Lecture 14
Final Projects

Jianwen Wei
Center for HPC, SJTU
Jun 2019
The project report for CS075

• Picking one or more projects to finish a report for your final scores of CS075.

• Length: >2 pages. Written in English.

• Due: **23:59 Jul 5th, 2019**

• Work in the project should be done independently and can be reproduced.

• Sample codes can be found [https://hpc.sjtu.edu.cn/info/1016/1697.htm](https://hpc.sjtu.edu.cn/info/1016/1697.htm). Code on github can be used in the project report.

• To get a high score, you can try
  – Finishing more than one project.
  – Finishing the "extra credits" part in projects
  – Fixing bugs in the project instructions.
  – Presenting what you learn in this course

• Send your report and feedbacks to TA: huhang123@sjtu.edu.cn. Read the docs [https://pi.sjtu.edu.cn/doc/](https://pi.sjtu.edu.cn/doc/) carefully before sending your issues to TA.
A project report should contain

• A brief introduction of SJTU π supercomputer.

• Background of your applications and motivation of your work.

• Experiment setups.

• Performance analysis.

• Summary
Project 1: OpenMP

★☆☆☆☆

- Running an OpenMP program to calculate the value of $\pi$ and comparing performance against the number of cores.

- Reference instructions:
  - [https://computing.llnl.gov/tutorials/openMP/](https://computing.llnl.gov/tutorials/openMP/)
  - [http://hpc.sjtu.edu.cn/info/1016/1477.htm](http://hpc.sjtu.edu.cn/info/1016/1477.htm)

- Extra credits
  - Comparing performance between cases with and without false sharing between threads.
  - Comparing performance between different compilers
  - Comparing performance between different OpenMP scheduler strategies
  - Running and analyzing performance of OpenMP programs other than $\pi$ calculation
Project 2: MPI
★★☆☆☆

- Running an MPI program to calculate the value of π and comparing performance against the number of cores.

- Reference instructions:
  - https://computing.llnl.gov/tutorials/mpi/
  - http://hpc.sjtu.edu.cn/info/1016/1477.htm

- Extra credits
  - Comparing performance between different compilers and MPI suites
  - Comparing performance between cases where MPI processes within a same node and among different nodes.
  - Running and analyzing performance of MPI programs other than π calculation
Project 3: RDMA

★★★★☆☆

• Running a RDMA program two nodes and evaluating RDMA performance in messages per seconds v.s. cpu utilization.

• Reference instructions:
  – https://community.mellanox.com/docs/DOC-2493
  – https://github.com/efficient/rdma_bench

• Extra credits
  – Communicating between more than 2 nodes.
  – Comparing performance between TCP/IP and RDMA when doing similar operations.
  – Trying more than one RDMA protocol and comparing their performance and programmability.
  – Running a real application (for example, scp) on RDMA and comparing performance between IP and RDMA.
  – Running RDMA Benchmarks
Project 4: Apache Spark
★★★★☆

• Starting a two-node Spark cluster then running SparkPi program to calculate \( \pi \).

• Reference instructions:
  – https://pi.sjtu.edu.cn/doc/spark/
  – https://github.com/LLNL/magpie

• Extra credits
  – Running other Spark programs.
  – Running other BigData Analysis stacks on Spark
  – Launching Spark programs via Magpie scripts
  – Using RDMA (for example IPoIB) for inter-node communication
Project 5: CUDA

★★★★☆☆☆☆☆☆☆☆

• Running the CUDA code to calculate the matrix-matrix multiplication and compare the performance with the sequential code to get the speedup.

• Reference instructions:
  – https://pi.sjtu.edu.cn/doc/samples/

• Extra credits
  – Optimizing the CUDA code using shared memory.
  – Optimizing the matrix-matrix multiplication algorithm with blocking.
  – Could the input matrix be with any size not only the square matrix?
Project 6: OpenACC

★★★★☆☆☆

• Add OpenACC directives to the matrix-matrix multiplication C code and run the OpenACC with PGI compiler on GPU card.

• Reference instructions:
  – https://pi.sjtu.edu.cn/doc/modules/

• Extra credits
  – Comparing performance with different compilers options.
  – Try different gang, vector size to achieve the best performance on GPU.
  – Running the OpenACC code on x86 multicore CPU.
Project 7: PyTorch CNN ★★★☆☆☆☆☆

• Run a CNN program on PyTorch.

• Reference Instructions
  – https://github.com/MorvanZhou/PyTorch-Tutorial

• Extra credits
  – Comparing performance between CPU and GPU.
  – Comparing performance between different settings of hyper parameters.
Project 8: PyTorch RNN
★☆★★★★☆

• Run a RNN program on PyTorch.

• Extra credits
  – Comparing performance between CPU and GPU.
  – Comparing performance between different settings of hyper parameters.