Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

A SIMULATION STUDY OF A PARALLEL PROCESSOR WITH UNBALANCED LOADS

Wade H. Shaw, Jr.
Timothy S. Moore
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

ABSTRACT

It has been well established that the performance of a parallel processor computer system is affected by many design alternatives and the underlying degree of parallelism in the workload. We look at the impact of workloads which load processors in the network unevenly to observe the performance degradation. We constrain the parallel processor architecture to the family of hypercube networks. Each node is loaded with some portion of the workload composed of CPU bursts and I/O, and allowed to run at its on pace until it completes. Since message transmission preempts node processing, communication between nodes complicates the concurrent operation of the network. We vary the degree of load balance, the processing node locality and the ratio of CPU burst time to message transmission time across a generic 16 node hypercube and use total processing time speedup as the performance criteria. Regression models indicate strong nonlinear correlation between the degree of load imbalance and job speedup and a linear effect due to CPU/IO intensity. The locality of workload is shown to be a minor but significant effect. The impact of the load balance, CPU/IO intensity and locality effects on algorithm decomposition is discussed.

1. INTRODUCTION

The advent of multiprocessor computer systems has resulted in evidence of decreased processing time for jobs that can be decomposed into parallel processes. This phenomenon has been tested to reveal significant but not perfect increases in process speedup as additional processing nodes are added. The realities of inter-node communication do not allow an N node parallel processor to achieve the theoretical linear speedup. That is, an N node machine actually produces something less than an N times speedup. We use a simple definition of speedup to be the ratio of the single processor execution time to the time measured with additional processors.

One area of specific concern is the effect of processor load balance on the performance of a job. This factor is important because processor load balance will significantly affect the choice of decomposition algorithms. The purpose of this research is to determine the effect of processor load balance on the execution time of a process executed in parallel on a loosely-coupled multiprocessor computer system.

Multiprocessor computing systems are divided into two general categories, tightly-coupled systems and loosely-coupled systems. Tightly-coupled systems usually have a large, shared memory through which the individual processors communicate. In loosely-coupled systems, each processor has its own local memory. An individual processor and memory module form a processing element, and the processing elements are connected through an interconnection network. The processors communicate with each other via messages sent through the interconnection network. An emerging architecture showing promise is the hypercube machine discussed at length by Wiley (1987). A 16 node hypercube is depicted in Figure 1.

We specify a means of characterizing processor load balance and locality, and use a simulation experiment to investigate the effect of the load imbalance on the speedup of a job executed in parallel on a loosely-coupled, hypercube system. Since the relative impact of communication time between nodes is known to dramatically affect performance, we investigate the imbalance of processor loads at two levels of CPU/IO intensity to insure that the effects of imbalance are isolated. We do not consider how to choose a decomposition algorithm; only the effects of choosing a poor decomposition algorithm.

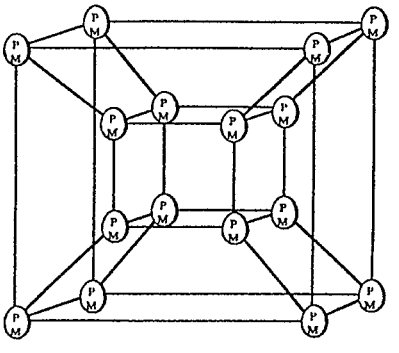

Figure 1: A 16 Node Hypercube

1.1 Research to Date

There are many factors against which multiprocessor performance can be evaluated. Recent performance evaluations have studied the effects of workload mix, program behavior, processor interconnection networks, redundant interconnection networks, memory management, and decomposition strategies. However, all of these studies were performed with balanced processor loads.

Nestle and Inselberg (1985) have shown that a tightly-coupled multiprocessor system can be modularly expanded while providing strictly linear increases in performance. These increases, they claim, are independent of the workload mix. They contrast their results to loosely-coupled multiprocessor systems which, they claim, cannot sustain linear increases in performance when running non-homogeneous workloads due to the inter-processor communication overhead. Although the claim is intuitive, no study was cited to support their claim.

Du (1985) performed a study where system structure and program behavior were the two main factors. This study, Du claims, is set apart from others by the fact that previous studies have usually ignored program behavior. His study evaluated the performance of a multiprocessor in which a crossbar was employed to interconnect processors to m commonly shared memory modules. A set of non-uniformly distributed probabilities, including a probability which represents a processor not generating any request, was used to model the program behavior. However, no distinction was made between processors. Several relations between the average processor utilization, average request completion time, and the effective memory bandwidth were obtained.

