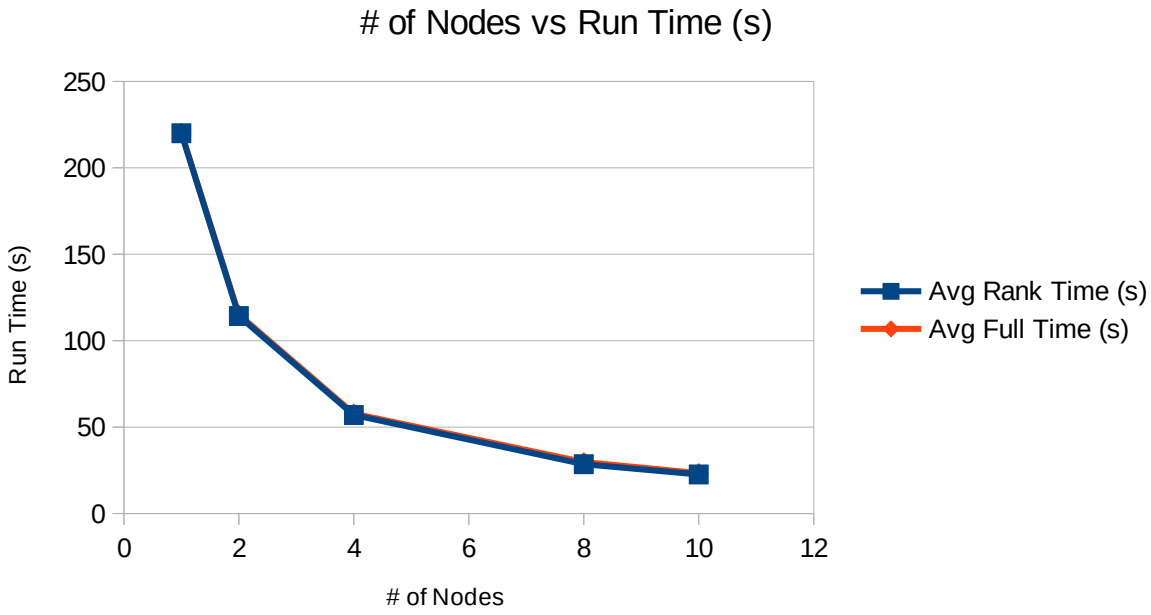
Michael Rivnak
COMP3450
Prof Deligiannidis
9 July 2020

Assignment LA4: Image Convolution

For this assignment we were asked to parallelize a convolution filter on an image. Even with a reasonable amount of parallelization, this task is still reasonably computationally intensive on large images. The image that was used in this testing was 4000x3000 pixels and even with 10 threads it took ~22 seconds to complete. The average times are shown in the graph below.



Clearly, parallelization has a considerable impact on performance, and this algorithm looks like it scales very well. I say this because of how nearly identical the full time and thread times were, thus it is likely that the additional overhead from sending data to and from threads will only have a significant impact on performance when extremely high thread counts are used.

As far as different convolution kernels are concerned, I tried adjusting the radius and found that a larger radius increase the blurring effect, this higher blur is likely to increase computation time a bit since it is operating on more pixels.

For applying a convolution kernel to a movie it is not as straightforward to parallelize as you might imagine. I worked with video files during my co-op last summer, and the issue is that compressed video formats do not store every frame, rather they store the differences between frames. This would make parallelizing by frames more difficult since you have to read each frame sequentially.