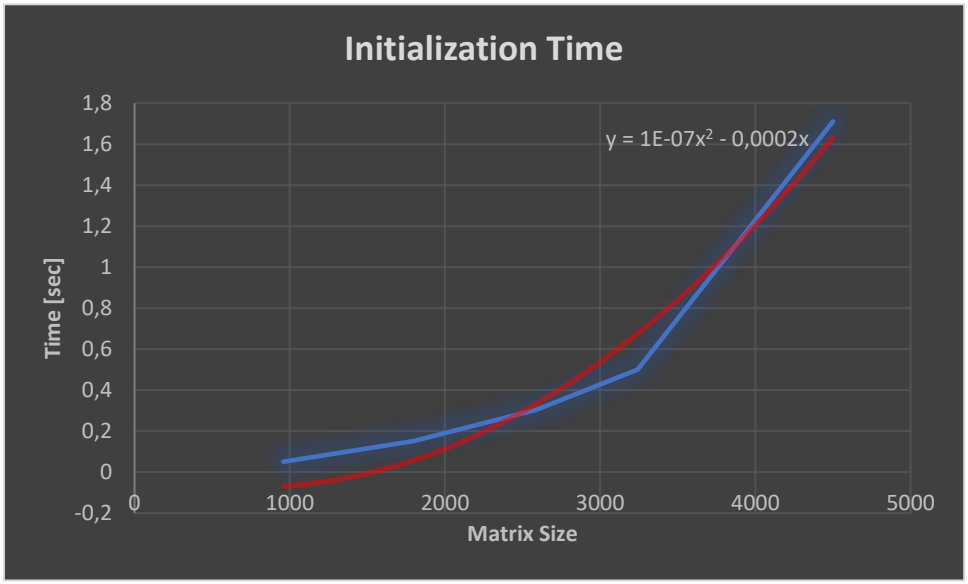
Distributed Systems

Practical Assignment 01

Marcel Zauder
16-124-836
User_04

3.1 Computation of Sequential References Times

3.2.1 Initialization Time

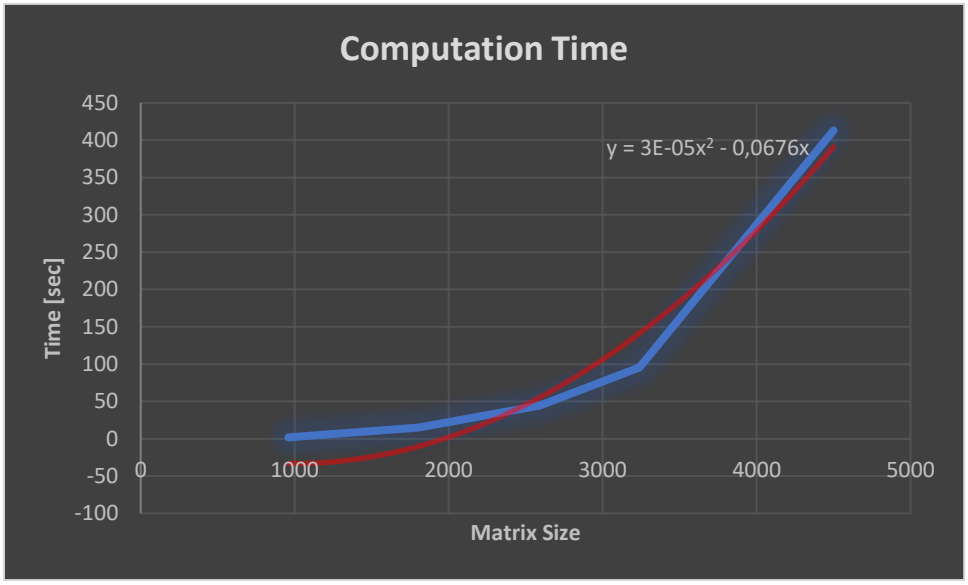


Matrix Size	Initialization Time [sec]
960	0,0512
1800	0,1524
2592	0,3056
3240	0,5004
4500	1,7108

