Michael Hoppal
November 9th, 2014
Program 4

For this program we revisited the Jacobi 1D problem that we looked at in openMP. In our results in openMP we saw that although we got some speed up as we used more threads there was a certain point in which the speed up flattened off. This was because of the memory wall that we hit as more threads were trying to use the cache.

In this program we tried to solve this issue by using MPI, which allows us to pass messages with data between multiple nodes or computers. In result we can split up the problem set and send it to multiple computers/nodes to accomplish work. Because of that we will not hit that memory wall as we did in openMP and will hopefully see a linear speed up.

To actually implement the MPI version of Jacobi you need to first think on how you are going to split up the work and pass data between them. Well you can split up the dataset evenly across the number of nodes that you are leveraging in MPI. But once you have done this, can each node process the data they have independently from each other? The answer is no as the edge elements of each dataset depend on the data that is either to the right or left of it being processed in other nodes since the equation of Jacobi depends on prev[i-1], prev[i] and prev[i+1]. This is where the message passing comes in. Each node is going to have to pass edge data to the nodes that are working on datasets to the left and right of it (unless the data set is the first or last). This is using ghost cell buffers.

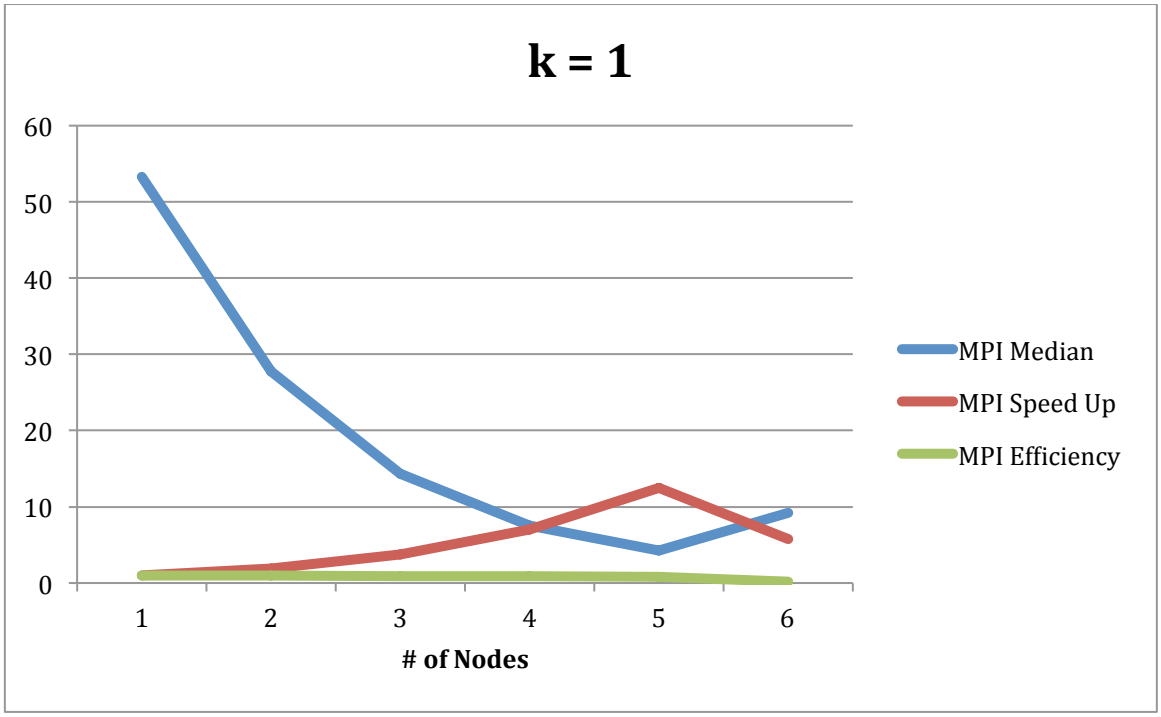
In each iteration do you have to pass data? Well if you only pass one element you will have to as the next edge element depends on the curr edge position + 1. So the nodes will have to communicate every iteration. To able to bypass communicating every iteration we can send more data. By doing this we can expand our calculation area by performing the Jacobi equation in the ghost cell buffer and than we will have more edge elements as we move down the table. How many iterations can we go before we have to communicate again depending on the buffer size we send? Well if we send k size buffer of the prev vector to our neighbors our neighbors will able to go k iterations before they will need data again. This is because as you move further down the data set those edge elements will need the calculated data from the neighbors as we can only perform a certain amount of Jacobi calculations in the ghost cell buffer.

