Fundamentals of Computing Systems Homework Assignment #7 - EVAL

1. EVAL Problem 1

   (a) **Relationship Between Instruction Count and Request Length**

   Except "IMG_REGISTER" operation, I plot the scatter plot between Request length and Instruction event count. The result is as follows:

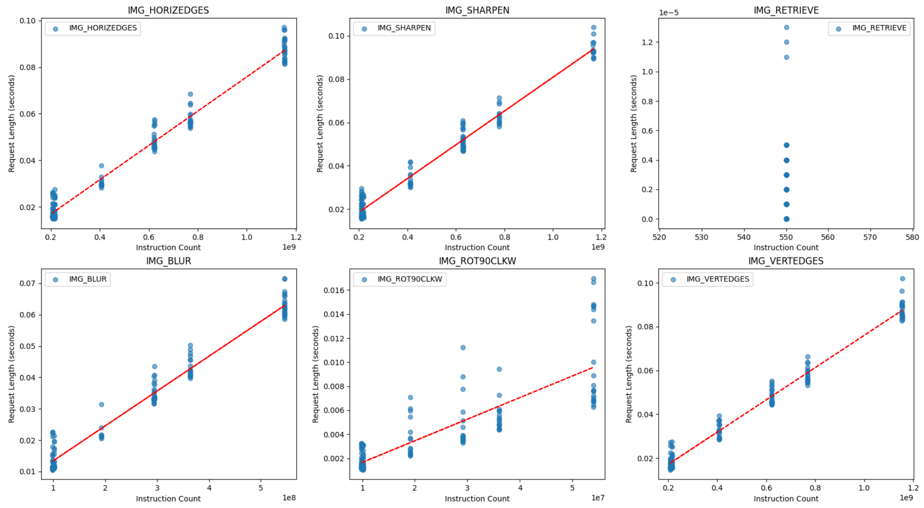

Figure 1: Relationship between Instruction Count and Request Length

The relationship between instruction count and request length, as shown in my plots, is linear for most operations except the "IMG_RETRIEVE" operation. This can be observed through the red line, which represents the linear regression model that studies the correlation between the x and y values. As the instruction count increases, the request length also grows. The scatter plot shows five major clusters, which are caused by differences in 5 images sizes, leading to variation in operation times and instruction counts. However, for the IMG_RETRIEVE operation, there is no significant correlation, as it simply sends the image directly to the client without additional processing, causing the data points to cluster in a similar area.

(b) **LLC Cache Miss Observation**

Except "IMG_REGISTER" operation, I plot the CDF plots of LLCMISS . The result is as follows:

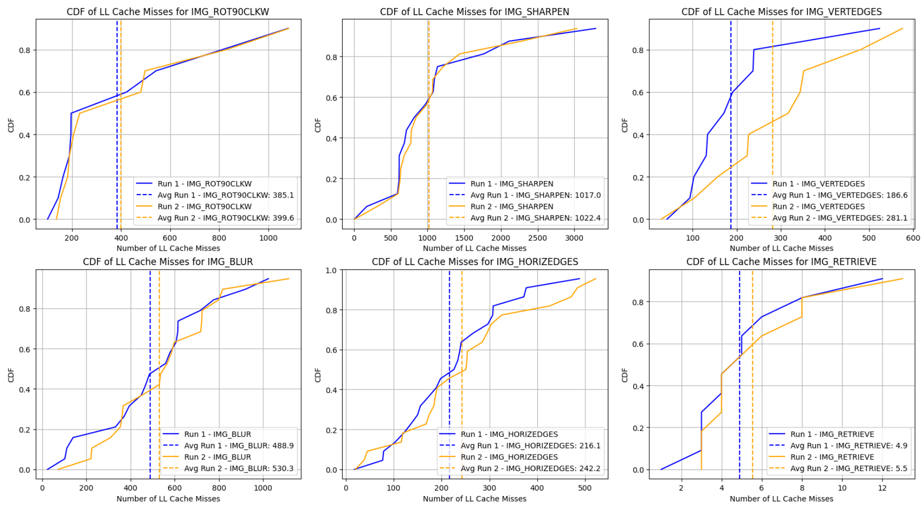

Figure 2: LLCMISS CDF Plots

Operation	**Run 1 (Min, Max)**	**Run 2 (Min, Max)**
IMG_BLUR	Min: 25, Max: 1024	Min: 73, Max: 1116
IMG_HORIZEDGES	Min: 18, Max: 488	Min: 20, Max: 522
IMG_RETRIEVE	Min: 1, Max: 12	Min: 3, Max: 13
IMG_ROT90CLKW	Min: 101, Max: 1080	Min: 137, Max: 1084
IMG_SHARPEN	Min: 6, Max: 3288	Min: 9, Max: 3033
IMG_VERTEDGES	Min: 42, Max: 524	Min: 29, Max: 576

Table 1: Operation statistics for Run 1 and Run 2

The two scripts for run1 and run2, differ significantly in sequence structure and operation distribution. Run1 features a more structured and consistent pattern, where operations like ‘r‘, ‘b‘, ‘s‘, ‘v‘, and ‘h‘ are repeated orderly. This structured approach indicates a controlled and systematic evaluation scenario. On the other hand, Run2 has operations in a more varied and sporadic order. This indicates a random workload that evaluates system behavior under unpredictable conditions.

