Korea Advanced Institute of Science and Technology
CS492 Spring 2021
Project 3

Jinwoo Hwang
jwhwang@casys.kaist.ac.kr

Due: May. 7th 11:59 PM KST

Rules

- Do not copy from others. You will get huge penalty according to the University policy. Sharing of code between students is viewed as cheating.
- Every code was written and will be tested in the provided docker environment. You need to implement your code in the given environment.
- Read carefully not only this document but the comment in the skeleton code.
- This is the version 1.0 document which means not perfect. Any specifications or errors can be changed during the project period with announcement. Stay tuned to Piazza.

Introduction

In this project, you are going to optimize the DNN graphs from the previous project by replacing the slow Python code with parallelization library on the CPUs.

In this project you will learn (1) how to exploit C language based library in Python and (2) how to use parallelization library and vector computation APIs.

Project Structure

This project contains the following files:

- `examples/`: Example files for various libraries
- `src/`: Reference codes of project 2
- `sample.jpg, sample_out.jpg`: Sample input, output image file
- `Makefile`: Makefile to run the code

Read the content of the `Makefile` carefully and make sure your code works with the following four commands since it will be used when grading your codes.

```
$ make {run_avx|run_avx_debug|run_openblas|run_openblas_debug}
```

Strategy

Ctypes

Ctypes is a foreign function library for Python. The C code should be compiled into shared object in order to import them into the Python code. Following command will create `libexample.so` from `example.c`.

```
$ gcc -shared -fPIC -o libexample.so example.c
```

Check the example in `example/ctypes`, how it is compiled and how it is imported in Python code.

Use BLAS libraries

There are several libraries that aims to optimize Basic Linear Algebra Subprograms (BLAS). OpenBLAS is one of the them which specifically aims optimizations on the CPUs. An example would be `sgemm(dgemm)` which stands for single(double)-precision general matrix-matrix multiplication. The function automatically vectorizes matrix multiplication with a single function call.

AVX & pthread

AVX is an extension to the x86 instruction set architecture. It collects several variables (e.g. eight single precision floats) in one data type to compute them in a single instruction. To compile the AVX code, you need to add `-mavx2` flag. Check `Makefile` in examples/AVX. Also, pthread is a well-known parallelizing library for employing threads in C. By combining these two libraries, you can achieve massive parallelism in the computation.

Implementation

Overview

Start from the reference solution for the Project 2 in the `src/`. You may also start from yours if you want. Optimize the computation of `dnn.py` by using the aforementioned libraries one at a time. The name of the files must be `dnn_openblas.py`, `dnn_openblas.c`, `dnn_avx.py`, `dnn_avx.c`. You will get full points only if your implementation outperforms the baseline code.

Also, implement a debug mode which will save the intermediate values from each layer into `intermediate/layer_#.npy`. You can assume that `intermediate` directory already exists. The code takes `--debug` flag and turning it on will set `DNNInferenceEngine.debug` to `True`.

- Library-supported parallelization - OpenBLAS (10pt)
- Manual parallelization - AVX & pthread (20pt)
- Debug mode(5pt)

Running Time Contest (10pt each, 20pt total)

In project 3, we are hosting an inference time contest. We will measure the inference time of the two versions of your code; OpenBLAS and AVX. The implementation that takes the shortest time will get the highest score. The first-place gets 10 points, the second gets 9 points, the third-place gets 8 points, 4-9th get 6 points, 10-15th get 4 points, 10-22th get 2 points.

Report (10 pt)

Write a detailed report that describes your code. Your report must include

- Key functions for parallelization (e.g., OpenBLAS - `cblas_dgemm`) and how they work
- Parallelization strategies: which parts in `dnn.py` and how you parallelized them
- Performance gain from offloading the code and analysis for each parallelization scheme.

The running time should be measured in the given KCloud CPU server. The purpose of the report is to check your understanding. Please write the answer clear.

Handin

You will need to submit a single tarball containing 4 files(`dnn_*.py, dnn_*.c`) and a report in PDF format. The name of the file should be `[student ID].tar`. Upload your tar file to KLMS.

Reference

- Ctypes - https://docs.python.org/3/library/ctypes.html
- OpenBLAS - https://github.com/xianyi/OpenBLAS/wiki/
- AVX - https://www.codeproject.com/Articles/874396/Crunching-Numbers-with-AVX-and-AVX