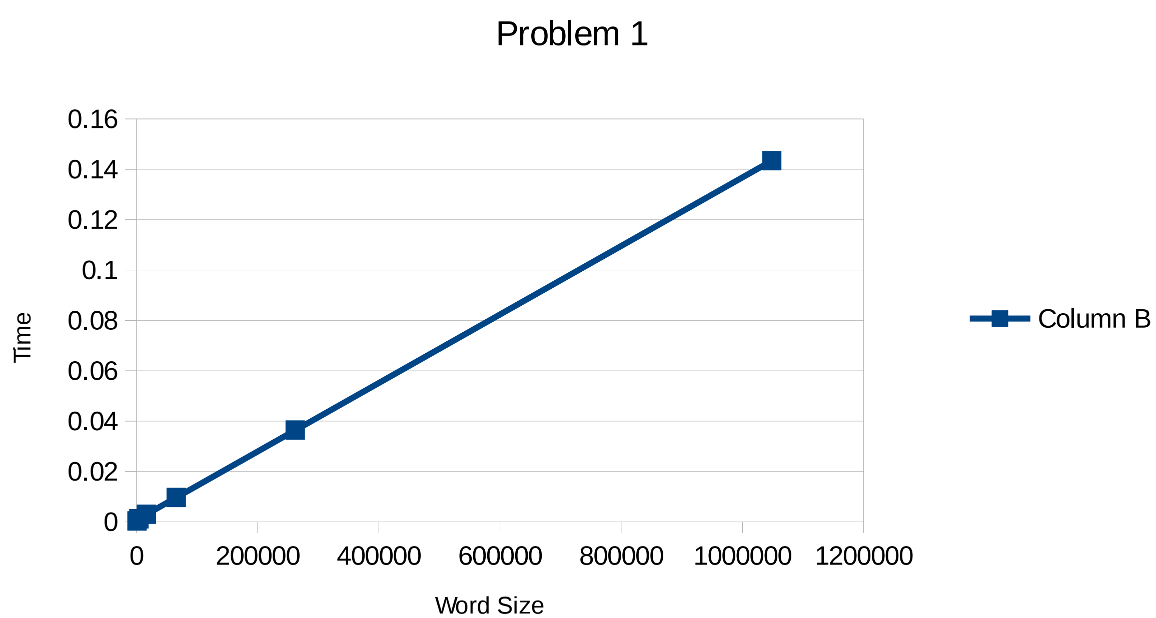
David Perry CS 525 HW#4 3/11/2014

1.)

Word Size	Time	Standard Deviation
1024	0.0003818035	1.801984e-05
4096	0.001153111	1.140639e-04
16384	0.002950191	3.985404e-04
65536	0.009628582	6.135993e-04
262144	0.03641074	5.342758e-04
1048576	0.1433638	5.079521e-04



t_s = 0.000603939516810275
t_w= 1.35851907980725e-07

2.)
a.)

Unidirectional Ring:

Word Size	**Time**	**Standard Deviation**
1024	1.641273e-03	3.719556e-04
4096	4.665184e-03	6.669531e-04
16384	1.189203e-02	1.982676e-03
65536	3.853662e-02	2.664039e-03
262144	1.455415e-01	2.538683e-03
1048576	5.732671e-01	2.267509e-03

Time Complexity:

$P(t_s + t_w*m)$

Estimated Values:

Word Size	**Time**
1024	0.0052021312
4096	0.008124524800000001
16384	0.019814099199999997
65536	0.0665723968
262144	0.2536055872
1048576	1.0017383488

Bidirectional Ring:

Word Size	**Time**	**Standard Deviation**
1024	8.431196e-04	1.863128e-04
4096	2.334642e-03	3.126104e-04
16384	6.391549e-03	8.243431e-04
65536	2.019982e-02	1.012009e-03
262144	7.306814e-02	1.249841e-03
1048576	2.868545e-01	1.198277e-03

Time Complexity:

$Ceil(P/2)(t_s + t_w*m)$

Estimated Values:

Word Size	**Time**
1024	0.0026010656

4096	0.004062262400000001
16384	0.009907049599999999
65536	0.0332861984
262144	0.1268027936
1048576	00.5008691744

The estimated values seem to be larger than the observed values by a factor of two. This occurs because the t_s and t_w values were calculated on the ping pong experiment. The ping pong experiment performed a round trip message passing. However, the ring only performs a one way trip. Therefore, the calculated values for t_s and t_w are twice as large as they should be.

