Fast Kronecker Matrix-Matrix Multiplications on GPUs

Introduction

In this document we describe the details of the artifact of the paper ”Fast Kronecker Matrix-Matrix Multiplications on GPUs” and how to reproduce key results of the paper. Our system is publicly available at https://github.com/abhijangda/fastkron and the experimental setup is available at https://github.com/abhijangda/fastkron-benchmarks . We also provide a docker container with source code at https://doi.org/10.6084/m9.figshare.24803229 .

Hardware Requirements

FastKron supports both single and multi-GPU systems. In our experiments we use a DGX-2 machine with 16 NVIDIA Tesla V100 GPUs connected using NVLINK 2. FastKron will work on any machine with an NVIDIA GPU, however, the results might differ. We can provide access to a machine of 4 NVIDIA Tesla V100 GPUs, where Single GPU results can be replicated but Multi GPU results will differ.

Prerequisites

We provide instructions for both docker and manual installation of prerequisites.

Download

Download the artifact zip file https://doi.org/10.6084/m9.figshare.24803229 and create unzip it.

```
unzip fastkron-ppopp-24-ae.zip
cd fastkron-ae
```

Docker Container

Install docker engine by following steps on https://docs.docker.com/engine/install/ubuntu/.
Install NVIDIA Container Toolkit by following steps on
https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html .

Then create the container, and cd to the directory:

```
docker build -t fastkron-ppopp-24-ae .
docker run -it --gpus all fastkron-ppopp-24-ae
```

Check if torch supports CUDA by checking if torch.cuda.is_available() returns True:

```
python
>>> import torch
>>> torch.cuda.is_available()
True
```

Manual Installation

Linux Installation: We recommend using Ubuntu 22.04 as the Linux OS. We have not tested our artifact with any other OS but we believe Ubuntu 20.04 should work.

Install Dependencies: Execute following commands to install dependencies.

```
sudo apt update && sudo apt install gcc linux-headers-$(uname
-r) make g++ git python3-dev wget unzip python3-pip
build-essential devscripts debhelper fakeroot cmake
```

```
sudo pip3 install matplotlib
```

Install CUDA: We need to install CUDA before proceeding further. In our experiments we used CUDA 12.1 on Ubuntu 22.04. We believe any CUDA version above 10 should work. CUDA 12.1 toolkit can be downloaded from https://developer.nvidia.com/cuda-12-1-0-download-archive?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=22.04&target_type=runfile_local
While installing CUDA please make sure that CUDA is installed in /usr/local/cuda.

Set NVCC Path and CUDA libraries path: We assume that nvcc is present in /usr/local/cuda/bin/nvcc. Please make sure that this is a valid path and this nvcc is from CUDA 12.2 by using nvcc --version. Then export this in your PATH variable. Also update LD_LIBRARY_PATH to include CUDA libraries.

```
export PATH="/usr/local/cuda/bin:$PATH"
export LD_LIBRARY_PATH="/usr/local/cuda/lib64:$LD_LIBRARY_PATH"
```

Check CUDA Installation: Executing nvidia-smi should give a list of all GPUs in the system.

Install Pytorch: Ubuntu 22.04 contains python 3.10. We believe any python 3.8+ should work. Install Pytorch using pip:

```
sudo pip3 install torch torchvision torchaudio
```

Check Pytorch CUDA Install: Check whether torch supports CUDA by running:

```
$ python
Python 3.11.5 (main, Sep 11 2023, 13:54:46) [GCC 11.2.0] on
linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import torch
>>> torch.cuda.is_available()
True
```

We should see True as the result of is_available()

Install GPyTorch: Install GPyTorch using pip

```
sudo pip3 install gpytorch
```

Install cuTensor: Install cuTensor from https://developer.nvidia.com/cutensor-downloads

Build and Install NVIDIA NCCL: Clone NCCL from https://github.com/NVIDIA/nccl

```
git clone https://github.com/NVIDIA/nccl
```

