Accelerating Deep Learning for Shared Facility Centers Using Tensorflow Framework Analysis Based on IBM POWER Platform

Oleg Nikitin[a] and Olga Lukyanova[a]

[a] *Computing Center of the Far Eastern Branch of the Russian Academy of Sciences, 65 Kim Yu Chena Ulitsa, Khabarovsk, 680000, Russia*

Abstract

In the present paper, the effectiveness of using various versions of the TensorFlow framework on the OpenPOWER platform IBM S822LC is discussed. Special emphasis is given to the conditions of shared facility centers that influence the possible solutions for deep learning research hardware and software. Different versions of TensorFlow were tested, including one specially compiled for the IBM POWER platform. It is shown that TensorFlow version 1.1.0 from Anaconda repositories is 23% less effective than the version from the IBM PowerAI package. At the same time, TensorFlow 1.2.1 surpasses all considered versions due to the efficient disposal of hardware. Thus, we may conclude that the novelty of the version of a deep learning framework is a significant factor and may surpass the special optimization techniques in terms of computing performance.

Keywords 1

Deep learning, high-performance computing, neural networks, TensorFlow, IBM POWER, CIFAR-10

1. Introduction

Modern fundamental and applied research requires the use of ever-greater computing resources, which are difficult to provide for each scientific organization. One of the effective ways to gain access to high-performance computing tools is to work with shared facility centers. The task of such organizations is to provide access to modern research facilities (in this case, high-performance computing systems) and provide appropriate consulting and analytical support.

In recent years one of the most important areas of high-performance computing applications is connected with the use of machine learning methods and, in particular, neural networks. The active development of machine learning (ML), deep learning (DL), and artificial intelligence (AI) technologies have led to the need for a modern computing infrastructure that can provide researchers with high-performance resources for specialized data processing. Systems dominating 5-7 years ago, the computing power of which was provided by a large number of multi-core central processors, are replaced by hybrid ones. In such systems, the main performance in floating-point operations is provided with various kinds of accelerators, and central processors play a rather coordinating and integrating role.

The availability of an optimized set of system and application software that can provide maximum performance and utilization of the resources of the computing system is an important criterion for the formation of an effective shared computing facility. Moreover, the issues of assessing the benefits of various approaches to installing, configuring, and organizing access to the resources of computing systems in a collective access format associated with multi-user and multi-tasking modes of computer systems operations are understudied. There are several approaches to solve this problem. Firstly, it is

VI International Conference Information Technologies and High-Performance Computing (ITHPC-2021), September 14–16, 2021, Khabarovsk, Russia
EMAIL: olegioner@gmail.com (A. 1); ollukyan@gmail.com (A. 2)
ORCID: 0000-0001-9139-5180 (A. 1); 0000-0001-6864-9875 (A. 2)

possible to use integrated software solutions already prepared by manufacturers of high-performance systems, consisting of various assemblies of applied software libraries, virtualization, and resource management tools, etc. These solutions are maximally adapted to the architecture of the hardware system and allow to obtain some performance advantages. However, versions of programs and libraries which are integrated into such software packages often are not the most relevant, thereby limiting users to utilize advanced algorithms and features implemented in older versions of systems.

Another common approach is the independent formation of the software environment of a computing cluster, taking into account its features and operating mode. This allows to adapt to the operating conditions of a system quickly. However, this requires more resources for the maintenance of the computing infrastructure.

Each solution has its advantages and disadvantages, though any supercomputer center considering the transition to the use of modern computing systems needs to make the appropriate choice. This paper is devoted to research in this direction. The paper considers the organization of the computing process for solving problems in the ML/DL/AI field in shared facility center and provides an overview of the approaches which are the most suitable for shared facility centers to organize a software and hardware environment for deep learning, in particular, it examines the preferred environment for the development and operation of neural networks using the TensorFlow framework on the OpenPOWER platform IBM S822LC for HPC Minsky.

1. Analysis of hardware and software environments for deep learning computing

The functioning of computer systems in a format of shared facility center requires organizing software and hardware environment, as well as for setting up and running application software packages on it. The joint execution of various tasks, using primarily the resources of the central processors and computations, including tasks in the field of machine learning with the priority usage of the co-processors resources, complicates the solution to this issue.

