Assignment High Performance Machine Learning Workshop

During the course we have seen Horovod, a distributed training framework. This consists of data-parallel training processes which can speed up training, by aggregating the gradients from the different workers. Let's see how to use this in practice. This will be based on the library Horovod with documentation at https://github.com/horovod/horovod

!! NOTE !!
These assignments cannot be done through Jupyter Notebook but have to be done through the LISA system on the command line. This means submitting to multiple nodes can take a while (try gpu – shared partition with the flag `-p`) You can use the virtual environment that was created during the workshop.

1. For the assignment, get the example from MNIST horovod running in the parallel folder on the MLCourse - github repository.
   Take a look at run.sh, and set the appropriate settings for LISA, for 1 GPU node.
   When the script is running, you should see the steps print and the loss like: `Step #10          Loss: 0.673573`
   While the script is running, start another terminal in Lisa, and run the following command:
   ```
   watch -n0.2 nvidia-smi
   ```
   How many GPU's are utilized, at what percentage? Can you think of a way to increase this percentage?
2. How many steps should be performed to cover the 40 epochs with a batch size of 128 (the number of training images is 60,000)? What is the time required for a full training run on a Lisa GPU node?

>> The total numbers of processes used by Horovod is the `-np` flag which is initially set to 1. You can scale this up by setting it for example it to 4, then it will utilize 4 GPU's

3. What is the validation accuracy for the 1/2/4 GPU versions?
4. Run the resulting Horovod code on 1/2/4 GPU's. What is the scaling efficiency (does it run twice as fast, when using twice the amount of GPU's) when going from 1 GPU to 4 GPU's? What could be the potential bottlenecks?
5. We can also scale to multiple workers, then we have to change the run script, to select multiple nodes, and change the `-np` flag, which are the total number of GPU's. Let's scale up to **2** workers. What happens to the batch size and learning rate? How can we keep the validation accuracy high when scaling up the batch size?
6. What is the best validation accuracy that you have achieved using 2 workers, and the same 40 training epochs?

Your accounts will be valid for a week, and we advise you to use the Lisa GPU partition for all experiments.