Also, a difference occurs because the estimation does not do any calculation on the time it takes for the operating system to block processes and and bring them out of a blocked state. These CPU cycles that the operating system uses are costly and are not at all represented in the estimated calculation. The estimation also assumes that all sends and receives happen at the exact same time. However, the order and time in which these operations actually occurs varies.

b.)
Scan:

Word Size	**Time**	**Standard Deviation**
1024	1.582694e-03	3.100412e-04
4096	4.105353e-03	6.679731e-04
16384	1.073942e-02	1.769316e-03
65536	3.808024e-02	8.447760e-03
262144	1.451507e-01	1.451507e-01
1048576	5.978804e-01	4.533968e-02

The MPI_Scan function is likely implemented using a unidirectional ring. The times observed when running the scan function are extremely close to the times observed in the unidirectional ring.

3.)

3(1) Ring

Unidirectional:

$P(t_s + t_w*m + l * t_h)$

Bidirectional:

$ceil(P/2)(t_s + t_w*m + l * t_h)$

3(2) 3-D Torus

The first step in the one-to-all broadcast will be to transmit the broadcasted message to $\sqrt[3]{P}$ processors in the x direction.

$$(t_s + m*t_w + l * t_h) * ceil((\sqrt[3]{P})/2)$$

Notice that the expense of the message transmission, $(t_s + m*t_w + l * t_h)$, only needs to be performed $ceil(\sqrt[3]{(P/2)})$ number of times. This is accomplished by broadcasting the message in a bi directional ring like fashion.

The next step is to broadcast the same message to $\sqrt[3]{P}$ processors in the y direction. This step requires:

$$(t_s + m*t_w + l * t_h) * ceil((\sqrt[3]{P})/2)$$

Again, the expense of the message transmission, $(t_s + m*t_w + l * t_h)$, only needs to be performed $ceil(\sqrt[3]{(P/2)})$ number of times. This is accomplished by broadcasting the message in a bi directional ring like fashion.

The last step is to broadcast the same message to $\sqrt[3]{P}$ processors in the z direction. This step requires:

$$(t_s + m*t_w + l * t_h) * ceil((\sqrt[3]{P})/2)$$

Again, that the expense of the message transmission, $(t_s + m*t_w + l * t_h)$, only needs to be performed $ceil(\sqrt[3]{(P/2)})$ number of times. This is accomplished by broadcasting the message in a bi directional ring like fashion.

Therefore, the total runtime of a the one-to-all broadcast is:

$$3 * (t_s + m*t_w + l * t_h) * ceil((\sqrt[3]{P})/2)$$

3(3)

My solution to this problem assumes that communication between the switching nodes only takes t_h time to complete, rather than t_s + m*t_w. Regardless, the computation is the same. Only the number of t_s + m*t_w would be altered to be the same as the number of (l * t_h).

The first step of the broadcast will require the communicating node to send a message to a processor on the other side of the root of the tree. This pass will take:

t_s + m*t_w + 2$\log_2$(p)(l * t_h)

The next step will require the communicating node and the node that last received a message to send a message to a processor on the other side of both nodes one level below the root. This can be accomplished in:

t_s + m*t_w + 2$\log_2$(p/2)(l * t_h)

Therefore, the k step of the broadcast will require:

t_s + m*t_w + 2$\log_2$(p/2^(k-1))(l * t_h)

A total sum of all these operations is:

$$\sum_{k=1}^{\log(p)} t_s + m * t_w + 2 * \log(p/2^{(k-1)}) * (l + t_h)$$

This can be simplified to:

$$\log(p)(t_s + m * t_w) + 2(l + t_h) \sum_{k=1}^{\log(p)} \log(p/2^{(k-1)})$$

And more so:

$$\log(p)(t_s + m * t_w) + 2(l + t_h) \sum_{k=1}^{\log(p)} \log(p) - \log(2^{(k-1)})$$

$$\log(p)(t_s + m * t_w) + 2(l + t_h)(\log(p)^2 - \sum_{k=1}^{\log(p)} k - 1)$$

$$\log(p)(t_s + m * t_w) + 2(l + t_h)(\log(p)^2 - \frac{\log(p)(\log(p) + 1)}{2} - \log(p))$$

4.)

