Heidelberg University
Institute of Computer Engineering
Computer Engineering Group

Holger Fröning
Bálint Soproni

Introduction to High-Performance and Distributed Computing
Winter Term 2023/2024

Exercise 8

- **Hand in via Moodle until 09:00 on Monday 22 January, 2024**
- **Include all names on the top sheet. Hand in a single PDF.**
- **Please compress your results into a single archive (`.zip` or `.tar.gz`).**
- **Please employ the following naming convention `hpc<XX>_ex<N>.{zip,tar.gz}`, where `XX` is your group number, and `N` is the number of the current exercise.**
- **A maximum of three students is allowed to collaborate on the exercises.**
- **In case an exercise requires programming:**
  - **include clean and documented code**
  - **include a Makefile for compiling**

Notes

- For this exercise you will be using the PyTorch tensor library for deep learning.
- As the name suggests, PyTorch is built on the Python programming language, which might lead you to think about performance implications. However, PyTorch uses highly optimized compute kernels in the back end for heavy lifting and is therefore very fast.
- All programming for this exercise will be done in Python.
- It's best to use Pytorch via Anaconda. Use `module load anaconda/3` to load it.
- PyTorch should be installed in the default conda environment. If you want to use some special features you may create a personal conda environment via `conda create --name <name>`
- Then load the newly created environment via `conda activate <name>`. You can then install any packages you may need using `conda install` or `pip install` (prefer conda if the package is available there).
- To install PyTorch to your new environment copy the conda command from https://pytorch.org/get-started/locally/. Set the CUDA version to 11.6.
- When using CUDA (GPU training) make sure to also `module load cuda/11.6`.

8.1 Training a CNN for CIFAR10

Try to finish this part of the exercise after the first week, as we can discuss interim results that way.

- Start with training a simple convolutional neural network (CNN) for the CIFAR10 classification task. Start with the script provided. After reporting loss, performance (time), and classification accuracy for this initial version, perform the following experiments:
  - Test different learning rates and mini-batch sizes, monitor and visualize the loss curve, find recommendations for a good training.
  - Test different model parameter counts, monitor and visualize accuracy → discuss when overfitting occurs.
  - For debugging, track at least once the ration of weight updates over weight magnitudes (i.e. for either of exercise a or b).
  - Move all training to a GPU, and measure speed-up.
- Notes: feel free to replace the CNN with another architecture (in case you are familiar with some). Keep in mind that training time is scarce, so keep model complexity low.

(30 points)

Holger Fröning
Bálint Soproni

8.2 Distributed Data-Parallel (DDP) training of a CNN for CIFAR10

- While the previous exercise was obviously already training in parallel (hopefully you analyzed CPU and GPU utilization), this exercise is now concerned with going "parallel" with regard to multiple GPUs.
  - Train on multiple GPUs on a single node using the `DistributedDataParallel` construct of PyTorch. Report speed-up of the training for 1-4 GPUs (use octane001 or 002).
    - Bonus points if you compare performance comprehensively against the `DataParallel` construct. Discuss reasons for performance differences.
  - Train on multiple GPUs on multiple nodes using the `DistributedDataParallel` construct of PyTorch. Report speed-up for 1-8 GPUs (use octane001 **and** 002), analyze time per epoch and number of epochs total.
    - Bonus points if you analyze and interpret the network traffic during training. You can use the provided function to get the total bytes sent by the network interface.
- For both:
  - Experiment with training hyperparameters such as learning rate and mini-batch size, report time, loss curve, and test accuracy.
  - Discuss observations.
  - Use CNN from previous exercise or model architecture of your choice.

(30 points + 20 bonus + 10 bonus)

8.3 willingness to present

Please declare willingness to present.

- Training a CNN for CIFAR10
- Distributed Data-Parallel (DDP) training of a CNN for CIFAR10

(30 points + 15 bonus points)

90 points + 45 bonus