Static Scheduler Assignment Answers

1. Please see plot graphs attached in zip file.

2. Why do you think some measurements are so erratic?

a. On observing the static_compare graphs, we can see that in the first 24 slides, the speedup of iteration-level sync is almost constant. Because of using mutex, we have achieved very less parallelism. On the other hand, the speedup of thread-level sync is more. It also increases with respect to number of threads against different intensities.

b. This graph is erratic because it's a measurement of how certain number of threads are performing better or worse against different values of n. For example, in slide 20, 2 threads are performing good against 1000 'n', but their performance starts to drop as 'n' increases. Which means we can ideally use more threads to increase performance (as apparent from consecutive graphs of similar intensity and 'n').

c. The remaining graphs help us in observing speedup against intensity across different number of threads and 'n'. The first initial graphs with lower 'n' don’t have much pick up, but we can see lot of improvement in speedup as soon as n climbs up from 10000 against 12 threads and more. This means that speedup is directly proportional to choosing number of threads and iterations (n). Hence, more number of threads and 'n' will give us better performance.

3. Why is the speedup of low intensity runs with iteration-level synchronization the way it is?

Speedup of low intensity in iteration-level graphs is low against any number of threads. We can also observe that the same is not true for graphs with higher intensities. According to Gustafson's law, we can only achieve a certain level of parallelism in the code, beyond which increasing any number of threads in the process doesn't increase the speedup. Hence, by increasing the workload instead, we have a better way of getting more work done in same amount of time and increasing speedup. This is the same behavior we can see in low intensity graphs, and hence the speedup is lower than those of higher intensity graphs.

4. Compare the speedup of iteration-level synchronization to thread-level synchronization. Why is it that way?

Speedup of iteration-level synchronization is comparatively much less than that of thread-level synchronization. Iteration level sync is implemented with help of mutex, which helps us to preserve data integrity of a global variable (by locking and unlocking it while writing data) but at the cost of losing parallelism. This results is lower speedup. Whereas, the thread-level sync exhibits true parallelism by independently working on their respective local variables without being dependent on each other (except for the last computation, which adds up all the values of thread level results). Therefore, we can see a better speedup in thread-level synchronization.