|  |  |
| --- | --- |
| **Name:** | *<name>* |
| **NetID:** | *<netid>* |
| **Section:** | *<class section>* |

**ECE 408/CS483 Milestone 3 Report**

|  |
| --- |
| 1. List Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images from your basic forward convolution kernel in milestone 2. This will act as your baseline this milestone. |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.17596ms* | *0.63716 ms* | *5.160s* | *0.86* | | 1000 | *1.6283ms* | *6.2782ms* | *46.817s* | *0.886* | | 10000 | *15.9715ms* | *62.4858ms* | *8m10.069s* | *0.8714* | |
| 1. **Optimization 1: *Use constant Memory to save the convolution kernel*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *I used constant Memory to save the convolution kernel, because constant memory can be accessed in shorter time.* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *This optimization requires us to save kernel in the constant memory can access the constant memory during inference. I think this optimization can work well because constant memory can be accessed in very short time.* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.366427 ms* | *0.75584 ms* | *5.420s* | *0.86* | | 1000 | *1.562 ms* | *5.724 ms* | *46.252s* | *0.886* | | 10000 | *14.8253 ms* | *56.9035 ms* | *8m10.362s* | *0.8714* | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *This implementation is successful and it improve the the op time slightly (~5%-7%) when the batch size become larger.* |
| * 1. What references did you use when implementing this technique? |
| *CUDA C++ Programming Guide* |
| 1. **Optimization 2:  *Tunning with restrict and loop unrolling*** |
| 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *Then I implemented restrict and loop unrolling because with ‘\_\_restrict\_\_’ keyword, the access of pointer can be optimized by compiler. And proper loop unrolling can let compiler make full use of registers.* |
| 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *I add \_\_restrict\_\_ keyword before*  *Using \_\_restrict\_\_ can let the compiler knows that two point cannot overlap, which can reduce the time of memory access. And using unroll can let the compiler use more registers which can be accessed in very short time. I think it can improve the performance because the more computation can happen betweeen registers and can synergize with the previous optimization.* |
| 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.18702 ms* | *0.51661 ms* | *4.882s* | *0.86* | | 1000 | *1.29562 ms* | *4.41658 ms* | *46.920s* | *0.886* | | 10000 | *11.1509 ms* | *43.0152 ms* | *8m3.109s* | *0.8714* | |
| 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *It turned out that this optimization works well, it improve the op time by about 30%* |
| 1. What references did you use when implementing this technique? |
| *CUDA C++ Programming Guide* |

|  |
| --- |
| 1. **Optimization 3: Tiled shared memory convolution** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *I implemented tiled shared memory convolution. I think it can improve the performance because it uses shared memory which can be accessed 50-100 times faster than the global memory.* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *This optimization requires every thread to load a tile of data from global memory to shared memory in parallel and then compute the convlution using the shared memory. I think this optimization can improve the performance since it make full use of computing power. And it can* synergize with previous optimization since we can constant memory access and loop unrolling is |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.19779 ms* | *0.64118 ms* | *4.830s* | *0.86* | | 1000 | *1.33216 ms* | *5.746 ms* | *47.148s* | *0.886* | | 10000 | *11.4665 ms* | *55.4457 ms* | *7m41.273s* | *0.8714* | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *Compare with the baseline, the performance improve a bit, but it can not outpeform previous optimization. It improved compare with baseline because it make use of shared memory, but it did not improve compare with previous optimization because the overhead of \_\_syncthreads().* |
| * 1. What references did you use when implementing this technique? |
| *CUDA C++ Programming Guide* |
| 1. **Optimization 4: Shared memory matrix multiplication and input matrix unrolling** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *<answer here>* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *<answer here>* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 |  |  |  |  | | 1000 |  |  |  |  | | 10000 |  |  |  |  | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *<answer here>* |
| * 1. What references did you use when implementing this technique? |
| *<answer here>* |
| 1. **Optimization 5: *Kernel fusion for unrolling and matrix-multiplication*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *<answer here>* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *<answer here>* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 |  |  |  |  | | 1000 |  |  |  |  | | 10000 |  |  |  |  | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *<answer here>* |
| * 1. What references did you use when implementing this technique? |
| *<answer here>* |
| 1. **Optimization 6: Fixed point (FP16) arithmetic** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *<answer here>* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *<answer here>* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 |  |  |  |  | | 1000 |  |  |  |  | | 10000 |  |  |  |  | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *<answer here>* |
| * 1. What references did you use when implementing this technique? |
| *<answer here>* |

|  |
| --- |
| 1. **Optimization 7: Using Streams to overlap computation with data transfer** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *<answer here>* |
| * 1. How does the optimization work? Did you think the optimization would increase performance of the forward convolution? Why? Does the optimization synergize with any of your previous optimizations? |
| *<answer here>* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 |  |  |  |  | | 1000 |  |  |  |  | | 10000 |  |  |  |  | |
| * 1. Was implementing this optimization successful in improving performance? Why or why not? Include profiling results from *nsys* and *Nsight-Compute* to justify your answer, directly comparing to your baseline (or the previous optimization this one is built off of). |
| *<answer here>* |
| * 1. What references did you use when implementing this technique? |
| *<answer here>* |