3.2.2 Sending Time

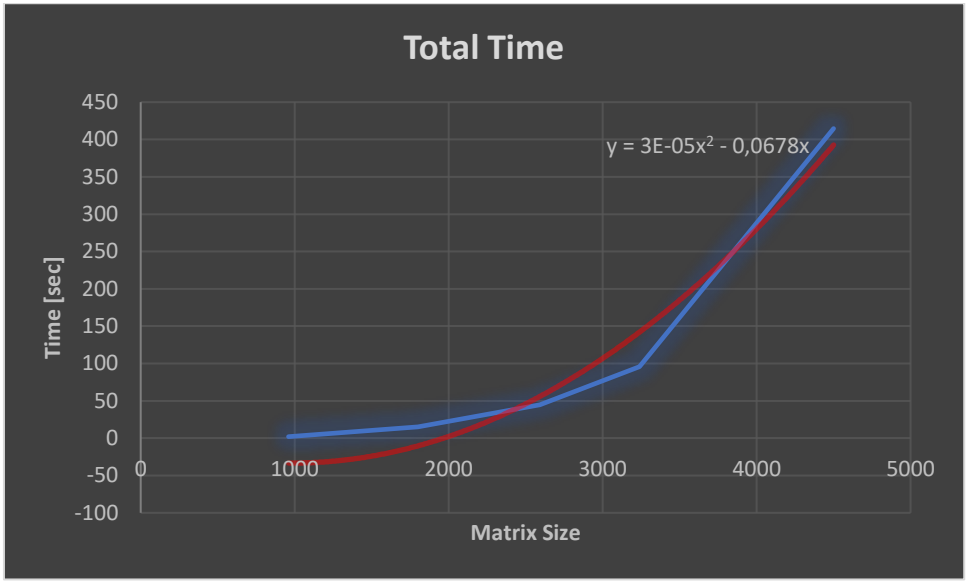
There is no sending in the sequential procedure.

3.2.3 Computation Time



Matrix Size	Computation Time [sec]
960	1,9536
1800	14,824
2592	44,4602
3240	95,3732
4500	412,9508

3.2.4 Total Time



Matrix Size	Total Time [sec]
960	2,0048
1800	14,9764
2592	44,7658
3240	95,8736
4500	414,6616

3.2 Computation of Parallel Times

3.2.1 Tables

i. 4 Workers

Matrix Size	Initialization Time	Sending Time	Computation Time	Total Time
1260	0,9044	2,117	0,7096	3,731
1728	0,9258	3,3672	1,6556	5,9486
2520	1,0348	6,2454	4,7676	12,0478
4140	0,9828	15,2366	17,433	33,6524
5040	0,9836	21,0324	30,8148	52,8308

ii. 12 Workers

Matrix Size	Initialization Time	Sending Time	Computation Time	Total Time
1260	0,96	2,3982	0,3472	3,7054
1728	0,9906	3,7224	0,5726	5,2856
2520	1,0596	7,5266	1,375	9,9612
4140	1,137	22,0286	6,031	29,1966
5040	1,0064	36,1374	11,463	48,6068

iii. 20 Workers

Matrix Size	Initialization Time	Sending Time	Computation Time	Total Time
1260	1,0746	2,9	0,4156	4,3902
1728	1,1022	4,4074	0,5536	6,0632
2520	1,1694	8,2476	1,0502	10,4672
4140	1,2338	23,013	3,508	27,7548
5040	1,124	41,1854	6,584	48,8934

iv. 28 Workers

Matrix Size	Initialization Time	Sending Time	Computation Time	Total Time
1260	1,1558	3,4254	0,4944	5,0756
1728	1,1356	4,7834	0,6216	6,5406
2520	1,2372	8,7378	1,0696	11,0446
4140	1,3608	26,7766	2,8848	31,0222
5040	1,1138	40,0966	4,9106	46,121

v. 32 Workers

Matrix Size	Initialization Time	Sending Time	Computation Time	Total Time
1260	1,129	3,7532	0,5434	5,4256
1728	1,1398	5,157	0,6818	6,9786
2520	1,2244	9,2518	1,0472	11,5234
4140	1,3154	28,3296	3,9762	33,6212
5040	1,1632	41,6344	5,6076	48,4052