In the CDF plots generated from these scripts, the differences between Run 1 and Run 2 are clearly visible. Generally, Run 1 (in blue) performs better, showing fewer cache misses than Run 2 (in orange). The average LL cache misses for Run 1 are consistently lower, which suggests that Run 1 benefits from more efficient system resource use. The differences in the minimum and maximum values for LL cache misses, as shown in the tables, indicate that Run 2 tends to have higher cache misses. This might be due to a colder cache or increased contention for resources.

The statefulness of the image server resources plays a significant role in these differences. Run 1 likely benefited from a warmed-up cache, reducing cache misses and improving performance. In contrast, Run 2 experienced more cache misses, potentially due to changes in resource availability or workload patterns. This indicates that server performance can be significantly influenced by its prior state. This emphasizes the need for optimizing cache usage and managing workloads to achieve consistent performance.

Combining these observations, it is evident that the structure of the workload (as seen in the two different scripts) and the state of the server significantly impact performance. To minimize such variability, it would be beneficial to implement better resource state management, such as optimized caching mechanisms. This would enhance consistency and efficiency across runs.

(c) **L1 Cache IMG_BLUR and IMG_SHARPEN**

I plot the Historgram Bin plots of ”IMG_BLUR” and ”IMG_SHARPEN” operation of L1MISS . The result is as follows:

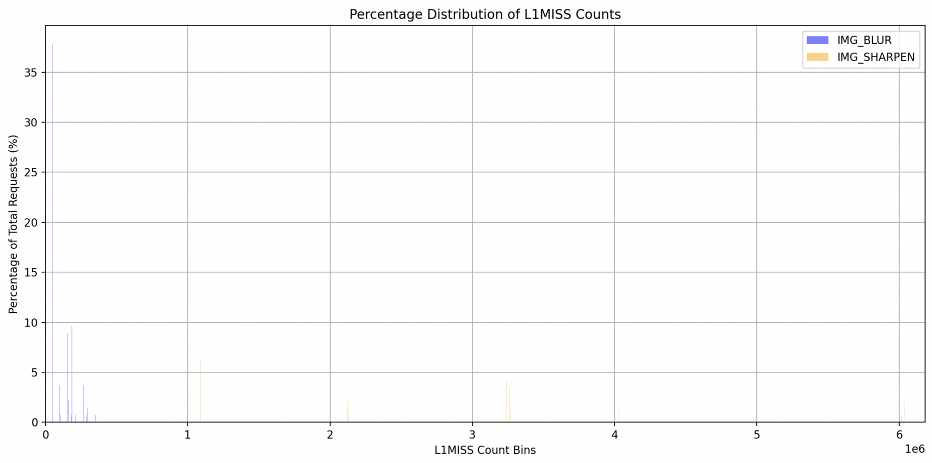

Figure 3: L1MISS IMG_BLUR and IMG_SHARPEN Histogram

The distribution for ‘IMG_BLUR‘ shows a few spikes in lower bins, indicating that it generates a notable number of L1MISS counts in those ranges. In contrast, ‘IMG_SHARPEN‘ exhibits a more concentrated distribution, with its counts appearing primarily in specific bins. This suggests that it may have fewer overall L1MISS occurrences than IMG_BLUR. IMG_BLUR data appears more spread across the lower bins, while ‘IMG_SHARPEN‘ has a sharper peak. This reflects the differences in memory access patterns and processing methods between the two functions. Overall, ‘IMG_BLUR‘ demonstrates a broader distribution of L1MISS counts, while ‘IMG_SHARPEN‘ shows a more focused distribution. This indicates distinct behaviors in memory usage.

BlurImage and sharpenImage functions, while performing similar operations on an image, exhibit vastly different L1 cache behaviors due to factors. Firstly, the memory access patterns differ significantly; the blurImage function accesses neighboring pixels in a straightforward manner, iterating over a 3x3 kernel, which results in a predictable and localized memory access pattern that is generally cache-friendly. In contrast, the sharpenImage function employs a more complex kernel that emphasizes the center pixel. This leads to less predictable access patterns and cache misses. Additionally, the blur operation averages kernel data, allowing for better cache utilization as the same data may remain in the cache for subsequent accesses. The sharpen operation, with its negative weights and more complex calculations, may require accessing different memory locations more frequently, further impacting cache performance. Moreover, the spatial and temporal locality of data access plays a crucial role; the blur function benefits from spatial locality by processing pixels in a linear fashion, while the sharpen function may not leverage temporal locality effectively due to its kernel's nature. Edge pixels also affect cache behavior, as memory access jumps can disrupt cache prefetch data. Lastly, the sharpen function involves more arithmetic operations, such as multiplication with kernel values. This can lead to different CPU pipeline behaviors and cache usage patterns. Overall, these differences in memory access patterns, data locality, kernel complexity, and arithmetic operations contribute to the contrasting L1 cache behaviors observed in the two functions.

To improve the operation with more L1 cache misses in the 'sharpenImage' function, consider optimizing memory access patterns by implementing techniques like loop tiling or blocking. This approach reorganizes the way data is accessed in memory, allowing for better cache utilization. Additionally, reducing the kernel complexity or using a smaller kernel size can minimize the number of memory accesses required. Finally, ensuring that data is accessed in a more linear fashion can enhance spatial locality, further reducing cache misses.