Changing gear for accelerating deep learning - the first year operation experience with DGX-1

Yu Wang[1], Janis Keuper[2], Milos Stanic[3], Moritz August[4]

[1]The Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, [2]Fraunhofer-Institut für Techno- und Wirtschaftsmathematik ITWM, [3]FluiDyna GmbH, [4]The Technical University of Munich

Abstract

The rise of GPU computing becomes one of the most important renovations in the computational technology. The recent phenomenal advancement and adaptation of deep learning in many traditional disciplines won't be possible without GPU computing. The Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities deployed several GPU systems, including a DGX-1 and Openstack cloud based GPU virtual servers (with Pascal 100) since the start of 2017. Our users tested scalability of Deep learning on DGX-1, trained Deep learning models for simulating Quantum experiments and performed numerical simulations of fluid motion, utilizing the multiple, NVlinked GPUs on DGX-1. These results demonstrate that GPU based computational solutions, such as DGX-1, are valuable computational assets of the Bavarian academic computational infrastructure.

Scalable Deep Learning

The training of Deep Neural Networks (DNN) is a very compute and data intensive task. Modern network topologies[3,4] require several exaFLOPS till convergence of the model. Even training on a GPU still requires several days of training time. Using multi-GPU system could easy this problem. However, parallel DNN training is a strongly communication bound problem [5]. In this study we investigate if the NVLINK interconnect with its theoretical bandwidth of up to 50GB/s is sufficient to allow scalable parallel training.

We used four popular Convolutional Neural Network (CNN) topologies to perform our experiments: AlexNet [1], GoogLeNet [2], ResNet [3] and InceptionNet [4]. The software-stack was built on NVIDIA-Caffe v0.16, Cuda 8 and cuDNN 6. We used the data-parallel training algorithm for multi-GPU systems [5], which is provided by the Caffe framework.

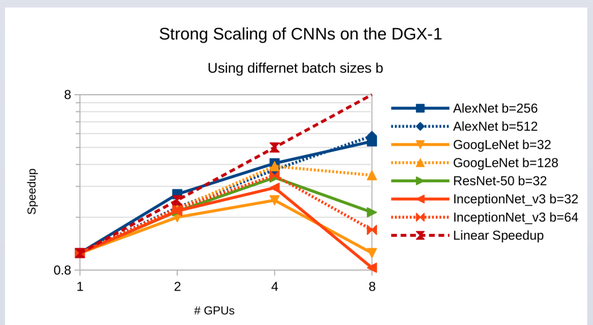

Figure 1 Strong scaling: Experimental evaluation of the CNN training speedup for different topologies and constant global batch sizes b. The smaller batch size is always the one used in the original publication, the larger batch is the maximum possible size given the 16GB memory per P100 GPU.

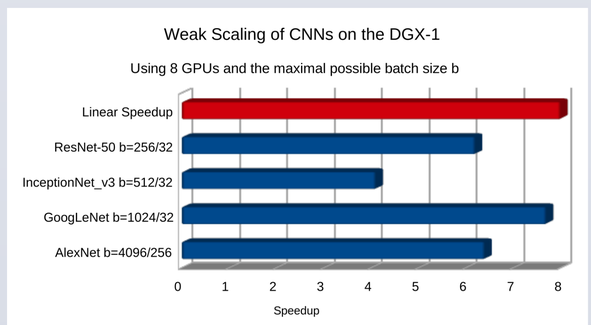

Figure 2 Weak scaling for 8 GPUS: Experimental evaluation of the speedup of CNN training using the maximum global batch size, compared to the maximum batch size of a singe GPU.

Figure 1 shows the results for a strong scaling of the CNN training. Notably, the parallelization shows to be efficient up to four GPUs, but drops significantly when scaling to eight GPUs. One might speculate that this might be caused by the NVLINK interconnection topology of the DGX-1 (shown in Fig 3), where the GPUs are split into two fully connected groups of four. However, looking at the results for AlexNet (which has the largest communication load) shows that the maximum possible batch size is actually the problem. As shown in [5], data-parallel splitting of smaller batch sizes causes inefficient matrix operations at the worker level.

GPU 2 GPU	0	1	2	3	4	5	6	7
0	507.77	36.84	36.84	36.82	36.84	16.91	17.55	17.58
1	36.85	506.4	36.8	36.8	17	36.84	17.07	17.05
2	36.8	36.83	510.73	36.83	17.55	16.91	36.83	17.51
3	36.82	36.79	36.82	510.73	17.55	16.95	17.29	36.84
4	36.82	17.02	17.58	17.56	507.81	36.82	36.83	36.81
5	17.22	36.81	17.36	17.26	36.82	508.05	36.83	36.83
6	17.54	17.17	36.85	17.52	36.84	36.85	509.13	36.84
7	17.59	16.89	17.53	36.8	36.85	36.85	36.83	507.43

Figure 3 Experimental evaluation of the actual bandwidth between single GPUs using message sizes like during training of Alexnet [1].

Large batch sizes can be preserved by a weak scaling approach, shown in figure 2. Using the maximum global batch size leads to better scaling performance. However, it should be noticed that increasing the batch size usually leads to a reduced generalization abilities of the trained model [5].