Bhuyan (1984) evaluated two loosely-coupled architectures, each having three types of interconnection networks: shared bus, crossbar, and a class of multistage interconnection networks called Omega networks. The probability that a message is accepted was used as a measure of the performance. The study showed that for a high rate of internal requests, an Omega network performed close to a crossbar, but at a considerably reduced interconnection cost.

Padmanabahn and Lawrie (1985) conducted an evaluation which focused on the effect of redundant path interconnection networks on performance. Their evaluation showed that redundant path networks provide significant fault tolerance at a minimal cost. In addition, improvements in performance and very graceful degradation were shown to result from the availability of redundant paths.

Jalby and Meier (1986) conducted a study in which memory management was the primary factor. They claim that as the memory organizations of large multiprocessor computers become more complex, data management in the memories becomes a crucial factor for achieving high performance. An architecture which combines vector and parallel capabilities on a two-level shared memory structure was studied via analyzing and optimizing matrix multiplication algorithms. The optimized algorithms yielded high efficiency kernals which can be used for many numerical logarithms such as LU and Cholesky factorizations.

Vrsalovic, Gerhinger, Segal and Siework, (1984,1985) present a model for predicting multiprocessor performance on iterative algorithms based on the decomposition strategy used. Each iteration was assumed to require some amount of access to global data and some amount of local processing. The application cycles were allowed to be synchronous or asynchronous, and the processor may or may not have incurred waiting time, depending on the relationship between the access time and the processing time.

The amount of global data accessed and the processing time incurred by the parallel processes were dependent upon characteristics of the algorithm and its decomposition. The decomposition of several algorithms was studied and several decomposition groups were identified. The Poisson partial differential algorithm was used to determine how decomposition affected the performance of the algorithm. This study is more directly related to the topic than the others presented. However, the decompositions that were evaluated resulted in balanced loads on the individual processors and the system evaluated was a tightly-coupled system.

Wiley (1987) claims that an evenly distributed load is essential for efficient parallel computing. In addition, factors such as communication time between processors are also important. While these claims are intuitive, no references are cited to support the statements.

Reed and Grunwald (1987) performed an evaluation on the Intel iPSC which relates directly to this research effort. They determined the message processing times for nearest neighbor nodes on the iPSC Hypercube. They characterized the transmission times in accordance with the following model:

$$S = L + Nt \qquad (1)$$

where S is the transmission time, L is the communication startup time (latency), t is the transmission time per byte, and N is the number of bytes transferred. They performed a least-squares fit of the data to the linear model with the following results:

L = 1.7 milliseconds, and

t = 0.00283 milliseconds.

As the research cited indicates, there are many factors against which multiprocessor performance can be evaluated. One such factor is the effect of processor load

balance on performance. The effect of the load balance will be important in determining which algorithm to use when decomposing programs into parallel processes. It is accepted that perfect balancing results in more efficient program execution. However, the effects of unbalanced processor loads have not been thoroughly researched and characterized. Consequently, there is minimal literature pertaining directly to the subject.

We feel that it is intuitive to suspect that a parallel processor will exhibit reduced speedup as the degree of load imbalance is increased to the extent that the execution time resembles the performance of a smaller machine. The major issue is the nature and severity of the load imbalance and locality effect; and, whether or not the effect is consistent across different processing to communication ratios.

1.2 Research Objectives

The objective of this research is to establish whether or not load imbalance can be used to characterize speedup of a process executing on a hypercube machine. We also wish to examine the impact of spatial differences of a given degree of imbalance and the impact of burst to IO time ratio. Simply stated, the research hypothesis is given as:

> Ho: There is no variability in process speedup explained by the degree of load imbalance, the locality of the load imbalance, the processing to communication ratio or any interaction on a hypercube parallel processing machine.

We test this hypothesis using a carefully controlled experiment where data is obtained by simulation of a generic process which can be characterized by CPU bursts and subsequent IO.

1.3 Scope and Limitations

We direct our investigations to the performance of a 16 node hypercube machine with statistically controlled processor and IO loads. This allows carefully controlled experiments to be performed. However, this approach does not necessarily predict the performance of any particular algorithm. Rather, we intend to develop a fundamental relationship between processor load imbalance and job speedup. This relation provides insight that explains the general nature of workload partitions and locality.

We model the workload in terms of CPU bursts of exponential length followed by transmission of a random length message to a randomly selected receiver node. The workload imbalance is modeled by varying the number of CPU bursts on each node. The locality of the workload is varied by spatial location of the node loads once a level of imbalance is specified. We choose two processing to communications ratios by setting the CPU burst to be 10 and 2 times the average time to transmit a message between nearest neighbor nodes. Clearly, the use of a single destination node for a message transmission is a conservative approach. Nevertheless, we feel this design serves as a good starting point.

