CSC4005 Project 4 Report

Jiarui Chen 120090361

Task 1: Sequential softmax implementation

In this part, a sequential softmax regression has been implemented. The output result is as follow:

```
Softmax Sequential
Training softmax regression
| Epoch | Train Loss | Train Err | Test Loss | Test Err |
|     0 |    0.35134 |   0.10182 |   0.33588 |  0.09400 |
|     1 |    0.32142 |   0.09268 |   0.31085 |  0.08730 |
|     2 |    0.30802 |   0.08795 |   0.30097 |  0.08550 |
|     3 |    0.29987 |   0.08532 |   0.29558 |  0.08370 |
|     4 |    0.29415 |   0.08323 |   0.29215 |  0.08230 |
|     5 |    0.28980 |   0.08182 |   0.28973 |  0.08090 |
|     6 |    0.28633 |   0.08085 |   0.28793 |  0.08080 |
|     7 |    0.28345 |   0.07997 |   0.28651 |  0.08040 |
|     8 |    0.28099 |   0.07923 |   0.28537 |  0.08010 |
|     9 |    0.27886 |   0.07847 |   0.28442 |  0.07970 |
Execution Time: 7002 milliseconds
```

At first, I implemented the matrix multiplication using for loops and memory locality, but at the end it takes about 17000ms. So, I use SIMD to accelerate the matrix multiplication and get the result of 7000ms.

Task 2: OpenACC softmax

In this part, an OpenACC implementation of softmax regression has been implemented. The output result is as follow:

```
Softmax OpenACC
Training softmax regression (GPU)
| Epoch | Train Loss | Train Err | Test Loss | Test Err |
|     0 |    0.35134 |   0.10182 |   0.33588 |  0.09400 |
|     1 |    0.32142 |   0.09268 |   0.31085 |  0.08730 |
|     2 |    0.30802 |   0.08795 |   0.30097 |  0.08550 |
|     3 |    0.29987 |   0.08532 |   0.29558 |  0.08370 |
|     4 |    0.29415 |   0.08323 |   0.29215 |  0.08230 |
|     5 |    0.28980 |   0.08182 |   0.28973 |  0.08090 |
|     6 |    0.28633 |   0.08085 |   0.28793 |  0.08080 |
|     7 |    0.28345 |   0.07997 |   0.28651 |  0.08040 |
|     8 |    0.28099 |   0.07923 |   0.28537 |  0.08010 |
|     9 |    0.27886 |   0.07847 |   0.28442 |  0.07970 |
Execution Time: 6605 milliseconds
```

We use `#pragma acc data copyin` to send the data into GPU, and using `#pragma acc parallel loop present` to start parallel by OpenACC.

The implementation is based on non-SIMD sequential implementation, so it has accelerated about 10,000ms.

Task 3: Sequential neural network implementation

In this part, a sequential nn training has been implemented. The output result is as follow:

```
NN Sequential
Training two layer neural network w/ 400 hidden units
| Epoch | Train Loss | Train Err | Test Loss | Test Err |
|     0 |    0.13465 |   0.04023 |   0.14293 |  0.04240 |
|     1 |    0.09652 |   0.03020 |   0.11593 |  0.03700 |
|     2 |    0.07343 |   0.02210 |   0.10038 |  0.03190 |
|     3 |    0.05810 |   0.01693 |   0.09058 |  0.02860 |
|     4 |    0.04665 |   0.01288 |   0.08342 |  0.02650 |
|     5 |    0.03910 |   0.01025 |   0.07944 |  0.02540 |
|     6 |    0.03255 |   0.00810 |   0.07549 |  0.02440 |
|     7 |    0.02810 |   0.00687 |   0.07357 |  0.02410 |
|     8 |    0.02435 |   0.00548 |   0.07189 |  0.02340 |
|     9 |    0.02133 |   0.00463 |   0.07082 |  0.02210 |
|    10 |    0.01869 |   0.00372 |   0.06978 |  0.02200 |
|    11 |    0.01670 |   0.00310 |   0.06907 |  0.02180 |
|    12 |    0.01496 |   0.00250 |   0.06849 |  0.02150 |
|    13 |    0.01347 |   0.00193 |   0.06785 |  0.02130 |
|    14 |    0.01210 |   0.00153 |   0.06747 |  0.02060 |
|    15 |    0.01126 |   0.00125 |   0.06742 |  0.02060 |
|    16 |    0.01012 |   0.00097 |   0.06709 |  0.02040 |
|    17 |    0.00944 |   0.00078 |   0.06706 |  0.02040 |
|    18 |    0.00868 |   0.00065 |   0.06688 |  0.02040 |
|    19 |    0.00797 |   0.00045 |   0.06667 |  0.02030 |
Execution Time: 483065 milliseconds
```

The nn training implementation also use the SIMD version of matrix multiplication.

Compile and Run

We can compile and run the programs by

```
cd ./project4
bash test.sh
```

Or run the compiled files by `sbatch` command on the cluster.

```
cd ./project4
sbatch sbatch.sh
```

Profiling Results

We use `nsys` to get profiling results in the docker container.

The profiling files of the programs are in the `profiling` folder.

Three of the tasks which have taken most time (unit: ns) are:

IMPLEMENTS	POLL	IOCTL	READ
softmax OpenACC	3104m	128m	14.8m