CSE 6230, Fall 2014

Lab 4

Ankit Srivastava, gtID: 902838136

October 2, 2014

1

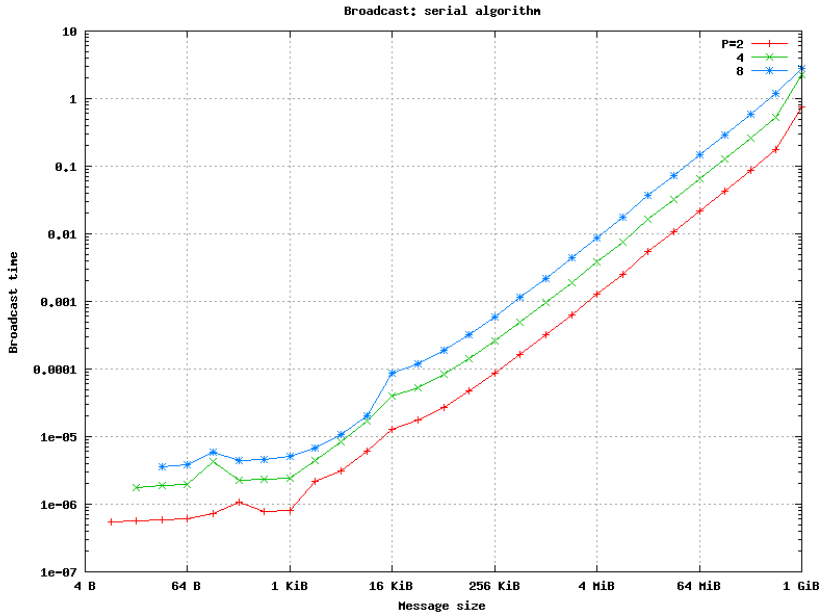

Figure 1

Question

Assume that the time $T(n)$ to send a message of size n words is $T_{msg}(n) = \alpha + \beta n$, where α is the – *message latency*, in units of time, and β is the – *inverse*

bandwidth, in units of time per word. Then, use the data of Figure 1 to estimate the values of α and β. You may either eyeball the plot or use the raw data used to generate these plots, which appear at the following links for two processors (`serial-2.dat`), four processors (`serial-4.dat`), and eight processors (`serial-8.dat`). Explain how you derived your estimate and report α in microseconds and β in microseconds per – *byte*. (Note that the plot shows message sizes along the x-axis in bytes, kibibytes, and mebibytes.)

Solution
In the serial case, node with rank 0 sends the complete message to all the nodes. Therefore, total time for broadcast should be:

$$T = T_{msg}(n) * (P-1) = (\alpha + \beta n) * (P-1)$$

where P is the number of processes. From the above equation, a set of simultaneous linear equations in α and β can be obtained, using the observed times, as follows:

$$\alpha + \beta n = \frac{T}{P-1} \tag{1}$$

Timings for all the different number of processes and different message sizes was reduced using (1) and a number of points were obtained. Following observations were made from observing the points:

1. Values remained more or less constant for message sizes less than or equal to 1 KiB.
2. Values followed a linear trend as message size increased beyond 1 KiB.

Based on above observations, I concluded that α dominates the times for message size less than equal to 1 KiB and β starts dominating as the size is increased beyond 1 KiB. For calculating the values of α and β, therefore, I divided the plot into two parts: $n \leq 1024$ and $n \geq 1024$. Therefore, T_{msg} was modified as:

$$T_{msg}(n) = \begin{cases} \alpha & \text{if } n \leq 1024 \\ \alpha + \beta n & \text{if } n \geq 1024 \end{cases}$$

Using the above model, following values were calculated:

$$\alpha = 0.6834422485 \ \mu s$$

$$\beta = 0.0005388419288 \ \mu s/byte$$

2

Question
Using your model of message time from Question 1, write down an analytical model of the time to execute the tree-based algorithm.

Solution
In the tree-based algorithm, depth of the message sending tree is $log_2(P)$. Therefore, using the model defined above, following can be used for predicting broadcast times using the algorithm:

$$T(n) = \begin{cases} \alpha * log_2(P) & \text{if } n \leq 1024 \\ (\alpha + \beta n) * log_2(P) & \text{if } n \geq 1024 \end{cases}$$

where P is the total number of nodes.

3

Question
Compare these data to your model from Question 2. How well do they agree?

