HPC LAB-2

V Sai Srikar SE20UARI164
B.Krishna Vamshi SE20UARI179
Aashish Joshua James SE20UARI002
Aditya Sake SE20UARI006
ADITYA SAI SE20UARI005

To Run the programs-:

Go to Lab2 folder-> and go Merge Sort Folder and to run Mergesort.c file

Run the following command-:

1. gcc mergesort.c -o mergeint -fopenmp
2. ./mergeint

Go to Lab2 folder-> and go to nQueens Folder and run nQueens.c++ the following way

1. g++ nQueens.c++ -fopenmp
2. ./nQueens <Number of Queens> <Number of Threads>

Here the function arguments are the number of queens and number of threads

Parallel code for-:

1.Merge Sort-:

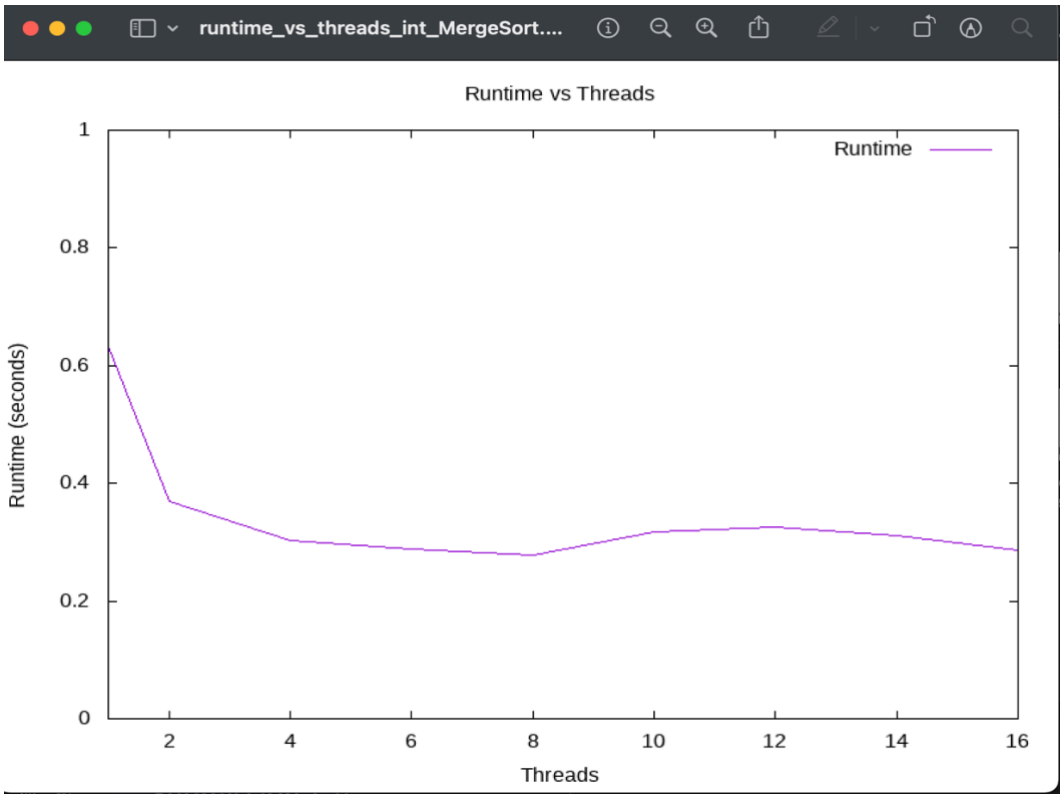
Merge Sort is a divide-and-conquer algorithm that works by dividing an array into smaller arrays and then merging them together in sorted order. Parallelizing Merge Sort can improve its performance, particularly for large arrays.

One way to parallelize Merge Sort is to use a technique called "parallel merge". This technique involves dividing the input array into multiple parts, sorting each part in parallel, and then merging the sorted parts back together in parallel.

