Jackson Dawkins, Sarah Malick, Spencer Provost, Abrar Basha
Group 3
Project 4 – cpsc 3620
Analysis of results

The dynamic workload allocation seemed to be optimal when run on both the CloudLab cluster, as well as the Palmetto Cluster. We came to this conclusion on a couple of bases. The execution time for every number of workers was better with the Dynamic implementation than with the static implementation, assuming the hardware was constant. The total sleep time for each worker was also significantly lower on both sets of hardware for the dynamic implementation.

The dynamic implementation of the code had a significantly lower variance, as well. This means that the work done by the processors doing work within the static implementations had much more variation than with the dynamic implementation.

There are a couple possible explanations for the large advantage seen by the dynamic implementation. The most prevalent is probably the design of the algorithm used. The algorithm took the work queue and split it evenly by work segments to the worker processes. This allows for the possibility that the randomness of the queue could possibly give one process much more or less work than the others. The alternative of this method would have been to calculate the total work that would need to be done to process the entire queue before splitting it up. This would have allowed the master process to send a more equal workload to each of the workers. The downside to this method is that it takes much more work at the beginning to allow for equal distribution.

We assume that based on the randomization of the work queue, there would be scenarios where static workload balancing method would be more efficient than the dynamic workload balancing. The randomization of the queue during testing simply didn't allow for this.

In both implementations, there is quite a bit of communication between processes. The communication between the processes of the dynamic implementation is definitely much more cumbersome than that of the static implementation. This would also slow down the dynamic implementation in many scenarios.