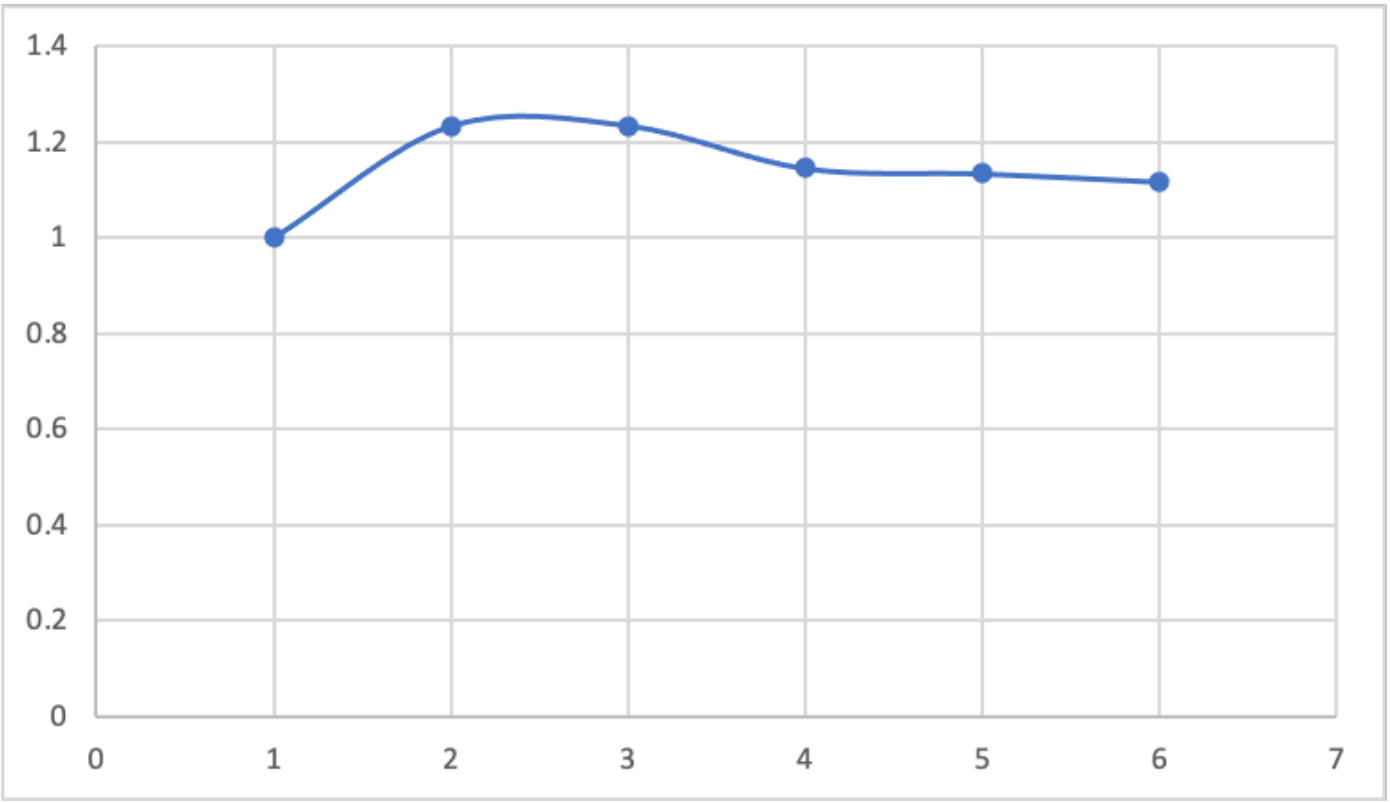
COMP409 Assignment-1 Q2



I am using a intel 4 core machine. The default parameter I used for testing this program and obtaining the above graph are:

- outputheight = 2048
- outputwidth = 4096
- Attempts = 500

I used a data division stategy for speeding up this program. Each thread terminates when their attempt reach 0.

Basically, I divide the output image vertically into several equal parts. Each part has a equal width. The number of divison is dynamically calculate based on the width of the output image and the max width of the input icons, which aims to gain a speed up while avoiding the overlap. You can see my code for detail.

I use this number to allocate the locks. For example, given the default parameter I listed above and the provided cats icons, the number of locks I used in this program is 7. Each lock locks a vertical part of the output image, two thread that are trying to draw on the same part will obtain only one lock on that part. But differnet threads that are trying to draw on the different part are able to draw at the the same time. There are some details to handle the overlap issue that might occur at the edge of each part.

In general, you can see that there are speed up for t from 2 to 6. I think this is because given a large ouput image and a large attempt, single thread need to fill the entire image one at a time, which has a rate or possibility of encountering the overlap slower. In contrast, having thread number greater than 1 will draw the image concurrently at the each part, which fill the entire image quicker so each thread will encounter the overlap at a high rate. This is the main reason why mutithreading will have a speed up in general.

However, as you can see from my graph. The speed up is not linear. The speed up I got is at highest between 2 to 4 threads. At 4 thread, the speed up tends to become asymptotic. I believe that this is because the speed up is limited or bounded by the number of locks I use. In this case, the maximum icons that can be drawed at a single time is 7. Given only two threads, they are able to draw at the different parts at most time. But given 4 to 6 threads, even more icons can be draw at a time, the number of locks I have is 7, which means there is a high probability that more than one threads will compete for the same lock, and some of them have to wait for the lock. This causes lock contention more frequenty and is why the speed up got decreased.

You can try to modify the default the parameter to use a bigger output image and gives a high attempt number, you may observe a more obvious speed up. This is because the number of lock I allocated is proportional to the width of the output image. Having more locks allow a higher concurrency on drawing the icons though may also bring some overhead of creating the locks.