REFERENCES

1. A. Beguelin and D.J. Vasicek, "Communication Between Nodes in a Hypercube", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
2. D. Bergmark, et al, "On the Performance of the FPS T-Series Hypercube", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
3. Y. Birk, et al, "A Simple Mechanism for Efficient Barrier Synchronization in MIMD Machines", *Proceedings of the 1990 International Conference on Parallel Processing*, pp. 195-198.
4. L. Bomans and D. Roose, "Communications Benchmarks for the iSPC", *Hypercube and Distributed Computers*, F. Andre and B.V. Verjus (Editors), Elsevier Science Publishers B.V. (North-Holland), 1989.
5. T.H. Dunigan, "Hypercube Performance", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
6. S.S. Fried, "Personal Supercomputing with the Intel i860", *BYTE,* Jan. 1991, pp. 347-358.
7. D.C. Grunwald and D.A Reed, "Benchmarking Hypercube Hardware and Software", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
8. M.L. Mathews, "Hypercube Software Performance Metrics", *Hypercube Multiprocessors* 1987, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
9. C. McCreary and H. Gill, "Efficient Exploitation of Concurrency Using Graph Decomposition", *Proceedings of the 1990 International Conference on Parallel Processing,* Vol. II, pp. 199- 203.
10. S. Nugent, "The iPSC/2 Direct-Connect Communications Technology", *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications*, 1988, pp. 51-60.
11. P. Pierce, "The NX/2 Operating System", *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications*, 1988, pp. 384-390.
12. J. Quinlan and D. Siewiorek, "Baseline measurements on the Mark II Hypercube", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
13. B. Stramm and F. Berman, "How Good is Good?", Tech Report CS90-169, UCSD, 1990.
14. H. Sullivan, and T.R. Bashkow, "A Large Scale, Homogeneous, Fully Distributed Parallel Machine, I", *Proceedings of the Fourth Symposium on Computer Architecture*, 1977, pp. 105-117.
15. D.W. Walker, et al, "A Comparison of the Performance of the Cal Tech Mark II Hypercube and the Elxsi 6400", *Hypercube Multiprocessors 1987*, Edited by M.T. Heath, SIAM, Philadelphia, Pa, 1987.
16. A. Zaafrani, H. Deitz, and M.T. O'Keefe, "Static Scheduling for Barrier MIMD Architectures", *Proceedings of the 1990 International Conference on Parallel Processing*, pp. 187-194.

Communication type	Message Size	Prediction Equation	R^2
SS/SD	Small	T = 174 (mean), 19.3 (standard deviation)	NA
SS/SD	Large	T= 306 + 0.353*L	0.985
SS/MD	Small	T = 341 + 213*D	0.8562
SS/MD	Large	T = 1620 + 0.364*L*D - 0.026*L	0.9997
MS/SD	Small	T = 126 + 76.5*S	0.9665
MS/SD	Large	T = 381 + 95.1*S + 0.366*S*L	0.9998
MS/MD	Small	T = 240 + 101*S + 158*D	0.9859
MS/MD	Large	T = 1217+ 136*S+ 0.35*L*D + 0.081*L*S*$\log_2$S	0.9963

Table 11: iPSC/860: Summary Formulas

The communication performance of the iPSC/860 is only slightly faster than that of the iPSC/2. In the SS/SD cases it appears to be about twice as fast, but in the SS/MD case the values appear to be approximately the same. However, the iPSC/860's node processors, the i860 chips, at 33MHz can perform arithmetic operations at a rate of about 20 (in scalar mode) to 100 (in dual-instruction mode) times that of the iPSC/2's 387 processors at 20 MHz [6]. This greater imbalance makes the issue of grain size even more important.

The work accomplished in this paper extends the measurements of communication from previous work that involves a single source and a single destination to messages from a single source to multiple destinations and multiple sources to multiple destinations. Several additional communication experiments are planned. For the multiple sender case, random nodes must be designated as the senders rather than smallest order nodes. The worst case for contention is where senders are adjacent in the highest dimension.

