|  |  |
| --- | --- |
| **Name:** | *Yen-Ting Liu* |
| **NetID:** | *ytliu2* |
| **Section:** | *AL2* |

**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 | *2.07946 ms* | *31.8139 ms* | *1.117 s* | *0.86* | | 1000 | *20.7417 ms* | *322.405 ms* | *9.517 s* | *0.886* | | 10000 | *207.877 ms* | *3242.94 ms* | *1 m 33.543 s* | *0.8714* |   *(baseline\_build-619d3ce95876a202d8bfbba3)* |
| 1. **Optimization 1: *Weight matrix (kernel values) in constant memory*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *Calculation for all outputs require kernel, and it does not change during grid execution. These characteristics make kernel array a nice candidate to move into constant memory with minimal code change.* |
| * 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? |
| *Constant memory resides in device memory and caches in the constant cache. In case of cache hit, the resulting memory requests are served at the throughput of the constant cache. Since kernel weight will not change throughout the inference, I believe constant memory can provide some improvements, and can work well with other optimization (this is my first 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 | *1.86608 ms* | *30.5911 ms* | *1.093 s* | *0.86* | | 1000 | *18.702 ms* | *314.776 ms* | *9.420 s* | *0.886* | | 10000 | *186.951 ms* | *3183.48 ms* | *1 m 32.670 s* | *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). |
| ***Op 1***    ***Op 2***    *In my following report sections, I will use numbers directly instead of screenshot SOL.*   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | ***SM [%]*** | ***Memory [%]*** | ***L1 Cache [%]*** | ***Duration [%]*** | | *Op 1* | *+10.40%* | *-36.58%* | *-37.80%* | *-10.36%* | | *Op 2* | *+0.81%* | *-43.65%* | *-45.45%* | *-3.11%* |   *The statistics shows marginal improvement when we move the kernel to constant memory. Memory bandwidth drastically decreases as expected, but L1 cache utilization also decreases for some reason.*  *(constant\_build-619d3b015876a201b3608beb)* |
| * 1. What references did you use when implementing this technique? |
| *CUDA Toolkit v11.5.0 Programming Guide* |
| 1. **Optimization 2: *Sweeping various parameters to find best values (block sizes, amount of thread coarsening)*** |
| 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *Nsight Compute shows my kernel launch only execute 4 blocks, which is less than the GPU’s 80 MPs. It also shows on average, each warp spends 3.3 cycles stalled on a fixed latency execution dependency. This calls for adjust block size.*  *Originally, I go for block size (32, 32), but dumping out the data dimension shows*  ***Op 1*** *(B=100, M=4, C=1, H=86, W=86, K=7)*  ***Op 2*** *(B=100, M=16, C=4, H=40, W=40, K=7)*  *Plug these numbers into kernel size calculation W-K+1 shows 80 and 36 respectively.*  *Op1 W\_out=80, Op2 W\_out=36* |
| 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). |
| ***Block size (4, 4)***   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total  Execution  Time | Accuracy | | 100 | *1.19454 ms* | *16.3916 ms* | *1.084 s* | *0.86* | | 1000 | *11.8864 ms* | *163.821 ms* | *9.314 s* | *0.886* | | 10000 | *119.357 ms* | *1695.41 ms* | *1 m 31.348 s* | *0.8714* |   ***Block size (8, 8)***   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total  Execution  Time | Accuracy | | 100 | *1.15893 ms* | *15.9704 ms* | *1.131 s* | *0.86* | | 1000 | *11.5606 ms* | *160.342 ms* | *9.702 s* | *0.886* | | 10000 | *114.973 ms* | *1603.01 ms* | *1 m 34.981 s* | *0.8714* |   Block size (20, 20)  Block size (36, 36)  Block size (40, 40) |
| 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 3: *<optimization name>***   ***(Delete this section blank if you did not implement this many optimizations.)*** |
| * 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 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 1000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 10000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | |
| * 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 4: *<optimization name>***   ***(Delete this section blank if you did not implement this many optimizations.)*** |
| * 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 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 1000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 10000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | |
| * 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: *<optimization name>***   ***(Delete this section if you did not implement this many optimizations.)*** |
| * 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 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 1000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 10000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | |
| * 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: *<optimization name>***   ***(Delete this section if you did not implement this many optimizations.)*** |
| * 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 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 1000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | | 10000 | *<op\_time>* | *<op\_time>* | *<exec\_time>* | *<accuracy>* | |
| * 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>* |