Thus, comparison and analysis of their performance, as well as an assessment and implementation of the most effective approach to setting up and operating software tools, are held to select the optimal operational mode of modern computing systems. Hereinafter in the paper, we examine the approaches to the implementation of an integrated software and hardware platform based on the OpenPOWER architecture for completing deep learning tasks using the TensorFlow framework in a shared facility center.

1.1. Hardware environment

The tasks of deep learning include dozens of the typical operations of matrix multiplication, which can be effectively performed in parallel. Currently, there are many technologies to speed up computation and parallelization. There are graphics processing unit (GPU, a specialized processing unit that was mainly designed to process images and videos), field-programmable gate array (FPGA, an integrated circuit which can be "field" programmed to work as per the intended design), and an application-specific integrated circuit (ASIC, a customized circuit for a very specific use, rather than general-purpose) [1]. For promising neural networks with a high degree of sparsity and compact data (1-2 bit resolution) [2] FPGAs will outperform the GPUs in speed due to the high degree of customization of computing. However, nowadays, FPGA acceleration loses in GPU speed for standard types of neural networks. ASIC accelerators are the most promising type of hardware for training deep neural networks, but they are not yet available for order and provided only through cloud services; also these accelerators do not allow universal computing and are adapted only for working with neural networks.

CUDA technology is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). It is called to solve the problem of parallelizing computing on GPUs. Before introducing CUDA, any GPU computing required writing low-level code, which limited the spread of GPU acceleration in deep learning tasks.

Today, the use of CUDA allows accelerating the training of neural networks by more than 50 times [3]. Thus we examine the acceleration performance of modern deep neural networks framework on the GPU-enabled hardware.

The shared facility center "Data Center of FEB RAS" (hereinafter - SFC "Data Center of FEB RAS"), with the resources of which the described computational experiments in this paper are conducted, is a shared facility center that provides resources for performing calculations of a wide class of problems in the interests of scientific and educational organizations. Therefore, the selected cluster nodes for it are Sitonica PW22LC servers (IBM Power Systems S822LC 8335-GTB) with the following configuration:

- two IBM POWER8 processors with a maximum frequency of 4.023 GHz,
- two NVIDIA Tesla P100 GPU co-processors,
- 256 GB DDR4 RAM,
- two Seagate 1TB 7200RPM hard drives and an InfiniBand EDR controller.

Below, we will consider options for installing software for the presented architecture used by the SFC "Data Center of FEB RAS" for solving deep learning problems using the TensorFlow framework as an example.

1.2. Software

The tasks of deep learning include dozens of the typical operations of matrix multiplication, which can be effectively performed in parallel. Currently, there are many technologies to speed up computation and parallelization.In matters of productivity and speed of deep learning frameworks, the version and method of installing the framework play an important role. So, for example, if we compare with the same version of TensorFlow installed using The Python Package Index (PyPI package management system, which used to install and manage software packages written in Python), thanks to the use of TensorFlow, imported from Anaconda library Intel®MKL-DNN, we see that installing TensorFlow using Anaconda (a distribution package for managing and deploying Python packages) increases the speed of neural networks up to 8.6 times [5].

In this paper, we consider the efficiency of launching neural networks developed with the use of the TensorFlow framework, depending on the version of the framework.

For the OpenPOWER system IBM S822LC for HPC Minsky, which the authors of this paper worked with, there are several options for setting up and installing the TensorFlow framework: provided as part of the IBM PowerAI software [6] and distributed through Anaconda repositories. Two of these options for the installation sources of the framework can affect system performance since the IBM version announced optimizations related to the features of CPU POWER 8, as well as the architecture of the Minsky platform.

There are several installation options for working with TensorFlow on servers based on Power8 processors (ppc64el architecture) [7]:

- Assembling from the source code,
- Use of versions compiled for this platform, in particular:
  - available in the Anaconda repository,
  - provided as part of the IBM PowerAI ML/DL software pack.

Often unexpected difficulties in the form of unmet dependencies and the need to adjust the build mode for the target architecture come along during the compiling of programs from the source code for the POWER architecture. This makes compiling a difficult task for AI developers, whose competence often does not include low-level system administration. The study of programs compiled from source code is beyond the scope of this paper. Therefore, we will focus on the use of installation types available to the average neural network developer.

