Addressing the straggler problem for parameter server(CSCI 5570 proposal)

authors

November 2, 2018

In our experiments, we have found that there are some straggler problems in the parameter systems. Hence, in this proposal, our group would propose some half-baked ideas. And at the end of this proposal, we will give a schedule about the implementation.

1 Introduction

In the past nine weeks, our group have implemented the baseline of the parameter server system. However, we observe some straggler problems. Under Bulk Synchronous Parallel (BSP) computational model, all workers have to wait until the slowest worker finishes the computation in each iteration. The straggler problem will affect the performance of the overall system, especially when the number of the workers increases. In such situation, the computation resource will be wasted for most of the workers are idle.
There are numerous reasons which lead to the straggler problems, e.g. unbalanced data distribution, hardware heterogeneity, resource contention and so on. In addition to straggler problems, we noticed that overhead of network communication will be a significant problem as the number of the workers increases. We need some optimization to eliminate the communication overhead.

2 Possible solution

The first possible solution is that we try to make the data balance across the workers.
The second solution is that we schedule the number of training data before each iteration. To achieve this goal, we will transfer a portion of the training data from straggler worker to another worker. To avoid massive data transport between workers, similar to redundancy strategy in consistent hashing, we can keep a backup data partition of next node in every node. So that each node is the backup node for its next node. When a straggler worker is detected, its

backup node can start work steeling if its possible. After this iteration, the workload can be transfer to backup worker permanently.
As for communication pattern optimization, one natural solution is to merge redundant messages on the worker side. Another idea is to use shared memory for retrieving/ updating parameters when servers and workers are on the same node, and this is called local zero-copy communication. Based on these two idea proposed in [FlexPS], we can further boost the systems performance. To make full use of local communication, we can scan the dataset to partition data by their related parameters key. To be specific, data should be partitioned to the node in which the parameters and data are most correlative. In this case most of the GET/PUT request will be sent to local server thread. In order to achieve this goal, we can define a correlation metric between data and parameter partition called correlative score. Top k data with highest correlative score with a parameter partition will be assigned to that parameter partition to ensure evenness of data partition. This is based on the assumption that connections between data and parameters are sparse and evenly distributed.

3 Schedule

1. **Week 1. Literature review.**
2. **Week 2. Implementation.**
3. **Week 3. Testing.**
4. **Week 4. Optimization.**