A ARTIFACT APPENDIX

A.1 Abstract

This artifact evaluation appendix helps AEC to reproduce the main evaluation results in our EuroSys'23 paper: Espresso: Revisiting Gradient Compression from the System Perspective. Specifically, we provide a guideline to reproduce the main experimental results including:

- E1: End-to-end training throughput of different models with NVLink-based GPU machines and 100Gbps cross-machine Ethernet. (Figure 11)
- E2: End-to-end training throughput of different models with PCIe-only GPU machines and 25Gbps cross-machine Ethernet. (Figure 12)
- E3: Espresso can select the compression strategy for a DNN model in milliseconds. (Table 4 & 5)

A.2 Description & Requirements

A.2.1 How to access. You can access the evaluation code of Espresso at GitHub repository https://github.com/zhuangwang93/Espresso.

A.2.2 Hardware dependencies. The largest scale experiments in this paper need eight GPU machines and each of them has eight V100 GPUs. NVLink-based GPU machines with 100Gbps cross-machine Ethernet and PCIe-only GPU machines with 25 cross-machine Ethernet are needed for Expr1 and Expr2, respectively. If the resources are not available, it is fine to perform experiments with 4 GPU machines. It is also fine to perform the PCIe-only experiments on NVLink-based machines by setting `NCCL_P2P_DISABLE=1`.

A.2.3 Software dependencies. CUDA 11.1 and torch 1.8.0. Note that some Python packages used by Espresso have dependencies on CUDA 11.1 and torch version should be $\geq$ 1.8.0. Other software dependencies are Debian 10 and NCCL 2.8.3. Their versions are recommended but not mandatory.

A.2.4 Benchmarks. The DNN models and the data-sets used in the evaluation are listed as follows.

Model	Dataset	Batch size
VGG16 [62]	ImageNet [18]	32 images
ResNet101 [24]	ImageNet [18]	32 images
UGATIT [30]	selfie2anime [57]	2 images
BERT-base [20]	SQuAD [53]	1024 tokens
GPT2 [52]	WikiText-2 [43]	80 tokens
LSTM [41]	WikiText-2 [43]	80 tokens

A.3 Set-up

Follow the install instructions in `README.md` of Espresso's GitHub repository. It takes around 20 minutes to complete the installation and download the datasets. Espresso is built upon BytePS and the code for the artifact evaluation is in the directory `./byteps/torch/examples`. There are altogether six DNN models used in our experiments and thus there are six folders in this directory for each model.

A.4 Evaluation workflow

A.4.1 Major Claims.

- *(C1)*: Espresso can greatly improve the training throughput of representative DNN models compared to start-of-the-art systems. This is proven by the experiments (E1 and E2) described in Section 5 whose results are illustrated in Figure 11 and Figure 12.
- *(C2)*: Espresso can select the compression strategy for a DNN model in milliseconds. This is proven by the experiment (E3) described in Section 5 whose results are illustrated in Table 4 and Table 5.

A.4.2 Experiments.

Experiment (E1): [5 human-minute + 30 compute-minutes]
It aims to prove that Espresso can outperform other start-of-the-art systems for training on NVLink-based GPU machines. The expected results are that the training throughput of DNN models with Espresso is obviously higher than that with other baselines.

[Preparation] BERT-base, GPT2, and UGATIT are used in this experiment. The script to run this experiment is in `./byteps/torch/examples`.

Check the name of the network interface of all GPU machines and the IP address of the root machine. Any machine involved in training can serve as the root machine.

[Execution] Suppose the IP address of the root machine is *root_ip* and the network interface name is *nic_name*. Then set `ifname=`*nic_name* and `DMLC_PS_ROOT_URI=`*root_ip* in `run_nvlink_models.sh` on all GPU machines. Follow the instructions in `./byteps/torch/examples/README.md` to run both the baselines and Espresso.

[Results] There are three baselines for AE: `BytePS`, `BytePS-Compress`, and `HiTopKComm`. The output logs include the training information, such as the compression algorithm, the compression ratio, and the baseline name. It also prints out the training throughput at the end of each epoch. The metrics are "images/sec" or "tokens/sec". The results are logged in `model_log`.

Experiment (E2): [5 human-minutes + 30 compute-hour]
It aims to prove that Espresso can outperform other start-of-the-art systems for training on PCIe-based GPU machines. The expected results are that the training throughput of DNN models with Espresso is obviously higher than that with other baselines.

[Preparation] VGG16, LSTM, and ResNet101 are used in this experiment. The script to run this experiment is in `./byteps/torch/examples`.

Check the name of the network interface of all GPU machines and the IP address of the root machine. Any machine involved in training can serve as the root machine.

To save the data-set download time, we just download a subset of ImageNet to measure the training throughput of VGG16 and ResNet101.

[Execution] Set `ifname` and `DMLC_PS_ROOT_URI` in `run_pcie_models.sh` on all GPU machines. Follow the instructions in `./byteps/torch/examples/README.md` to run both the baselines and Espresso.

[Results] There are three baselines for AE: `BytePS`, `BytePS-Compress`, and `HiTopKComm`. The output logs include the training information, such as the compression algorithm, the compression ratio, and the baseline name. It also prints out the training throughput at the end of each epoch. The metrics are "images/sec" or "tokens/sec". The results are logged in `model_log`.

Experiment (E3) [1 human-minutes + 1 compute-minutes]

It aims to prove that the compute time of Espresso to select a compression strategy for a DNN model is in milliseconds. We will show the compute time for the six DNN models used in E1 and E2.

[Preparation] The strategy selection algorithm is implemented in `./byteps/torch/mergeComp/scheduler/`

[Execution] Follow the instructions in this folder to run the script.

[Results] The results have two types of information. One type is the model information, such as the DNN model name, the model size, and the number of tensors. The other type is the compute time to select the compression strategy and the time to find the best CPU offloading solution. Note that the results are better than the numbers listed in Table 4 because we have further optimized the implementation of Espresso.