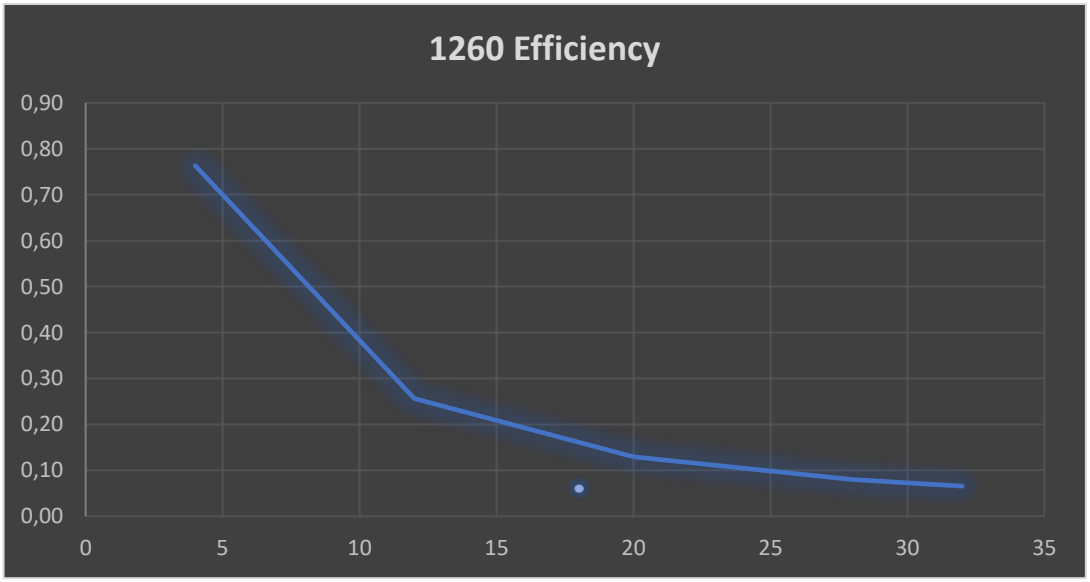
3.2.2 Plots

i. Matrix Size 1260

a. SpeedUp

We can see that the SpeedUp in the beginning is approximately constant but when using more workers it is lessened. This is because the sending time increases the more workers are used which is negatively influencing the total time for the computation as well as the speedup.
Therefore when the matrix size is small it is better to use few workers, because otherwise the communication between the workers takes to much time in comparison to the actual computation time.