After we figured out how to write the MPI Jacobi program we went through experiments with different ghost cell buffers, K, and number of nodes we are leverage, N. We used the same data set for each experiment. These results are shown below along with the analysis.

Sequential Time: 53.26

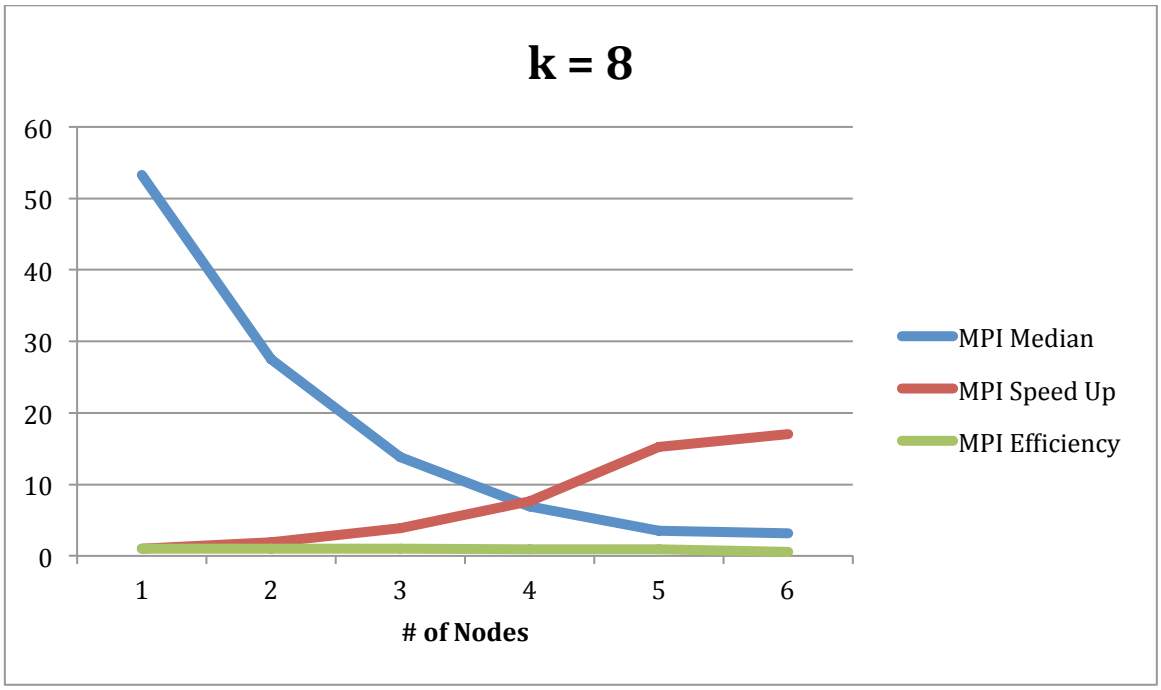
K = 1 (1 Ghost Cell on each side)

N Value:	Median	Speed Up	Efficiency
1	53.26	1	1
2	27.73	1.920663541	0.960331771
4	14.32	3.719273743	0.929818436
8	7.56	7.044973545	0.880621693
16	4.26	12.50234742	0.781396714
32	9.22	5.776572668	0.180517896