2. RESEARCH METHOD

In order to investigate the effects of load imbalance it is necessary to characterize load imbalance and locality metrics. Using these metrics, an experimental design was set up so that the metrics were varied over a sufficiently wide range to observe the impact on process speedup. Since the metrics are quantitative, we use regression techniques to determine the nature and significance of the main and interactive terms.

2.1 Imbalance Metrics

Two metrics are proposed. First, a raw measure of unbalance (B) computed as the coefficient of variation of individual processor loads. This metric is computed as:

$$B = \frac{\sigma_b}{\mu_b}, \qquad (2)$$

where σ_b is the standard deviation of the processor loads over the nodes in the hypercube and μ_b is the mean load. This metric is dimensionless and allows direct comparison of B for alternative processes. We compute each statistic based on the number of CPU bursts at each node. The mean is therefore a constant across all experimental units.

A secondary independent variable is locality. The concept of locality is used to characterize the node loadings with respect to node location. For example, assume that nodes 0 and 1 each have 45% of the load of a given job, and the remaining 10% is distributed evenly among the other nodes. This loading scheme will be characterized by a value for the degree of imbalance and a value for the degree of locality. Now assume the same case except that nodes 0 and 15 each have 45% of the load. In this case, the degree of imbalance will be the same, but the degree of locality will be different because nodes 0 and 15 are not directly connected as is the case for nodes 0 and 1. Locality is characterized by calculating L(i) for each node and calculating the coefficient of variation of the L(i). L(i) is given by:

$$L(i) = \sum_{j=1}^{n} l(ij)p(j), \quad \text{for all } i,\ i \neq j \qquad (3)$$

where l(ij) is the number of hops required to transfer a message from node i to node j, p(j) is the percentage of the load computed by node j and n is the number of nodes.

The coefficient of variation of locality

is again used to compute a dimensionless quantity and represents the degree to which a load is 'clumped' together. As in Equation 2, the locality metric (L) is given by:

$$L = \frac{\sigma_1}{\mu_1}, \qquad (4)$$

where the standard deviation and mean of L(i) over n nodes is used.

2.2 Experiment Design

A designed experiment was used to reduce experimental error. The general linear model is:

$$S = \mu + R + B + L + RB + RL + BL + RL + RBL + error \qquad (5)$$

where S represents the observed process speedup calculated by dividing the observed execution time into a uniprocessor time, μ is the experiment average, R is the ratio of average processor burst time divided by average message transmission time, B is the load balance metric (Equation 2), L is the locality metric (Equation 3) and RB, RL, BL, and RBL are the interactions of these terms.

Table 1 shows the experimental design where each case was simulated for R values of 10 and 2. Data was obtained by setting the total number of CPU bursts for a generic process to 256 where each burst was distributed as a negative exponential with a mean of 16.14 milliseconds for R=10 and 3.23 milliseconds for R=2. The IO time was set to a random variable determined by the length of a message distributed uniformly between 100 and 1024 bytes and the timing equation developed below. Each node processes its load independently of the other nodes but must suspend process execution if a message is routed through it.

Each experimental unit composed of a degree of imbalance, burst to message time ratio and locality was simulated so that batch means of 10 runs with 10 jobs each were used to obtain an execution time average. In all, 3200 jobs were simulated. It is noteworthy that an additional case exists not shown in Table 1 which represents the single processor case where only one node is loaded with all the CPU bursts. This case corresponds to a single processor machine with a known behavior of 256 * 16.14 = 4132 millisecond execution time for R=10 and 826.4 milliseconds for R=2. Case 1 represents the perfectly balanced case where B and L are 0.

2.3 The Simulation Model

A crucial aspect of this research effort was to model the time required transmit a message between nodes. In the case of nearest neighbor transmissions this problem has been researched as shown in Equation 1. However, Equation 1 was estimated based on nearest neighbor transmissions and does not account for intermediate processing time at nodes along the sender/receiver path. The hypercube architecture uses a fixed routing algorithm which sets the intermediate nodes given a sender and receiver node.

To estimate the message handling time a linear relation was used and is given as:

$$M = \mu + \beta_1 HX + \beta_2 I + error, \qquad (6)$$

where M is the time to send a message of X bytes in length over H hops via I intermediate nodes. It is noted that I = H-1 since the number of intermediate nodes is directly proportional to the hops required for the message to arrive at its destination.