Build NCCL by specifying specific SM and COMPUTE version. For Tesla V100, we can use below command

```
make -j src.build
NVCC_GENCODE="-gencode=arch=compute_70,code=sm_70"
```

Install NCCL by creating .deb package and installing the package

```
make pkg.debian.build
```

```
cd build/pkg/deb/
sudo apt install ./libnccl*
```

Clone the repository: Clone the FastKron repository with its submodules and switch to ppopp-24-ae branch:

```
git clone https://github.com/abhijangda/fastkron.git
--recurse-submodules
git checkout ppopp-24-ae
```

Clone benchmark repository: Clone the fastkron-benchmark repository with its submodules:

```
git clone https://github.com/abhijangda/fastkron-benchmarks.git
--recurse-submodules
```

Getting Started

We will now build FastKron and execute tests. In docker container the FastKron directory is available at `/fastkron` and `/fastkron-benchmarks` . Go to fastkron directory

```
cd <fastkron-dir>
```

Setup cmake: Create build directory, setup cmake and cd to build:

```
mkdir <fastkron-dir>/build
cd build
cmake ..
```

Execute a SingleGPU Test: We can execute one of single GPU test as below:

```
make gen-single-gpu-kernels
make run-single-gpu-no-fusion-tests -j
```

Execute a MultiGPU Test: We can execute one of the multi GPU test as below:

```
make gen-multi-gpu-tests-kernel
```

```
make run-multi-gpu-nccl-no-fusion-tests -j
```

If all above tests run fine and do not give any error then we have successfully setup the benchmarking.

Execute all Tests (Optional): We can execute all tests from parent directory

```
cd ..
python tests/run-tests.py
```

Step by Step Instructions

[Time 2 - 3 hours]
We will now reproduce results in Figure 9, Table 3, Figure 11, Figure 10, and Table 5. These commands generate Figures as PDF in the benchmarks directory and Table as CSV in the benchmarks directory. Change to the benchmark directory:

```
cd <fastkron-benchmarks-dir>
```

[Time 30 mins] Figure 9: We will reproduce the Figure 9 and compare FastKron against GPyTorch and COGENT. Inside the fastkron-benchmarks directory execute:

```
python run_benchmarks.py -fk-dir <fastkron-dir> -fk-bench-dir <fastkron-benchmarks-dir> -bench Figure-9
```

Above command will generate single-gpu-flops.csv . We can generate graph by

```
make Figure-9.pdf
```

[Time 15 mins] Table 3: Inside the fastkron-benchmarks directory execute:

```
python run_benchmarks.py -fk-dir <fastkron-dir> -fk-bench-dir <fastkron-benchmarks-dir> -bench Table-3
```

Above command will generate Table-3-float.csv for Float results and Table-3-double.csv for Double results

[Time 40 mins] Figure 10: Inside the fastkron-benchmarks directory execute:

```
python run_benchmarks.py -fk-dir <fastkron-dir> -fk-bench-dir
<fastkron-benchmarks-dir> -bench Figure-10
```

Above command will generate real-world-benchmarks.csv. Generate graph by

```
make Figure-10.pdf
```

[Time 40 mins] Figure 11: Inside the fastkron-benchmarks directory execute:

```
python run_benchmarks.py -fk-dir <fastkron-dir> -fk-bench-dir
<fastkron-benchmarks-dir> -bench Figure-11
```

Above command will generate multi-gpu-flops-64.csv and multi-gpu-flops-128.csv . Generate the left part of Figure 11 as

```
make Figure-11-64.pdf
```

Generate the right part of the Figure 11 as

```
make Figure-11-128.pdf
```

[Time 30 mins] Table 5: We will now execute the integration of FastKron in GPyTorch for SKI, SKIP, and LOVE GPs. First, download the 1GB UCI dataset here: https://1drv.ms/u/s!Agc23QHWOw-TjCvWAsX7zEV5u6LG?e=ioRZ0X. Extract the zip file to a folder. Execute the command:

```
python gps-Table-5.py <uci-path> 10
```

This will print the Table 5 for each Gaussian Process.