If TensorFlow from the Anaconda repository is a pre-compiled publicly available version for ppc64el architecture, then PowerAI is an IBM-distributed software pack optimized for use on Power Systems S822LC servers. PowerAI is a collection of the most popular open-source frameworks for deep learning, along with supporting software and libraries in a single installable package. It helps to simplify the installation process of neural network tools and consists of a set of binary distributions of several custom-tuned neural nets (software) and some associated custom NVIDIA GPU DL libraries compatible with machine learning tasks. As a software suite, PowerAI is designed to run on a single one of IBM's highest-end Power servers, the Power S822LC for high-performance computing (HPC), but also to scale up from one to many supercomputing clusters. IBM PowerAI software is "curated, tested, and pre-packaged distribution of the major deep learning frameworks"[6]. PowerAI includes all the most important modern frameworks for deep learning [4].

This paper discusses PowerAI version 1.4.0, which includes the following packages: Caffe – 1.0.0, Caffe (IBM) – 1.0.0, Caffe (NVIDIA) – 0.15.14, Chainer – 1.23.0, DIGITS – 5.0. 0, TensorFlow – 1.1.0, Theano – 0.9.0, Torch – 7. IBM has taken steps to optimize TensorFlow as part of PowerAI for usage on the IBM Minsky platform. These efforts should lead to better performance of this framework compared to the general version Anaconda.

2. Materials and Methods. Test task and network architecture

The classic problem of classifying images into 10 classes based on one of the most popular data sets for computer vision problems – CIFAR-10 [8] is taken as a test problem. This data is a set of 60,000 images 32 * 32 pixels in size with marked belonging to one of 10 classes. The task is to achieve the most accurate recognition of the presented image's class.

The classification task of CIFAR-10 is solved using a convolutional neural network containing 2 convolutional and 2 fully connected layers (Figure 1). Available startup options for 1 and 2 GPUs on the IBM site are examined.

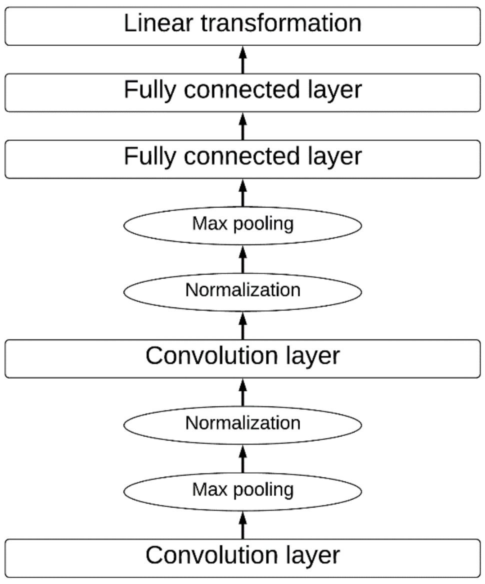

Figure 1: The deep convolutional neural network used for the CIFAR-10 task.

Thus, on the IBM Power Systems S822LC platform, TensorFlow version 1.1.0 from Anaconda and PowerAI are compared, as well as TensorFlow version 1.2.1 from Anaconda, to perform the task of classifying images using a convolutional deep neural network.

Below we compare three options of the TensorFlow framework installation:

- Anaconda -- version 1.1.0 (Av1.1.0),
- PowerAI -- version 1.1.0 (PAIv1.1.0),
- Anaconda -- version 1.2.1 (Av1.2.1).

3. Results. Evaluation of TensorFlow acceleration on IBM Power system

We compare two versions of TensorFlow (1.1.0), as well as version 1.2.1, which was the latest available on servers with the ppc64el architecture at the time of the relevance of the PowerAI version

1.4.0. According to IBM, TensorFlow version 1.1.0 of PowerAI is superior to the standard TensorFlow version 1.1.0. At the same time, the performance of the more recent Anaconda versions of TensorFlow could be improved.

3.1. Key indicators of neural network training

An analysis of the network training efficiency with a batch size of 2048 and a stopping criterion of achieving an error of 0.7, which is corresponded with an accuracy of about 80%, is given below.

Table 1 shows the following indicators for various launch options: total training time until the stopping criterion is reached in seconds; the accuracy that the network showed on the test sample after stopping training; the number of steps that have been spent for various setups; and the time taken by one training step. The best results for options with 1 and 2 GPUs are shown in italic and bold, respectively.

Table 1
Key indicators of neural network training using various versions of the TensorFlow framework. 1 or 2 is the number of GPUs.

