Celeste Shen and Sandy Zhu
University of California at Santa Cruz

During the internship, the objective was to find the “best” performing hash function for partitioning different types of graphs. This means balancing data as evenly as possible across all partitions. We measured this by comparing the load imbalance of each hash function.

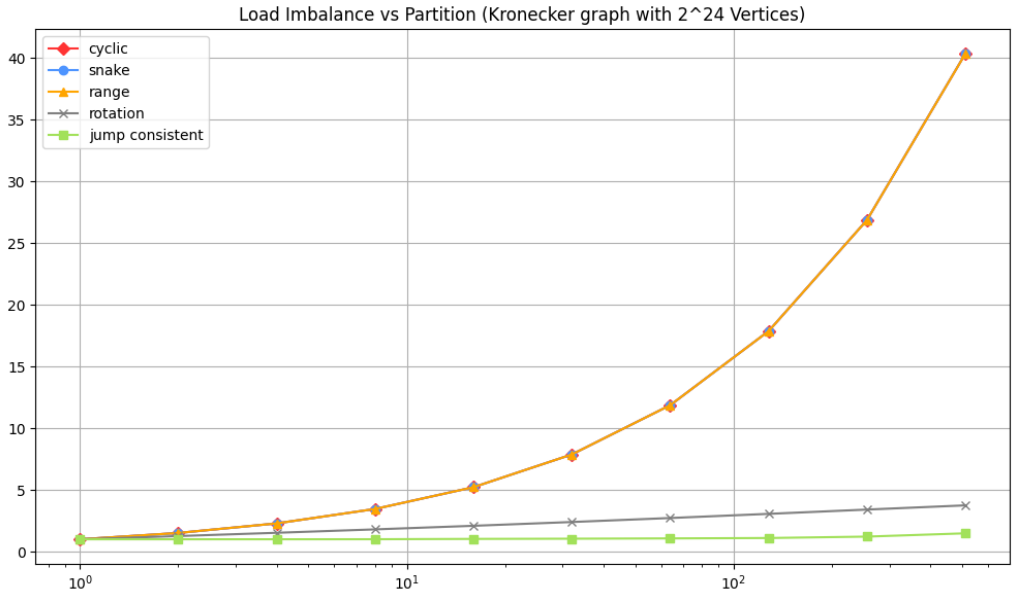

Figure 1. Load Imbalance vs Partition Graph with scale 24 and 5 hash functions

In a non-shuffled kronecker graph, we found that jump consistent hash performed the best, while cyclic, snake, and range seem to grow exponentially as the number of partitions increases. Rotation also looked good as well, not better than JCH but it was consistent compared to the others.

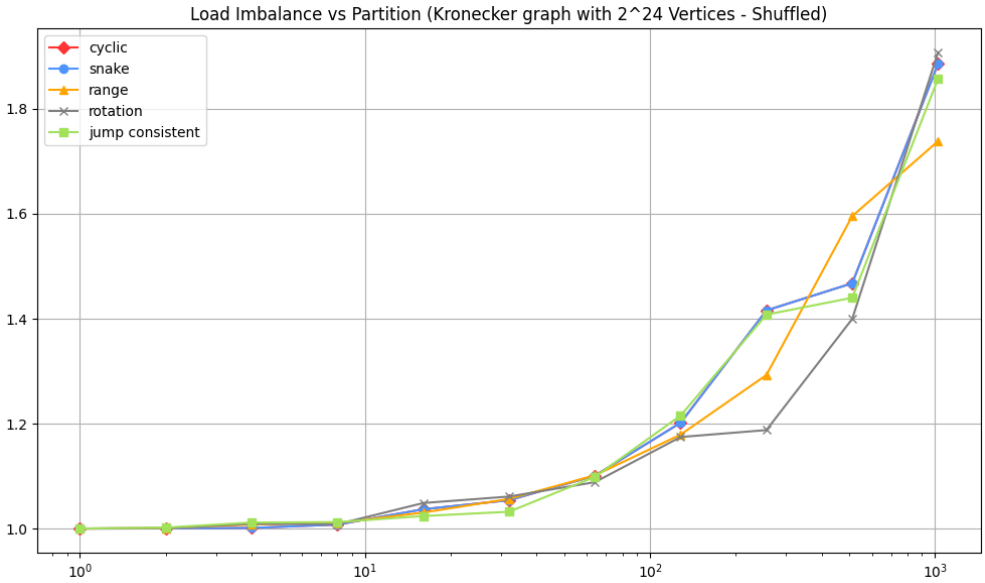

Figure 2. Load Imbalance vs Partition Graph with scale 24 and 5 hash functions (SHUFFLED)

On the contrary, in a shuffled graph, the load imbalance of each hash function seems to drastically change. Jump consistent hash seems to have a much more average performance. Range has the best performance at the most partitions, despite having the worst load imbalance in figure 1. Rotation had a low load imbalance until we reached higher partitions, where the load imbalance jumped significantly.

