UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Programming Languages
(CSCI 3055)

Assignment 3: Concurrency in Clojure

April 9th, 2016

Luisa Rojas (100518772)

OBJECTIVE

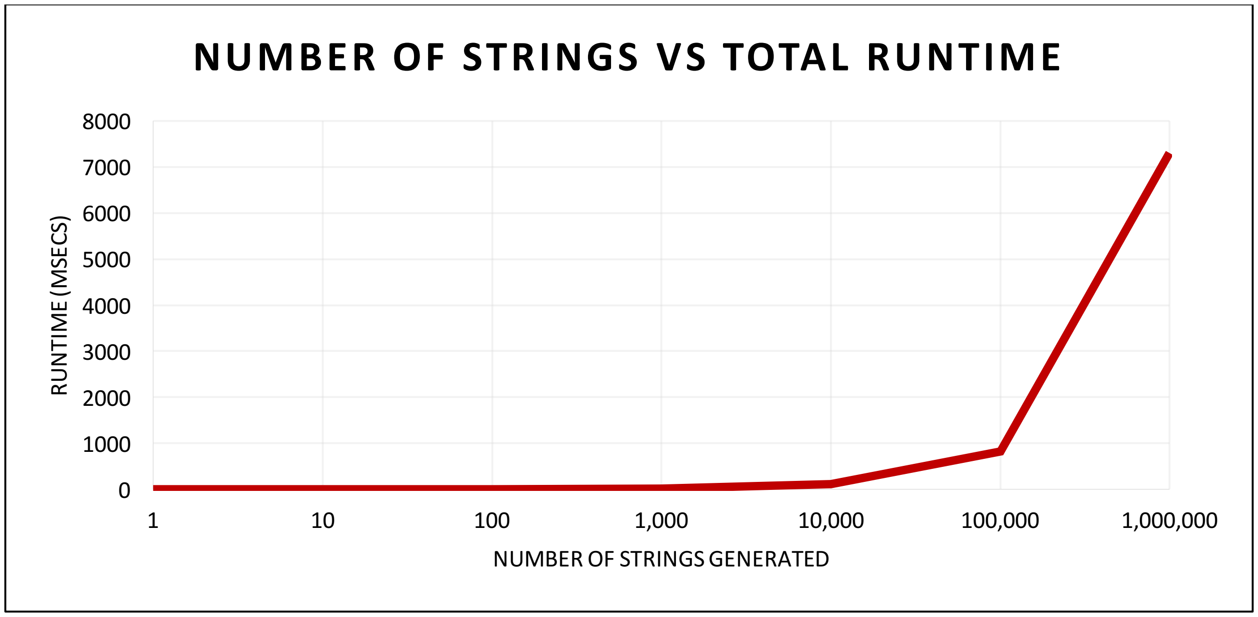
The purpose of this assignment is to study the effect of concurrency in Clojure as well as the choice of number of threads for each observation done.

DESCRIPTION

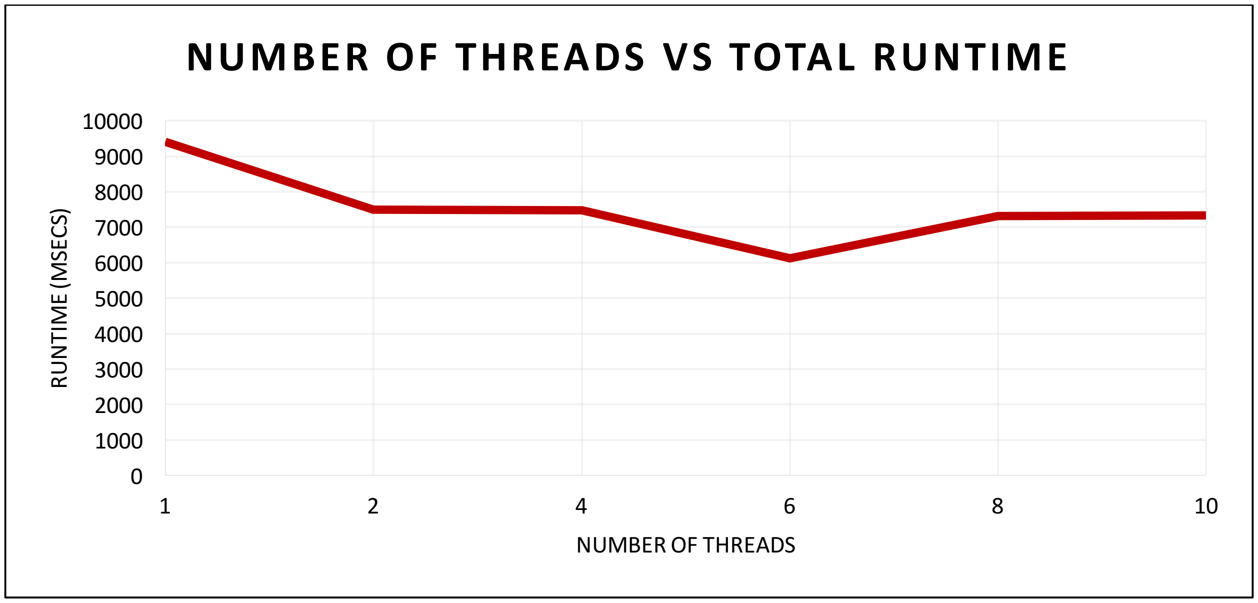
In order to achieve the objective described above, a series of N strings of length 15 (characters) were generated. Then, for each string created, the number of distinct characters were counted and then the grand total was aggregated from these.

This process was performed using a sequential and a threaded implementation; the time it took for it to complete the task for both scenarios was recorded.

It's important to note that the "task" being timed only consists of the count of distinct characters per string and its aggregation; the random string generation was not taken into account for this purpose. These were the results:



As expected, an exponential increment in the runtime as the number of strings generated increases can be observed; the reason for this is that, the more strings are created, the more separate values there are to calculate and aggregate.



When implementing threads, the number of threads was taken into account for observational purposes rather than the number of strings generated. Instead, the number of strings created was set to a constant value of 1,000,000.

In the case of the concurrency implementation, a strong pattern is not very obvious or noticeable; however, it could be concluded that the most optimal thread number for this calculation is 8. Technically, concurrency should have helped run the program in a more efficient way, since threading would allow each portion of it to be executed in parallel.

More detailed results can be observed below:

SEQUENTIAL	
N	**Time (msecs)**
1	1.319426
10	3.962381
100	7.534452
1000	24.100678
10000	114.792626
100000	825.558029
1000000	7302.089817

THREADED	
K	**Time (msecs)**
1	9403.779768
2	7491.386379
4	7477.752955
6	6129.617075
8	7312.724026
10	7337.065449