TU Wien

Data Science

Basic of parallel computing

Exercise 1

Mal Kurteshi 11924480

May 5, 2021

1 Introduction

In this report i am presenting my solution regarding the first assignment in the course *191.114 Basics of Parallel Computing*.

2 Tasks

2.1 Parallelize the Computation of the Julia Set

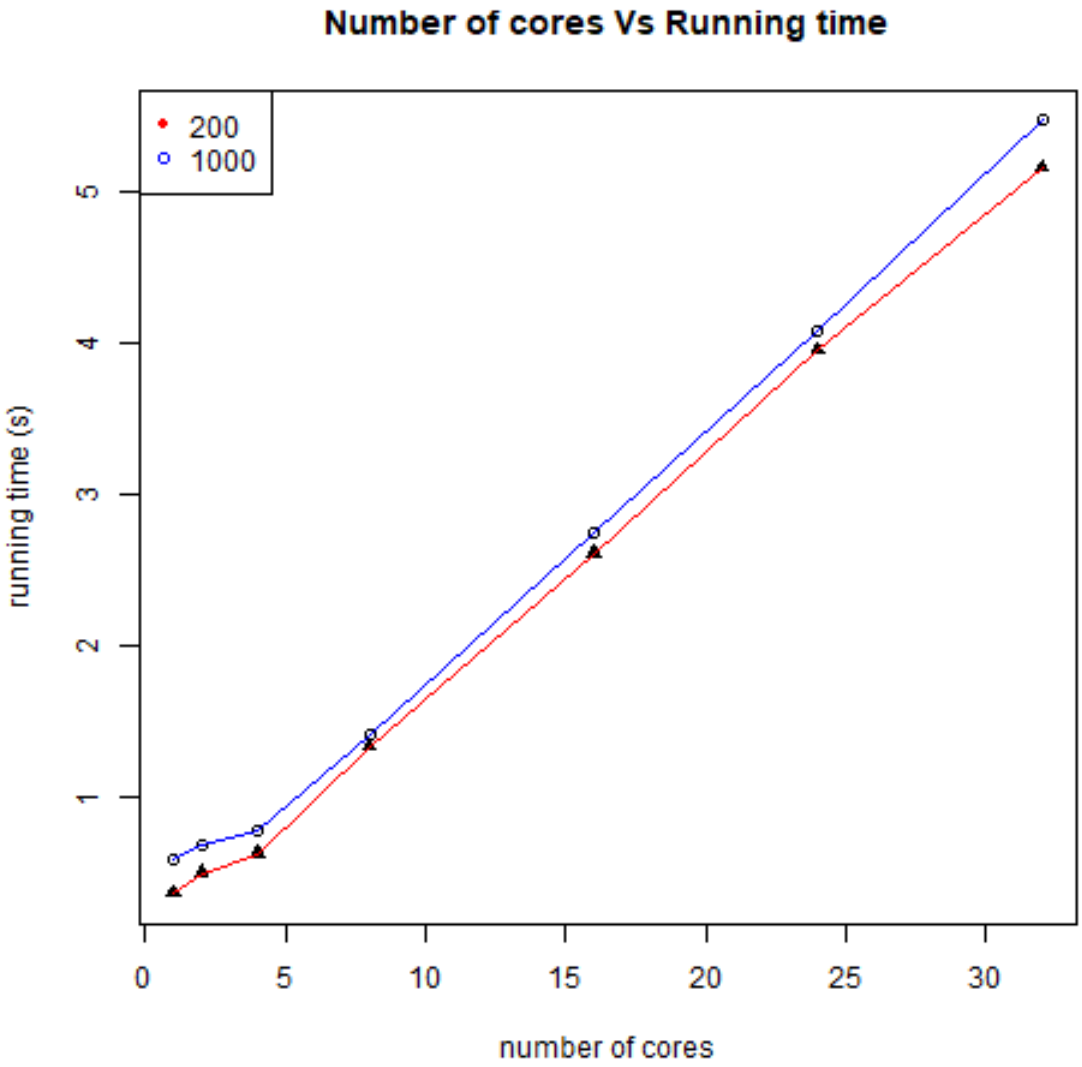
Solution under this task are under julia_par.py python script, in which is provided the parallelism task for the julia set visualisation using python.

