|  |  |
| --- | --- |
| **Name:** | *Diego Kleiman* |
| **NetID:** | *diegoek2* |
| **Section:** | *AL1* |

**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 | *0.180023 ms* | *0.638175 ms* | *0m1.248s* | *0.86* | | 1000 | *1.60305 ms* | *6.13152 ms* | *0m10.200s* | *0.886* | | 10000 | *15.8424 ms* | *61.2369 ms* | *1m35.258s* | *0.8714* | |
| 1. **Optimization 1: W*eight matrix (kernel values) in constant memory (1 point)*** |
| * 1. Which optimization did you choose to implement and why did you choose that optimization technique. |
| *I chose the optimization “Weight matrix (kernel values) in constant memory.” I chose it because storing the convolution kernel in constant memory is supposed to decrease the number of cycles it takes to read a value from the mask as compared to reading it from global memory. The convolution mask is a good candidate for constant memory because it is small and it is always constant for the forward pass.* |
| * 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 optimization works by reserving enough constant memory in the device and then using cudaMemcpyToSymbol from the host to transfer the mask to this region of memory. Then, we replace all global reads from the convolution mask into constant memory reads (that is, we remove the device\_mask argument and use the symbol that we copied the mask into). Constant memory is cached, so the number of cycles it takes to read a value from here is smaller as compared to global memory. For this reason, I thought that this optimization would improve the performance of the kernel. This is the first optimization, so it does not synergize with any other 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 | *0.149252 ms* | *0.609631 ms* | *0m1.169s* | *0.86* | | 1000 | *1.45413 ms* | *6.48258 ms* | *0m9.985s* | *0.886* | | 10000 | *14.3962 ms* | *64.451 ms* | *1m34.163s* | *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). (Next page) |
| *This optimization was not successful overall in improving the performance if we look at the sum of the op times for batch size 10,000. The sum of op times increased from 77.08 ms to 78.85 ms. This is probably because the GPU still has to read the input matrix from global memory, so there is not enough time saved from reading the mask from constant memory. According to Nsight-Compute, for the input layer call, the SM throughput increased from 77.80% to 84.63% and the memory throughput decreased from 91.72% to 72.29% (because there are fewer global memory reads).*  *Profiling results from nsys (CUDA statistics only):*  *Generating CUDA API Statistics...*  *CUDA API Statistics (nanoseconds)*  *Time(%) Total Time Calls Average Minimum Maximum Name*  *------- -------------- ---------- -------------- -------------- -------------- --------------------------------------------------------------------------------*  *78.7 1089649488 6 181608248.0 18682 590417294 cudaMemcpy*  *14.3 197669531 6 32944921.8 322262 194350825 cudaMalloc*  *5.6 78115572 8 9764446.5 945 63484106 cudaDeviceSynchronize*  *1.2 16690715 6 2781785.8 18807 16573414 cudaLaunchKernel*  *0.2 2523441 6 420573.5 85065 799854 cudaFree*  *0.0 346657 2 173328.5 171975 174682 cudaMemcpyToSymbol*  *Generating CUDA Kernel Statistics...*  *Generating CUDA Memory Operation Statistics...*  *CUDA Kernel Statistics (nanoseconds)*  *Time(%) Total Time Instances Average Minimum Maximum Name*  *------- -------------- ---------- -------------- -------------- -------------- --------------------------------------------------------------------------------*  *100.0 78088581 2 39044290.5 14607117 63481464 conv\_forward\_kernel*  *0.0 2912 2 1456.0 1376 1536 do\_not\_remove\_this\_kernel*  *0.0 2816 2 1408.0 1344 1472 prefn\_marker\_kernel*  *CUDA Memory Operation Statistics (nanoseconds)*  *Time(%) Total Time Operations Average Minimum Maximum Name*  *------- -------------- ---------- -------------- -------------- -------------- --------------------------------------------------------------------------------*  *91.7 995553661 2 497776830.5 406012837 589540824 [CUDA memcpy DtoH]*  *8.3 89536710 6 14922785.0 1440 48000977 [CUDA memcpy HtoD]*  *CUDA Memory Operation Statistics (KiB)*  *Total Operations Average Minimum Maximum Name*  *----------------- -------------- ----------------- ----------------- ----------------- --------------------------------------------------------------------------------*  *1722500.0 2 861250.0 722500.000 1000000.0 [CUDA memcpy DtoH]*  *538919.0 6 89819.0 0.004 288906.0 [CUDA memcpy HtoD]*  *Profiling results from Nsight-Compute (next page):* |
|  |
| * 1. What references did you use when implementing this technique? |
| *I followed the same steps as in Lab 4 (3D convolution) to allocate constant memory and copy the convolutional mask.* |
| 1. **Optimization 2: *<optimization name>*** |
| 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 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>* |