We also ran our load imbalance function on already built graphs, such as a static twitter graph and a road graph:

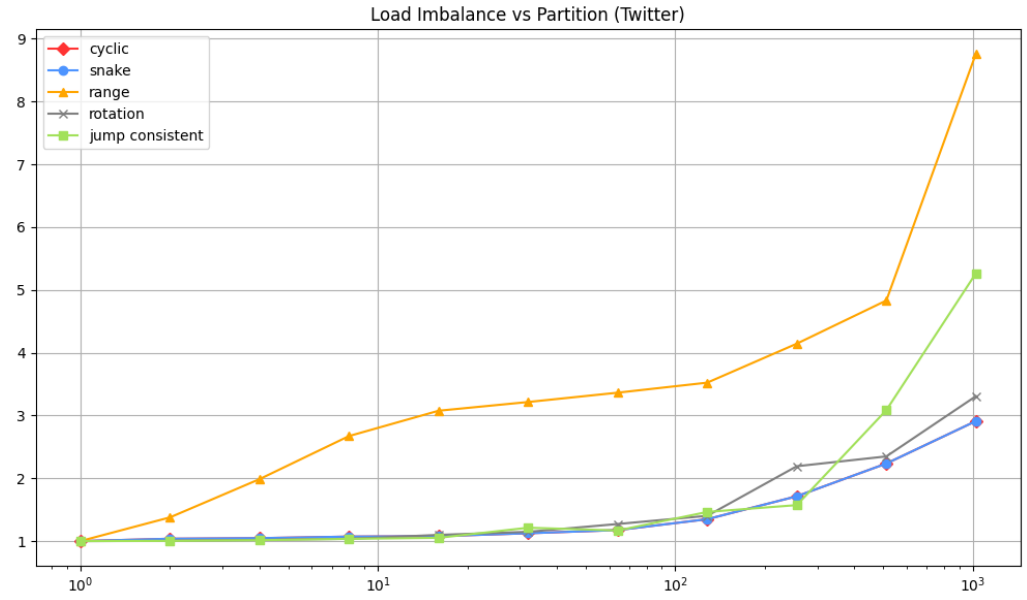

Figure 3. Load Imbalance vs Partition on Twitter graph

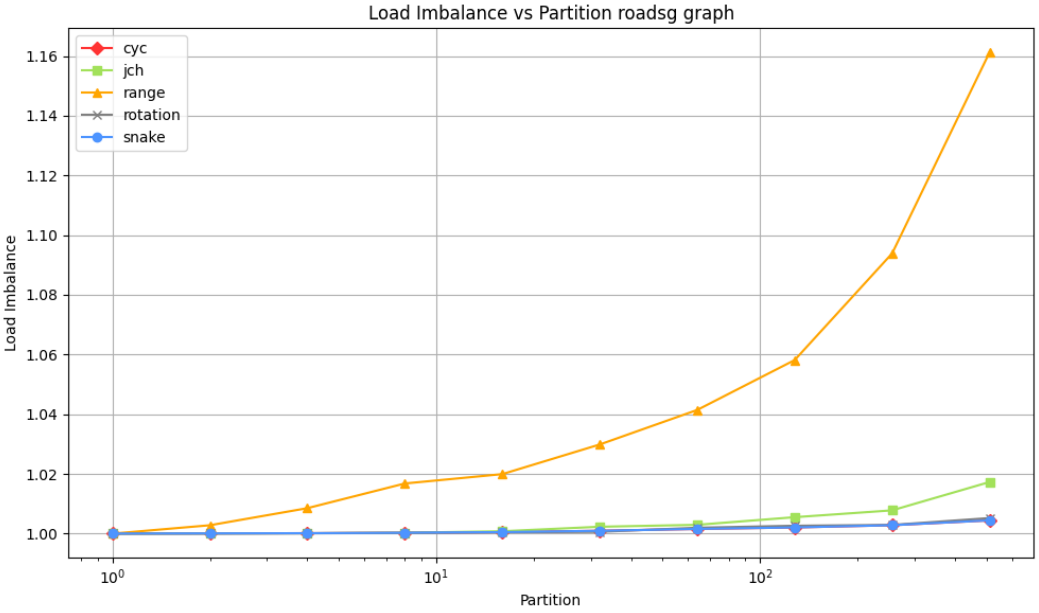

Figure 4. Load Imbalance vs Partition on Road graph

Looking at figure 3 and 4, it was surprising to us to see that snake, cyclic, and rotation seemed to perform the best on both graphs, while previously having the worst load imbalance, like in figure 1 and 2 (except rotation). We observed that depending on the