2.2 Compute Speed-up and Parallel Efficiency for 2 Instance Sizes

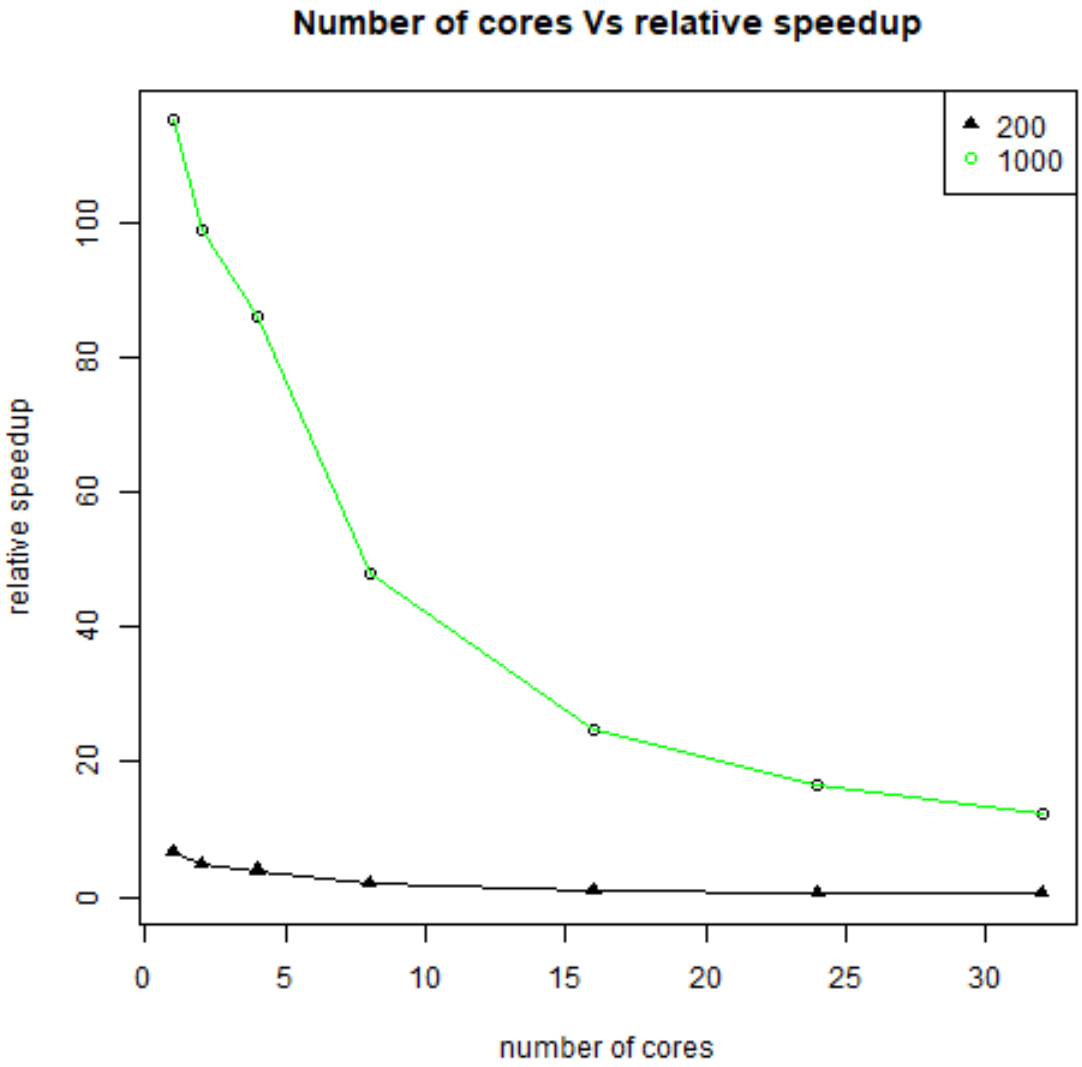
In the table below we have the results from the experiment where we are analysing the scalability of the parallel in which we based on the table we calculated the speed-up and the parallel efficiency based on the formula from the course slides.