Shifting gears: gear-train simulations on the DGX-1 using nanoFluidX

Based on Smoothed Particle Hydrodynamics (SPH) method, nanoFluidX (nFX) code is a GPU-based CFD software for numerical simulations of fluid motion, developed by FluiDyna GmbH. nFX is primarily used for simulations of gear- and power-train components in the automotive industry allowing quick execution of transient, multi-phase simulations in complex moving geometries.

Figure 4: Example of a realistic geometry simulation of a single-stage gearbox done in collaboration with Magna Engineering Centre St. Valentin, Austria.

The SPH method is based on an algorithm which is perfectly suited for parallelization as it involves large number of simple computations repeated over regions that are spatially independent. This allows for easy distribution of tasks over threads and efficiently harnesses the power of the GPUs.

Performance and scaling of the nFX code on DGX-1 are shown in Figure 5 and Figure 6. The chosen test case for scaling and performance tests is a single gear immersed in an oil sump. The case contains 8,624,385 particles, which at maximum number of GPU's results in approximately 1 million particles per GPU device. Each case ran for exactly 1000 steps, resulting in minimum run time of 37.78 s and maximum of 2 min 54 s.

It has been noted that scaling on GPU's is heavily influenced by the relative load put on each card. In reality this transfers to the issue of having an upper limit on the acceleration of the simulation for a limited size of the case. As a counter-example, one can imagine having a case with 100 million particles, and scaling would likely be almost ideal in the range 1-10 GPU's, but would likely drop off to about 80% at 100 GPU's.

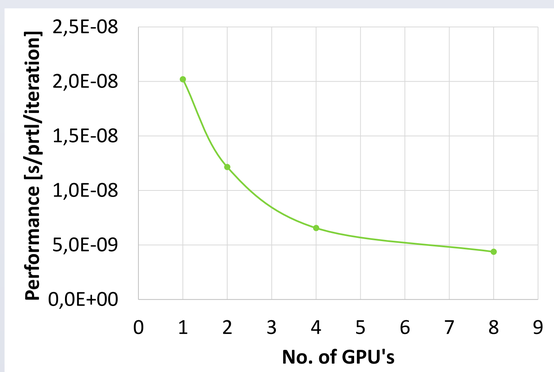

Figure 5 nFX code performance [s/particle/iteration]. It is noticeable that the scaling drops off as number of particles per GPU decreases. This is common behavior under sub-optimal load of the cards as communication becomes more prominent while the GPU memory is under-utilized.

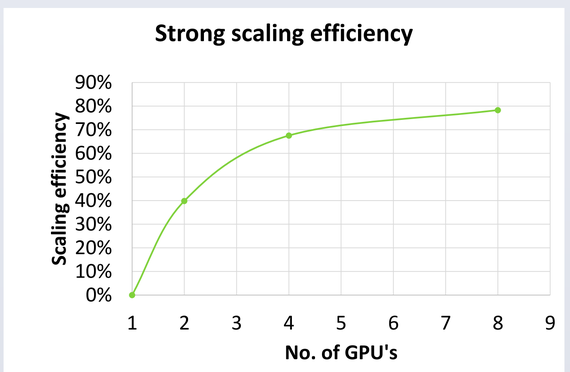

Figure 6 Strong scaling efficiency, calculated as 1 minus the ratio of the single GPU performance to the current number of GPU's. We see that the performance tops off at around 80% efficiency, corresponding to the relative performance drop from Figure 5.

Deep learning models for simulating Quantum experiments

This work aims at developing deep learning models to automatically optimize degrees of freedom and predict results of quantum physics experiments. In order for these algorithms to be broadly applicable and be compatible with the quantum mechanical particularities, i.e. measurements influence the results, we take a black-box perspective and, for instance, do not assume the error measure representing the experiment's result to be differentiable.

August and Ni have recently introduced an algorithm [6] for the optimization of protocols for quantum memory. The algorithm is based on Long Short-Term Memory (LSTM) recurrent neural networks that have been successfully applied in the fields of natural language processing and machine translation. . Tackling this problem from a different perspective, August has now casted it to a reinforcement learning setting where the agent's policy is again represented as an LSTM.

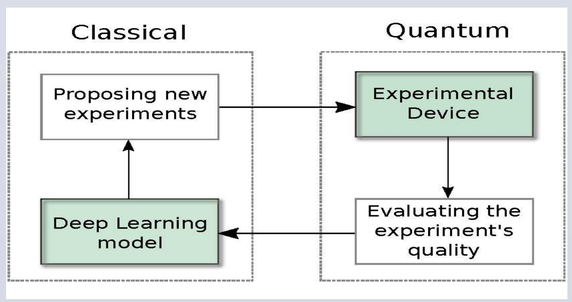

Figure 7 A conceptual illustration of the interaction between a reinforcement learning agent parameterized by a deep learning model and the quantum environment, e.g., a quantum experiment.

Reference

[1] AlexNet: Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[2] GoogLeNet: Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[3] ResNet: He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] InceptionNet: Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[5] Keuper, Janis, and Franz-Josef Preundt. "Distributed training of deep neural networks: theoretical and practical limits of parallel scalability." Machine Learning in HPC Environments (MLHPC), Workshop on. IEEE, 2016.

[6] August, Moritz and Ni, Xiaotong. "Using recurrent neural networks to optimize dynamical decoupling for quantum memory." Phys. Rev. A 95, 012335