Head 1	Av1.1.0		Av1.2.1		PAIv1.1.0	
	1	2	1	2	1	2
Training time (sec)	1680	1753	*845*	**548**	1458	1425
Test accuracy	0.803	0.809	*0.811*	0.812	0.810	**0.817**
Number of steps	*561*	583	572	561	581	**551**
Time by one step (sec)	0.268	0.293	*0.109*	**0.070**	0.219	0.249

Table 1 shows that the version of the TensorFlow framework provided as part of the PowerAI software pack surpasses (by 15-23%) the corresponding version of TensorFlow from the Anaconda open repository. At the same time, the most recent version of TensorFlow 1.2.1 significantly surpasses all other installation options. Besides, this version achieves efficient parallelization on 2 GPUs. There is no apparent stable increase in the training speed under the same launch conditions for TensorFlow 1.1.0 with the use of 2 GPUs. At the same time, TensorFlow version 1.2.1 on two GPUs achieves 54% faster training speed than on one.

Also, analyzing Table 1, we can see that training on two GPUs led to slightly greater accuracy of the network in the test sample. This is natural since parallelization into two accelerators leads to the separation of the batch size and further averaging of the obtained error gradients. This procedure leads to a better generalization of the network. Thus, given the same accuracy for the training set, the accuracy on the test set differs.

A mutual comparison of training speeds with different settings is shown in Table 2.

Table 2
Relative acceleration of network training, depending on the application of different versions of the TensorFlow framework and the number of GPUs used. 1 or 2 is the number of GPUs.

Head 1		Av1.1.0	Av1.1.0	PAIv1.1.0	PAIv1.1.0	Av1.2.1
		1	2	1	2	1
Av1.1.0	2	-4%	-			
PAIv1.1.0	1	+15%	+20%	-		
PAIv1.1.0	2	+18%	+23%	+2%	-	
Av1.2.1	1	+99%	+107%	+72%	+69%	-
Av1.2.1	2	+207%	+220%	+166%	+160%	+54%

As can be seen in Table 2, the use of TensorFlow version 1.2.1 in combination with several GPUs increases the training speed up to 220%.

3.2. Evolution of the neural network error parameter

Figure 2 shows a comparison of the evolution of the neural network error parameter for different versions of the framework.

Here we can see that the dynamics of the network error parameter for Av1.1.0 and PAIv1.1.0 occurs similarly, but the PAIv1.1.0 option learns faster. This suggests that IBM experts made efforts to optimize the version of TensorFlow from IBM PowerAI, in comparison with the similar version built by the open-source community.

Nevertheless, the option with TensorFlow version 1.2.1, available through Anaconda repositories at the time of the PowerAI relevance, significantly outperforms both options of TensorFlow version 1.1.0. The launch with two accelerators is especially representative. It can be seen that in this realization at step 380, an acceptable network loss function value of less than 1 is already obtained. Thus, the results are sufficient for use in some practical applications. Such a loss value is obtained, first of all, due to the efficiency of the parallel operation of two accelerators in the TensorFlow framework version 1.2.1.