Table 1: Measurements for task 2.2

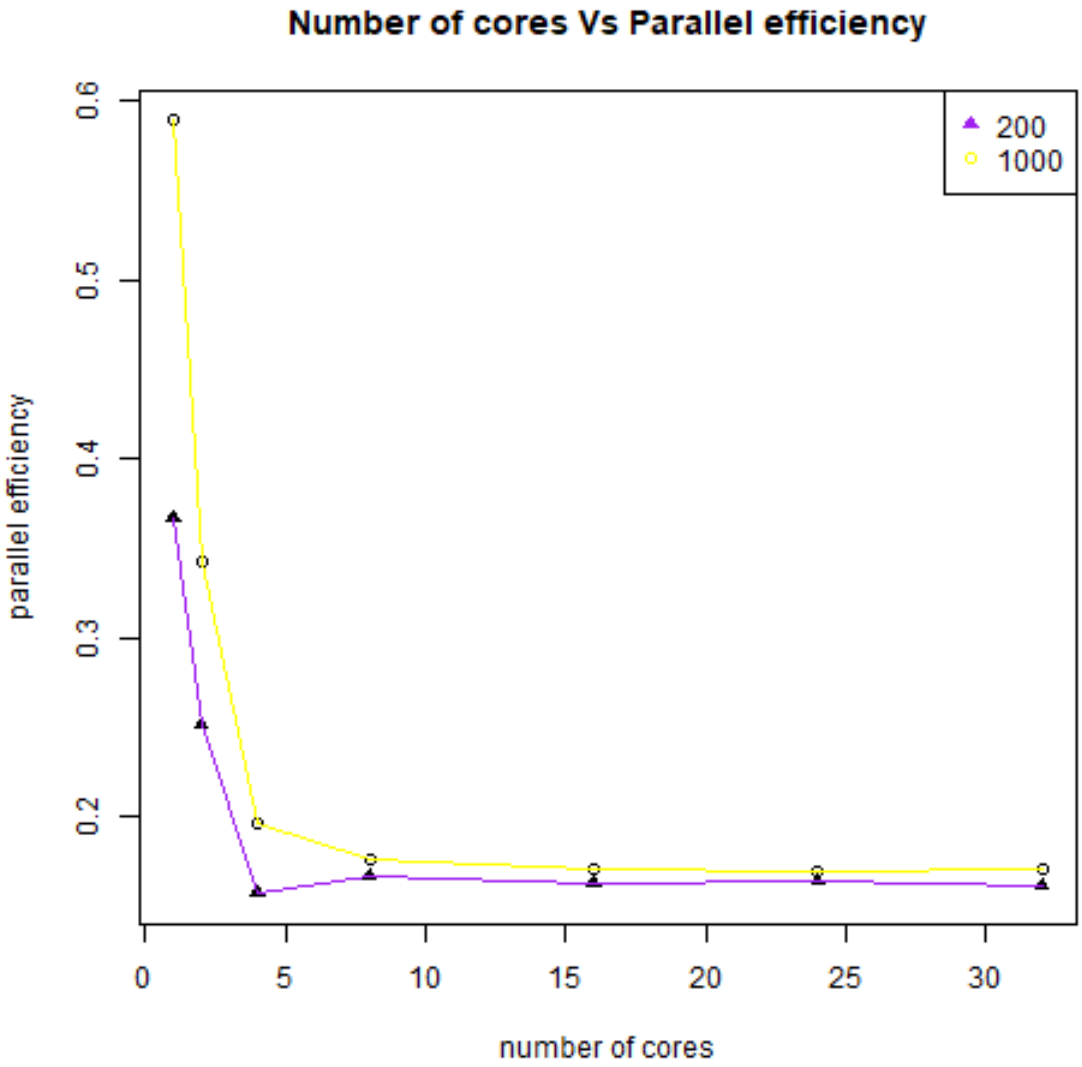
size	**p**	**mean runtime(s)**	**speed-up**	**par.eff**
200	1	0.366925567	6.706745882	0.366925567
200	2	0.502540933	4.896867837	0.251270467
200	4	0.6300252	3.905996988	0.1575063
200	8	1.3370399	1.840540835	0.167129988
200	16	2.613323533	0.94166547	0.163332721
200	24	3.948258967	0.62328144	0.16451079
200	32	5.160963533	0.476825019	0.16128011
1000	1	0.588887667	115.2358136	0.588887667
1000	2	0.686187633	98.89561703	0.343093817
1000	4	0.787918633	86.12684931	0.196979658
1000	8	1.409412667	48.14838905	0.176176583
1000	16	2.7464312	24.70877457	0.17165195
1000	24	4.080774	16.62943094	0.17003225
1000	32	5.470557333	12.40475975	0.170954917



