# Static loop scheduler

**Question: Report time and speedup across a range of precision, intensity, and synchronization mode.**

The report is attached with the name “**static\_sched\_plots.pdf**”.

**Question: Why do you think some measurements are so erratic?**

It is due to the fact that with increase in a value for the number of threads, the performance decreases. This becomes worse for iteration mode than thread mode. This is due to that fact that the iteration is no better than sequential due to mutex lock every time the value of integral is calculate.

**Question: Why is the speedup of low intensity runs with iteration-level synchronization the way it is?**

The iterator level synchronization is the same as the sequential execution of a program. This is due to acquiring a lock for every value that is computed locally. Hence, the speedup is very less regardless of the number of threads that have been used to run the program. This give us a deep insight on how the locks should be placed for making a parallel program more effective.

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

For higher intensity and higher threads, iterator level synchronization gains low speedup than the thread level synchronization. This is due to the way of coding fact that thread level synchronization requires locks is number of threads. However, iterator-level-synchronization requires locks to compute the value of integral across all the threads irrespective of the number of threads, where is the number of steps for the integral function.

# Dynamic loop scheduler

**Write a dynamic loop scheduler to compute numerical integration. Use dynamic sched.cpp as a base code. Write the code so that it outputs the integral value on stdout and the time it takes to make the computation on stderr.**

The code is placed in a file named “**dynamic\_sched.cpp”.**

**Report time and speedup across a range of precision, intensity, and synchronization mode.**

The report is attached with the name “**dynamic\_sched\_plots.pdf**”.

Note: Due to non-availability of the cluster for job execution, I have run the code for 1 to 8 and 16 threads on my local machine and not 12 threads. I have run the dynamic scheduler for 16 threads separately on the HPC and plotted it in a file named “**dynamic\_sched\_plots\_16.pdf**”.

**Question: Compare performance at 16 threads across the different synchronization mode. Why are the speedup this way?**

For higher granularity and smaller number of elements, chunks achieve a better performance over threads. Whereas for higher granularity and more number of elements, threads achieve a higher speedup than the chunk synchronization due to the fact that for large number of elements, threads have to pick tasks from a worker thread queue, update local variable less number of times for than chunk synchronization updating the shared variable acquiring locks which increases the tine take and hence the low performance. The iterator is the sequential form and cannot more achieve speedup or cannot achieve speedup as much as the chunk or thread synchronizations.

**Question: For thread level synchronization, compare the performance at 16 threads of different n and intensity. Why are the plots this way?**

As I was not able to run the whole of bench\_dynamic.sh, my plots are divided into sets of 1 to 8 threads and 16 thread plots.

# Extra Points:

### Gantt Chart

The modified code for Gantt chart is placed in GanttChart folder. I have made the program to output the start and execution time of a thread for the task. But, I was not able to use GNUPlot to create a gant chart. So, I have manually created Gantt charts for a run of both static and dynamic schedulers. I have placed those files in **GanttChart/Reports** folder.