In order to estimate Equation 6, a benchmark program was executed on an Intel iPSC Hypercube. A hypercube with 32 nodes was used to measure the time to transmit a

Table 1: Experiment Design Node Loading (CPU Bursts)

	Node Number															
Case	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
2	24	22	20	18	16	14	12	10	1	11	13	15	17	19	21	23
3	1	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
4	32	30	28	26	1	3	5	7	17	8	6	4	2	27	29	31
5	1	2	3	4	5	6	7	8	32	31	30	29	28	27	26	17
6	38	6	6	6	6	38	6	6	38	6	6	38	6	6	6	38
7	38	38	38	6	38	6	6	6	38	6	6	6	6	6	6	6
8	72	8	8	8	8	8	8	8	8	8	8	8	8	8	8	72
9	72	72	8	8	8	8	8	8	8	8	8	8	8	8	8	8
10	79	7	7	7	7	7	7	7	7	7	7	7	7	7	7	79
11	79	79	7	7	7	7	7	7	7	7	7	7	7	7	7	7
12	100	4	4	4	4	4	4	4	4	4	4	4	4	4	4	100
13	100	100	4	4	4	4	4	4	4	4	4	4	4	4	4	4
14	121	1	1	1	1	1	1	1	1	1	1	1	1	1	1	121
15	121	121	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	241	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

random length measure of between 5 and 1024 bytes from node 0 to every other node. For each unique receiver node, 20 data points were collected where each data point was the average of 100 transmissions from node 0 to the receiver and 100 transmissions from the intended receiver to node 0. In all, 124,000 transmissions were generated. A data set consisting of 620 observations (20 for each receiver node), the number of hops (H) and the measured time was generated.

Equation 6 was estimated using linear regression to yield the following relation:

$$M\ (ms) = 1.23 + 0.0009XH + 0.485I. \quad (7)$$

The model's R-square was 0.9939 and each coefficient significant at the 99.9% level. The latency of 1.23 milliseconds is lower than the 1.7 reported by Reed and Grunwald (1987) and the 0.9 microseconds per byte is considerably higher than their estimate. We suspect that the earlier work included nearest neighbor transmissions only. The .485 millisec delay experienced at each intermediate node represents the low level protocol to hand-off the message to another communications channel and is not dependent on message length. This time is somewhat lower than the latency time at the receiver and sender ends of the path but represents a major culprit in explaining the less than theoretical speedup obtained in practice.

Using Equation 7 as the function which maps message length to transmission time, a simulation model described in Figure 2 was constructed. We chose to model the parallel processor using the SLAM (1987) simulation language and executed the model on a VAX 11-785. The hypercube is modeled as a single user system with 16 nodes declared in the cube. The cube and the 16 nodes are unique SLAM Resources while communication channels are modeled as single server Activities preceded by a Queue. Each channel uses a unique activity and queue file number which facilities routing of entities through the network via a lookup table. Basically, the simulation proceeds as follows:

a. A job enters the system and waits for the hypercube.

b. When the cube becomes available, it is allocated to the first waiting job.

c. The time the hypercube is allocated is recorded as the job start time.

d. The job is replicated into 16 processes.

e. Each process is assigned a processor identification, a number of CPU bursts, and a an average burst duration depending on the selection of the case in Table 1.

f. Each process waits for the node to which it is assigned.

g. When the node becomes available, the node processes one burst of exponentially distributed length and initiates a single IO of random length (100-1024 bytes) to a randomly selected receiver node (not including itself). The number of bursts remaining for that node is decremented.

h. The node that processed and initiated the IO is freed and available for the next CPU burst.

i. The process entity is replicated as a message entity with higher processing priority. The process entity returns to the originating node to wait for processing time.

j. Using the randomly selected receiver assignment, a lookup table is used to determine the next channel required by the message to reach its intended receiver.

k. The message waits in the appropriate channel queue.

l. When the channel becomes available, the message is transmitted using Equation 7 to determine the time of transmission. Note: at this level all transmissions are nearest neighbor (one hop).

m. The receiving node is preempted.

n. If the receiving node is not the final destination, it uses up 0.485 milliseconds of hand-off time, computes the next intermediate node, looks up the channel number and retransmits the message as in step 1. The intermediate node is freed.

o. If the receiving node is the final destination, it processes the message using one-half the latency found in Equation 7 and the entity is terminated.

p. When all bursts have been completed and all messages have been processed, the time the job has been in the system is recorded and the hypercube is freed for the next job. Ten jobs are simulated to form one of ten data points in the experimental unit average.