In this plot we have presented the mean running time based on the number of cores for groups of size parameter 200 and 1000. For both of the cases we see that the plot shows an linear incremental based on the number of cores, so based on this we can confirm that by the raises of the number of cores the running time increases, its important to note that in my experiment we can see that until the number of cores 4 we do not have such an exponential growth of running time and also for the case of size 200 in the number of cores 32 we have a slightly fall of increment. As conclusion number of cores affects the growth of running time.



In this plot we are presenting the relative speed-up based on number of cores for two groups of size parameter 200 and 1000. In the plot we can see that by the increase of number of cores we have a downgrade of the relative speed-up parameter for both of the group sizes. Is important to note that based on the plot we can conclude that the increase of the number of cores lowers the speed-up parameter.



In the plot we can see the relation in between the parallel efficiency parameter and the number of cores for two groups of size parameter 200 and 1000, we can see from the plot that also the parallel efficiency gives low values as the number of cores increases, so by the raise of the core number the parallel efficiency gets low but important to note that after the number of cores grater than 5 it takes a stabilisation in terms of values for both cases of size parameter.

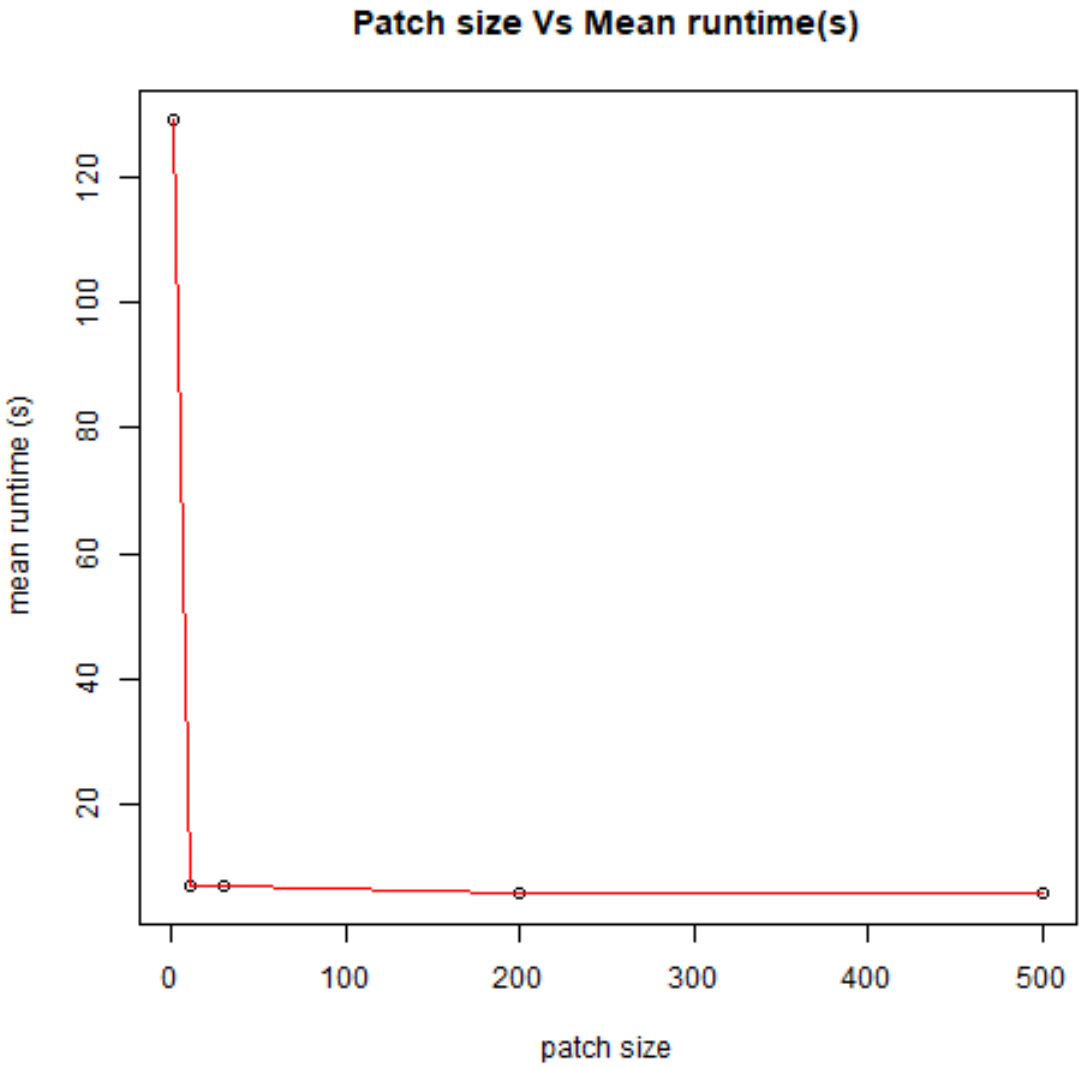
2.3 Influence of Patch Size

In this section we will analyse the influence of the patch size in our experiment. For the experiment we are keeping the number of cores fixed 32, the size parameter 1000

and then we are looking forward to mean runtime for different patch sizes.

Table 2: Measurements for task 2.3

size	p	patch	mean runtime(s)
1000	32	1	128.9545737
1000	32	10	7.100259133
1000	32	30	7.0519919
1000	32	200	5.862768633
1000	32	500	6.060787233



From the plot we can see the relation in between mean running time in seconds and the patch size. We can see than when the patch size is 1 we have a big value of running time compared to other patch size numbers, important to note that is the patch size is not 1 than we have quite small running times which means that the calculation is very efficient in terms of the running time.

2.4 Finding the Best Patch Size

In this section we are conducting an experiment where now the size parameter is fixed 800, number of cores 16, and we are incrementally increasing the number of patches from 1 to 30. Below we can see that table of the results from this experiment and also the graphical presentation of output.

