Assignment-3

Deadline: Feb 17, 2020

All the assignments have to be supplemented with a **brief write-up with the following details** (wherever necessary):

1. Context:
    - Brief description of the problem.
    - Complexity of the algorithm (serial).
    - Possible speedup (theoretical).
    - Profiling information (e.g. gprof).
    - Optimization strategy.
    - Problems faced in parallelization and possible solutions.
2. Hardware details: CPU model, memory information, no of cores, compiler, optimization flags if used, precision used.
3. Input parameters. Output. Make sure results from serial and parallel are same.
4. Problem Size vs Time (Serial, parallel) **curve**. Speedup curve. Observations and comments about the results.
5. If more than one implementation, curves for all algorithms in the same plot.
6. Wherever necessary use log scale and auxiliary units.
7. Problem size vs. speedup curve.
8. No. of cores vs. speedup curve for a couple of problem sizes.

Problem-1

Write a basic parallel code (openMP) for integration using trapezoidal rule. Serial/parallel code was discussed in the class.

Use the parallel code to calculate PI and verify the implementation.

Try to optimize your parallel code (more than one version by implementing small changes in the basic code) and compare the performances.

In addition to 8 points discussed above, make comments about your observations and the most optimized implementation.

Measure performance in MFLOPS/sec.

Problem-2

Write a **serial code** and **parallel code** (using openMP) for the following :

a. Summation of two vectors (serial and parallel comparison for accuracy of results).
b. Multiplication of Two vectors followed by summation.

In the report you must include important observations (point wise and brief), challenges faced while parallelization and how you addressed those issues. How you could improve performance by following different strategies for the same problem.

Make sure go from small problem sizes (2^6) to big problem size at least (2^28).

Problem-3 (starting with your codes developed in previous assignment)

Parallel Matrix Multiplication (MM)

- simple parallel implementation (different loops, different cases with different openMP directives/pragmas)
- Create n x n matrix; take n in multiples of 2 while increasing the size of matrix [e.g. 4, 8, 16, 32, 64 go up to 528 or even 1024 (serial will take large amount of time)].
- now implement parallel **Block MM algorithm**
- Change size of block to see the effect.