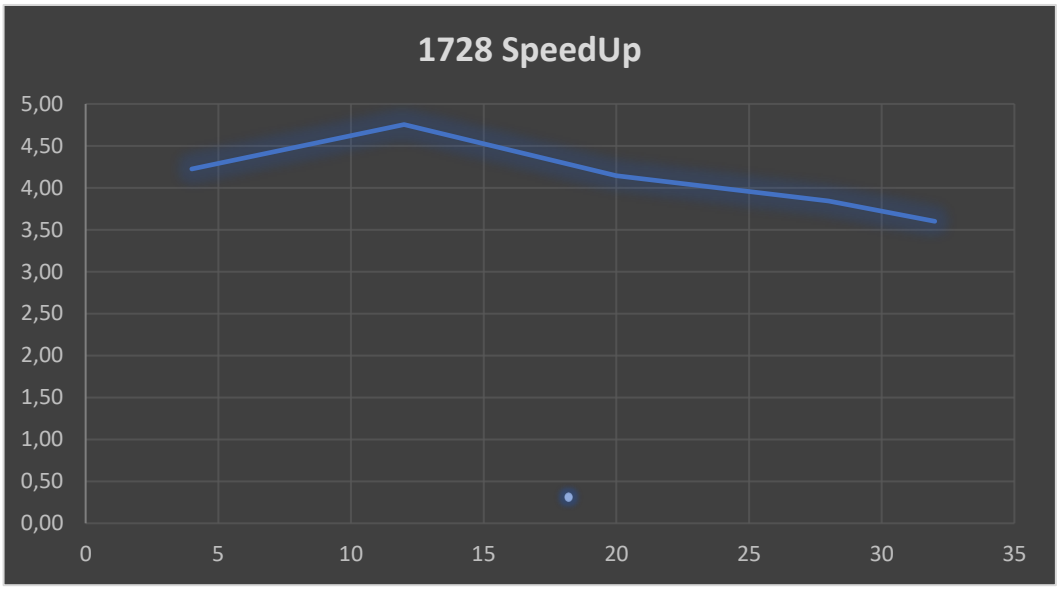
b. Efficiency



When looking at the efficiency plot it is obvious that less worker are more efficient than when using a lot of workers. This is mainly because when more workers are working on the same task they have to synchronize and eventually wait for each other, therefore if the speedup is not significantly increased the more workers are added the less efficient one worker would be.
In this example even when using 4 workers, one worker would not reach the efficiency of one worker.

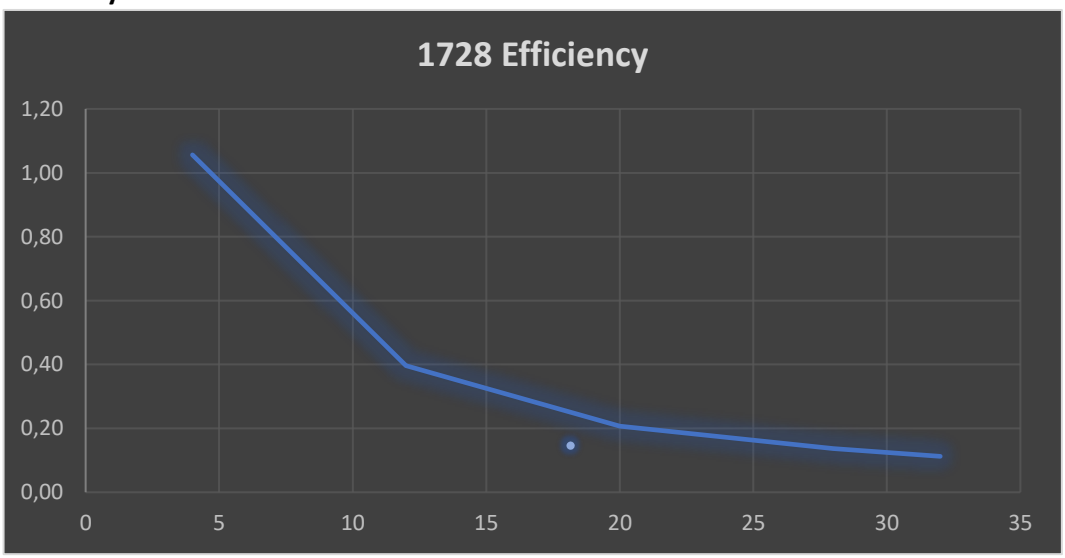
ii. Matrix Size 1728

a. SpeedUp



We can see that the SpeedUp in the beginning is increasing but when using more workers it is lessened. This is because the sending time increases the more workers are used which is negatively influencing the total time for the computation as well as the speedup.
Therefore for this matrix size it is faster when using 12 workers. When using 20 workers the communication takes too much time, but when using 4 workers each of them still needs to do plenty of work.