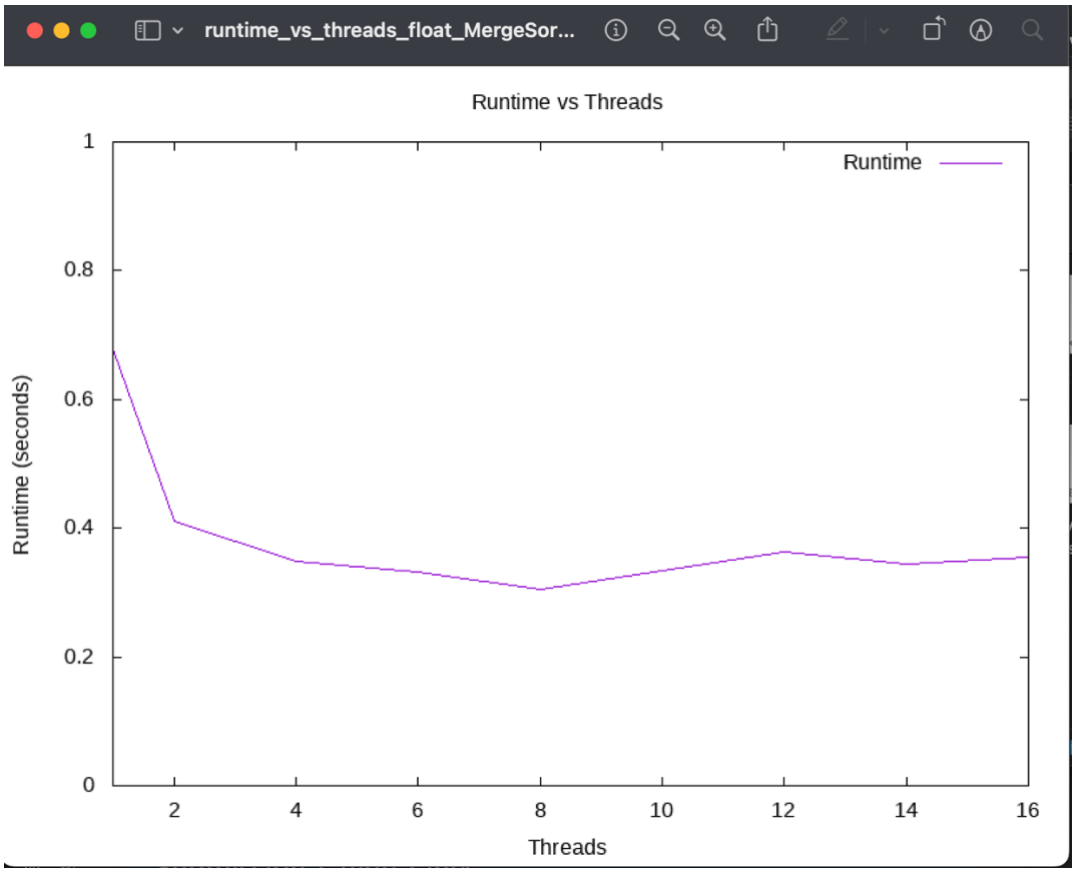
For comparisons a hybrid approach is to use a combination of centralized and distributed comparison methods, where a centralized unit handles comparisons between large sub-arrays, and distributed comparison methods are used for smaller sub-arrays.

Below can we can see Merge Sort Technique by considering bottom-up approach which has been parallelised and run for different datatypes mainly-: int, float and double.