nature of the graph, the load imbalance for each hash function could change drastically, so it wasn't conclusive which hash function is overall the best one.

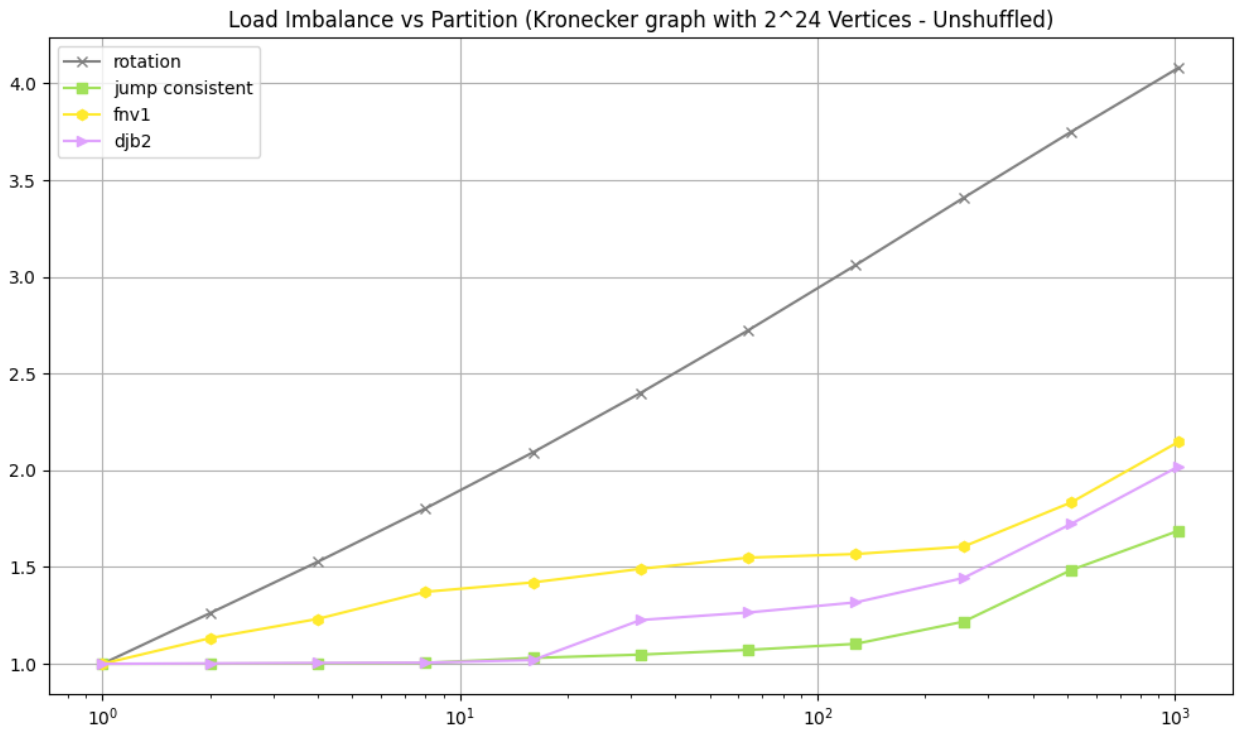

Figure 5. Load Imbalance vs Partition on unshuffled graph

We also incorporated different hash functions. We tried one called "djb2" and "fnv1". Looking at figure 5, we can tell that rotation has the worst load imbalance in an unshuffled graph, while the JCH hash still performs with the lowest loading imbalance. Fnv1 and djb2 perform well but not better than JCH.

