|  |  |
| --- | --- |
| **Name:** | *Xing Shen* |
| **NetID:** | *Xings2* |
| **Section:** | *ZJU* |

**ECE 408/CS483 Milestone 3 Report**

|  |
| --- |
| 1. List Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images from your basic forward convolution kernel in milestone 2. This will act as your baseline this milestone. Note: **Do not** use batch size of 10k when you profile in *--queue rai\_amd64\_exclusive*. We have limited resources, so any tasks longer than 3 minutes will be killed. Your baseline M2 implementation should comfortably finish in 3 minutes with a batch size of 5k (About 1m35 seconds, with nv-nsight). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.193803ms* | *0.897425ms* | *2.353s* | *0.86* | | 1000 | *1.54005ms* | *7.83892ms* | *11.314s* | *0.886* | | 5000 | *8.45031ms* | *43.2912ms* | *1m0.117s* | *0.871* | |
| 1. **Optimization 1: *Tiled shared memory convolution (2 points)*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *This optimization can utilize GPU shared memory efficiently, and in that case the running time should be reduced.* |
| * 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? |
| *The first optimization, and I believe that this can increase the performance of the forward convolution. It will not synergize with previous optimization.* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.213878ms* | *0.843126ms* | *1.794s* | *0.86* | | 1000 | *2.02368ms* | *8.31878ms* | *11.293s* | *0.886* | | 5000 | *10.0167ms* | *41.0016ms* | *55.566s* | *0.871* | |
| * 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). |
| *We could found that the runtime of conv\_forward\_kernel is smaller than m2, which means shared memory will increase the performancy.* |
| * 1. What references did you use when implementing this technique? |
| *Lecture, textbook.* |
| 1. **Optimization 2: *Weight matrix in constant memory (0.5 points)*** |
| 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *Weight matrix in constant memory*  *The matrix in CNN is fixed, so use constant memory will easily increase the performance.* |
| 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 believe it will increase the performance because we just change the memory type into constant memory. This type of memory could be read more quickly, leading to a shorter time.* |
| 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.173929ms* | *0.660685ms* | *1.686s* | *0.86* | | 1000 | *1.77945ms* | *7.13333ms* | *11.830s* | *0.886* | | 5000 | *8.79106ms* | *32.3771ms* | *54.501s* | *0.871* | |
| 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). |
| *Constant memory can be accessed faster than global memory, so the increase of performance is obvious.* |
| 1. What references did you use when implementing this technique? |
| *Textbook, Lecture.* |

|  |
| --- |
| 1. **Optimization 3: *Using Streams to overlap computation with data transfer (4 points)*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *Stream is a part without appearing in MPs, so I try to use this optimization in order to cement my knowledge.* |
| * 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? |
| *Using Streams, we can overlap the data transfer, which means different part of data could be transferred at the same time, which obviously increase the performance.*  *This optimization is based on ms2, since we change the arrangement of functions, and we delete two functions, which means OP time cannot be used to judge the performance. Also, there is no synergize with another optimization.* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | */* | */* | */* | */* | | 1000 | */* | */* | */* | */* | | 5000 | */* | */* | */* | */* | |
| * 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). |
| *Since the stream is small, and if the stream is larger, there will be error with the code. In that case, the performance actually is almost the same as the ms2.* |
| * 1. What references did you use when implementing this technique? |
| *Textbook, lecture.* |
| 1. **Optimization 4: Input channel reduction: atomics (2 points)** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *I choose this optimization because it could synergize with the pervious optimization perfectively.* |
| * 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? |
| *Using atomic reduction, each output element on feature map, a thread will be used to calculate the convolution values.*  *This optimization can synergize with previous optimizations.* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.185732ms* | *0.707704ms* | *1.799s* | *0.86* | | 1000 | *1.79141ms* | *7.16269ms* | *10.891s* | *0.886* | | 5000 | *8.83665ms* | *35.6465ms* | *53.014s* | *0.871* | |
| * 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). |
| *From the total execution time, we could easily find that using atomicAdd is much faster than not use it.* |
| *Also from the nsys, we could find the time of conv\_forward\_kernel is shorter than the original.* |
| * 1. What references did you use when implementing this technique? |
| *Textbook, Lecture.* |
| 1. **Optimization 5: *FP16 arithmetic (4 points)*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *FP16 arithmetic*  *Since this will reduce the size of data, which may cause reduction on data transfer.* |
| * 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? |
| *Use FP16 will change the data type from float to half, which will reduce the data transfer time and calculate time. In that case, the performance of total code will increase.*  *Besides, this optimization will also slightly increase the accuracy.* |
| * 1. List the Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 5k images using this optimization (including any previous optimizations also used). |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.564685ms* | *1.90058ms* | *1.611s* | *0.86* | | 1000 | *7.18244ms* | *19.0528ms* | *12.059s* | *0.887* | | 5000 | *18.1903ms* | *90.2399ms* | *53.424s* | *0.8712* | |
| * 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). |
| *Actually it improves accuracy slight, and it will short the total execution at specific batch size.*  *I use this optimization based on ms2, so we compare it with* |
| * 1. What references did you use when implementing this technique? |
| *Textbook, lectures.* |

Total points: 2 + 0.5 + 4 + 2 + 4 = 12.5

The code is in the folder optimize.