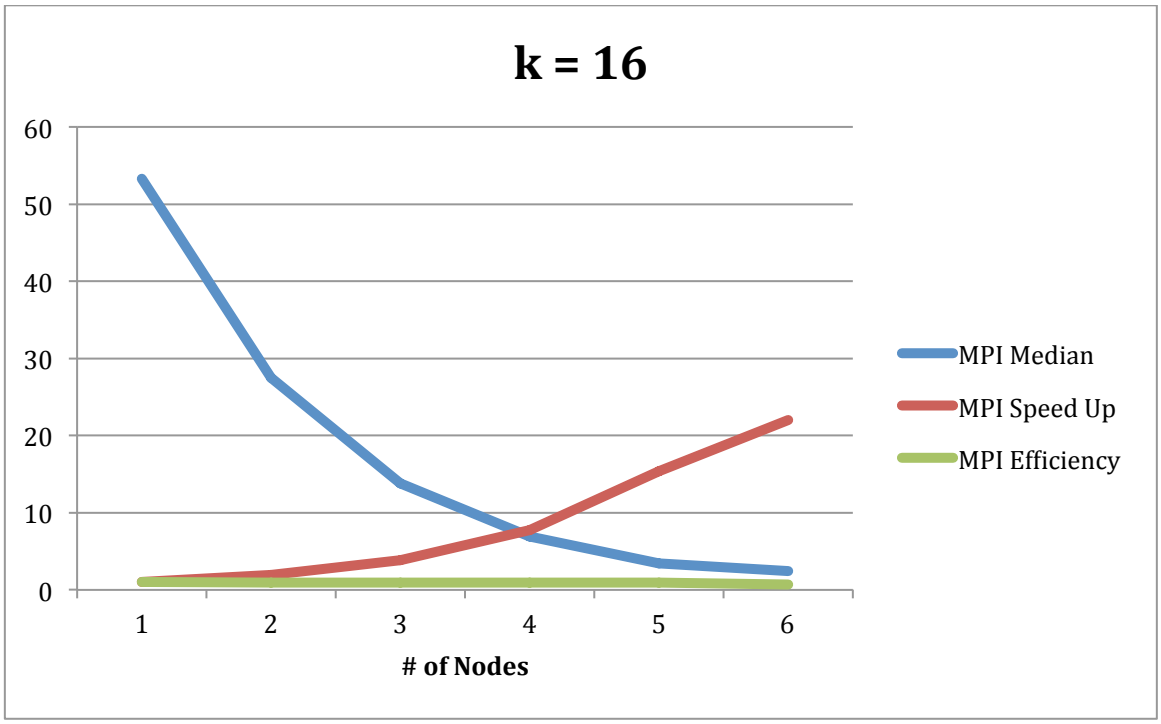
K = 8 (8 Ghost Cell on each side)

N Value:	Median	Speed Up	Efficiency
1	53.26	1	1
2	27.48	1.938136827	0.969068413
4	13.79	3.862218999	0.96555475
8	6.94	7.674351585	0.959293948
16	3.49	15.26074499	0.953796562
32	3.13	17.01597444	0.531749201