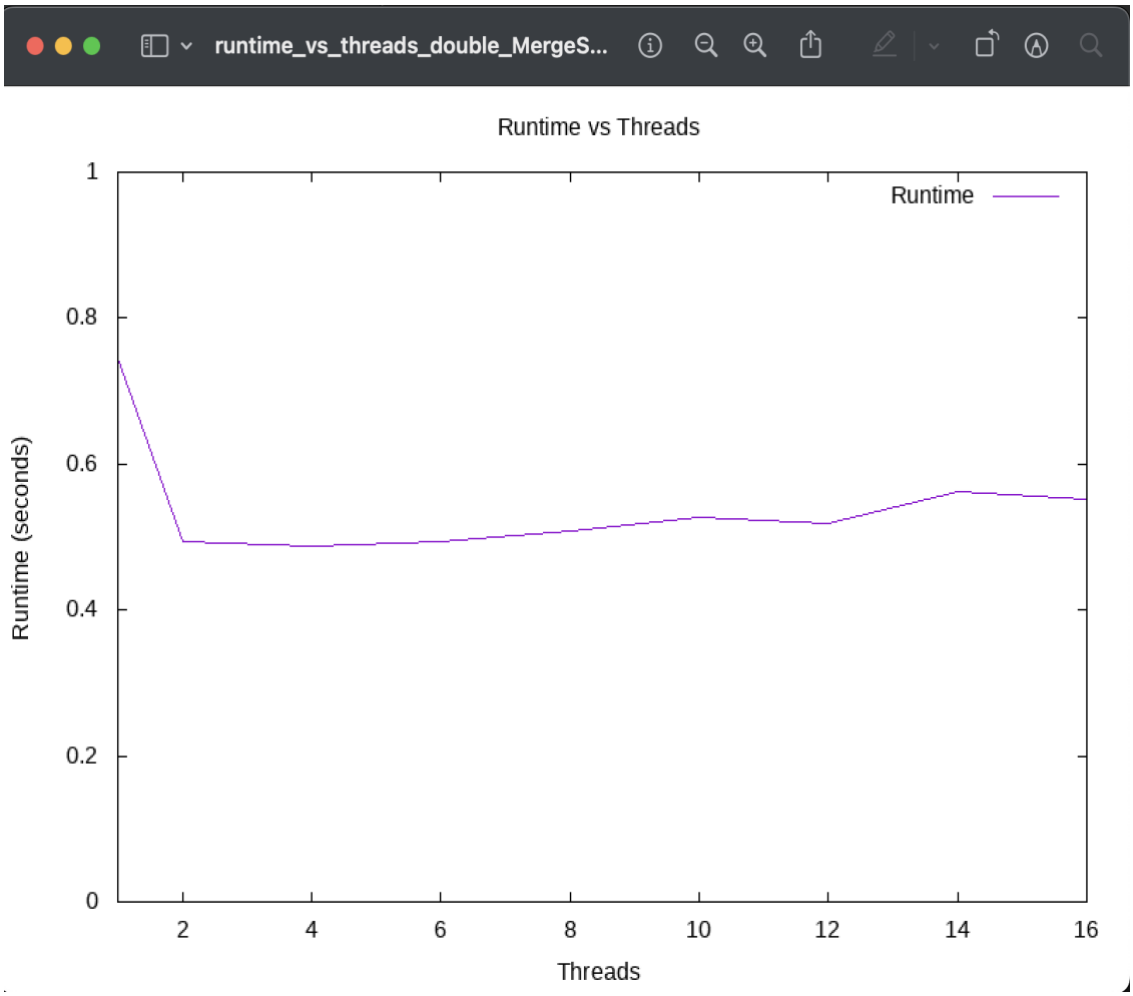
The graphs consist of Runtime Vs Numbers of Threads. From the graphs we can see that the runtime decreases as we increase the number of threads and this is because adding in more number of threads increase the amount of parallelisation capacity.