The resulting simulated job execution times were considered to be accurate reflections of actual hypercube performance for several reasons. First, the balanced case measurements were reasonable and corresponded to actual experience with the hypercube. Second, when the degree of load imbalance was maximized the execution time did in fact move to the known uniprocessor time. Third, the progression of execution times as the load imbalance was increased was reasonable and produced a speedup profile which agrees with engineering judgment and intuition. Finally, each component of the simulation was tested and desk checked to insure compliance with the design specifications.

3. EXPERIMENT RESULTS

The raw execution times and the calculated speedup statistics are shown in Table 2. Figure 3 depicts the data with respect to B, the load imbalance metric. It is evident that extreme variability is present and that there is overwhelming

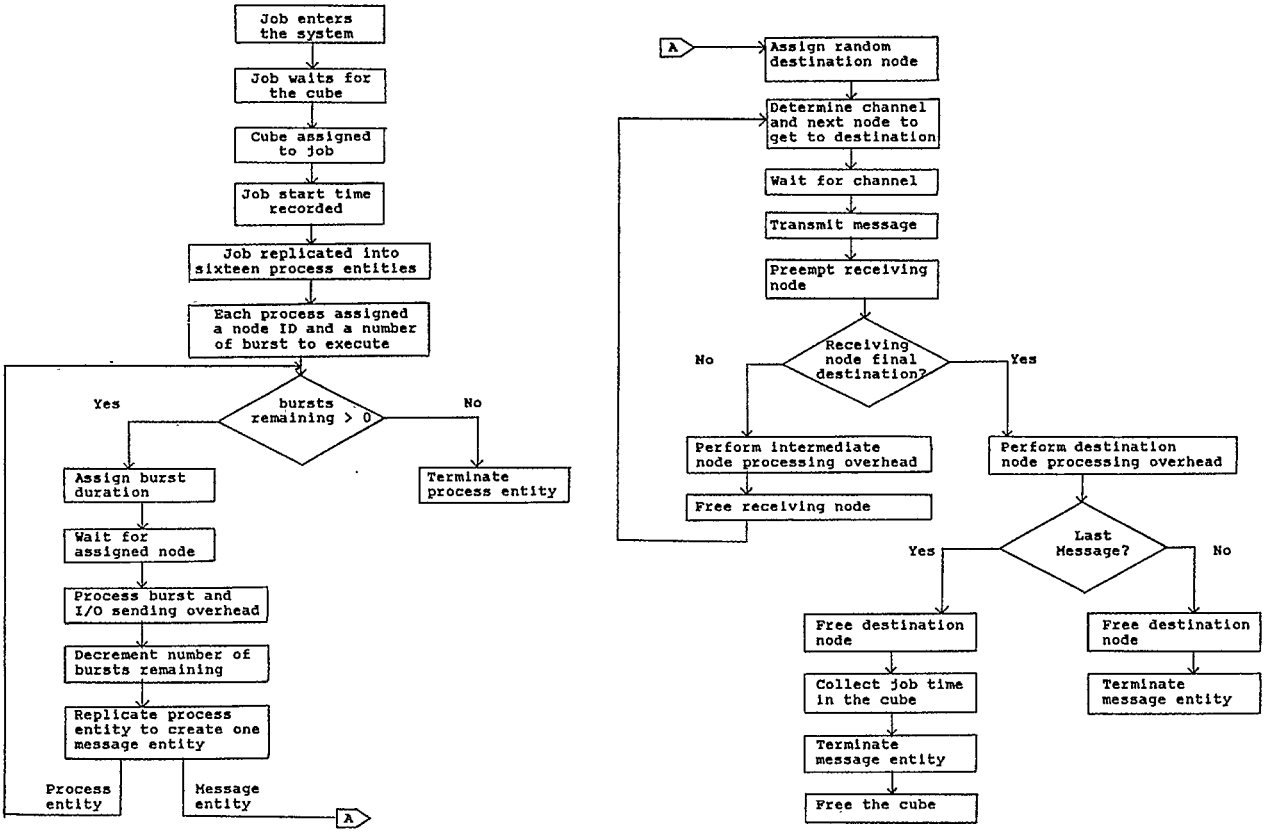

Figure 2: Simulation Model Flow Diagram

Table 2: Simulation Results, 16 Node Hypercube

Case	B	L	Time (ms)		Speedup	
			R=10	R=2	R=10	R=2
1	0.00	0.00	429.2	110.2	9.6	7.5
2	0.37	0.02	503.2	127.8	8.2	6.5
3	0.37	0.09	496.3	129.4	8.3	6.4
4	0.78	0.05	652.5	164.4	6.3	5.0
5	0.78	0.19	664.0	167.9	6.2	4.9
6	0.96	0.06	779.8	188.7	5.3	4.4
7	0.96	0.19	780.9	194.8	5.3	4.2
8	1.37	0.00	1306.0	303.7	3.2	2.7
9	1.37	0.22	1309.0	322.5	3.2	2.6
10	1.54	0.00	1417.0	331.5	2.9	2.5
11	1.54	0.25	1442.0	350.2	2.9	2.4
12	2.05	0.00	1792.0	413.1	2.3	2.0
13	2.05	0.34	1818.0	438.8	2.3	1.9
14	2.56	0.00	2162.0	493.5	1.9	1.7
15	2.56	0.42	2178.0	525.8	1.9	1.6
16	3.75	0.48	4075.0	936.6	1.1	0.9
Uni	4.00	0.52	4131.8	826.4	1.0	1.0