Communication measurement is the first phase of developing a complete system performance model. Experiments should be designed to measure costs of context switching, synchronization routines, global operations and node to host interactions. Also, this work needs to be duplicated on other multiprocessing systems.

Type	Prediction Equation	R^2
Small	$T = 240 + 101*S + 158*D$	0.9859
Large	$T = 1217 + 0.35*L*D + 0.081*L*S*\log_2 S + 136*S$	0.9963

Table 9: iPSC/860 Communication Costs: Multiple Sources to Multiple Destinations

IV. Summary and Future Work

The goal of this paper was to create formulas that would be helpful to iPSC users in writing efficient parallel programs and to developers of support tools for parallel programmers or anyone interested in predicting performance. This paper has presented formulas to model communication costs in four message passing scenarios: single source/single destination, multiple source/single destination, single source/multiple destination and multiple source/multiple destination. The operating system handles messages less than 100 bytes in length differently from larger messages, and as a result separate predictive equations are required for each. The formulas were developed for both the iPSC/2 and iPSC/860. The results explained in Tables 1-9 are summarized in Tables 10 and 11. The communication types abbreviations indicate the number of sources and destinations. For example SS/MD stands for Single source/Multiple Destinations.

T = time in microseconds
L = message length in bytes
S = number of senders
D = dimension of receiving subcube
small messages are less then 101 bytes

Communication type	Message Size	Prediction Equation	R^2
SS/SD	Small	$T = 394$ (mean), 13.8 (standard deviation)	NA
SS/SD	Large	$T= 739 + 0.362*L$	0.988
SS/MD	Small	$T = 296 + 267 * D$	0.8266
SS/MD	Large	$T = 1610 + 0.378 *L*D - 0.052*L$	0.9992
MS/SD	Small	$T =342 + 268*S$	0.9919
MS/SD	Large	$T = -58.3 + 502*S + 0.456*S*L$	0.9994
MS/MD	Small	$T = 285 + 347*S + 158*D$	0.9986
MS/MD	Large	$T = 721 + 854*S + 0.41*L*D + 0.067*L*S*\log_2 S$	0.9983

Table 10: iPSC/2: Summary Formulas

contention, and if all measurements were perfect, the conjectured equation for large messages in the case of multiple senders would also be the product of message length and small message formula. Because of contention factors, the expected actual results are greater than those given by the formula:

$t = (c_0 + c_1(D-X) + c_2S)L + c_4.$

The Independent Variables

Through the work on single source messages, it was clear that not all the independent variables were relevant. Hops and message type were eliminated from further consideration. The number of senders **S** was added. The other single variables considered included: **R,** the number of receivers and **D** the subcube dimension (also the log of R)**,** and **L**, the message length**.** In contrast to the single sender case, receivers, rather than its log, showed a straight plot.

The projected equations for large messages indicated a need for a variety of product terms in the SAS generated prediction equation. For large messages, the message length was a factor in many components. It was unclear at the start of the experiment how contention would factor in to the equation. To achieve the best model possible, the following terms were input for consideration by the statistical package: $S*\log_2 S$, $L*S*\log_2 S$, D*L, D*L*S, L*S, L*R, S*R, S*D, L*S*R, S/D, $\log_2 S*D$.

Communication Cost Formulas

Multiple regression on the data created the prediction equations shown in Tables 8 and 9. For small messages, the prediction equation is of the form $T = c_o + c_1S + c_2D$ where S is the number of senders and D is the dimension of the receiving subcube and is nearly the same as the one conjectured above. Only the term X, the log of S, did not offer a significant contribution. X is probably absorbed in the factor S since they both measure the same phenomena and X’s contribution is minuscule in comparison to that of S.

