Assignment Report- Optimization Engineer (Computation)

`LLA  = calculating_LLA(time_steps, satnames, timeperiod)`

Input Parameters:

Start Date(`time_period = date_time(2019, 12, 9, 12, 0, 0)`),

Time Steps(`5*24*60- every minute for 5 days`)

Satellite Names(`30sats.txt`)

Output Parameters:

LLA = Array of Arrays containing position vectors at every timestep(3 float values every second)

Explanation:

1. At every time step, the function calculates Julian dates from calendar dates
2. Calculates position vectors of the satellite
3. Converts the position vectors in Spherical coordinates to Latitude, Longitude, Altitude

`satellite_times = find_entry_exit_times(LLA, bottom_left, top_right)`

Input Parameters:

LLA = Array of Arrays containing position vectors at every timestep(3 float values every second)

Area of Interest = area inside the rectangle whose diagonal coordinates are `bottom_left, top_right`

Output Parameters:

Satellite entry exit times = An array of a dictionary(key-value pairs) corresponding to each satellite's entry and exit times into the region of interest

Explanation:

At every timestep, the entry time of the satellite is saved and the exit time is updated

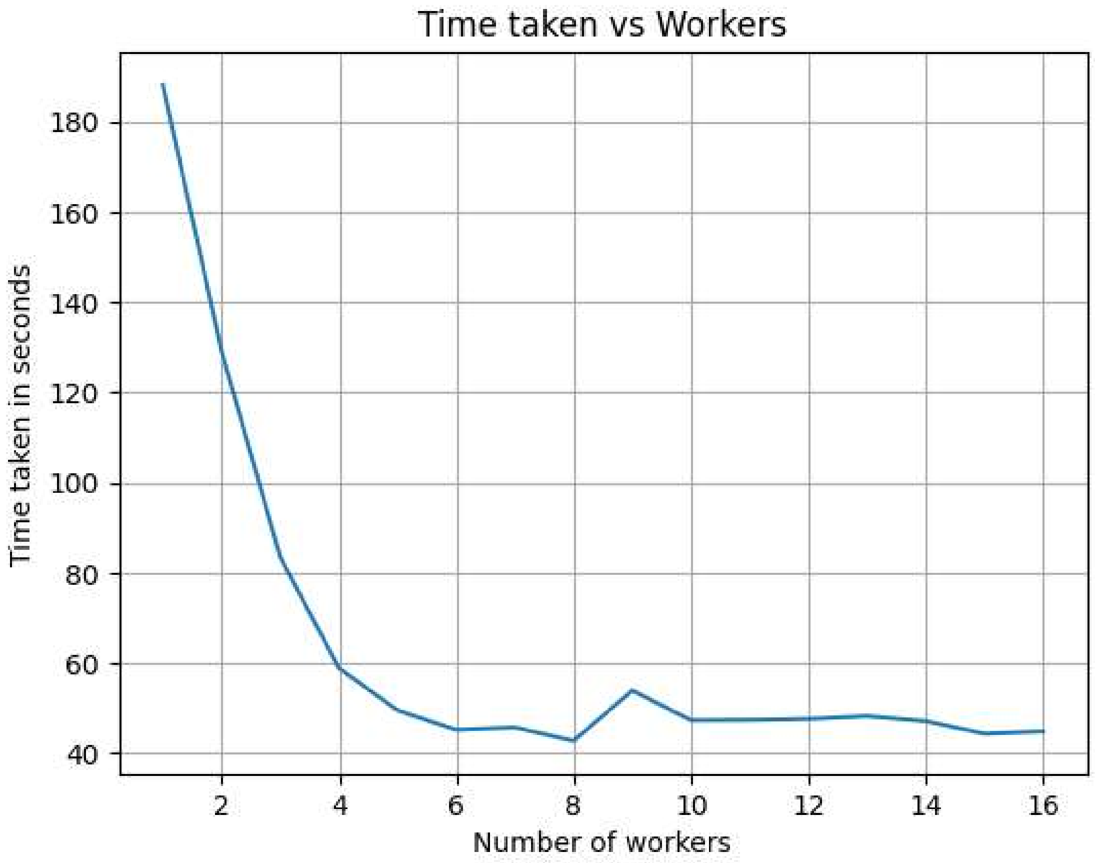
`saving_entry_exit_times(satellite_times, time_period, bottom_left, top_right)`