evidence of nonlinear effects. Based on the evidence a 16 processor hypercube with a coefficient of variation of load (B) around 1.5 and a CPU/IO intensity of 10 performs like the theoretical 4 node machine. Clearly, the penalty for load imbalance is severe!

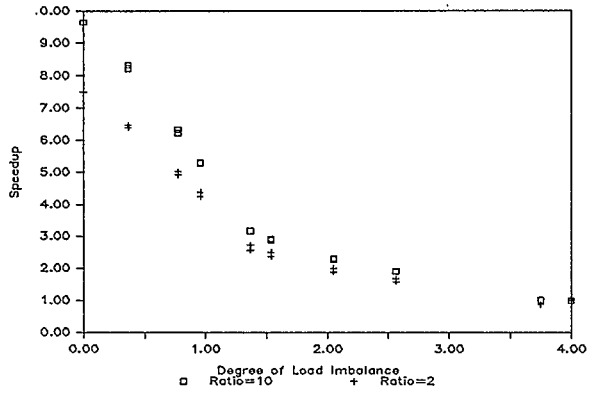

Figure 3: Average Speedup Measurements versus Degree of Imbalance (B)

Figure 4 displays the data with respect to the load locality metric (L). Again, the effect is evident and nonlinear. At this point we argue that the inclusion of a nonlinear term in Equation 5 is necessary. Therefore, we introduce the square of B and L into the model as a simplw way to estimate nonlinear effects and restate the hypothesized relationship to be a polynomial fit of degree 2. Other transformations could be used; however, the curvature of the line appears to obey a power law which is straightforward in its estimation.

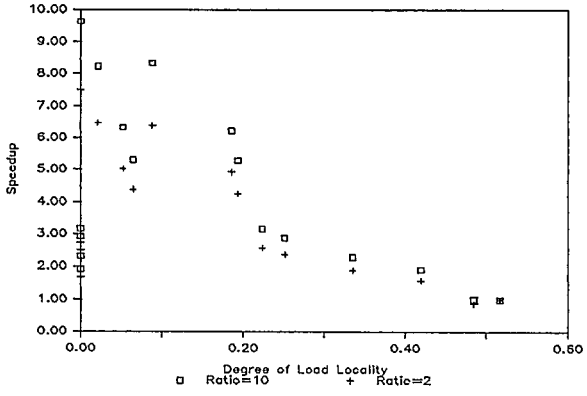

Figure 4: Average Speedup Measurements versus Degree of Locality (L)

We estimated Equation 5 with the additional squared B and L terms using least squares and refer to this model as Model 1. We then removed each term not significant at the 99% level and re-estimated the relation to yield Model 2. Again, deleting non-significant terms, we estimate the simplest model referred to as Model 3. Table 3 shows the estimated coefficients. Figure 5 compares the observed data with predictions based on Model 1, the full featured model. The model predicts speedup quite well as evidenced by an R-square of 98.4%.

Table 3: Model Coefficients

Term	Least Squares Estimates		
	Model 1	Model 2	Model 3
Constant	7.46**	8.16**	8.13**
R	0.25**	0.10**	0.10**
B	-4.89**	-4.91**	-4.84**
L	1.54*		
RxB	-0.11*		
RxL	-0.25		
BxL	-5.93**	-0.30	
RxBxL	0.15		
B^2	1.05**	0.80**	0.74**
L^2	29.57*		
Model R^2	0.984	0.961	0.960