For large messages, the equation has the form $T = c_0 + c_1*S + c_2*L(D + \log(S)*S)$, where L is message length, D is the dimension of the receiving subcube, and S is the number of transmitters. This is not quite the same form as the theoretical model of $T = (c_0 + c_1(R-\log S) + c_2S)L + c_4$. The additional term c_1*S can be explained as the overhead incurred by each sender for initiating the broadcast. There is also an additional factor, log(S), which indicates that the time is not linearly dependent on the number of senders. In a theoretically perfect situation, there would be no contention for many senders to broadcast to the same set of receivers. However, in actuality, contention will rise as the number of senders increases. This is reflected in the term log(S)*S, showing that the influence of the number of senders is non-linear.

Type	Prediction Equation	R^2
Small	T = 285 + 347*S + 158*D	0.9986
Large	$T = 721 + 854*S + 0.41*L*D + 0.067*L*S*\log_2 S$	0.9983

Table 8: iPSC/2 Communication Costs: Multiple Sources to Multiple Destinations

Type	Prediction Equation	R^2
Large	T = 381 + 95.1*S + 0.366*S*L	0.9998

Table 6: iPSC/860 Communication Cost - Multiple Source/Single Destination

D. Multiple Source/ Multiple Destination Message Passing

Multiple source to multiple destination communication time data points were collected from both the iPSC/2 and the iPSC/860. Both small and large message data points were collected from both systems. All combinations of 2,4,8,16, and 32 senders and receivers were tested with fewer senders than receivers. Since iPSC message passing does not support multicasting from a cube to a smaller subcube when the number of senders is larger than the number of receivers, more than one send operation is required.

The message passing situation becomes much more complex when there are multiple transmitters of multicast messages. At some of the stages, more than one message must be transmitted across a single channel. Multiple transmissions then occur, each incurring its own cost. For example consider the case where there are 4 sending nodes and 8 destinations. In our experimentation the senders are placed in the subcube with lowest node numbers. Table 7 shows what happens at each stage. Numbers in parenthesis shows the id of the node originating message. After stage 2, each node contains 2 messages that it must transmit to its neighbor with opposite least significant bit.

Stage	Senders	Receivers
1	0, 1, 2, 3,	4(0), 5(1), 6(2), 7(3)
2	all	0(2), 1(3), 2(0),3(1), 4(2), 5(3), 6(0), 7(1)
3	all	0(1), 1(0), 2(3), 3(2), 4(1), 5(0), 6(3), 7(2) 0(3), 1(2), 2(1), 3(0), 4(3), 5(2), 6(1), 7(0)

Table 7: Sending messages from 4 nodes to 8 nodes

For small messages, the important factor is no longer the subcube dimension (or equivalently the number of stages), but the number of total transmissions. For our experimental model where senders are placed in the subcube with smallest ids, a single message is sent at each stage until all nodes have a message to pass. Then with each stage the number of messages to be transmitted from each node doubles. In passing messages from 8 nodes to a cube of 32 nodes, the transmissions at each stage are, respectively, 1, 1, 1, 2, 4 and the total number of transmissions is 9. Let S denote the number of transmitters, R the number of receivers, $X = \log_2(S)$, and $D = \log_2(R)$. Notice that the final sequence of transmissions in the stages is: $1+2+4+...+2^{X-1} = S - 1$. The remaining number of single transmissions per stage is D - X and so the total number of transmissions is (D-X)+S-1. In a perfect setting with no contention, the formula for the communication cost of small messages from S transmitters to 2^D receivers should be: $t = c_0 + c_1(D-X) + c_2S$. The overhead cost for queueing more than one transmissions is represented by c_2.

