CIS6007 Parallel and Distributed Systems  
Assignment 2  
Name: Yumit Demir  
Student Id: 2218012  
Program: SE3

<https://github.com/yumitdemir/CIS6007-Assignment-2>

Code Structure:

Circle.cs

This class represents a circle with coordinates and a painted status.

Painter.cs

This class represents a painter that can paint circles. It keeps track of painted circles using a concurrent dictionary.

PaintManager.cs

This class manages the painting process. It distributes circles among multiple painters and starts the painting process.

Program.cs

This is the main program that generates circles, creates a PaintManager, and measures the time taken to paint all circles with different numbers of workers.

Evaluation of Task 1

1. Is this problem able to be parallelized?

Yes, it can be parallelized because each circle can be painted independently by different workers.

2. How would the problem be partitioned?

The problem can be partitioned by assigning subsets of circles to different painters, either by evenly distributing circles or dynamically assigning them from a shared pool as painters finish their tasks.

3. Are communications needed?

Minimal communication is needed, mainly to track which circles have been painted. This could be handled using a shared data structure, like a thread-safe collection, to ensure painters don't paint the same circle twice.

4. Are there any data dependencies?

No, there are no direct data dependencies between painters since each circle can be painted independently. However, a thread-safe mechanism is needed to prevent multiple painters from painting the same circle simultaneously.

5. Are there synchronization needs?

Yes, synchronization is needed to ensure that painters don't access or modify the same circle simultaneously. This can be handled using thread-safe data structures like a ConcurrentDictionary or locks to prevent race conditions.

6. Will load balancing be a concern?

Yes, load balancing can be a concern, especially if the circles have varying complexity or painting requirements. If some painters finish their tasks faster than others, it may lead to inefficient resource usage. Dynamic task assignment can help mitigate this issue by allowing idle painters to pick up new tasks.

Task results and observations







This shows that increasing the number of workers significantly reduces the time taken to complete the task. However, there are diminishing returns as the number of workers increases beyond a certain point, as seen in the smaller difference in time between 20 and 100 workers compared to the difference between 5 and 20 workers. This suggests that while parallelizing the task with more workers improves efficiency, there is a point where adding more workers yields diminishing returns due to factors like overhead and resource contention.