* Significant at 0.05 level
** Significant at 0.01 level

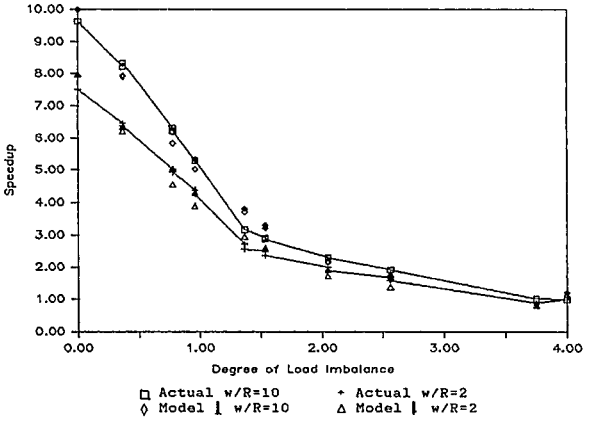

Figure 5: Actual versus Model 1 Predictions

Model 1 establishes that the ratio of CPU burst time to message time is highly significant but not really involved in any interaction. That is, the ratio's effect is a scaler which tends to adjust the curve up or down by a factor of .25 milliseconds per unit of R. The balance metric and the locality metric both enter the model as linear and nonlinear operators. The impact of locality appears minimal and involved in a balance interaction. Apparently, locality by itself does not influence speedup to any great extent. We tested this question further by varying the locality over four settings for a high and low level of the balance metric for both values of R. Figure 6 indicates confirmation of the regression analysis: locality does not affect speedup very much!

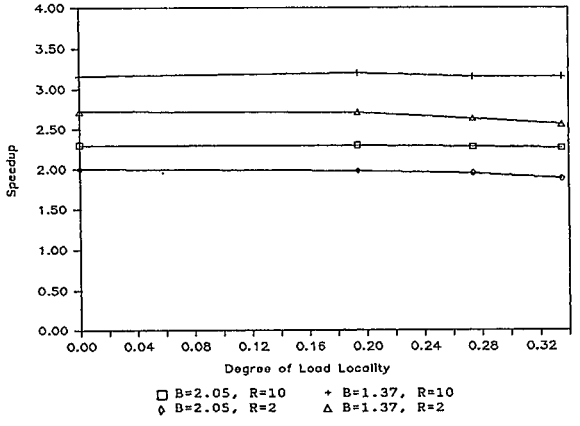

Figure 6: Effects of Load Locality (L)

Using the results of Model 1, the non-significant terms were removed to yield simple Models 2 and 3. The Models' R-square remain high (96.1 and 96%) indicating little loss of explanatory power as terms are

removed. Figure 7 depicts the actual observations versus predictions using the simplest model, Model 3. Interpretation of Model 3 is straightforward: the ratio of CPU time to transmission time contributes 0.1 milliseconds per unit across all levels of imbalance; the imbalance metric (B) basically governs the shape of the speedup degradation by subtracting out 4.84 millisecs for every unit increase in B adjusted by adding the square of B times 0.74 milliseconds. The penalty for imbalance is severe initially but tapers off as the square term adds back speedup to terminate in the uniprocessor performance.

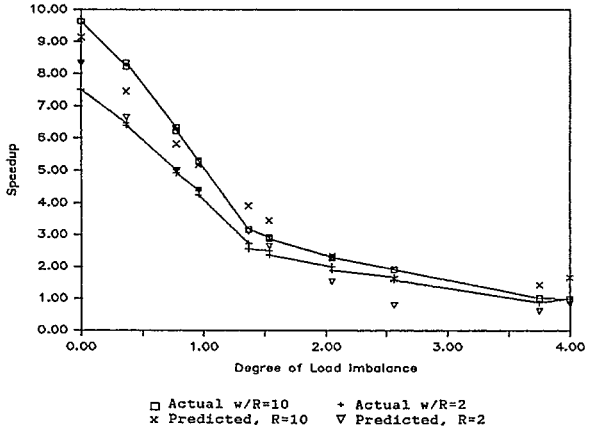

Figure 7: Actual versus Model 3 Predictions

Recalling that the imbalance metric is the ratio of the load standard deviation to the load average; it appears that as the standard deviation reaches the hypercube average, performance suffers dramatically. Furthermore, as the IO load becomes more dominant, the speedup is initially worse and subject to the same imbalance phenomenon. Locality appears to be of minimal impact and involved in statistically significant interactions which are difficult to explain from an engineering point of view. In short, the null hypothesis is resoundly rejected. There is definitely a relationship between balance, locality and IO intensity which characterizes speedup phenomenon very well.

4. CONCLUSIONS

It is apparent that load imbalance severely impacts a parallel processor's performance. The adverse effects are acute when even minor aberrations from a balanced load are allowed. The effect of load locality is minor and enters the speedup model primarily as an interactive term. This would suggest that locality effects, though minor, influence speedup behavior in ways that depend on the degree of imbalance. The intensity of IO is significant and affects the speedup across all levels of locality and imbalance.