Above is the graph for int datatype



Above is the graph for Float Datatype



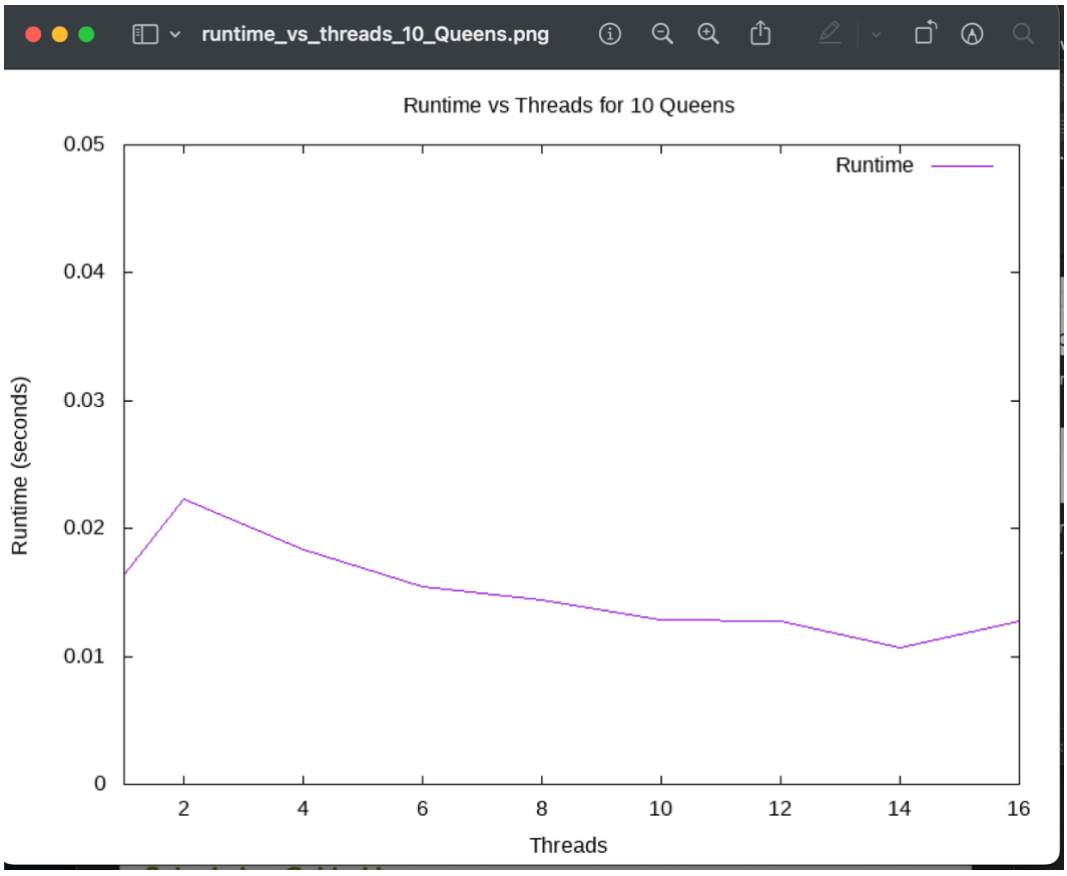