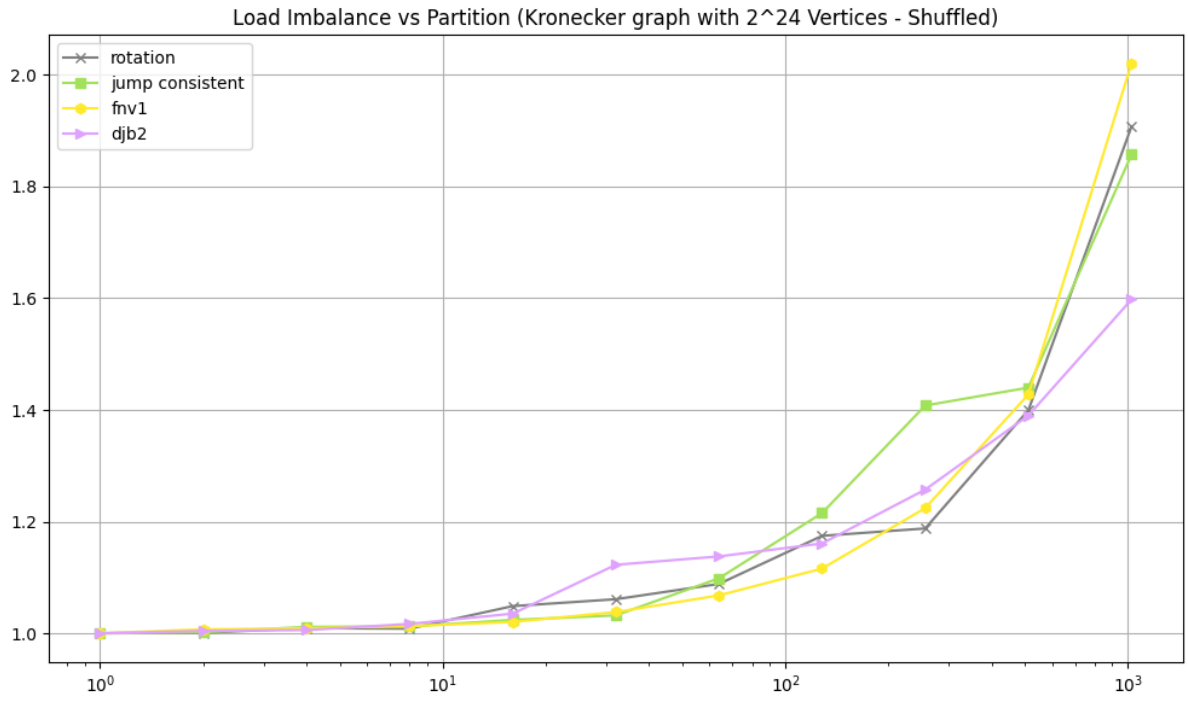

Figure 6. Load Imbalance vs Partition on shuffled graph

In figure 6, we can see that djb2 performs very well at higher partitions. Fnv1 performs well at a lower partition count. Overall, djb2 seems to perform the best in this shuffled kronecker graph.