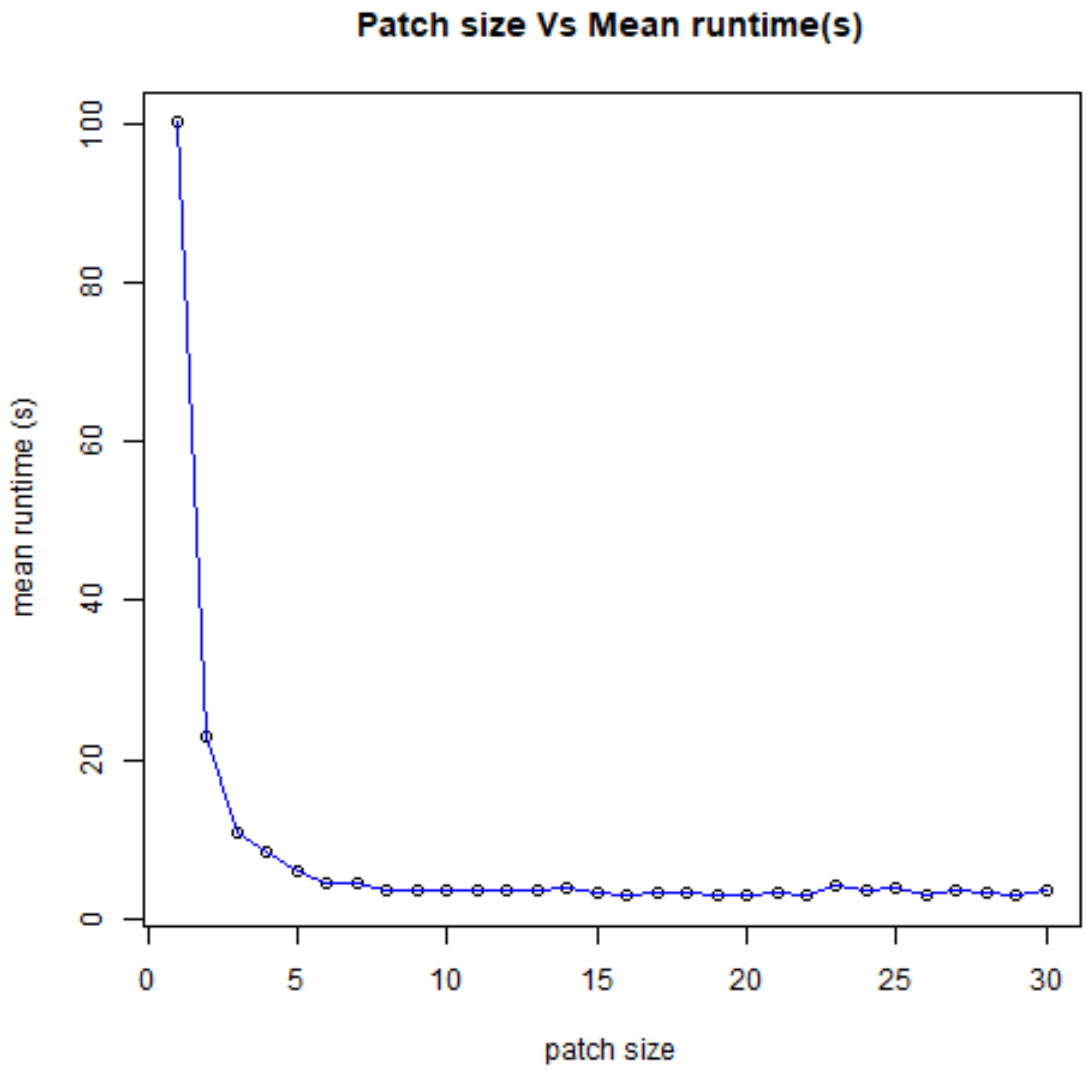
Table 3: Measurements for task 2.4

size	**p**	**patch**	**mean runtime(s)**
800	16	1	100.2281534
800	16	2	22.91150533
800	16	3	10.6634488
800	16	4	8.464866767
800	16	5	5.931327167
800	16	6	4.399542833
800	16	7	4.532331233
800	16	8	3.677068367
800	16	9	3.5174344
800	16	10	3.451052567
800	16	11	3.674077567
800	16	12	3.564871767
800	16	13	3.535040633
800	16	14	3.709654967
800	16	15	3.330507033
800	16	16	2.897853167
800	16	17	3.359166233
800	16	18	3.331343833
800	16	19	3.069417533
800	16	20	2.873782533
800	16	21	3.3599341
800	16	22	3.055777067
800	16	23	4.1069467
800	16	24	3.418405333
800	16	25	3.832624467
800	16	26	3.066856867

Continued on next page

Table 3 – continued from previous page

size	p	patch	mean runtime(s)
800	16	27	3.6072816
800	16	28	3.210179233
800	16	29	2.9436385
800	16	30	3.479451067



In this plot we can see that by the increase of the patches the running time goes down, as we can see after the patch size 8 our values of running time find a stability,

so in this case we could say that for our experiment the patch prefered size would be equal or grater than 8. As a conclusion we see that the grater the number of patches the more lower and stable would be the running time.