SIT315- Seminar 4- Multithreading

Activity 1 - Decomposition techniques

1. The best technique is to use the **Exploratory Decomposition**, meaning decomposition according to a search of a state space of solutions. Thus, the puzzle will be stored in a matrix where there is possibility for either 4 next moves (if the blank square is in the middle), 3 next moves (if the blank square is a corner) or 2 next moves (if the blank square is an edge). Each possibility is then allocated to processor where each processor would traverse through each next move for every solution. Here the beginning configuration is divided to n different possibilities which are tested individually on separate threads and finally each thread results in a solution. It is also important to terminate the search when a solution is found to avoid long run time.

2. The best technique is to use the **Data Decomposition**, meaning this split the page to different sections and generate several threads(N) for the number of different sections. The sections could be for different text sections of the page. We then check each section for the count of each word separately.

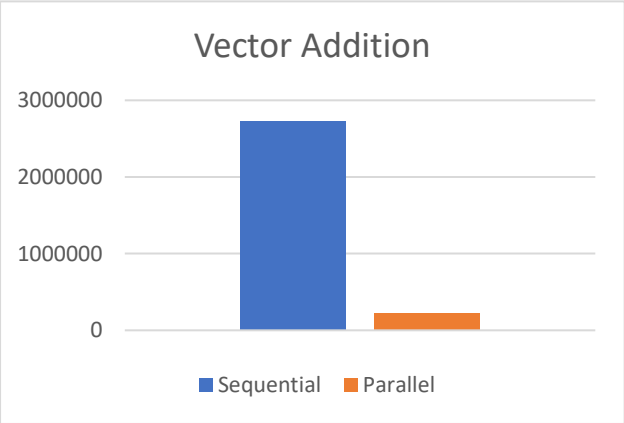
3. Assuming that the input array is sorted binary search could use R**ecursive decomposition**. Here we would split the array in half and take the half and divide back in half until we reach a solution. This can be done recursively until the subset of the array consists of only one element which is when a level of simplicity has been reach and can call back to the precious recursions.

Activity 2 - Parallel Vector Addition

Git hub link -

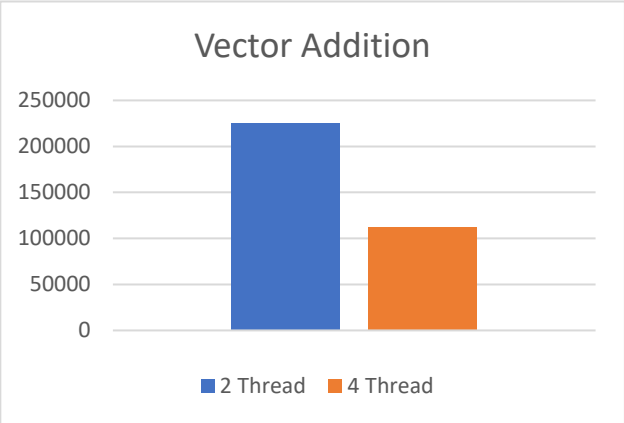
Sub tasks	**Sequence or Parallel**
Declare three vectors	Sequence
Allocate random data to vector one and two	Parallel (Divide to two tasks) - Generate random vector one - Generate random vector two
Carry out addition of vector one and two	Parallel (Divide to decided number of threads) - The data in the array is partitioned equally to the given number of threads and added separately
Compute time taken to carry out function	Sequence

Comparison of sequential and parallel execution time:



As seen the Parallel method is extremely faster than sequential.

Comparison of Parallel with two and four threads



As seen using 4 threads is almost twice as faster than 2 threads.