In the single source case, the conjectured equation for large messages was derived by multiplying the small message equation by the message length. If there were an unbounded number of message buffers, if all message sends were precisely synchronized, if there were no

Type	Prediction Equation	R^2
Small	T = 296 + 267 * D	0.8266
Large	T = 1609 + 0.378 *L*D - 0.052*L	0.9992

Table 3: iPSC/2 Communication Costs of Single Source to Multiple Destinations

Type	Prediction Equation	R^2
Small	T = 341 + 213*D	0.8562
Large	T = 1620 + 0.364*L*D - 0.026*L	0.9997

Table 4: iPSC/860 Communication Costs - Single Source/ Multiple Destinations

C. Multiple Source/Single Destination Message Passing

Since all the sends were generated independently from each node, and the receiver had to initiate one receive command for each incoming message, the conjecture was that the prediction equation would involve a factor of S, the number of senders. For small messages, it was expected that the equation would be of the form c_0+c_1S and for large messages $c_0+(c_1+c_2L)S$.

The Independent Variables and Communication Cost Formulas

Our work with the previous cases and our theoretical model led us to consider message length, number of sources, and the interaction term of message length and number of sources. These three variables for large messages gave us an equation with R^2 of 0.9998. Again message length is not a factor for message lengths under 100 bytes.

Multiple regression on the data supported the theoretical formulas proposed and created the prediction equations shown in Tables 5 and 6. L is the message length in bytes, S is the number of transmitters and T is the time given in microseconds.

Type	Prediction Equation	R^2
Small	T = 342 + 268*S	0.9919
Large	T = -58.3 + 502*S + 0.456*S*L	0.9994

Table 5: iPSC/2 Communication Cost of Multiple Source/Single Destination

Type	Prediction Equation	R^2
Small	T = 126 + 76.5*S	0.9665

Table 6: iPSC/860 Communication Cost - Multiple Source/Single Destination

Despite the predictions of only 2 independent variables, length and subcube dimension, a conservative approach to developing the formulas was taken and all reasonable independent variables were included in the statistical analysis. For each independent variable a graph or plot of that variable versus communication time was created. The plots were inspected to determine any correlation that may exist and to discover independent variables that may interact with each other.

Message Type - The data showed no difference in the communication time between the synchronous and asynchronous messages across different lengths and subcube dimensions. This result is expected since the measurements gathered represent the time it takes to interpret the message send command, create the data path, send the message, and notify the receiver that the message has been received. For a synchronous send the sender may encounter a delay before it can execute its next instruction (see appendix). This delay is not part of the measurement calculated by the receiver.

Hops - The number of hops did not significantly effect communication time. This again indicates that the hypercube acts as a completely connected set of nodes.

Receivers - The plot of number of receivers show a curved line indicating the function was not linear, but perhaps a logarithmic function. The corresponding log function gives the subcube dimension.

Subcube Dimension - The plots showed a direct linear relation between the dimension of the receiving subcube and the communication time. An example of these plots for small messages from the iPSC/2 is provided as Figure 1. The influence of subcube dimension also shows up on the plots of message length vs. communication time in the form of multiple data lines.

Message length - The plots of message length against communication time show some interesting properties. The plot of messages from the iPSC/2 (Figure 2) shows a distinct discontinuity between messages of size 100 and 200 bytes. This is expected (See appendix) because of the different treatment given very small messages in the iPSC/2. To isolate the very small messages, other plots sized for messages ranging in length from 50 to 100 bytes were made. The one for asynchronous messages is provided as Figure 3. The random pattern of the plot indicates that for very small messages, the size of the message has little effect on communication time. A plot of larger sized messages from the iPSC/860 is provided as Figure 4. This clearly shows that communication time is proportional to message length for larger messages. Also seen in the figure, evidenced by the appearance of several separate lines, is the influence of subcube dimension.

LD product - The relation between message length and subcube dimension suggests the need for an interaction term in the regression equation. The interaction term chosen is the product of message length and subcube dimension. This product, the LD product, is plotted for large asynchronous messages recorded from the iPSC/860 as Figure 5. The apparent straight line of this plot implies that this form is well suited for least squares regression.

