

## CSE 512/CS 554 Homework Assignment 2 — Solutions

1. Construction of the Empire State Building required about two person-millennia of effort, yet the project was completed in only about one year. How was this possible? What must the average workforce have been? The peak workforce was about 3500. What “efficiency” does this imply for the overall project?

*Answer:* The building could be completed in one year by having many workers working simultaneously in parallel. The average workforce was about 2000 workers. If the peak workforce was 3500, then the overall efficiency was  $2000/3500 \approx 0.57$ . The full 3500 workers could not be gainfully employed all of the time due to precedence constraints (e.g., the 100th floor cannot be started until the 99th is in place).

2. Two people can make up a bed in less than half the time it takes for one person to make up the bed. Explain this apparent example of superlinear speedup.

*Answer:* With two persons, one on either side of the bed, no access to “remote data” is required. With only one person, time is wasted going from one side of the bed to the other. Thus, the speedup anomaly is due to better data locality in the “two-processor” case.

3. Suppose you have misplaced your glasses in an eight room house, and it takes one minute per room for one person to look for them. Is it possible to achieve superlinear speedup if others help by searching multiple rooms simultaneously? If so, give an example; if not, explain why. Assuming that all rooms are equally likely to contain the glasses, is it possible to achieve superlinear speedup in the *expected* time to find the glasses? Explain your answer.

*Answer:* Yes, superlinear speedup is achievable in some cases with multiple searchers. For example, suppose the rooms are numbered 1 through 8, and the glasses are actually in room 8. If a single person searches the rooms in increasing order, 8 minutes will be required to find the glasses. If two persons search, agreeing to start at either end and meet in the middle, then one of the searchers will find the glasses in room 8 in only 1 minute, for a speedup of 8. Note, however, that this result is dependent on the particular search pattern and location of the item sought. Other arrangements could have yielded sublinear speedup, or even a slowdown, with the parallel search requiring longer than the serial search. In general, if all rooms are equally likely to contain the glasses, then the expected time for one person to find them is 4.5 minutes. The expected time for two searchers is 2.5 minutes (a speedup of 1.8), and so on down to 1 minute for eight searchers (a speedup of 4.5). Thus, the speedup never exceeds  $p$ , and superlinear speedup in the expected time is impossible.

4. (a) For the  $n \times n \times z$  3-D finite difference grid example given in Chapter 4 on *Parallel Performance*, what is the tradeoff point in  $n$  for which a 1-D parti-

tioning performs better or worse than a 2-D partitioning. That is, derive an expression for the value of  $n$  as a function of  $t_c$ ,  $t_s$ ,  $t_w$ ,  $z$ , and  $p$  such that a 1-D partitioning performs better for smaller values of  $n$  and a 2-D partitioning performs better for larger values of  $n$ .

*Answer:* From Lecture 3,

$$T_p^{1D} = t_c \frac{n^2 z}{p} + 2t_s + 4t_w n z,$$

and

$$T_p^{2D} = t_c \frac{n^2 z}{p} + 4t_s + 8t_w n z / \sqrt{p}.$$

Tradeoff point occurs when these two quantities are equal. Setting them equal and solving for  $n$  yields

$$n = \frac{t_s}{2t_w z (1 - 2/\sqrt{p})}.$$

(b) What is the asymptotic value for the tradeoff point as  $p$  becomes very large?

*Answer:* Tradeoff point approaches  $t_s/(2t_w z)$  as  $p \rightarrow \infty$ .

(c) For the values  $t_c = 20$ ,  $t_s = 1000$ ,  $t_w = 5$ , and  $z = 10$ , what are the largest and smallest values for the tradeoff point in  $n$  for  $9 \leq p < \infty$ ?

*Answer:* Plugging in these values for the parameters, we have

$$n = \frac{1000}{2 \cdot 5 \cdot 10 (1 - 2/\sqrt{p})} = \frac{10}{1 - 2/\sqrt{p}}.$$

For  $p = 9$ , the tradeoff point is  $n = 30$ . As  $p \rightarrow \infty$ , the tradeoff point approaches  $n = 10$ .