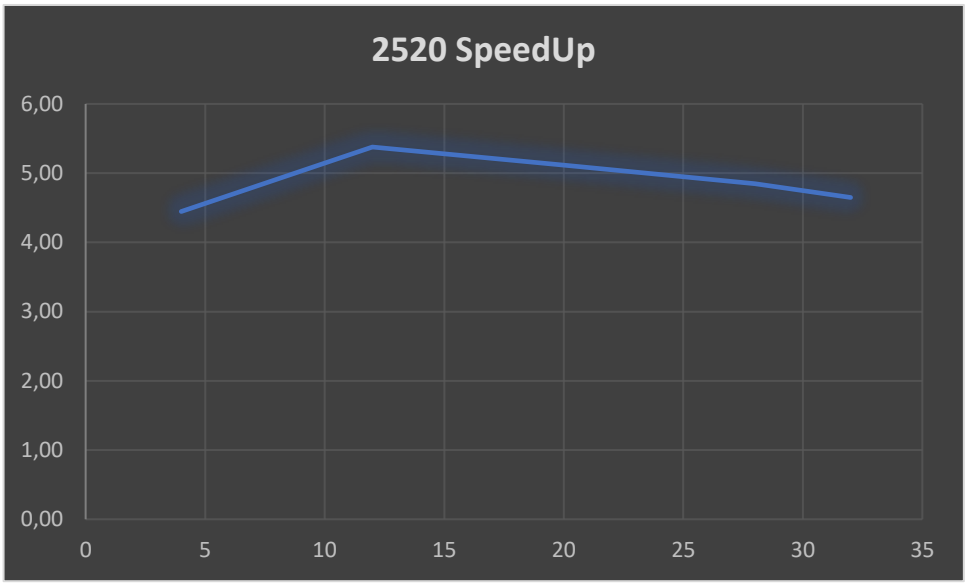
b. Efficiency



When looking at the efficiency plot it is obvious that less worker are more efficient than when using a lot of workers. This is mainly because when more workers are working on the same task they have to synchronize and eventually wait for each other, therefore if the speedup is not significantly increased the more workers are added the less efficient one worker would be.
In this example when using 4 workers, one of them is nearly as efficient as when using one worker.

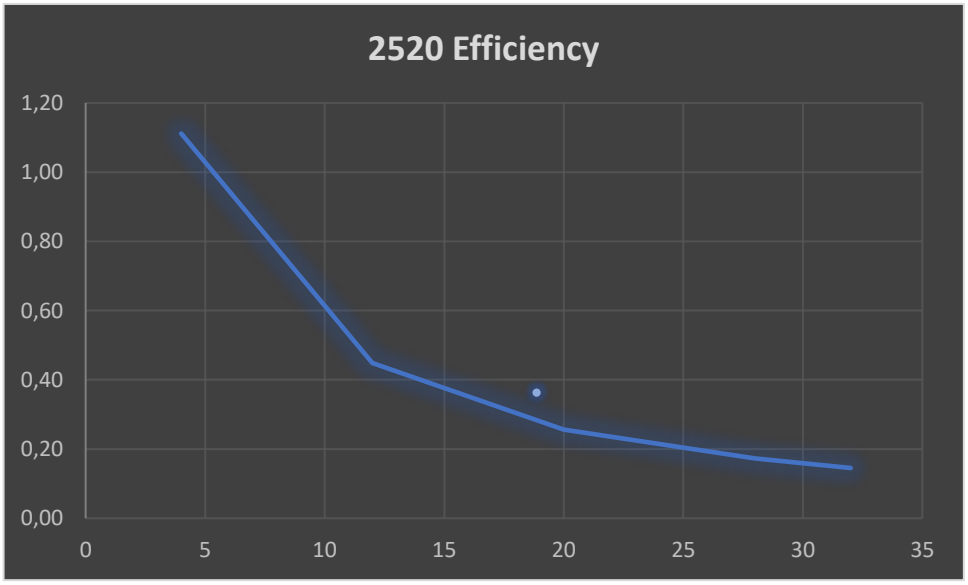
iii. Matrix Size 2520

a. SpeedUp



We can see that the SpeedUp in the beginning is increasing but when using more workers it is lessened. This is because the sending time increases the more workers are used which is negatively influencing the total time for the computation as well as the speedup.
Therefore for this matrix size it is faster when using 12 workers. When using 20 workers the communication takes too much time, but when using 4 workers each of them still needs to do plenty of work.