Communication Cost Formulas

Multiple regression on the data supported the theoretical formulas proposed and created the prediction equations shown in Tables 3 and 4. Both equations are equivalent to the conjectured models given above. All times are given in microseconds. L is message length in bytes and D is the dimension of the receiving subcube. The R^2 value for small messages of 0.8266 is less than other R^2 values found in this study. This is because for small messages on the iPSC/2 system all actual times are under 1.42ms. With a clock resolution of only 1 ms and small average times, the prediction is not as accurate as for larger messages. In the equation for large messages on the iPSC/2 system, the negative coefficient of the length term shows that as message length is increased the effect of the additional delay for messages is dampened.

Communication Cost Formulas

Regression analysis on the data generally supported the expectations. For small messages, the only significant term remaining is the constant. On the iPSC/2, communication time values ranged from a minimum of 375 usec to a maximum of 430 usec. On the Ipsc/860 the range was 140 to 220 usec. For large messages, the theoretical model is supported fully; the only independent variable is the length. The R^2 value, which indicates the goodness of fit for a regression equation, was computed for large messages. When the equation perfectly fits the data, R^2 is 1.0.

The results presented in Tables 1 and 2 are comparable to those presented in [4] for single sender / single receiver message passing. T is time in microseconds and L is message length in bytes.

Type	Prediction Equation	R^2
Small	T = 394 (mean), 13.8 (standard deviation)	NA
Large	T= 739 + 0.362*L	0.988

Table 1: iPSC/2: Single Source/Single Destination

Type	Prediction Equation	R^2
Small	T = 174 (mean), 19.3 (standard deviation)	NA
Large	T= 306 + 0.353*L	0.985

Table 2: iPSC/860: Single Source/Single Destination

B. Single Source/Multiple Destination Message Passing

The e-cube algorithm (see appendix) sends the message over each channel in turn from highest numbered channel to lowest. For example if the receiving subcube is of dimension 3 and the message originates from node zero three stages are required:

Stage	Senders	Receivers
1	0	4
2	0,4	2,6
3	0,2,4,6	1,3,5,7

At each stage only one hop is active, so hops should not be a part of this equation. The predicted equation for small messages should be of the form $T = c_0 + c_1 D$ where c_0 and c_1 are constants and D is dimension of the subcube receiving the message. c_0 can be interpreted as the overhead cost for initializing the send, including interpreting the send command and notifying the DCM of the need to handle the message. c_1 includes the cost of creating the data paths and sending the message in each dimension.

For larger messages, this process is repeated for each "buffer" of data. The theoretical form would then be: $T = c_0 + c_3(c_1 + c_2 D)L$ where L is message length, c_3 is a factor of length required for overhead of each buffer and incorporating buffer size. c_0 is the offset due to overall overhead costs.

The Independent Variables

elapsed time for the multiple source broadcasting was computed as the difference between the last receive time and the earliest broadcast start time.

III. Analysis and Results

The results are given for four different traffic patterns: single source to single destination communication, multiple source to single destination, single source to multiple destination communication and multiple source to multiple destination communication. The analysis was conducted in two stages using the SAS statistical analysis package. The first stage involved evaluating the effects of single independent variables (e.g. message length) on the dependent variable, communication time. When indicated by the theoretical predictions, log or second order terms for independent variables were introduced and independent variables were combined as products or quotients to form interaction terms. This stage established the proper form for each relevant independent variable.

The second stage of the analysis used the information from the first stage to help derive prediction equations. This stage consisted of two steps. In the first step, SAS was given data on all of the original independent variables and the relevant interaction terms for a given data set. These were combined with a specified acceptance level (0.05) to derive a first order prediction equation. Variables whose effect on communication time was either negligible or accounted for by another variable were weeded out by this step. The information gained in this step identified the functional forms of the prediction equations. In the second step, multiple regression was conducted using SAS to generate the predictive equations.

For each of the following sections, the theoretical foundations are established; independent variable determination is described; and the communication cost formulas are presented.