Solution
Using the model defined above, for tree-based algorithm, predicted times were calculated and written to `tree-2-predicted.dat`, `tree-4-predicted.dat`, and `tree-8-predicted.dat`. The plots comparing these values with observed timings are shown below in Figure 2, Figure 3, and Figure 4 for 2, 4, and 8 processors respectively.

The plots for predicted timings follows the plots for observed timings very closely in all three cases.

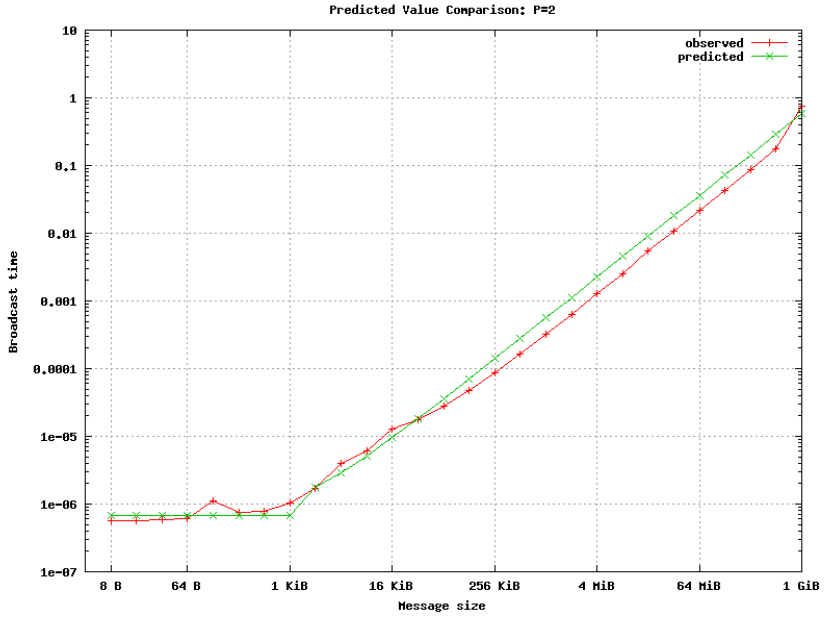

Figure 2

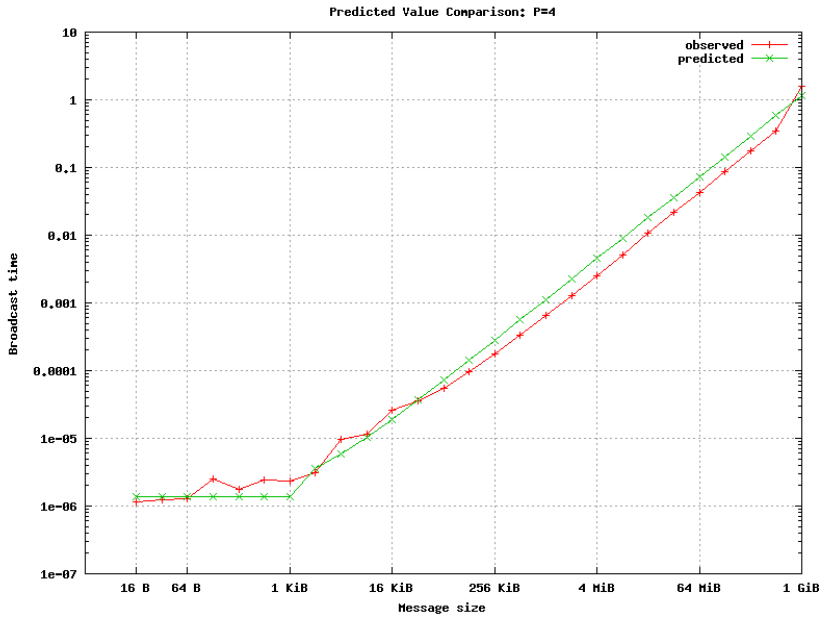

Figure 3

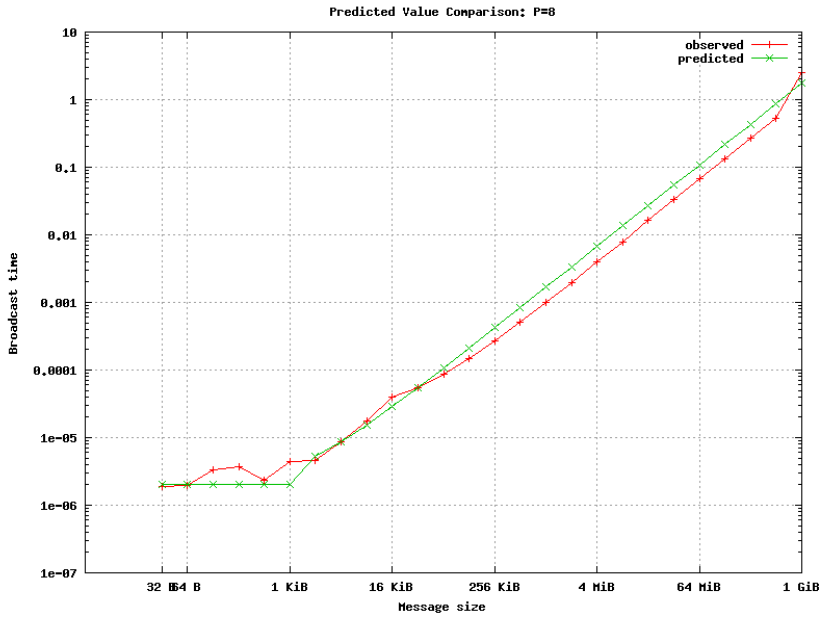

Figure 4

4

Question
Complete the "all-gather" code, using only point-to-point MPI operations (either blocking or non-blocking, as you wish). See the notes below on compiling and testing your code. You may make the same assumptions as we do in the tree-based algorithm: power-of-two number of processes and the number of processes evenly dividing the message length.

Solution