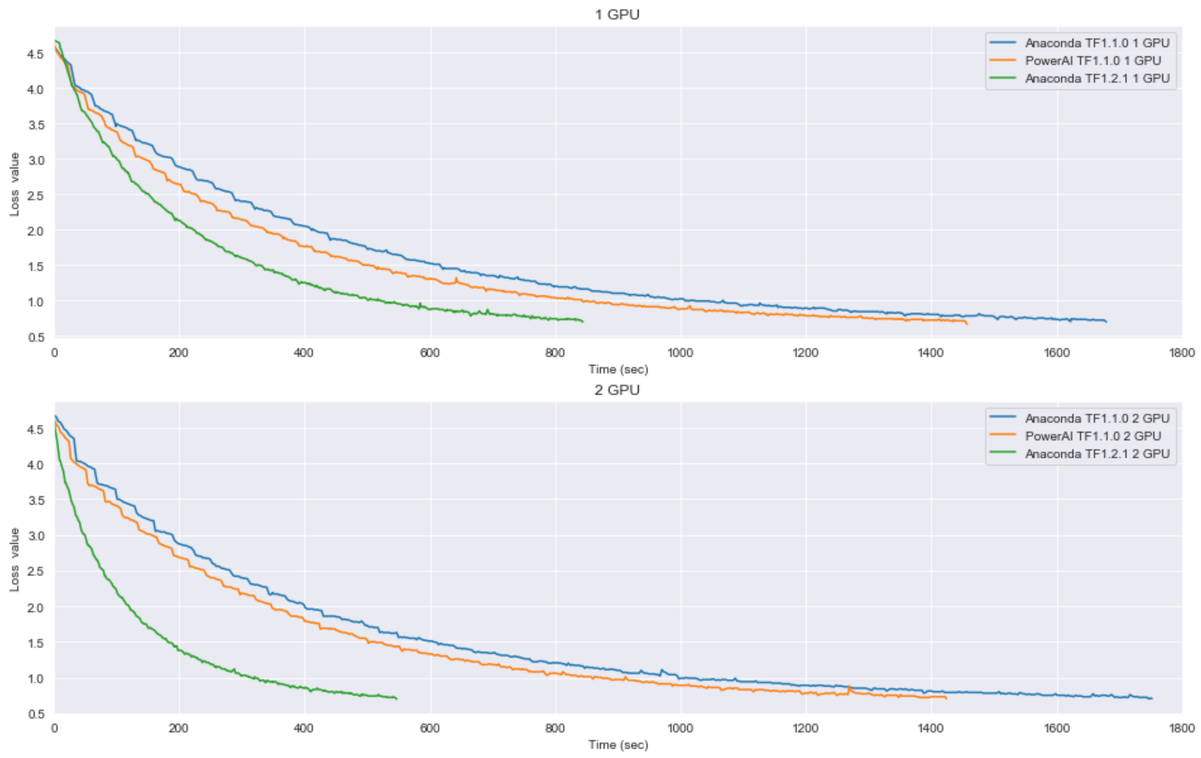

Figure 2: Different TensorFlow versions learning curves (top – one GPU option, bottom – two GPU).

3.3. The dynamics of the learning process and execution time

Figure 2 shows that for the options Av1.1.0 and PAIv1.1.0, the dynamics of the learning process on one and two GPUs do not change much, while for the Av1.2.1, training on two accelerators is significantly (54%) faster. Note that TensorFlow version 1.2.1 (Av1.2.1) is available simultaneously with PowerAI 1.4, which included TensorFlow version 1.1.0 (PAIv1.1.0). We can conclude that it is necessary to install the newest version of the framework compared to a specially optimized one.

Figure 3 shows the dynamics of the execution time of each step of the training.