5. (a) Develop a performance model (parameterized by  $t_s$ , etc.) for the execution time of the basic single-node broadcast algorithm given in the section of Chapter 1 on *Collective Communication*, for a message of length  $L$  on a hypercube with  $p = 2^k$  nodes.

*Answer:* Each of the  $k = \log p$  steps takes time  $t_s + t_w L$ , so

$$T_p^{(a)} = k (t_s + t_w L).$$

(b) Develop an analogous performance model for a pipelined single-node broadcast algorithm in which the message of length  $L$  is broken into  $q$  pieces, which are pipelined along the same spanning tree. Determine an expression for the optimal number of pieces  $q$ .

*Answer:* Assuming that a given processor can communicate on all its links simultaneously, then we have a pipeline with  $k$  stages, each of which takes time  $t_s + t_w L/q$ . Thus, the time to process  $q$  items is

$$T_p = (k + q - 1)(t_s + t_w L/q).$$

Differentiating this expression with respect to  $q$  and setting the derivative equal to zero, we obtain the equation

$$\frac{d}{dq} T_p = t_s - t_w (k - 1) L/q^2 = 0,$$

which we can solve for  $q$  to obtain the optimal number of pieces

$$q = \sqrt{\frac{t_w (k - 1) L}{t_s}}.$$

Substituting this value for  $q$  into the previous formula, we obtain the optimal time

$$T_p^{(b)} = 2\sqrt{t_s t_w (k - 1) L} + t_s (k - 1) + t_w L.$$

If a given processor can communicate on only one link at a time, then the total time required is multiplied by  $k$ , but the optimal value for  $q$  does not change.

(c) Develop an analogous performance model for single-node broadcast in a hypercube in which the message of length  $L$  is broken into  $k$  pieces, which are broadcast using  $k$  edge-disjoint spanning trees.

*Answer:* Assuming that a given processor can communicate on all its links simultaneously,

$$T_p^{(c)} = k(t_s + t_w L/k) = t_s k + t_w L.$$

Again, if a given processor can communicate on only one link at a time, then the total time required is multiplied by  $k$ .

(d) Among the three preceding algorithms, determine the tradeoff points in  $L$ , as a function of the remaining parameters, for which each algorithm is best. For the algorithm of (b), assume that the optimal value for  $q$  is used.

*Answer:* We assume that a given processor can communicate on all its links simultaneously. Since algorithm (a) is a special case of algorithm (b) with  $q = 1$ , (a) cannot be faster than (b) if the optimal  $q$  is used in the latter. Algorithm (a) is also never faster than algorithm (c). Thus, the only possible tradeoff is between algorithms (b) and (c). We can determine this tradeoff by equating their running times

$$2\sqrt{t_s t_w (k - 1) L} + t_s (k - 1) + t_w L = t_s k + t_w L$$

and solving for  $L$  to obtain

$$L = \frac{t_s}{4 t_w (k - 1)}.$$

When the message length is less than this value, algorithm (b) with optimal  $q$  is faster, and when the message length is greater than this value, algorithm (c) is faster.

(e) For each of the three algorithms, plot the time to broadcast a message of length  $L$ ,  $1 \text{ Kbyte} \leq L \leq 1 \text{ Mbyte}$ , in a hypercube having  $p = 1024$  processors, a message latency of 5 msec, and a message transfer rate of 100 Mbyte/sec. Use a log-log scale.

*Answer:* See Figure 2. For the particular parameter values given, the tradeoff point between algorithms (b) and (c) is

$$L = \frac{t_s}{4 t_w (k - 1)} = \frac{5 \times 10^{-3}}{4 \times 10^{-8} \times (10 - 1)} \approx 13,889.$$

(f) Your answers to the preceding questions depend on your assumptions about the degree of concurrency in using communication links. How much difference does it make if a given processor can communicate on only one link at a time or on all links simultaneously?