Above is the graph for Double data type

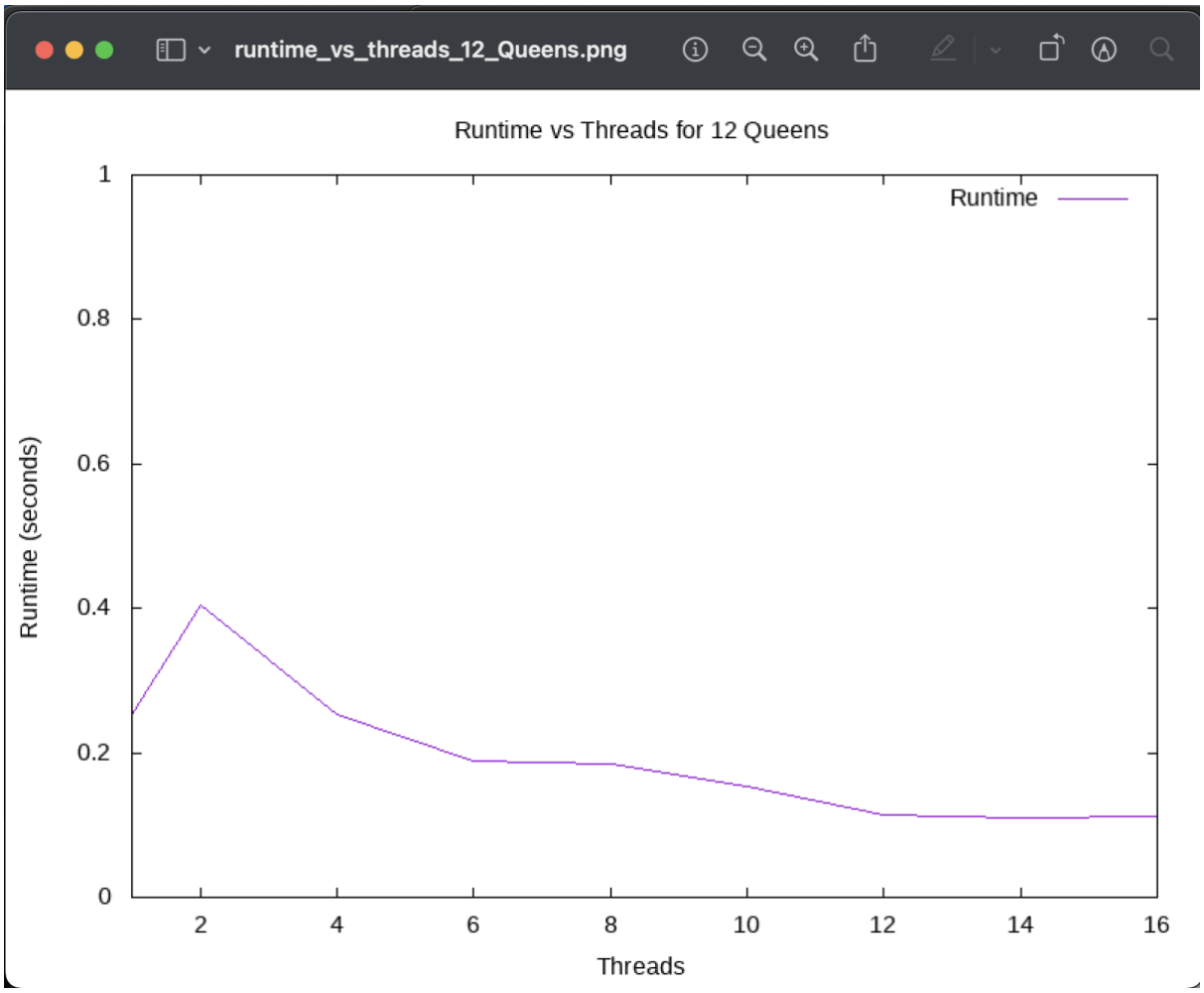
2.Nqueens-:

