Zeb Becker, Alex Racapé
Professor Barker
Distributed Systems
12/3/23

Checkpoint 2

Since the last checkpoint, we made significant changes to our design, and we are now implementing a distributed system for federated learning. This is another framework for distributed machine learning, except data is decentralized and privately maintained by each peer. In our architecture there is a central coordinator machine that keeps track of the workers, and the workers train separately on their own dataset. They push updates to the coordinator in batches, and the coordinator merges all the updates into a single model that is then broadcast out to the workers. Since the last checkpoint we coded the majority of the project in the worker and coordinator classes, and our scripts for deployment are much better.

The main challenge we encountered was that the coordinator has to synchronize things and wait for enough workers to respond in order to proceed onto the next round of training. We came up with several options for dealing with this using RPCs. The first option was for the coordinator to keep sockets open and stall before responding to the workers requests. The second option was for workers to send their update then poll to check for the new merged model. The first option would make the coordinator a bottleneck as the number of workers scales, and the second option would add significant network traffic, especially if training times vary a lot between workers. We ultimately implemented a third option which was to add an RPC server to each worker. This means that the coordinator can notify workers when a new updated model is ready, avoiding the need for polling and stalling responses. Workers keep this server running in a background thread, and it is used to watch for notifications from the central server.

We also set up model instantiation, training, and a test problem for the system to work with. For now we are using the classic MNIST digit dataset for image classification, but in the future we have the flexibility to adapt our system to almost any problem. We have started to test our system, and we have trained a model with a single worker communicating with the coordinator for several iterations. Going forward, we want to add some features for load balancing and shutting down the training process. We also want to perform more rigorous testing.