Mark Randles
Lab #2
2004-03-22

The results for this lab were to be expected. The actual implementation of the algorithim was pretty straight forward. First a single processor implementation was formed. From that the statically allocated version of the multiple processor program was formed. Also from the single processor version a dynamically allocated version of the multiple processor program was formed.

After the programs were developed and verified to output the correct data, they were each run for 5-33 processors, or 4-32 worker processes, incrimenting by 2 between each process. The single processor number was found by running the program interactivly on the cluster. The speedup was caculated and then plotted against the number of processors.

The results were a bit suprising, but not to be expected based on the actual computations needed to solve the problem. The statically allocated solution was much slower then dynamically allocated solution, however each was much faster then the single processor solution.

This dicotomy in the different data allocation types stems from the fact that the dataset in question has parts that are harder to caculate then other parts. In the statically allocated solution afew of the processes will get the very computationally heavy part of the graph, while a few other processes will get the computationally easy part of the graph. The dicotomy in computation times is what drives down the performace of the statically allocated solution, as there are processes that idle much longer then others.

Now simple logic would present that the dynamically allocated solution is much slower because there is more communication overhead. However this isn't the case. The communication overhead isn't a immediate limiting factor. The dynamically allocation provides that each process is kept at pretty much peak utilitation for the entire run time. Only at the end as the last few datasets were finished, did processes need to wait.

The results from this lab showed in the barest sense that the granularity of the data provided to the workers really does affect the performance of the program. If you have a computationally intensive caculation to make, and it doesn't affect the entire dataset equally, then a small granularity is in order. However, as was shown in lab 1, if the load on each worker would esentally be the same for their part of the dataset, then a large granularity is in order. A more indepth study of the actually computation vs. communication time would reveal some of the intricincies that present themselves in the division of the dataset.