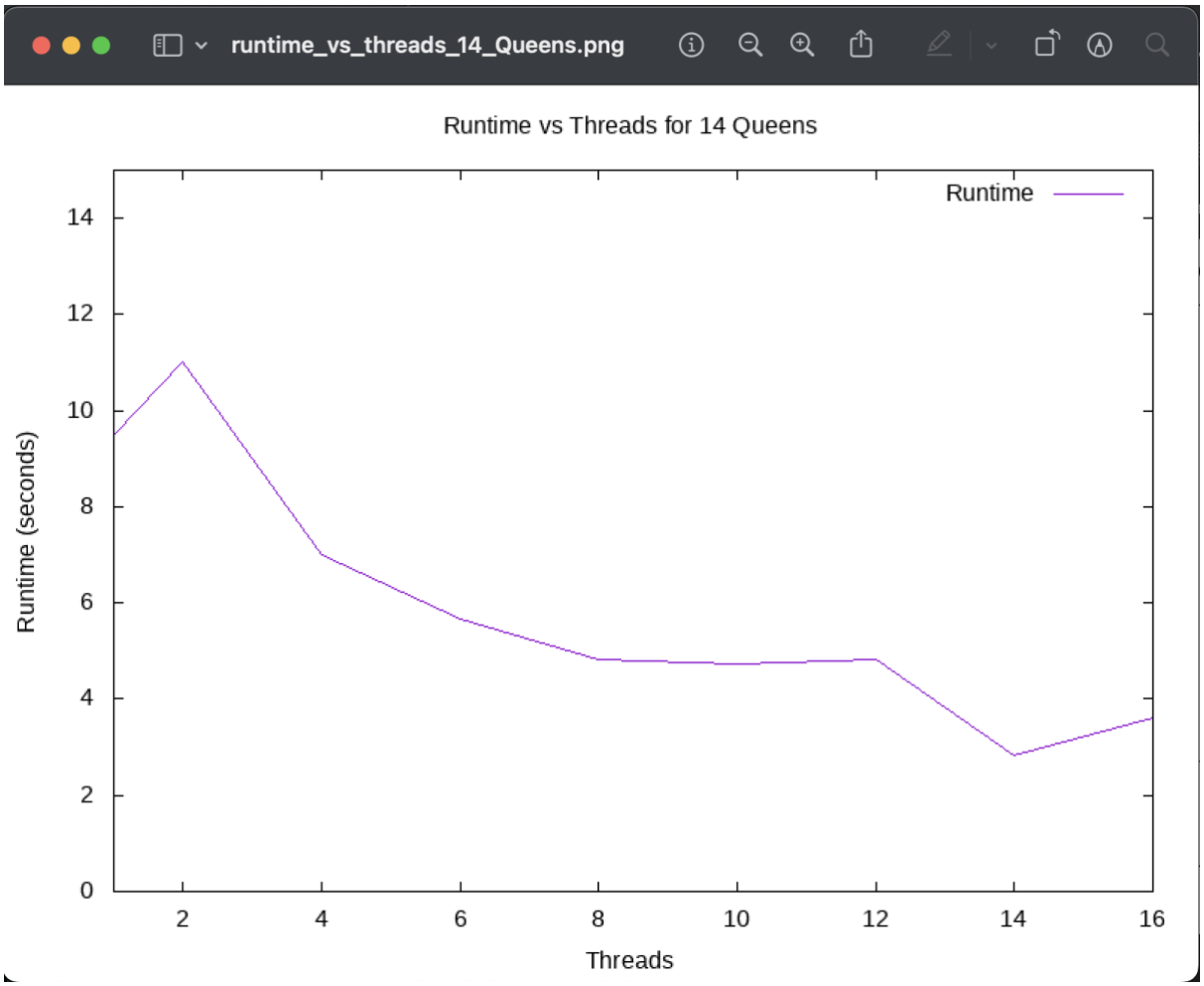
The N-Queens problem is a classic puzzle that involves placing N queens on an NxN chessboard such that no two queens threaten each other. Specifically, no two queens can be in the same row, column, or diagonal. The problem is to find all possible solutions for a given N.

