A. Artifact Appendix

A.1 Abstract

This artifact appendix helps the readers reproduce the main evaluation results of the OSDI' 21 paper: PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections. The artifact evaluation includes the following experiments:

E1: An end-to-end performance comparison between PET and other frameworks. **(Figure 7)**
E2: An operator-level performance comparison on different backends, including cuDNN/cuBLAS, TVM, and Ansor. **(Figure 9)**
E3: Searching time. **(Section 8.4)**
E4: A performance comparison across different optimization policies, including fully-equivalent transformations, partially-equivalent transformations and joint optimization using both. **(Figure 10)**
E5: A performance comparison using different heuristics. **(Figure 11)**

A.2 Artifact check-list (meta-information)

- **Runtime environment**: Linux Debian 4.19.160-2
- **Hardware**: Dual Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz, NVIDIA V100 GPU
- **Metrics**: Inference time
- **Experiments**: Described in Subsection A.1
- **How much disk space is required (approximately)?**: The disk space of our server is approximately 150GB.
- **How much time is needed to prepare workflow (approximately)?**: The installation of PET only takes minutes. However, PET requires some pre-requirements, including CUDA, cuDNN, TVM, etc. The whole installation may take less than an hour from scratch. However, we can provide a publicly available server with all the pre-requirements installed.
- **How much time is needed to complete experiments (approximately)?**: The searching time of the Inception-v3 model in E1, E3 and E4 may take up to 1-2 hours, while other models should take a few minutes for each. E2 can take up to 24 hours if all the auto-tuning process is required, but we also provide the TVM-generated kernels we used in our paper so that E2 can be completed within a few minutes if needed. E5 should take about half an hour.
- **Publicly available?**: Yes
- **Core licenses (if publicly available)?**: Apache License, Version 2.0
- **Workflow frameworks used?**: TensorFlow, TVM, ONNX, TensorRT, TASO

A.3 Description

A.3.1 Hardware dependencies

This artifact depends on a NVIDIA V100 GPU.

A.3.2 Software dependencies

This artifact depends on the following software libraries:

- PET uses cuDNN and cuBLAS libraries as backend. Our evaluation uses CUDA 10.2 and cuDNN 7.6.5.
- CMake (https://cmake.org/) with a minimum version of 3.9 is required.
- Python 3 and the following packages are required:
  - numpy
  - onnx
  - torch
  - decorator
  - protobuf
- TensorFlow, TensorRT, TASO TVM and Ansor are used as baseline DNN frameworks in E1 and E2. Our evaluation on these baseline uses TensorFlow 1.15, TensorRT 7.0.0.11.

A.4 Installation

A.4.1 Install PET from source

- Clone code from git
- Install requirements
  - TVM: see A.4.5
- Install PET
  - `mkdir build; cd build; cmake ..`
  - `make -j`
- `Set the environment for evaluations`
  - `export PET_HOME=path_to_pet_home`

A.4.2 Install TensorFlow

- Install pre-compiled libraries
  - `pip install tensorflow==1.15.0 tensorflow-gpu==1.15.0 torch torchvision onnx==1.7.0`
- Install onnx-tf from source (the pre-compiled version depends on TF2)
  - `git clone https://github.com/onnx/onnx-tensorflow.git`
  - `cd onnx-tensorflow`
  - `git checkout v1.7.0-tf-1.15`
  - `pip install .`

A.4.3 Install TensorRT

- Download and extract the latest TensorRT 7.0 GA package for Ubuntu 18.04 and CUDA 10.2
  - `cd ~/Downloads`
  - `tar -xvzf TensorRT-7.0.0.11.Ubuntu-18.04.x86_64-gnu.cuda-10.2.cudnn7.6.tar.gz`
  - `export TRT_RELEASE=`pwd`/TensorRT-7.0.0.11`

    - `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$TRT_RELEASE/lib`

A.4.4 Install TASO

- see https://github.com/jiazhihao/TASO/blob/master/INSTALL.md for detailed installation instruction
- `git clone --recursive https://www.github.com/jiazhihao/taso`
- `cd taso`
- `export TASO_HOME=/path/to/taso`
- `mkdir build; cd build; cmake ..`
- `sudo make install -j 4`
- `cd ../python`
- `python setup.py install`

A.4.5 Install TVM & Ansor

- See https://tvm.apache.org/docs/install/from_source.html for detailed installation instruction
- `git clone --recursive https://github.com/apache/tvm tvm`
- `mkdir build`
- `cp cmake/config.cmake build`
- `cd build`
- `cmake ..`
- `make -j4`
- `cd python; python setup.py install --user;`

A.5 Experiments workflow

The following experiments are included in this artifact. All DNN benchmarks use synthetic input data in GPU device memory to remove the side effects of data transfers between CPU and GPU.

A.5.1 End-to-end performance (E1)

This experiment reproduces Figure 7 in the paper.

Prerequisite: generate ONNX models

- `cd $PET_HOME/models-ae`
- `./generate_onnx.sh`

TensorFlow & TensorFlow XLA. The TensorFlow & TensorFlow XLA results of the 4 models are available in the `tensorflow_ae` folder. The following command lines measure the inference latency of TensorFlow and TensorFlow XLA, respectively:

- `cd $PET_HOME/tf-ae`
- `./run.sh`

TensorRT. The TensoRT results of the 4 models are available in the `tensorrt_ae` folder. The following command lines measure the inference latency of TensorRT:

- `Load TensorRT environment (add library path to LD_LIBRARY_PATH)`
- `cd $PET_HOME/trt-ae`
- `./run.sh`

TASO. The TASO results of the 4 models are available in the `taso_ae` folder. The following command lines measure the inference latency of TASO:

- Load TASO environment
- `cd $PET_HOME/taso-ae`
- `./run_e2e.sh`

PET. The PET results of the 4 models are available in the `pet_ae` folder. The following command lines measure the inference latency of PET:

- `cd $PET_HOME/pet-ae`
- `./run_e2e.sh`

A.5.2 Operator-level performance (E2)

This experiment reproduces Figure 9 in the paper. The scripts are available in the `operator_ae` folder. The experiments of TVM and Ansor will take a very long time. We can provide the log files of the searching processes if needed (the log files are too large to upload).

cuDNN/cuBLAS. The following command lines measure cuDNN/cuBLAS results for the 4 operator-level benchmarks:

- `cd operator_ae/cudnn`
- `./run.sh`

TVM. The following command lines measure TVM results for the 4 operator-level benchmarks:

- `operator_ae/autotvm`

Ansor. The following command lines measure TVM results for the 4 operator-level benchmarks:

- `operator_ae/ansor`

A.5.3 Searching time (E3)

This experiment reproduces Section 8.4 in the paper. The scripts are available in the `pet_ae` folder. The same commands for PET in E1 print the searching time at the same time.

- `cd $PET_HOME/pet-ae`
- `./run_e2e.sh`

Note that our evaluation platform for AE has different CPUs with the platform we used for the paper, so the searching time could be different. But they should be within the same scale.

A.5.4 Different optimization policy (E4)

This experiment reproduces Figure 10 in the paper. The scripts are available in the `pet-ae` folder. The following command lines measure the results:

- `cd $PET_HOME/pet-ae`
- `./run_policy.sh`

A.5.5 Different heuristic parameters (E5)

This experiment reproduces Figure 11 in the paper. The scripts are available in the `pet-ae` folder. The following command lines measure the results:

- `cd $PET_HOME/pet-ae`
- `./run_param.sh`