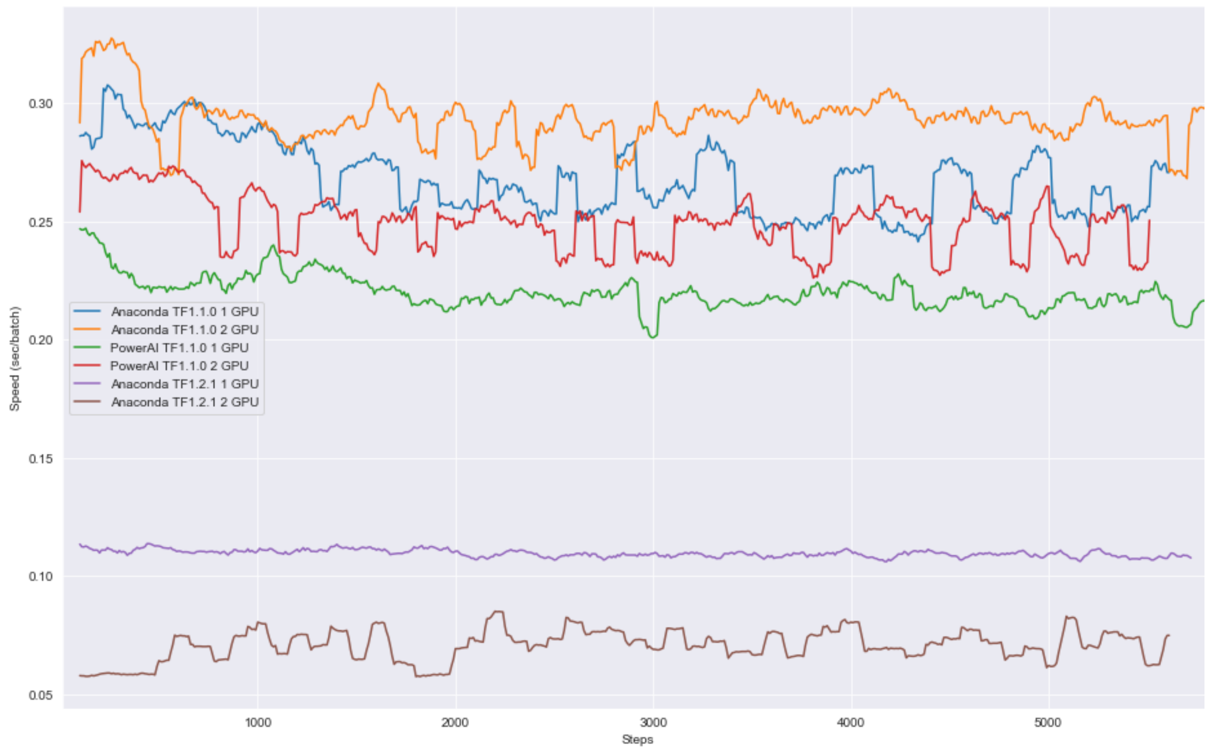

Figure 3: Step performance dynamics for different versions of TensorFlow.

It could be seen that for the TensorFlow version 1.1.0, parallelization slows down the execution of one step. At the same time, the TensorFlow version 1.2.1 performs 1 step much faster when using two accelerators instead of one.

The fact that using the TensorFlow version 1.1.0 with two accelerators, a neural network trains slower than using one, is at first glance unexpected. But, a detailed analysis of hardware performance during task execution allows us to better understand the causes of suboptimal network operation with TensorFlow version 1.1.0.

3.4. GPU utilization

Figure 4 shows GPU utilization during various tasks.

Here we can see that for options Av1.1.0 and PAIv1.1.0, GPU underload and large load gaps are observed. In launches using one GPU with a batch size of 1024, utilization does not rise above 94% and, on average, keeps at about 50-60%. When running on two GPUs, the framework version 1.1.0 distributes tasks in such a way that the load is no more than 35%. This leads to a slowdown in one step.

At the same time, network launches with TensorFlow version 1.2.1 showed high utilization (up to 100%), both in launches with one and two GPUs. This leads to the fact that the learning steps with two GPUs are performed 54% faster than with one.

The described non-optimal network operation with the Av1.1.0 and PAIv1.1.0 options may raise questions. Perhaps the test task is not large enough to load one GPU at full capacity.

This can lead to meaningless parallelization. Therefore, we conduct tests using a batch size of 8192. As a result of the verification, in general, we can say that with such a batch size the program code becomes unstable and often fails. Nevertheless, we can estimate the average execution time of one step for different versions of the framework and study GPU utilization.

