**Performance Analysis – ECE 569 HW4**

**Rishab Verma** [**-rishv1995@email.arizona.edu**](mailto:-rishv1995@email.arizona.edu)

**Date 3/17/2021**

The Table below showcases difference between kernel function start and end time.





**Analysis-1: Discuss your findings based on experiments 1 and 2 for Versions 0 and 1, Global vs Shared memory implementation. (1 page limit)**

**Answer:**

From experiments 1 and 2, we can clearly see the benefits of Shared memory implementation of histogram generation. Looking at the graph, when the inputs are randomized with random atomic addition operations, shared memory- privatized implementation has ~30% higher execution rate and consistently out performs the global memory implementation.

Shared memory in general, allows reducing memory latency and is an ideal method/source for atomic operations as well. When we add in atomic operations in shared memory this memory is available at thread block level, therefore reducing memory access. On top of these benefits, creating private copies and then merging them in global memory, significantly reduces our memory operations on read and writes on the global memory(unlike version 0).

Additionally, when we run our add atomic operations on shared memory (on private copies), there is significantly lower write/read contention as threads are now only competing against the threads in the block only. This therefore reduces serialization for thread operations i.e. threads have a smaller “waiting” period. In version 0, this is not the case and the serialization from atomic operations has a longer wait period as now threads from the grid are in contention for write/reads on same address.

While Experiment 1 only has ~30% performance improvement for version 1, Experiment 2 is where shared memory with privatized copy implementation of histogram generation really showcases it’s superior performance. Version 0 has a much better perfromance benefit ~700% when the atomic operations are happening on the same address for a large data set. This benefit can be observed again due to the utilization of shared memory application.

**Analysis-2: Explain your implementation approach for Version-2. Discuss the superiority of your approach (Version 2) compared to versions 0 and 1. What is the parallelization opportunity that you exploited or what drawbacks of Versions 0 and 1 did you resolve? Does performance vary based on the nature of the distribution of the data? For which test cases does Version-2 perform better than Versions 0 and 1? Discuss any relevant execution time trends. (1.5 page limit)**

**Answer:**

**I was unsuccessful in implementing version 2 of Assignment4. My intention for this implementation was to Utilize a sorted histogram array implementation which utilized shared memory accesses on privatized copies. I intended on sorting the input array and running threads on bin strides.**

**Short Answer**

**How many global memory reads are being performed by each kernel? Explain.**

**Answer:**

**Version 0 – num elements**

**Version 1 – num elements**

**How many global memory writes are being performed by each kernel? Explain.**

**Answer:**

**Version 0 – num elements**

**Version1 - num elements/num bins**

**How many atomic operations are being performed by each kernel? Explain.**

**Answer:**

Version0 - Num elements

Version1 – Num elements/num bins