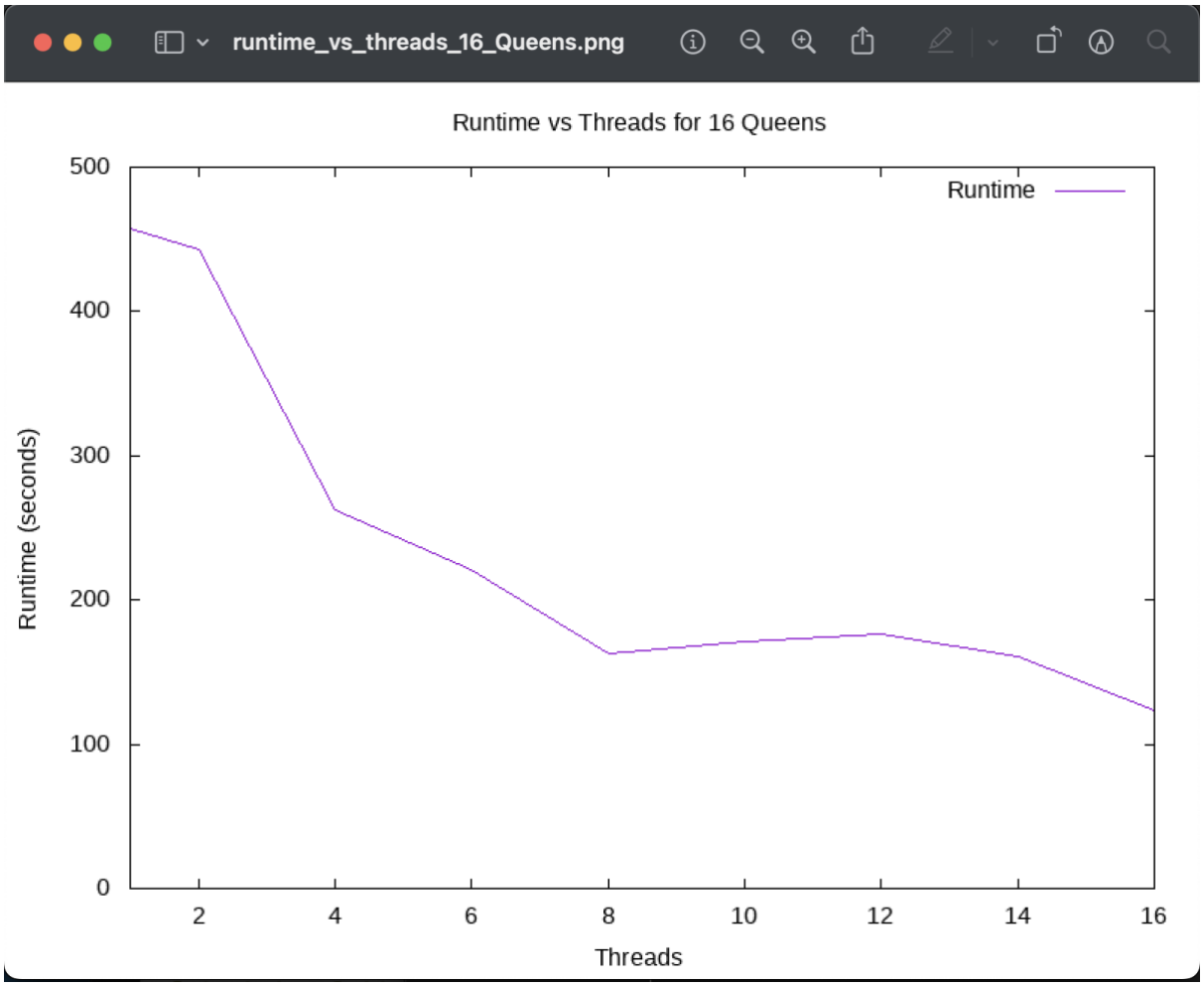
To parallelize the N-Queens problem, one approach is to divide the state space tree into smaller sub-trees and assign each sub-tree to a different thread. Each thread can then explore its own sub-tree in parallel, searching for solutions.