*Answer:* If a given processor can communicate on only one link at a time, then the time for algorithm (a) is unaffected because each processor need not communicate on more than one link at a time anyway. However, algorithms (b) and (c) are seriously degraded, with the total time multiplied by  $k$ , and they both become inferior to algorithm (a).

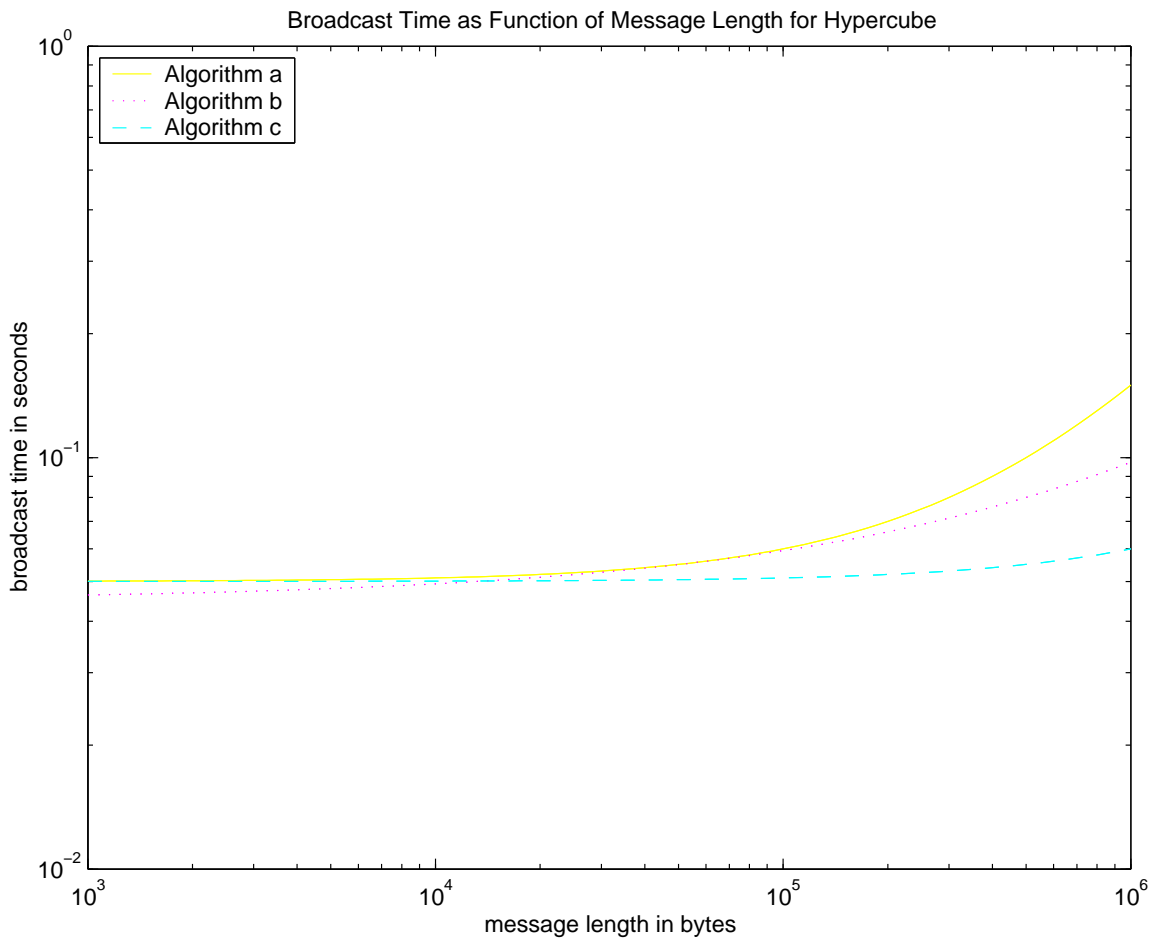


Figure 1: Broadcast time as function of message length on hypercube using three different algorithms.