A. Single Source/ Single Destination Communication

Based on the message buffering structure of the iPSC described in the appendix, two predictive equations are projected: one for small messages (≤ 100 bytes) and one for larger messages. For small messages, the ready availability of 100 byte buffers leads to the expectation that length is not a factor. This leaves only the possibility that communication time is dependent on the number of channels required to reach the destination, or equivalently, the distance, or number of hops, between the sender and receiver. However, other research [4] and our own discussions with Intel Corporation have lead us to expect that the influence of hops is negligible. Therefore, the expected equation for small messages should be of the form $t = c_0$, where c_0 is a constant. For large messages, hops is also not expected to be influential but length of the message is expected to be an important factor due to the need for the receiver to allocate an appropriate number of buffers before it can accept an incoming message. Therefore, in this case, the equation should be of the form $t = c_0 + c_1L$, where c_0 and c_1 are constants and L is the message length in bytes.

<u>The Independent Variables</u>

When developing the formulas from the statistical data, the only relevant variables are Message Length and Hops. Because there is only one message at a time being sent, there is no contention and therefore no other variables influence the results.

Message Length - A plot of this variable against communication time for small messages has a random pattern implying length has no influence. For larger messages, the plot shows that communication time is proportional to the message length.

Hops - For both small and large messages, the plots of hops against communication time display random patterns, implying that this variable has no significant influence. The regression analysis also supported this conclusion.

performed 100 times for each node and an average value is used as *adjust*. Node 0 sends each node its corresponding adjustment time. Now, anytime that Node *i* reads its clock, it determines the time with respect to the root node by adding the adjustment value to its time.

B. Communication Time Calculation

To develop the prediction equations given in the next section, many communication time data points were calculated for each of the four communication situations. Data points were calculated based on different combinations of message length, number of senders, number of receivers, subcube dimension and location of senders and receivers in their subcube. Accuracy of the calculations was enhanced by using the average of repeated tests as a single data point.

Single Source/Single Destination

Two methods of data collection were used in this case. In the first method, each node's clock time was adjusted with respect to node 0 in the allocated cube. Node 0 sent messages one at a time to each of the other nodes in the cube. Each node then determined the time the message arrived. The message contained the time when the message was initiated by node 0, and each receiving node computed the elapsed time of the send.

For small message sizes, the elapsed time is difficult to measure because the time of the send can be close to any amount of error incurred when adjusting the node clock times. Therefore, data was also collected by performing round-trip message exchanges. Node 0 sent messages to another node. The receiving node immediately echoed the message back to the sender. Node 0 computed the elapsed time by recording the starting and ending times of the round-trip exchange. One-half of the elapsed time is the time for sending messages one-way.

For both of these experiments, the sends were repeated 100 times to each node, and the averages were reported. Repetitions are necessary because the clocks are accurate to only 1 ms. Both methods gave nearly identical means, but the standard deviation was larger with the one-way method. The reliance on an additional variable, the clock adjustment, could cause such a variance. We report on the round-trip results for the small messages. The measurements for the iPSC/2 are essentially the same as those reported by Bomans and Roose[4].

Single Source/Multiple Destination

The broadcast time was measured by the following procedure. First the adjust values for the node clocks with respect to the root node (0) were computed. The root broadcasted a message to all nodes in its subcube. Each receiver recorded the time it received the message. Because the message sent contained the time when the root node initiated the broadcast, each node computed the amount of time it took to receive the message. The largest elapsed time is recorded as the amount of time it took to deliver the message to the entire subcube.

Multiple Source/Single Destination

The experiment synchronized the nodes using the system call GSYNC, and then the receiving node (0) immediately recorded the starting time and the other nodes issued their send. Node 0 waited to receive messages from nodes 1 through N. After all messages are received, the ending time was recorded. Node 0 computed the elapsed time of receiving the set of messages. No clock adjustment was necessary because all time values were read only on node 0.

