# 15-418 2021 Spring

Diego San Miguel (dsanmigu)
Nathan Ang (nathanan)

# Distributed Machine Learning Using MPI and CUDA

## Final Project Proposal

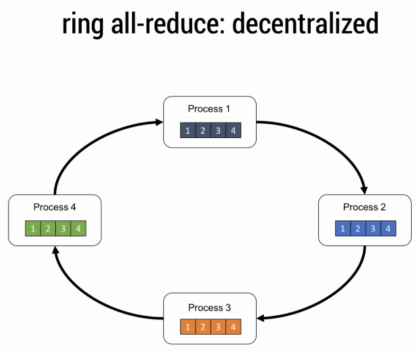
April 7th 2021

## Summary

For our project, we will implement distributed Machine Learning using Message Passing Interface and CUDA in efforts to speed up the training time of a simple neural network. Specifically, we will use MPI to enable parallel gradient computation and Ring-AllReduce for convergence. We will implement N-bounded delay Bulk Synchronous Parallel (BSP) Execution, as well as exploring and analyzing different approaches to the challenges it entails, such as consistency and synchronization. If time permits, we will be using CUDA in efforts to speedup the batch gradient computation.

## Background

One of the most significant challenges of deep learning is the very large amount of time necessary to train models. Given that the performance for deep learning distributed among multiple systems has the capability to outscale the performance of a singular machine, it is no surprise that distributed machine learning is a very active field of research.




In distributed machine learning systems, we partition the data amongst all nodes in our system, splitting the computational and storage requirements for every node. Nodes then communicate to each other their model parameters after every iteration or epoch through an algorithm called All-Reduce allowing them to compute more accurate parameters without having to do all the computation on the given data.

Ring All-Reduce is a decentralized communication method between nodes that allows them to efficiently communicate their model parameters to each other without bloating the network with unnecessary communication. It works in two phases:

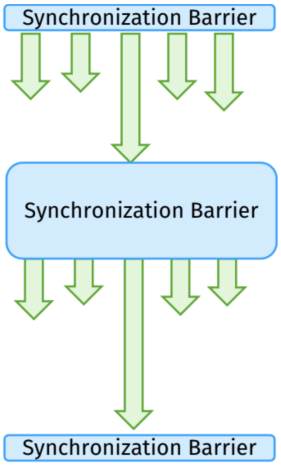
1) Each process p sends data to the process (p + 1) % P (P = num of nodes). The array of data of length N being shared is divided by p, where each process is responsible for sending one of the data chunks. This process of sharing occurs p - 1 times and each time a process receives a piece of data it applies the reduce operator to it.

2) Once each process holds a complete reduction of their share of the data, the share-only step begins. Each process forwards the data it has in a ring-like fashion p - 1 times such that each process will finish with all of the reduced data.

This algorithm, executed in a bulk synchronous parallel (BSP) model, can be wasteful of computing resources. This is because there can be no communication during computation, and no computation during computation. In order to combat this, we can perform N-bounded delay bulk synchronous parallel execution. Essentially, every node is allowed to proceed up to N epochs or episodes ahead of the slowest node if necessary, in order to more efficiently use its resources and time.

## Challenges

Distributed Machine Learning is currently a very active field of research due to its various challenges.




One significant challenge is communication overhead. Because we are distributing work amongst different machines, communication between threads at each iteration is necessary for useful results. As a result, scaling out is a concern when it comes to Distributed Machine Learning. To tackle this, we plan to implement an efficient ring All-Reduce with MPI, which will reduce the communication overhead.

With Bulk Synchronous Parallel (BSP) Execution, synchronization is a challenging topic. In the BSP model, machines often work on their own computations, and then synchronize at a barrier to ensure consistency. However, we can encounter stragglers, which will cause significant slowdown during barriers. As mentioned before, we plan to explore this challenge with analysis of various levels of synchronization strictness with N-bounded delay.

Somewhat related to the synchronization topic is the tradeoff between throughput and progress. This is a key design decision to explore, as the relaxation of synchronization will allow for higher throughput, but may result in less progress per iteration. A poor balance between the two will result in lower performance. This is a twist on the 15-418 concept of throughput which we will explore.

Another challenge is the fact that it is possible for distributed machine learning to simply not be the best solution for some machine learning problems. Pure performance is one thing, but the complexity of implementing distributed machine learning is also a factor, hence the growing interest in creation of distributed machine learning frameworks and APIs. We plan to explore this experimentally to see how the performance of our completed Distributed Machine Learning implementation will perform against a sequential implementation on the same problems.

## Resources

We plan to use MPI on the GHC machines where each MPI process will be on a different machine in the cluster. If we get past what we plan to achieve, we intend to then use the RTX cards on each GHC machine, in order to run a simple CUDA implementation of a neural network, and achieve even greater speedup of a deep neural network training process.

Testing and training data necessary for performance analysis will be gathered from an open source kaggle dataset.

## Goals and Deliverables

- What we plan to achieve
    1. Setup MPI to use multiple different machines in the GHC cluster
    2. Implement the ring-allreduce algorithm using MPI so that it can communicate between different machines in the GHC cluster
    3. Modify the ring-allreduce algorithm to support a N bounded delay BSP execution model
    4. Implement a simple neural network, using a library or our code from a previous ML course, and modify it to work with the distributed system.
    5. Find suitable large datasets to test our system on
    6. Analyze the performance of our distributed ML system against a sequential and single node implementation. Chart these performances and calculate speedup
    7. Analyze different sections of the system to determine bottlenecks and any system characteristics causing less than linear speedup

- What we hope to achieve
    1. Implement the neural network in CUDA and use the RTX cards on the GHC machines to run our neural network training in a SIMD parallelized fashion
    2. Analyze performance differences between our CUDA implementation of a neural network and our sequential CPU implementation of a neural network

## Platform Choice

Parallel Computing Architectures: MPI and (hopefully) CUDA

Primary Language: C++

## Schedule

April 9th - April 15th

- Conduct research on N-bounded delay BSP parallelism
    - Reach goal: Schedule a meeting with 15-440 Professor Heather Miller to speak in depth about BSP parallelism and all-ringreduce
- Research how to get an MPI program running on multiple GHC machines rather than multiple different cores of the same GHC machine
    - Reach goal: Successfully configure an MPI program to do this
- Find a dataset to test our system on

April 16th - April 24th (Checkpoint Due):

- Find an implemented neural network to test the system with
- Implement all-ring reduce using MPI
- Begin researching how to implement N-bounded delay BSP through MPI
    - Reach goal: Get a starter version of this working

April 26th - May 2rd:

- Finish implementing N-bounded delay BSP
- Measure speedup and performance of system with different number of nodes and bound sizes, compare to sequential single node implementation

- Create graphs visualizing our speedup and performance
    - Reach goal: Begin researching and speculating what is limiting our performance, and how the system could theoretically be improved

May 3th - May 10th (Reach goals):

- Implement a simple neural network in CUDA
    - Single hidden layer with sigmoid activation units
    - Softmax output layer
    - Parallelize feed forward and backpropogation process
- Analyze speedup of CUDA neural network relative to sequential premade neural network
- Analyze speedup of whole system relative to sequential single node system
- Make progress on report, detailing what may be causing less than perfect speedup

May 11th - May 13th (Report Due):

- Create and host a website through github pages
- Finish project report on hosted website