Completed implementation of the "all-gather" code in `bigvec.c`.

5

Question
Plot the `serial-8.dat`, `tree-8.dat`, and `bigvec-8.dat` together. You should see that the tree-based method is faster for "small" messages, whereas the scatter+allgather method if faster for "large" messages. What is the cross-over point (message size) between the tree and scatter+allgather methods?

Solution
The obtained plot for `serial-8.dat`, `tree-8.dat`, and `bigvec-8.dat` can be seen in Figure 5.

From visualizing the data, it can be seen that the cross-over point is between message sizes 128 KiB and 256 KiB. By plotting a straight line between these two points for both the tree as well as scatter+allgather implementation, the cross-over happens at the message size of about **171 KiB**.

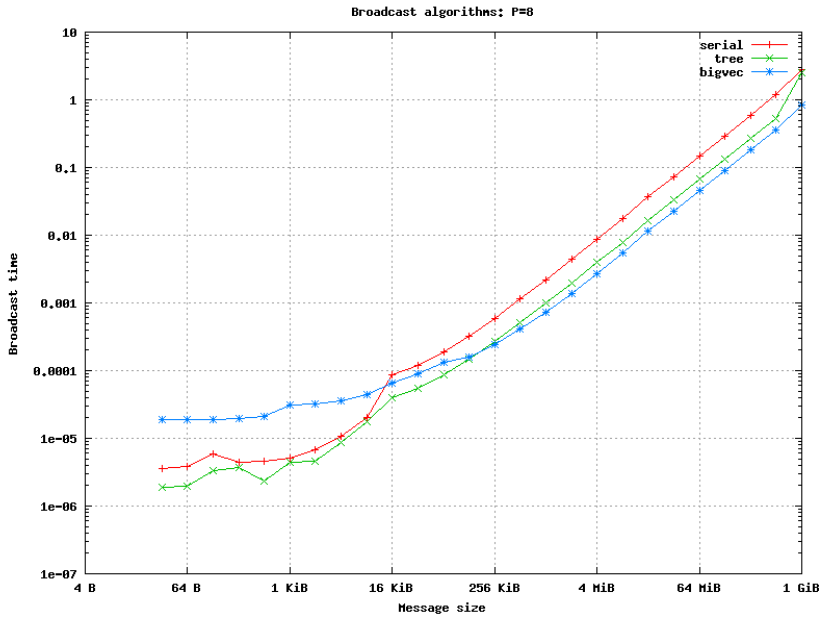
Broadcast algorithms: P=8
Broadcast time
Message size
serial
tree
bigvec
10
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
4 B
64 B
1 KiB
16 KiB
256 KiB
4 MiB
64 MiB
1 GiB

Figure 5