Jordan Itzkovitz

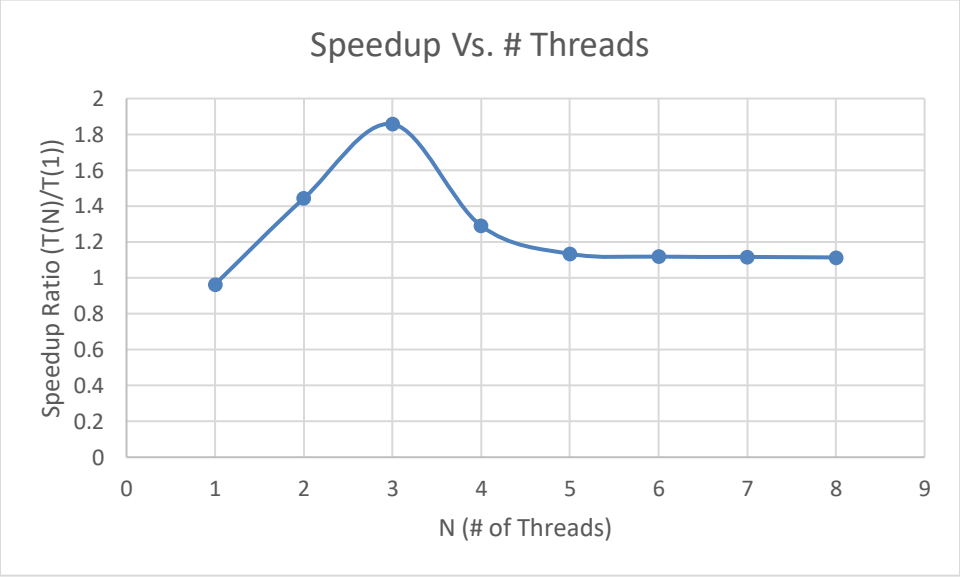
260687254

COMP409

Wednesday February 6th 2019

Assignment 1

Q1)



The plot above describes the relative speedup of the image processing algorithm in accordance with the number of threads being used for computation. As shown in the graph, the algorithm increases in performance as the number of threads utilized increases up to 3 threads, at which point there is a decline in performance as the number of threads increase to 8. These experiments were performed on a quadcore processor, meaning that each of the 4 CPU's are assigned their own thread when up to 4 threads are being used. For greater than 4 threads used there is an overhead associated to thread switching between each CPU that is assigned more than 1 thread, thus causing a slow down in performance. Moreover, this algorithm is implemented in such a way that the *outputimage* is synchronized, thus only allowing a single thread to write to it at a time. This is done due to the fact that pixels in a *BufferedImage* may be grouped together into a single array slot, and thus writes can affect each other dangerously. This is likely why there is a decline in speedup after 3 threads – each thread ends up waiting on the other threads to write to *outputimage*, and the more threads there are the more waiting may incur. In general, the algorithm assigns each thread a different position of the image to perform computations on. After completing such computations for the output image pixel, each thread is assigned a new position N positions away from it's current one – N being the number of threads

assigned to the overall computation. Thus if each pixel held its own array position then no synchronization would be needed. The other shared resources, including the input image and kernel, are only ever being read from, and thus do not require synchronization.