Both solutions assume that transmission between any two processors in a network only takes (t_s + t_w * m) time. This means that the number of hops, and time per hop is ignored.

Ring:

The scattering of the messages in the ring behaves similar to to the behavior of a binary tree. The initial processor passes a certain number of m/p size messages to the middle processor. The next step they pass a certain number of m/p size messages to the processor a fourth of the way through the ring respectively. This is shown mathematically below:

Step 1: t_s + (m/p * t_w) * p/2
Step 2: t_s + (m/p * t_w) * p/4
.
.
.
step k: t_s + (m/p * t_w) * p/(2^k)

Therefore, this computation can be represented as:

$$\sum_{k=1}^{\log(p)} t_s + (m/p * t_w) * (p/2^k)$$

This can be simplified to:

$$\log(p) t_s + (m/p * t_w)(p) \sum_{k=1}^{\log(p)} (1/2^k)$$

$$\log(p) t_s + (m/p * t_w)(p)(1 - 1/p)$$

$$\log(p) t_s + (m/p * t_w)(p - 1)$$

Now the computation for the gather must be done. Since every ring just sends the last message it received to the node in front of it the time to perform the gather is:

$$(p-1)(t_s + (m/p) * t_w)$$

Therefore, the total time for the communication is:

$$(p-1)(t_s + (m/p) * t_w) + \log(p) t_s + (m/p * t_w)(p-1)$$

Now considering the old algorithm does basically the same thing as the scatter described above (only with message size m) the time for its communication would be:

$$\log(p)(t_s+(m*t_w))$$

Therefore, it to figure out the size m must be in order for the new algorithm to be faster solve the inequality:

$$\log(p)(t_s+(m*t_w)) \;>\; (p-1)(t_s+(m/p)*t_w)+\log(p)t_s+(m/p*t_w)(p-1)$$

2-D torus

The first part of the broadcast consists of scattering information amongst the processors in the first row of size root(p). This is performed in a fashion similar to a binary tree. Therefore the computation goes as follows:

$$\sum_{k=1}^{\log(\sqrt{p})} t_s+(m/p*t_w)(p/2^{(k)})$$

Simplifies to:

$$\log(\sqrt{p})*t_s+(t_w*m/p)(p-\sqrt{p})$$

Now each processor in the row will distribute messages to its column. This communication is represented as:

$$\sum_{k=1}^{\log(\sqrt{p})} t_s+(m/p*t_w)(\sqrt{p}/2^{(k)})$$

Which simplifies to:

$$\log(\sqrt{p})*t_s+(t_w*m/p)(\sqrt{p}-1)$$

Now the all gather must be completed. This communication is performed by every processor simply passing its information on to the other processors in the network. This communication takes;

$$(p-1)(m/p*t_w+t_s)$$

Therefore, the total communication takes:

$$\log(\sqrt{p})*t_s+(t_w*m/p)(p-\sqrt{p}) \;+\; \log(\sqrt{p})*t_s+(t_w*m/p)(\sqrt{p}-1) \;+\; (p-1)(m/p*t_w+t_s)$$

Assuming the the old algorithm simply passed messages of size m on to each processor the communication would take:

$(p-1)(t_s + m * t_w)$

The calculation to find the size of m when the new algorithm is faster would be:

$(p-1)(t_s + m * t_w) >$
$\log(\sqrt{p}) * t_s + (t_w * m/p)(p - \sqrt{p})$ + $\log(\sqrt{p}) * t_s + (t_w * m/p)(\sqrt{p} - 1)$ +
$(p-1)(m/p * t_w + t_s)$