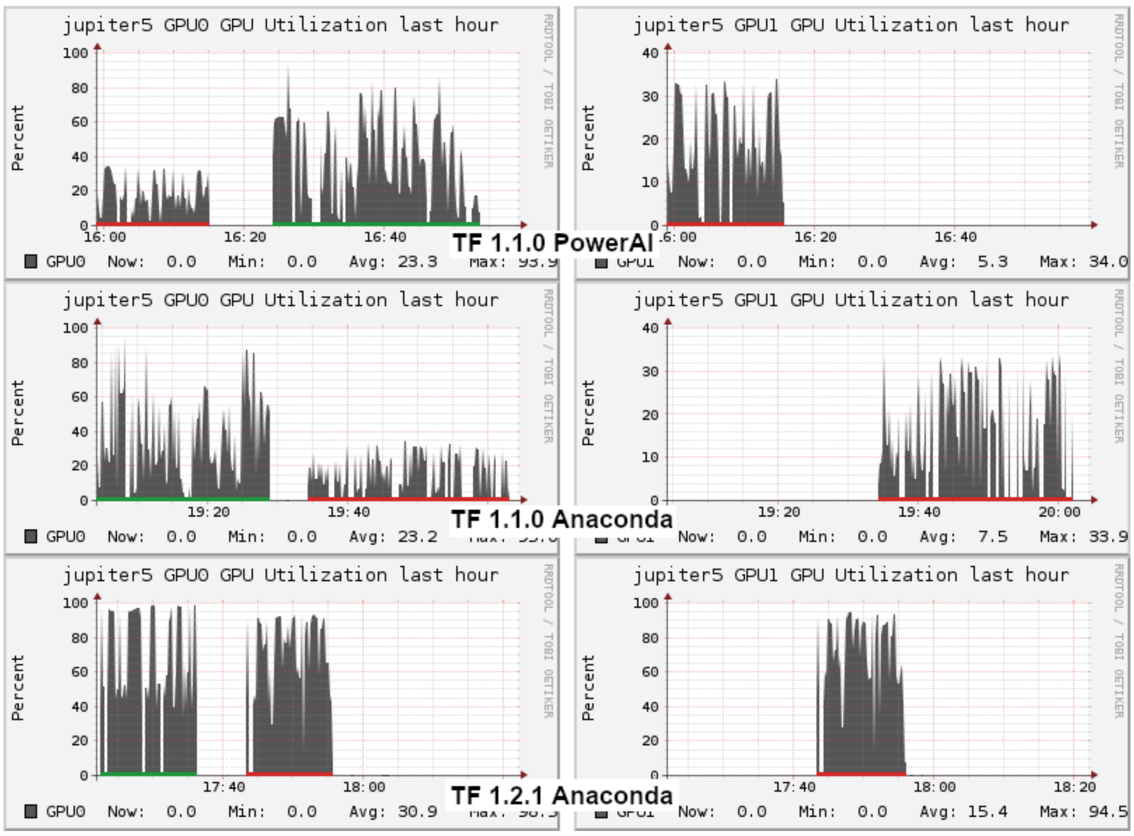

Figure 4: GPU utilization for different tasks and framework versions. 2 GPU tasks are underlined in red, 1 GPU tasks are underlined in green.

The average execution time of one step is shown in Table 3. It can be seen that the execution time of one step in Table 3 is significantly longer than in Table 1. Nevertheless, the ratio of acceleration due to the use of two GPUs has remained approximately the same.

Table 3
Average execution time for 1 training step.

TensorFlow option	1 GPU (sec)	2 GPU (sec)	speed-up, %
Av1.1.0	1.068	1.157	-8
PAIv1.1.0	0.902	1.014	-12
Av1.2.1	0.431	0.289	32

At the same time, Figure 5 shows that, despite the increase of the batch size, it is not possible to load the GPU for options Av1.1.0 and PAIv1.1.0 completely. However, utilization has increased significantly and reached 85%. When starting options Av1.1.0 and PAIv1.1.0 with two GPUs, utilization continues to reach high values, but gaps between loading blocks increase. TensorFlow version 1.2.1 continues to distribute the load efficiently and optimally utilizes the GPU, with both one and two accelerators.

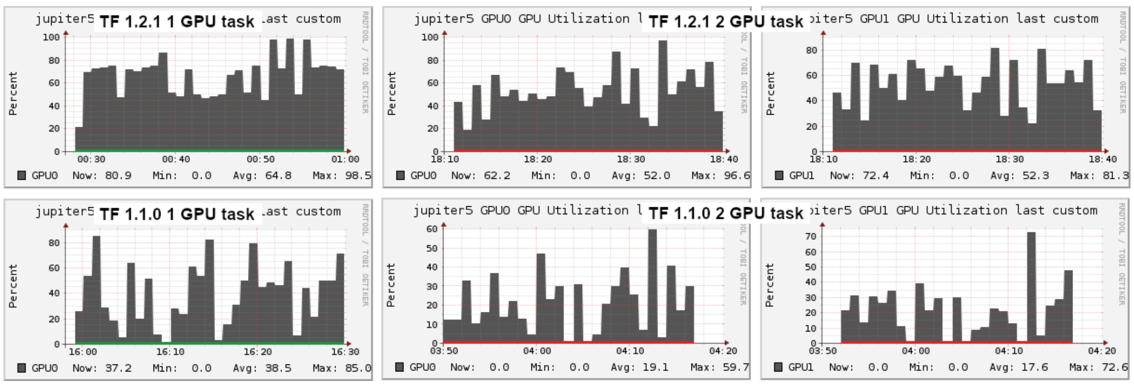

Figure 5: GPU utilization for high load tasks. Comparison of options Av1.1.0 vs Av1.2.1. 2 GPU tasks are underlined in red, 1 GPU tasks are underlined in green.

