DIT346

Ebrahim Jahanshiri &
Jonatan Hellgren

March 2021

1 Problem 1

Since the script is basically completely parallelizable due to the fact that it is basically only performing the Monte Carlo simulation, we can assume that the proportion of the code that is parallelizable is close to 1. This is becuase the portion that the programe that is running in the loop is not dependant on any previously computed part. With this assumption Amdahl's law will generalize in to a straight line that lies at the theoretical maximal speedup, which is the number of processors used. We can prove this by setting the proportion of the code that is possible to be parallelziable f equal to one in the equation for Amdahl's law:

$$S(s) = \frac{1}{(1-f) + \frac{f}{s}}$$

$$(f = 1) \Rightarrow \frac{1}{(1-1) + \frac{1}{s}} = s,$$

and thus the speedup s becomes the amount of processors used.

To measure the time it takes to execute the script for a certain amount of workers we execute the script 10 times for each number of worker and then use the average time, this is to to mitigate the randomness that occurs when executing scripts. We noticed while timing the script that speedup is highly dependant on the amount of iterations used to compute pi in the Monte Carlo simulation. In Figure 1 we can see what speedup we get when using 1 million and 10 million iterations.

As we can see in Figure 1 the measured speed up is far away from the theoretical speed up. This could be the case due to the fact that we have to assign workers and split up the computation, which makes it less effective per core when utilizing more then one. We can even see that for 1 million iteration it actually slows down when we are using a higher amount of workers. This could be explained by the fact that the efficiency might decrease when utilizing more workers.

To evaluate the efficiency of the script when utalizing more workers we can calculate the parallel efficiency, for P processors it is defined the following way:

$$\text{efficiency} = s\frac{1}{P} = \frac{T_i}{T_P}\frac{1}{P},$$

where T_i is sequential timing and T_P is run time based on P processors. So with our script we get that the following efficiency when utilizing 32 workers:

$$\text{1 million iterations: } 1.93\frac{1}{32} \approx 0.06$$
$$\text{10 million iterations: } 9.44\frac{1}{32} \approx 0.29,$$

so we can see that the script utilizing the workers more efficiently when the number of iterations is higher.

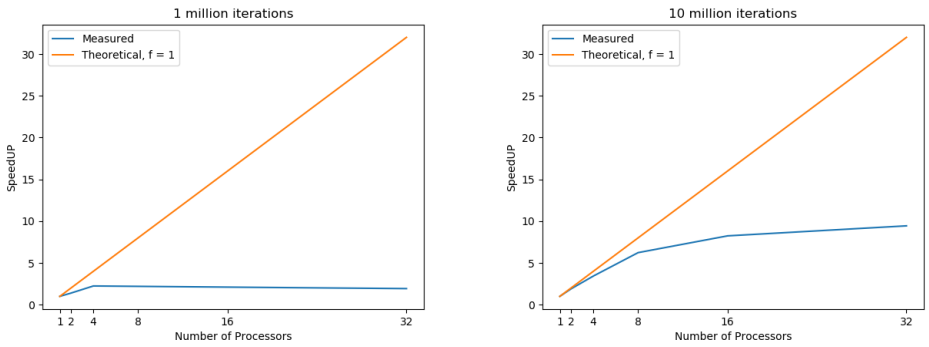

Figure 1: Here we can see how the speedup changes with regards to the amount of cores used to parallelize the script. The blue line is the measured time and the orange line is the theoretical time using Amdahl's law and assuming the entire script is parallelizable.

2 Problem 2

2.1 Discussion

Calculating the speedup for 1300GB data $T_i = 20, T_P = 12$ Therefore, we have:

$$S = 20/12 = 1.6.$$

We can now compute how much the parallelizable part needs to be sped up the reach this total speed up, if we assume that the fraction of paralizable code is 0.75 we get the following equation:

$$1.6 = \frac{1}{(1-0.75) + \frac{0.75}{s}}$$
$$\iff 1 = 1.6\left(0.25 + \frac{0.75}{s}\right) \implies s = 2,$$

and we get that a speedup factor of 2 is necessary. If the efficiency of the script is perfect this means that we need to double the amount of cores, if not we would need to increase it even more based on the efficiency.

So in both scenarios we can assume for it to be possible to complete the previous day analysis before 8:00 the next day, this is the case if the script that executes the analysis is parallelizable enough. If that is the case we can simply just add more cores unit the time limit is plausible.

The amount of cores that is necessary to add would be dependent on the proportion of the code that is parallelizable, with less amount of parallelizable proportion more cores would have to be added. Possibly we could even increase the size of the memory (ram) so that the entire dataset is able to fit in it, this would decrease the latency and thus increase the speed up. However the to different scenarios comes with their own respective pros and cons.

With scenario (a), buying new compute severs, it would be necessary to at least double the amount of compute power we already have to cut the duration of the analysis in half. This would however only solve the problem momentarily since it is specified that the complexity of the analysis is increasing. This would probably be a big initial investment to purchase the hardware and installing it, and it would make future upgrades an inevitability thus making it not that economically efficient. It could however be more economically efficient in the long run depending on how much cloud services cost. Another benefit would be security since the data never leaves the company's premise and we have virtually no data transfer cost.

Now considering scenario (b) instead, here we will have a more flexible solution to the problem since we do not invest in a server but instead are renting one. This makes hitting the deadline almost trivial since it is rather easy to increase the compute power that we are renting, the same thing can be done if the complexity of the analysis would increase. By looking at the options Microsoft Azure are offering we can see that the prize for severs that could handle this task isn't that expensive, some proposed examples can be seen below. Here we also need to consider that the security could become an issue when we are transferring the files and that this transfer of files also could become an additional cost.

2.2 Examples of computer servers

Looking at Linux server from Microsoft azure rented from Norway.

With virtual CPUs, assuming we don't need entire dataset on the ram at the same time.

- E48ds v4: 48 vCPU:s, 384 HB RAM, 1800 GB storage, 4.827 US$/h
- E64 v3: 64 vCPU:s, 432 GB RAM, 1600 GB storage, 4.814 us$/h
- E64ds v4: 64 vCPU:s, 384 HB RAM, 2400 GB storage, 6.436 US$/h
- E80isd v4: 80 vCPU:s, 504 GB RAM, 2400 GB storage, 9.1801 us$/h

With physical CPU on the Azure.

- M64ds v2: 64 Cores, 1024 GB RAM, 2048 GB storage, 9.9537 US$/h
- M64mds v2: 64 Cores, 2048 GB RAM, 2048 GB storage, 14.786 US$/h
- M128ds v2: 128 Cores, 2048 GB RAM, 4096 GB storage, 19.076 US$/h
- M192ids v2: 192 cores, 2048 GB RAM, 4096 GB storage, 22.926 US$/h