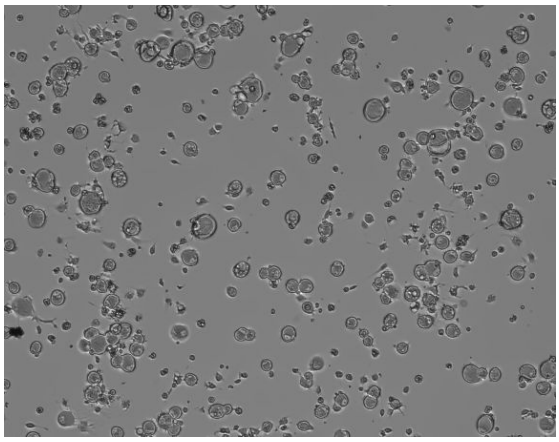
Matt Howa
Lee Leavitt
HPC Final Project
12/20/2018

Parallel Cell Profiling

Introduction

In the lab lee works in he runs test on cells by adding chemicals to them then flashing a light of a specific wavelength through them to test the effects the chemicals had on the cells. They then do this over and over capturing a picture at each step. In order to determine the effect on the cell, the mean intensity of each cell is calculated for each picture taken. These results are then taken into another program to run other algorithms on. The extent of our project starts and ends with calculating the mean intensity of each cell for each frame. Our intent is to parallelize this using OpenMP and MPI in order to speed up the calculations. Using the program they currently have can take up to 2 hrs on files that are about 4GB. The process is cpu heavy making the Computer unusable for the span of the calculations. Our hope is to speed up this process so that the lab does not see as much down time when running their tests.



This is a single image we would calculate the mean intensity for each cell on

Our Approach

We will try to parallelize using MPI. We will separate the individual images uniformly among the processors. Then when calculating the mean intensity per cell (well call this a Region of Interest or ROI), we will try to use OpenMP to parallelize at the thread level. Images are independent of other images and rois are independent of each other so we have no data dependence that is preventing us from doing either of our ideas.

Problems and Solutions

Due to the file format being a Nikon proprietary raw file we needed a special library to read the images and do our calculations. The library is only available in python so we had to write the meat of the program using it. This introduced a new issue. Thread level parallelism

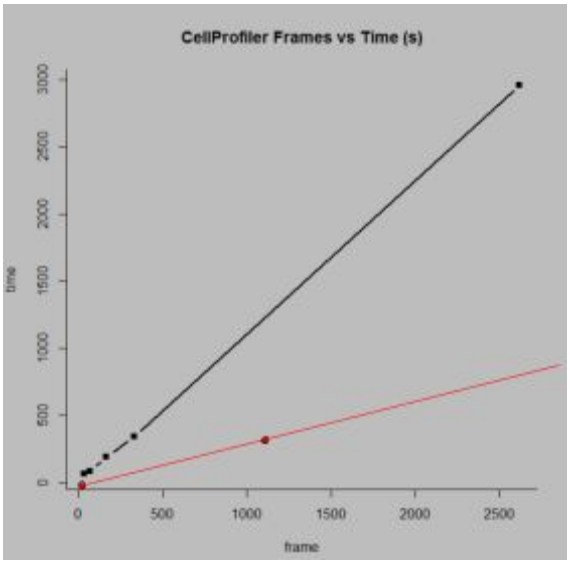
does not exist in standard python. So in order to take advantage c++ and openmp we wrote a swig wrapper around the code for calculating each ROI's mean intensity.

Results:

After a lot of hacking and initial learning curve, I was able to compile and link the swig wrapper in order to make use of openmp. However, even though the code linked and ran fine I was never able to get the OpenMP threads to work. No matter how I set the thread count I never saw more then 1 thread per process working. I added the library to the c++ file and compiled using the proper OpenMP threads. Infact I made sure the library worked by using some of the built in functions. Nonetheless, I could not get it to work properly and online documentation is scarce. I am confident that with more time and tinkering it would eventually work, however this is much as I can get to in the time we have.

With that said the MPI only implementation shows a huge speedup, which you can see in the graph below. For a 59 frames without parallelism would execute in about 25 seconds and with MPI it would finish in just under 9 seconds.

When looking at a much larger file, +1100 frames without parallelism would take hours in the lab, took under 300 seconds. Now these aren't great measurements as I did not get the exact timings from Lee due to communication issues, I will say the improvement is massive.



In red you can see the the new results against the black which is the old results. Again due to lack of communication I do not know how accurate Lees results (in black) are but still a considerable speed up can be seen. Another think to note is this is a pure MPI only implementation. I believe if I was able to get the OpenMP library to spawn of more threads the speed up would be much, much better.

Conclusion

I have learned a great deal this semester using MPI and OpenMP. It was exciting to use what I learned and apply it to a real world application. Although I was not able to complete all the goals I had when I set out on this project, seeing the results was a positive thing in the end. It seemed obvious we would see a speedup however seeing it work in practice was great. Lee knew the problem scope from working in the lab over the years however it was all new to me (Matt). Being able to dissect the problem and apply a solution was fun and rewarding. I had a lot of trouble getting the c++ code wrapped with swig, however that was also a very rewarding experience once it compiled and was able to be used within the python code. With all that said I believe with time constraints and my ability I completed this project as well as i could. Please see the code base ReadMe in order to install and run the program. Please note you will need root access to the machine you run it on. Below is a link to the Google Drive with some sample data used

Data:

https://drive.google.com/drive/folders/1x6FWT0fptIsdHOpZl9D3vnuKLtdHEQ3Y?usp=sharing