b. Efficiency



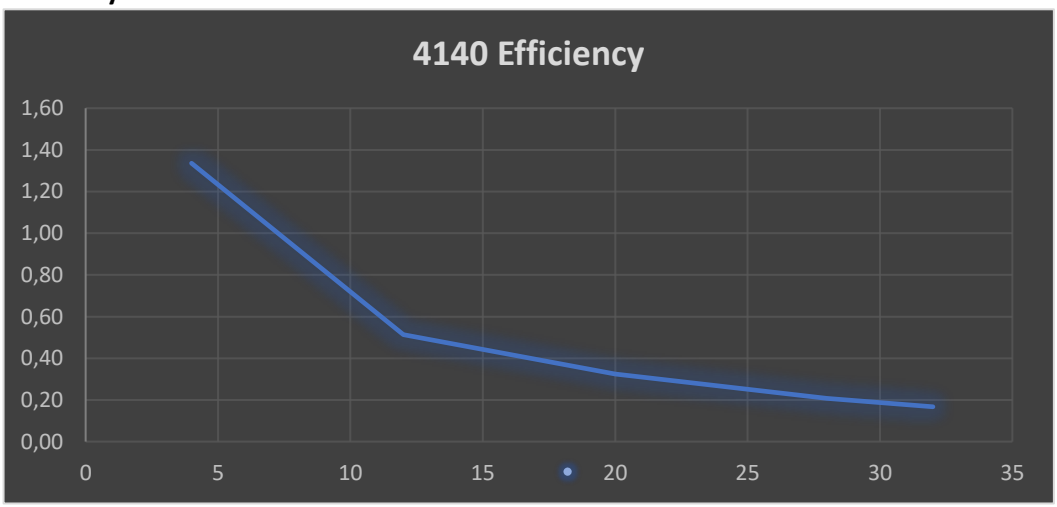
When looking at the efficiency plot it is obvious that less worker are more efficient than when using a lot of workers. This is mainly because when more workers are working on the same task they have to synchronize and eventually wait for each other, therefore if the speedup is not significantly increased the more workers are added the less efficient one worker would be.
In this example when using 4 workers, one of them is sligtly more efficient as when using one worker.

iv. Matrix Size 4140

a. SpeedUp

For the matrix size of 4140 it is faster when using 20 workers because the sending time is not so significant but for less workers there is still a huge amount of work to do. When using more workers the sending time is too significatn tob e efficient and the speedup decreases.
Therefore for this matrix size 20 workers are optimal because a size 4140 leads to a huge amount of work.

b. Efficiency



When looking at the efficiency plot it is obvious that less worker are more efficient than when using a lot of workers. This is mainly because when more workers are working on the same task they have to synchronize and eventually wait for each other, therefore if the speedup is not significantly increased the more workers are added the less efficient one worker would be.
In this example when using 4 workers, one of them is significantly more efficient as when using one worker.

v. **Matrix Size 5040**

a. **SpeedUp**

For the matrix size of 5040 it is faster to use 28 workers because it is a lot of work to compute the result. When using 32 workers the sending time gets too significant so that the speedup decreases.
Therefore for a huge amount of work it is better to have more workers because the sending time does not affect the speedup as much anymore.
The decrease of the speedup when using 20 workers could be due to some delay while sending between the workers.

b. **Efficiency**

When looking at the efficiency plot it is obvious that less worker are more efficient than when using a lot of workers. This is mainly because when more workers are working on the same task they have to synchronize and eventually wait for each other, therefore if the speedup is not significantly increased the more workers are added the less efficient one worker would be.
In this example when using 4 workers, one of them is nearly as efficient as one and a half workers.