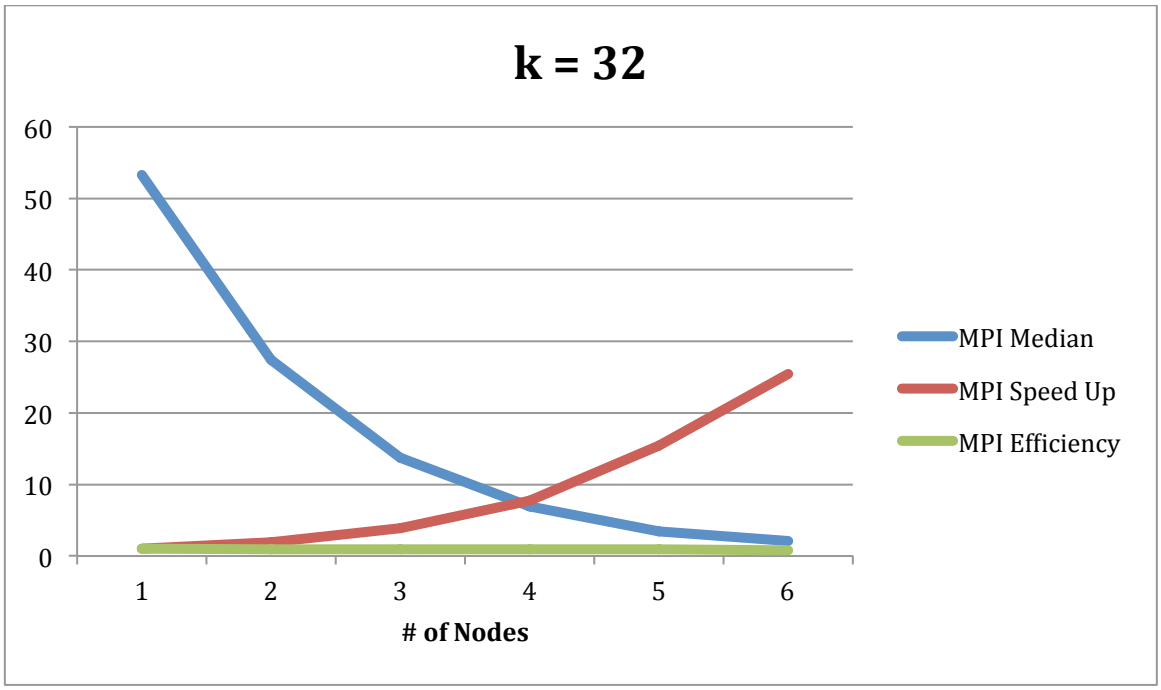
K = 16 (16 Ghost Cell on each side)

N Value:	Median	Speed Up	Efficiency
1	53.26	1	1
2	27.47	1.938842373	0.969421187
4	13.79	3.862218999	0.96555475
8	6.91	7.707670043	0.963458755
16	3.47	15.34870317	0.959293948
32	2.42	22.00826446	0.687758264



K = 32 (32 Ghost Cell on each side)

N Value:	Median	Speed Up	Efficiency
1	53.26	1	1
2	27.47	1.938842373	0.969421187
4	13.75	3.873454545	0.968363636
8	6.89	7.730043541	0.966255443
16	3.46	15.39306358	0.962066474
32	2.09	25.48325359	0.796351675



Analysis

For these experiments we used k buffer sizes: 1, 8, 16, 32 and n nodes: 1, 2, 4, 8, 16, 32. As we talked about previously the bigger the k value the less communication we have to do and you don't take a hit on how much you are sending since the packet size will stay the same. Also the more n nodes we use the more communication that will take place but there will be more work being done since more nodes are preforming calculations on the data set. So in theory the more data you pass and the more nodes that you use the better speed up you should get.

Now looking at the results we see in the first experiment that we have a linear speed up till we hit 32 nodes and then it dips down. Why is that? This is because not only are there more nodes to communicate with but when we go up to 32 nodes it acquires two boards on the computer while the other n values do not. This results in the communication to take longer as it is incurring inter board communication cost. This is why you see the dip down on the speed up.

Now as you pass more data, increase k, the less you have to communicate with your neighboring nodes and in result should see faster and faster results. We see this trend in our experiments as we add more to the ghost cell buffer our speed up gets larger and larger as we add more n nodes.

Will this trend always continue? No it will not. This is because all you are pass more data although you are communicating less and less you are performing k^2 more calculations since you are performing the Jacobi calculation in the ghost cell buffer on both sides of n's data set. So there is a happy median between how much

communication can be saved by passing more data and how much more calculations need to be performed. Although our experiments do not show this as we add more and more to k our speed up will start to flatten off and start decreasing because of the increase of calculations that need to be done.