Multiple Source/ Multiple Destination

All combinations of 2,4,8,16, and 32 senders and receivers were tested with fewer senders than receivers. Since iPSC message passing does not support multicasting from a cube to a smaller subcube, when the number of senders is larger than the number of receivers, more than one send operation is required. Our experiment is limited to the single multicast case.

After the adjust values for the node clocks with respect to the root node (0) were computed, each node in the cube recorded its own start time, then broadcasted a message to a subcube of other nodes. After a node received all of its designated messages, it recorded its ending time. The communication in distributed memory systems involve multicasting, a process in which a single sender multicasts the same message to multiple receivers. As an example, many applications require “barrier” synchronization, a logical point at which all or a subset of all of the processing nodes wait until they receive a common signal to resume processing [3] [16]. In a distributed memory MIMD system, the synchronization notice may be a message multicast to the necessary subset of processes. Some schemes [9] may pass additional data during synchronization. Multiple source/single destination message passing may occur in a situation where all active processors signal a master processor at a synchronization point. In this case, only one node is receiving messages, and all other nodes in the cube send a message to it.

Data for this study was collected on both the iPSC/2 and the iPSC/860 hypercube systems. Both iPSC series machines can have up to 128 nodes. Data was collected on 16-node and 32-node configurations. Each node of the iPSC/2 used in this study has a 386/387 16 Mhz processor, 4 Megabytes of memory and 16 Kilobytes cache memory. The nodes run version 3.3 of the NX operating system. Each node of the iPSC/860 used has a i860 40 Mhz processor, 8 Megabytes memory, 8 Kilobytes data cache and 4 Kilobytes instruction cache. This system also runs NX version 3.2.2.

The first phase of developing a performance model for iPSC hypercubes is to measure the costs of the range of message passing routines. There are five parameters of the send calls: message type, buffer id of message to be sent, length of message, identification of receiving node(s) and process id of receiving process. The only parameters to affect communication time are length of message and identification of receiving node. The receiving node’s id can affect communication time in two ways: distance from sending node and number of destinations. If the id of the receiving node (*node)* is negative, the message is sent to more than one node. If *node* is -1, the message is sent to all nodes; other negative numbers cause the message to be sent to surrounding subcubes of various sizes.

Developing a comprehensive cost model for communications costs involves running experiments with both asynchronous (isend) and synchronous (csend) system calls, using a variety of message lengths and sending the messages to subcubes of sizes ranging from one node to *n* nodes. In addition to exercising all the message passing options from a single node, a complete model must show the consequences of many nodes executing simultaneously. From all these experiments a good picture of the message passing costs can be created. Our results produced separate formulas for each of the different communication situations and for the iPSC/2 and the iPSC/860. Because of the different protocols for large and small messages, separate formulas for traffic consisting messages no greater than 100 bytes and for traffic consisting of large messages are given.

A. Clock Adjustment

Each node in the hypercube has its own internal clock, with a resolution of 1 millisecond. These node clocks are not perfectly synchronized and, therefore, will produce different values when sampled at the same time. To overcome this difference, an adjustment value, *adjust,* is determined for each clock. When *adjust* is added to the associated clock time, the sampled time will be identical to any other node’s adjusted time measured at the same instant. For this study, each clock in the cube is adjusted with respect to the root node (0).