Input Parameters: Satellite entry exit times, Area of Interest

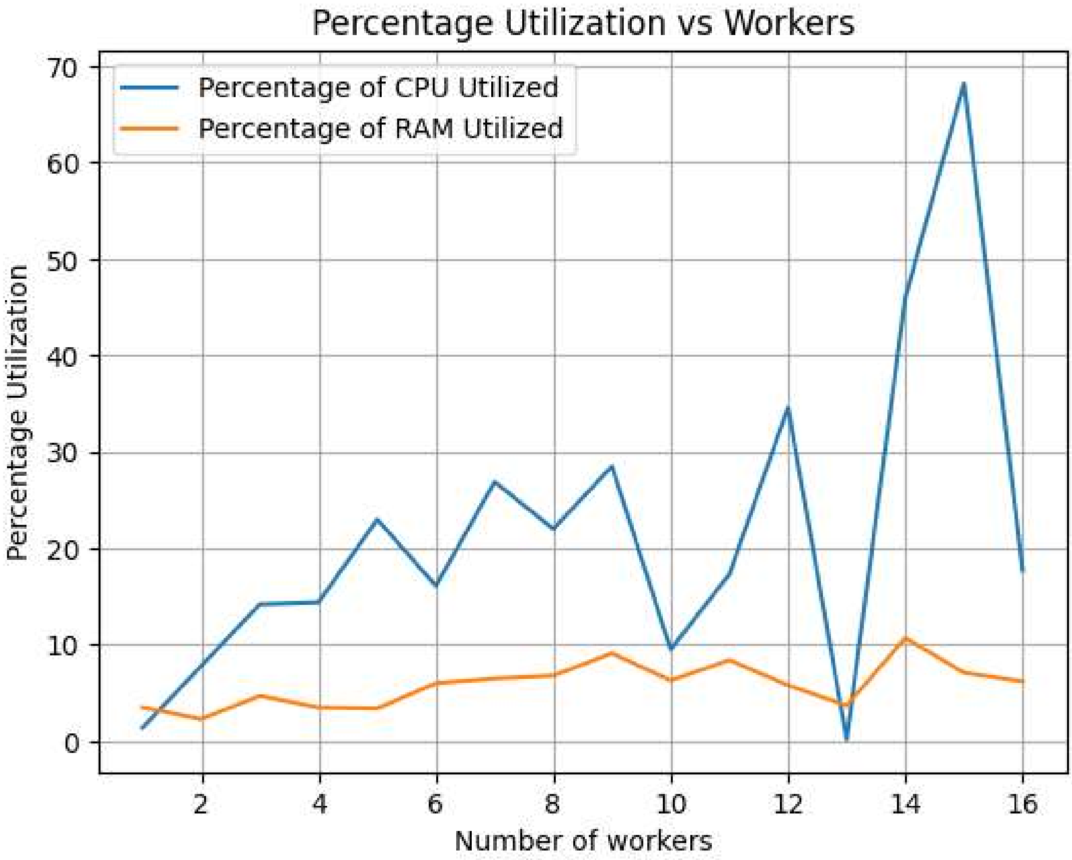
Explanation: Creates `results.txt` file

`calculating_LLA and find_entry_exit_times` are parallelised using Ray Libray in Python for distributed computing

Results:



As the number of workers increases the time taken decreases as shown for the above graph



As the number of workers increases, the CPU utilization and Memory utilization percentages shoot up as shown in the above plot

I could only run the code for 30 satellites because of a lack of computational resources for 27000 satellites