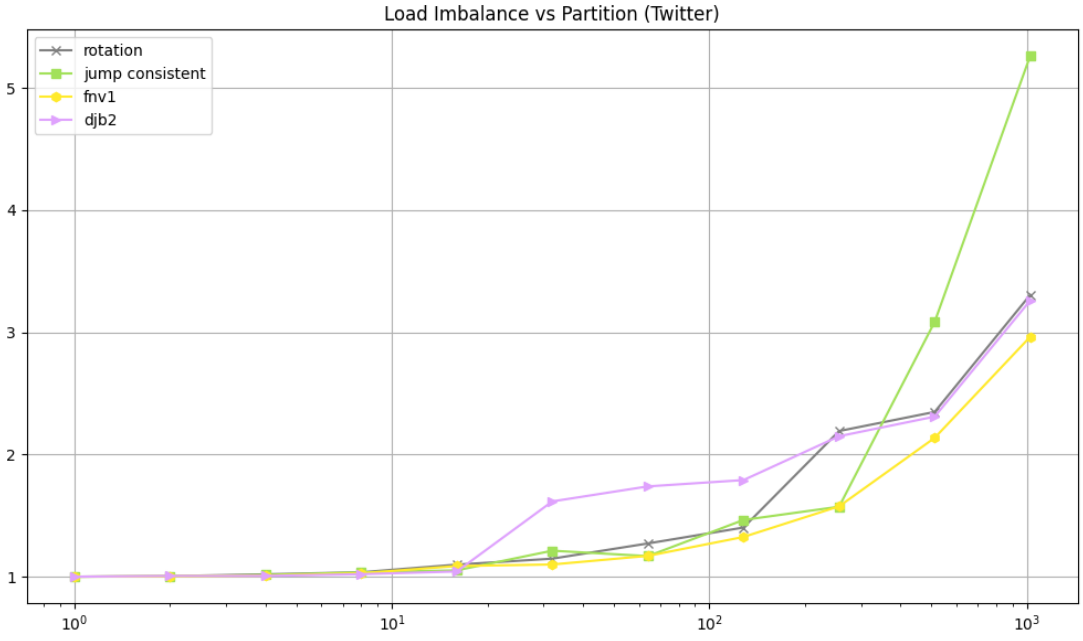

Figure 7. Load Imbalance vs Partition on Twitter graph

On a static graph like the Twitter graph, in figure 7 we can see that fnv1 and djb2 outperform jch at most partition counts. Fnv1 performs overall best in terms of load imbalance on this graph.

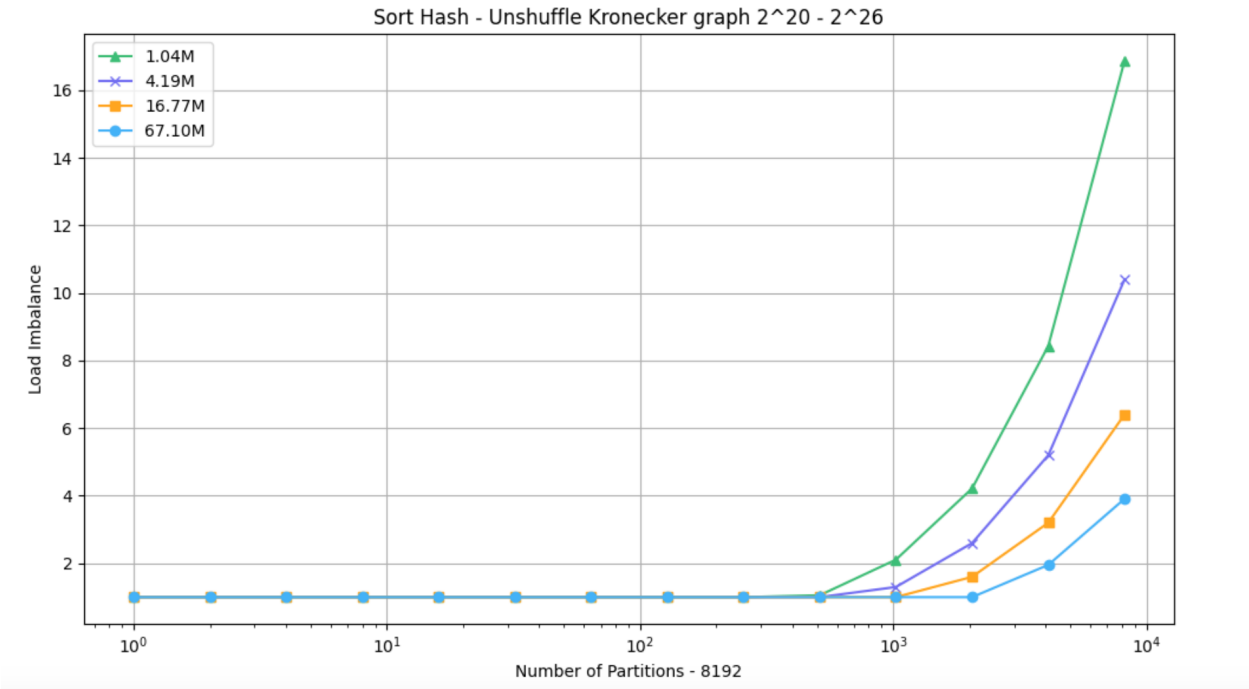

Figure 9. Sort Hash - Unshuffle Kron graph scale 2^20 - 2^26

We also came up with our own hash function, ‘Sort hash’. It first sorts the vertices by their out-degrees in a descending order. Then it identifies the partition with the minimum out-degree and assigns the next vertex to these partitions. This helps us achieve a low loading imbalance and the ‘sort hash’ has the lowest load imbalance compared to the rest. As shown in figure 9, we can see that as the scale of the graph increases, the load imbalance decreases.