The algorithm to calculate the adjustment for node *i* begins by reading the clock at node 0 (T_{start}). The value of T_{start} is sent to node *i* and on receipt of this time, node *i* reads its clock (T_{rec}) and immediately sends its time back to node 0. On receipt of T_{rec}, node 0 again reads its clock (T_{end}) and estimates the one-way communication cost as ONEWAY = $(T_{end} - T_{start}) / 2$. If the clocks were perfectly synchronized, T_{rec} would be T_{start} + ONEWAY. The adjustment is therefore given as T_{start} + ONEWAY - T_{rec}. To ensure a more accurate adjustment value, the sequence is

Hypercube Communication Performance

I. Introduction

One key issue in the creation of an efficient parallel program is the establishment of the proper grain size. The optimal granularity is dependent on the architecture of the target system. Systems with powerful processors and distributed systems having expensive (in terms of delay time) interprocessor communication require a coarser grain than systems with possibly less powerful processors but fast interprocessor communications. In creating tools for program restructuring, or in designing parallel programs, accurate modeling of the performance of the target multiprocessor system is critical. Construction of a cost model entails careful analysis of components in the execution environment. These include characteristics of the architecture, the operating system, and the language environment. Memory hierarchy, cache policy, port contention for shared memory, bus or switch contention, and routing policy are examples of architectural features that affect parallel performance. Scheduling policy, context switching, communication mechanisms, and low-level and barrier synchronization mechanisms are examples of operating system and language features that are significant performance factors. It is necessary to construct measurement-based cost models and abstract cost models for various system characteristics.

The two most significant components of performance in parallel processing systems are processing time and communication time. The goal of this work is to construct a performance model to accurately predict the costs of communications for Intel's iPSC/2 and iPSC/860 hypercube systems and to present a procedure for determining similar metrics for other systems. Measurements and model formulas are reported for communication costs for the spectrum of message passing routines available on iPSC systems.

This study combines previous work analyzing single message point-to-point communications with 3 additional communications situations: (i) single sender to multiple destinations (1:N); (ii) multiple senders to a single destination (N:1); (iii) multiple senders to multiple destinations (M:N) Previous work has looked only at the 1:1 case because communication time was calculated on a single clock as one-half or the round trip time[4]. To extend timings to the other three cases, we developed a practical method for synchronizing timing measurements among independently-clocked processing nodes.

II. The Experiment

Distributed memory MIMD systems (such as the Intel hypercubes) have the highest potential for adverse impact from communications delay due to their relatively slow communication speed. We have attempted to measure costs and develop a model for multicast communication costs of the two Intel systems. Many theoretical performance models for multiprocessors have been proposed[13], but when efficient programs must be developed, detailed performance measurements that accurately represent the underlying architecture are required. Several studies analyze the processor **execution** performance of Intel hypercube systems [7] [8] [5] [12] [2] [15]. Less work has been directed toward analyzing the impact of that interprocessor communication on overall system performance. Those studies dealing with communication costs [4] [1] have been limited to point to point single messages between two processing nodes because each processing node has its own internal clock and it is difficult to measure accurately the elapsed time from one node to another. As a result, these studies have been based on sending messages which loop back to the sender, restricting the timing to a single clock measuring round trip delay. Although these studies have provided valuable models for the single sender, single receiver case, much

Hypercube Communication Performance

Abstract

Performance information is essential to the design of efficient parallel programs. Whether the programmer has total control over determining the parallel execution of processes or whether some automatic means are used to parallelize serial or partially parallel code, the cost of the overhead due to parallelization must be known. This is especially important in systems where the costs of such mechanisms are high. In distributed memory systems, message passing is the primary means of communication between parallel processes, and message passing tends to be quite expensive. Users of such systems wrestle with a variety of questions related to performance issues. How is the data to be distributed among the processors? What grain size (amount of code executed serially on a processor) is required for efficient processing? How should the processes be assigned to the system nodes? Approximate answers to these question can be given only if the performance characteristics of the system are known.

This paper presents a step towards the development of a performance model for two distributed memory multiprocessor systems, the Intel iPSC/2 and iPSC/860 hypercubes. From actual timing measurements, we developed formulas to estimate the communication costs on the four possibilities for message passing among multiple processors: a single sender to a single receiver, single sender to multiple receivers, multiple senders to a single receiver, and multiple senders to multiple receivers. A practical method for synchronizing timing measurements among independently clocked processing nodes was developed to do this analysis.