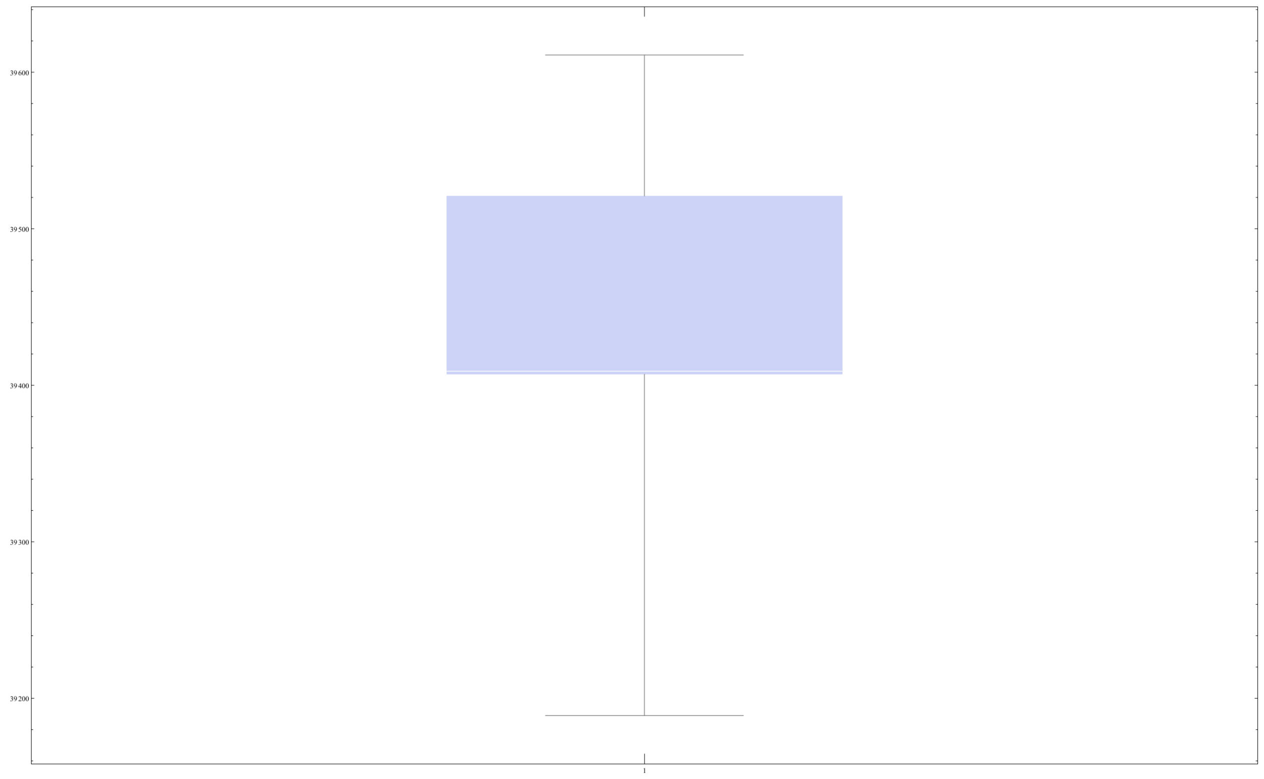
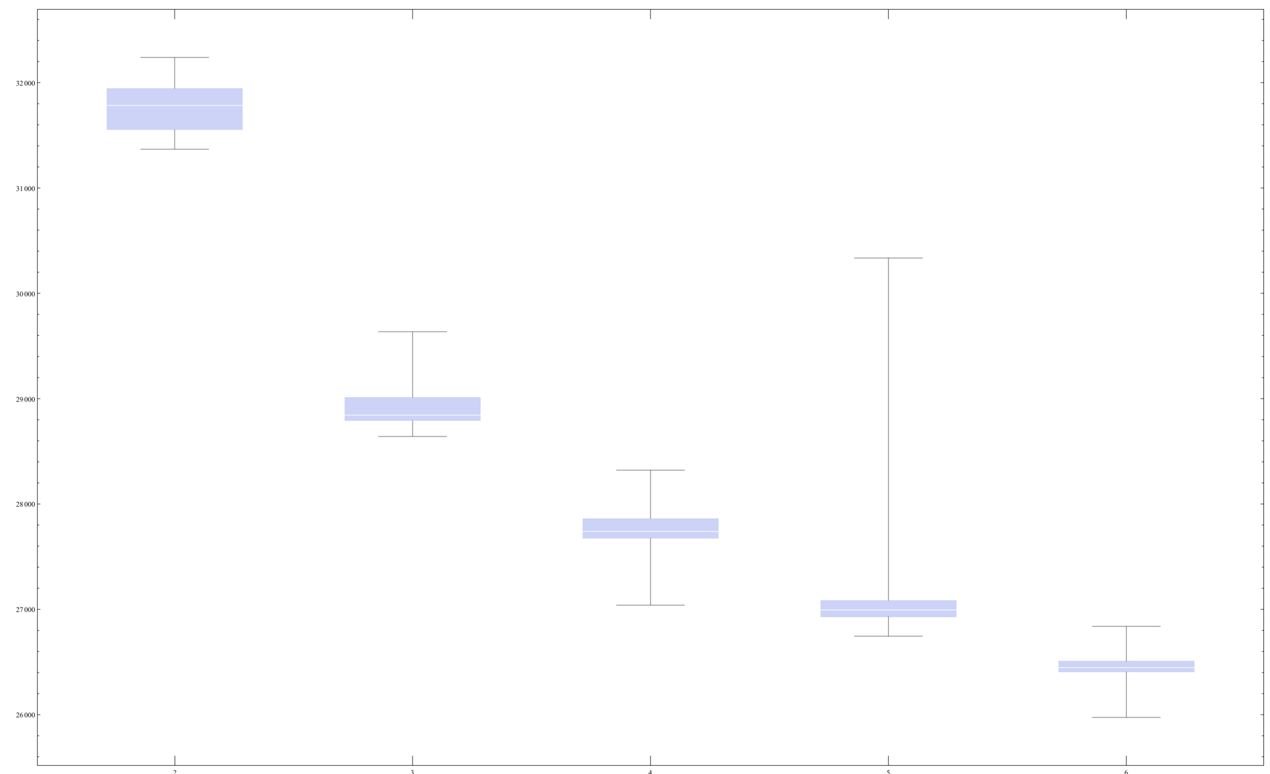
Computer Operating Environments Report