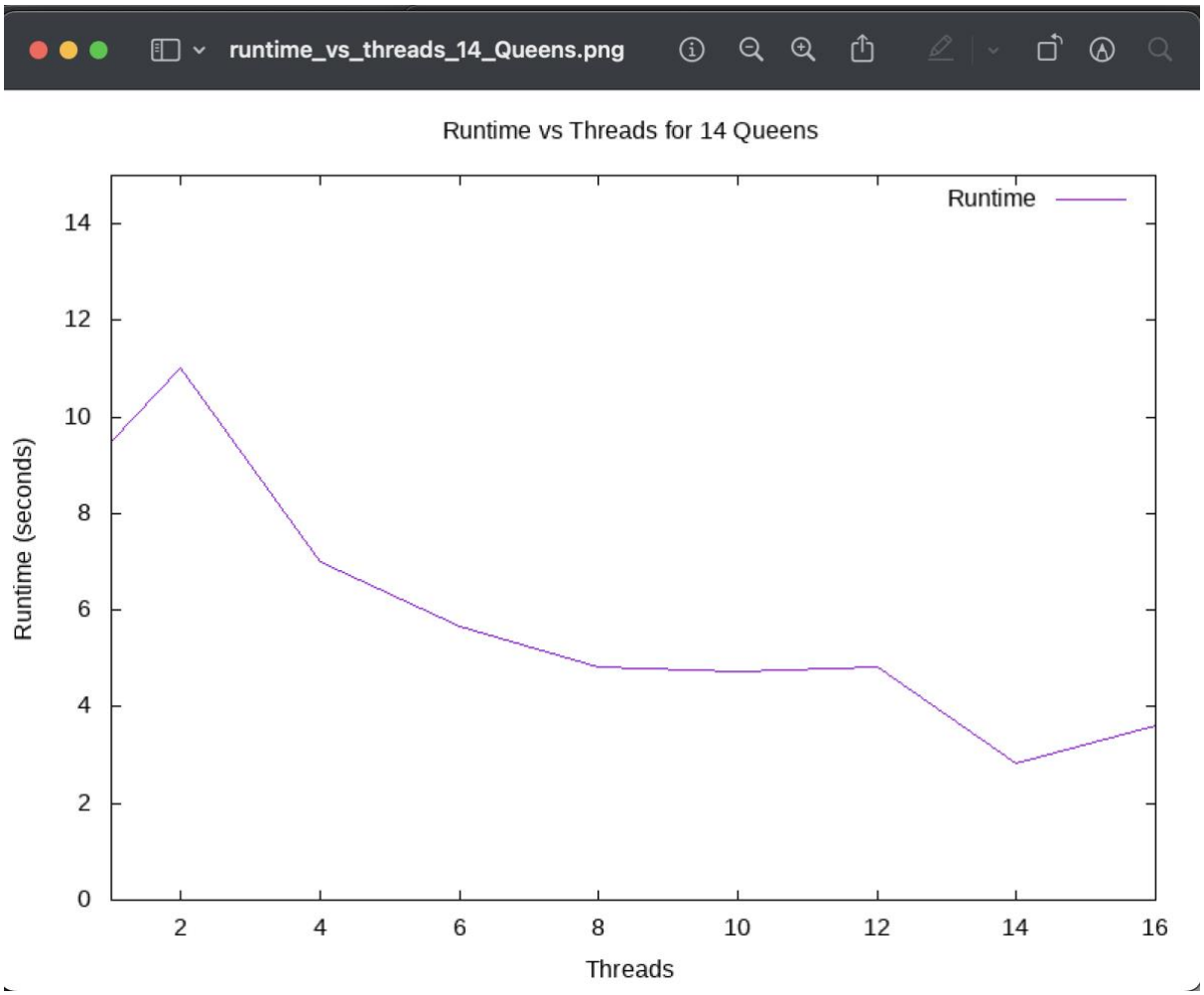
Below we can see graphs where we have Runtime Vs Number of Threads and for different values of N.A common Observation is that the run time decreases with increase in number of threads.




runtime_vs_threads_12_Queens.png
Runtime vs Threads for 12 Queens
Runtime
Runtime (seconds)
Threads
0
0.2
0.4
0.6
0.8
1
2
4
6
8
10
12
14
16


runtime_vs_threads_14_Queens.png
Runtime vs Threads for 14 Queens
Runtime
Runtime (seconds)
Threads
0
2
4
6
8
10
12
14
16


runtime_vs_threads_16_Queens.png
Runtime vs Threads for 16 Queens
Runtime
Runtime (seconds)
Threads
500
400
300
200
100
0
2
4
6
8
10
12
14
16


runtime_vs_threads_14_Queens.png
Runtime vs Threads for 14 Queens
Runtime
Runtime (seconds)
Threads
0
2
4
6
8
10
12
14
16