SVD: Programming Assignment 1

Part 1: Parallelization and Validation: due Friday, February 17, 11:59PM

- Parallelize the SVD code from the provided sequential code SVD.cpp using OpenMP
- **Compile sequential code**: Type "g++ -O3 SVD.cpp –o SVD" or "icc –O3 SVD.cpp –o SVD"
- **Run sequential code: Fi**rst generate an input matrix by running "python randomMatrix.py <M N >", providing values for M and N that are square (for simplicity). That is, if you want a 16x16 matrix, then type "python randomMatrix.py 16 16". Then you can run the sequential code by typing "./SVD 16 16 –dm" to generate output for Matlab (see file header).
- **Compile parallel code:** g++ uses –fopenmp flag, icc uses –qopenmp flag
- **Validation:** We are providing Matlab or Octave Validation scripts where you can compare against their calculations. For Matlab, type "Matlab < ValidationM.m". If you prefer, you can use the open source Octave, the flag "-do" to SVD and the validation program ValidationO.m.
- **What to turn in:** Provide a report file presenting the behavior of your code with different matrix sizes, in powers of 2 starting at 32. How does varying number of threads affect result? How does varying loop level affect result?
- Optional arguments to SVD:
  -t : print the number of iterations and execution time
- -p : print the U,V,S result
- -dm or -do : generate the matrix files for Validation.m

SVD Programming Assignment 2

Part 2: Performance Optimization: due Friday, March 3, 11:59PM

- Evaluate performance optimizations to improve performance
  - Change to parallelization strategy
  - g++ vs. icc (for icc, type “module load intel”)
  - -O3 vs. lower optimization levels
  - Locality optimizations
  - SIMD AVX instructions
  - Others
- What to turn in:
  - Report describing impact of optimization strategies (good and bad).
  - Grade will be based on effort and insight, with some points awarded for final performance.