Machine used for Benchmark:

Machine: MC8176 in Room 3516
Processor: Intel Core i5-2400 at 3.1GHz
Memory: 4.00 GB

Results:
I have included the full size .pdf of both of these graphs alongside this file for a clearer view.





For every addition of a thread, the program was run 10 times to get the results shown. The time taken in milliseconds is up the left and the number of cores along the bottom.

As it can be seen from the diagram, there is a drastic difference between using a single thread and two. But everything past this offers very little performance gain. Despite the fact that the diagram shows that six cores offers the best performance, this is not necessarily the case. Due to the arbitrary number of loops (in this case 2000), the number of pin numbers is slightly decreased when the amount of threads does not divide equally into 2000. Considering this fact, the amount of threads for best performance is either four or five. The mean difference in time is negligible at only 456.4 milliseconds.

This is due to several factors. The first and foremost is the amount of cores that the processor itself holds. The CPU mentioned above is said to be quad-core [Intel, 2011], this suggests that four threads would be best to fully maximise the number of cores in use. However, five cores offers similar results due to the way the tasks are split. The for loop used to generate the pin numbers for the application has its upper limit divided by the number of cores currently in use, followed by a second for loop to activate an equal number of threads every time round the loop. From this, exactly 2000 pin numbers are generated in the same way they are when the number of threads is as low as one or two, offering a more equal comparison.

Secondly, with the addition of extra threads, more pin numbers can be found at once. Whilst one thread takes the time to open the file and read the pin numbers into a Set, the remaining threads are able to get a new pin number ready for checking against the stored pin numbers. This could be one the factors resulting in the small gain in performance with the fifth thread, albeit unlikely since each time the file is only opened and read once before all pin numbers are found and written to the file.

Due to the overall simplicity of the task each thread must accomplish, the length of time the threads are running is relatively small. This allows for more threads to be generated more quickly and the channel list to be populated faster.

Lastly, the increase in speed is likely mostly due to the memory architecture and size of the cache of the CPU. Every time a pin number is retrieved the CPU places it in the cache if it can not yet be checked against the current list of pin numbers. As mentioned before, the task is relatively simple for each thread to accomplish and the memory requirement of each thread minimal, meaning that the physical memory of the system is seldom used in favour of the cache. Thus as more threads are added, the more pin numbers it is able to store at any given time allowing for faster checking since it does not have to wait for the thread to retrieve a new pin number.

References:

Intel Core i5 Processor Specification, 2011. [online]. Available from: http://ark.intel.com/products/52207/Intel-Core-i5-2400-Processor-6M-Cache-up-to-3_40-GHz [Accessed 11 May 2014].