Thus, even an increase in the size of the task does not lead to a full load of the GPU, as well as an efficient allocation of resources between the two accelerators for TensorFlow version 1.1.0, which means that this version of the framework does not fully unleash the full potential of the IBM Power Systems S822LC system.

4. Conclusions

We examined various options and approaches for organizing hardware and software for performing the tasks of training neural networks in the conditions of shared facility centers. The most preferred hardware setup for shared facility centers includes GPU accelerators.

The performance of various versions of the software environment was demonstrated as a result of the launch of test tasks on the servers of the SFC "Data Center of FEB RAS" under various settings. In particular, TensorFlow version 1.2.1 and 1.1.0 from the Anaconda repository were compared, as well as TensorFlow version 1.1.0 from the IBM PowerAI software pack. It is shown that the PowerAI version of TensorFlow exceeds the corresponding TensorFlow release installed from the Anaconda repository by 15-23%. At the same time, TensorFlow version 1.1.0, installed from both sources, did not show effective parallelization for performing tasks on several GPUs.

The most relevant of the considered versions of the framework is TensorFlow 1.2.1. The operation of a neural network with the same launch conditions, using this version of TensorFlow, allowed increasing the speed of training significantly. Training of a neural network on one GPU on TensorFlow version 1.2.1 leads to two-fold acceleration, and using two GPUs in parallel allows training the network 3.2 times faster, compared with the TensorFlow version 1.1.0. Paralleled launch on two GPUs effectively accelerates network training and leads to an increase in productivity of 54%.

Considering the single period in which all three versions of the framework were available, we can conclude that it is necessary to install the latest version of the framework compared to a specially optimized one. It is preferable to install the TensorFlow framework from Anaconda repositories in the face of lagging updates that occur as part of the IBM PowerAI software pack.

In this release, IBM experts combined the paradigm of customized assembling and delivering through Anaconda repositories. This approach should accelerate updates in new releases of PowerAI. IBM PowerAI 1.6 includes Tensorflow1.14, which is the most current stable version of TensorFlow in general releases. In further research, we will consider a comparison of TensorFlow 1.14 from PowerAI 1.6 and Anaconda shared repositories on the IBM POWER hardware.

5. Acknowledgements

The work was done with partial support by the Russian Foundation for Basic Research, project No. 18-29-03196. The studies were carried out using the resources of the Center for Shared Use of Scientific Equipment "Center for Processing and Storage of Scientific Data of the Far Eastern Branch of the Russian Academy of Sciences" [9], funded by the Russian Federation represented by the Ministry of Science and Higher Education of the Russian Federation under project No. 075-15-2021-663.

6. References

[1] K. Sato, C. Young, D. Patterson, An in-depth look at Google's first Tensor Processing Unit (TPU), 2017. URL: https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles\\-first-tensor-processing-unit-tpu.

[2] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. Tat Liew, K. Srivatsan, D. Moss, S. Subhaschandra, G. Boudoukh, Can FPGAs beat GPUs in accelerating next-generation deep neural networks?, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '17, Association for Computing Machinery, New York, NY, 2017, pp. 5-14, doi:10.1145/3020078.3021740.

[3] NVIDIA DGX-1: Deep learning server for AI research. URL: https://www.nvidia.com/en-in/data-center/dgx-1/.

[4] M. Quartly, PowerAI introduction, 2018. URL: https://www.slideshare.net/ssuseraff9d9/power-ai-introduction.

[5] J. Helmus, TensorFlow in Anaconda, 2018. URL: https://www.anaconda.com/tensorflow-in-anaconda.

[6] IBM unveils new AI software, reduces barriers for data scientists to fuel cognitive development, 2017. URL: https://www-03.ibm.com/press/us/en/pressrelease/52346.wss.

[7] S. I. Mal'kovskii, A. A. Sorokin, S. P. Korolev, A. A. Zatsarinnyi, G. I. Tsoi, Performance evaluation of a hybrid computer cluster built on IBM POWER8 microprocessors, Programming and Computer Software 45(6) (2019) 324-332, doi:10.1134/S0361768819060057.

[8] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images. University of Toronto, 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[9] A. A. Sorokin, S. V. Makogonov, S. P. Korolev, The information infrastructure for collective scientific work in the Far East of Russia, Scientific and Technical Information Processing 44(4) (2017) 302-304, doi:10.3103/S0147688217040153.