Damian Franco

Meiling Traeger

CS-542

**Final Project Update**

*Progress:*

Our project is focusing on learning one-sided communication with MPI and teaching others within the class about our findings. We stated in the proposal that we will be specifically focusing on developing an understanding of the Post-Start, Complete Wait (PSCW) Generalized Active Target Synchronization technique which is still a great priority of ours, but our efforts have expanded. We now have the knowledge to try to understand and teach other synchronization techniques such as the Fence and Lock/Unlock techniques. Our main matrix-matrix multiplication implementation will still be using PSCW synchronization.

So far, we have done an intensive deep-dive into the intricacies of how one-sided communication in MPI operates and the components of it. We both have compiled a singular list of our individual understandings of one-sided communication and have discussed with each other the best way to understand how it works, the difference between two-sided and one-sided communication in MPI, and the implementation of one-sided communication with a matrix-matrix multiplication example. We have combined the seemingly countless resources of research papers, online articles and example implementation code into a list which we will include in the final paper’s resource section.

We have both put in a great amount of time into understanding the topics as well as actually implementing the code for our matrix-matrix multiplication example. First, we wanted to compare how one-sided communication performs compared to two-sided communication so we implemented a two-sided, very simple matrix-matrix multiplication. This code is fully implemented and will be provided via a Github repository alongside the final paper and poster. We have run 10 tests and took the median runtime for those and placed it in a plot to show the runtime with different data sizes as the input. We are currently developing the code for the one-sided communication matrix-matrix multiplication using the PSCW synchronization technique and have a promising development. The program is currently running with no runtime errors. Currently, we have some minor calculation bugs which we will smooth out before running multiple attempts and comparing runtimes with the two-sided communication. Since reading multiple articles has shown that two-sided may be more efficient than one-sided in scenarios such as matrix-matrix multiplication, we are extremely excited to see how both implementations compare since some articles or papers say one-sided communication performs better and some say the opposite. We will have the plots for the implementation’s performances provided in the final paper and poster.

*Future Work:*

To conclude our work with this project, we will continue to debug the code for the one-sided communication matrix-matrix multiplication example and run the program multiple times to get a median runtime. After, we will take the runtimes and plot them against the two-sided implementation and gather results. We expect, and already experienced, issues to come with the runtimes and implementations, but will take them with open minds because it will provide us with valuable learning experience of our topic. We will then compile our results and research into a paper and poster to teach others about one-sided communication and its use cases. Our goal is to make an easy to understand user experience when teaching others during our poster presentation about one-sided communication, synchronization techniques and implementations. We look forward to continuing gaining knowledge about one-sided communication in MPI and informing others about it!