University of Tromsø

Assignment 1- Tensorflow

INF-3203

Camilla Stormoen

Department of Computer Science

March 7, 2017

1 Introduction

This report describes the evaluation of a Tensorflow computation with different number of workers. The tensorflow computation is MNIST, which is a simple computer vision dataset containing images of handwritten digits.

2 Experimental evaluation

2.1 Experimental setup

The MNIST is a simple computer vision dataset of imgages of handwritten digits. In this implementation, we're going to train a model to look at the imgaes and predict what digits they are. The MNIST computations are configured with a batch-size of 100, a learning-rate at 0.5 and 3 training-epochs. These tests are run from 1 worker up to 10 workers and is run 30 times each. Computation workload is distributed in the cluster using these computers: *Dell Precision T3600 with Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz, 36GB RAM and 1.00 TB DISK*, *Dell Precision T3500 with Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, 12GB RAM and 500GB DISK* and *HP Compaq 8000 SFF with Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz, 8GB RAM and 250GB DISK*.

2.2 Experimental results

Results from computations is shown in Figure 1 and show number of workers from 1 to 10, runned 30 times each. As the figure show, computation time increase when number of workers increase.

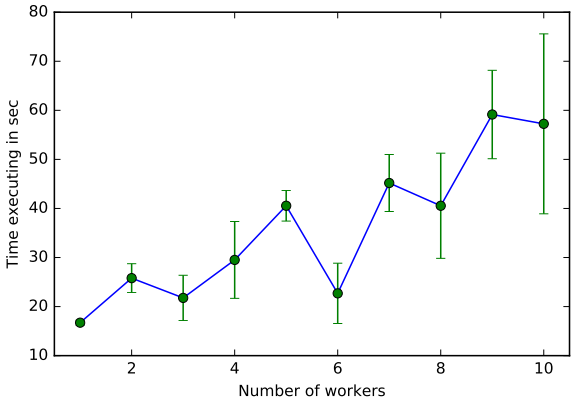

Figure 1: Shows time executing MNIST computation per worker with an errorbar with standard deviation of 30 runs per number of workers.

2.3 Discussion

As we can see from Figure 1, more workers doing computation tends to use more time. One reason for this can be that there is a lot of communication between workers and the chief and this could possibly be a bottleneck. It's the chief that collects all the results when the workers are done. Also the chief only return the result when every worker is done, and some workers finish before others. It will always be the slowest worker that will have the final time-result.

Another reason can be the tensorflow implementation. Overall time went drasticly down when changing learning-rate from 0.001 to 0.5 and training-epochs from 20 to 3. Depending on learning-rate, batch-size and number of

training epochs, there is a possibilty to decrease execution time, however more workers would maybe not solve the problem about distribution. Since the MNIST implementation contains a lot of computations, there is reason to think that this implementation contains a lot of work that is not efficient to distribute in thoughts of problem size, learning-rate, batch-size and training-epochs. Or maybe the worksize isn't big enough to be distributed in a efficient way. A thing to mention, is that the tensorflow version is also outdated with regards to functions that are not necesarry any longer because of their new API. So the implementation use some time to initiate flags and parameters.

The errorbars from Figure 1 also shows that there's a big difference in execution-time with a high number of workers, especially with 8 or 10 workers. A lot of worker can lead to a lot of communication and overhead on the chief who is gathering the results from workers. In this implementation, it looks like running the code with one worker is the best solution, however 3 or 6 worker can also be good.

As the testing went from 1 to 10 workers, throughout the day there was more activity on the cluster which probably lead to a overhead on the cluster because other students running their computations on the cluster aswell. This may affect the result as we also can see from the errorbars.

3 Conclusion

This report concludes the experimental evaluation of a MNIST computation. With this configuration, will not tend to be more effective by distribute the computations. Results shows that this implementation of MNIST is not a good implementation for distribute the problem.