/misc/cylshell/s3dkq4m2.html