The more IO involved in a process compared to CPU processing, the worse the speedup characteristics. This is intuitive since IO preempts node processing and introduces overhead which a single processor would not experience. What is not intuitive is that the IO load does not interact with other terms. Apparently, higher IO loads cause a consistent worsening of performance regardless of the locality or imbalance of the load.

The findings of this research have serious impact on algorithm decomposition strategies. Given a known CPU to IO load, the balanced case speedup can be determined by simulation or benchmarking. As soon as processor imbalance is allowed, dramatic performance degradation can result. This research indicates that imbalance could not be overcome by locality. However, we did not model the affinity one node might have for another in terms of its IO. If such an affinity were known, we predict that intelligent spatial loading, even if unbalanced, would be useful. We suspect though, that simple relocation of unbalanced loads may never recover the inherent loss of speedup caused by the unbalanced condition.

The use of the balance metric (B) and the locality metric (L) are simple statistics which can be used to model any process which can be monitored during execution. The simulation approach allows a statistical approach to predicting process performance which we feel provides a convenient framework for analysis. Sensitivity analysis is possible with multiple simulation runs.

Several issues remain to be investigated. First, what happens when the messages generated by a node must be sent to all other nodes? Clearly, this situation will worsen the IO load and may change the interpretation of the analysis. Second, does the dimension of the hypercube affect the performance as the load becomes unbalanced? That is, would load imbalance on an 8 node or a 64 node machine be similar to the 16 node case. We suspect that the initial cube dimension will have an effect so that lower dimensioned cubes are more adversely affected. However, we make these comments with caution since experience has indicated counter-intuitive results! These issues are the subject of current research and the findings are forthcoming.

Finally, we suggest the use of animated simulation models to "watch" the dynamic nature of load imbalance, locality, IO intensity and message passing affinity. We propose to develop an animated model of the present simulation and demonstrate its effectiveness.

ACKNOWLEDGMENTS

We wish to thank Captain Cathy Lamanna for her assistance in benchmarking the Intel iPSC Hypercube and the Department of Electrical Engineering at AFIT for their support and use of computing resources.

REFERENCES

Bhuyan, L.N. (1984) "On the Performance of Loosely-Coupled Multiprocessors," Proceedings of the 11th Annual IEEE Computer Architecture Symposium. 256-262. IEEE Press, New York.

Du, H. (1985) "On the Performance of Synchronous Multiprocessors," IEEE Transactions on Computers, 34 (5): 462-466.

Jalby, W. and Meier, U. (1986) "Optimizing Matrix Operations on a Parallel Multiprocessor with a Hierarchical Memory System," Proceedings of the 1986 International Conference on Parallel Processing. 429-432. IEEE Press, New York.

Nestle, E. and Inselberg, A. (1985) "The Synapse N+1 System: Architectural Characteristics and Performance Data of a Tightly-Coupled Multiprocessor System," Proceedings of the 12th Annual IEEE Computer Architecture Symposium. 233-239. IEEE Press, Silver Spring, Md.

Padmanabahn, K. and Lawrie, D. (1985) "Performance Analysis of Redundant-Path Networks for Multiprocessor Systems," ACM Transactions on Computing Systems, 3 (2): 117-144.

Pritsker, A. (1986) Introduction to Simulation and SLAM II, John Wiley, New York.

Reed, D. and Grunwald, D. (1987) "The Performance of Multicomputer Interconnection Networks," *Computer*, June 1987, 63-73.

Vrsalovic, D. and others. (1985) "The Influence of Parallel Decomposition Strategies on the Performance of Multiprocessor Systems," Proceedings of the 12th Annual IEEE Computer Architecture Symposium. 396-405. IEEE Press, New York.

Vrsalovic, D. and others. (1984) "Performance Prediction for Multiprocessor Systems," Proceedings of the 13th International Conference on Parallel Processing. 139-146. IEEE Press, New York.

Wiley, P. (1987) "A Parallel Architecture Comes of Age at Last," *IEEE Spectrum*, June 1987, 46-50.

AUTHORS' BIOGRAPHIES

WADE H. SHAW, JR. is an Assistant Professor of Electrical and Computer Engineering at the Air Force Institute of Technology in Dayton, Ohio. He completed his Ph.D. in Engineering Management at Clemson University. He holds a B.S. in Electrical Engineering and a M.S. in Systems Engineering. Dr. Shaw has published numerous research papers on a variety of subjects including Computer Performance, Simulation, Decision Support Systems, Software Engineering and Project Management. He actively pursues teaching, research and consulting in simulation, computer performance evaluation and management information systems. Dr. Shaw is a member of several professional organizations including the IEEE, IIE, DSI, TIMS, ORSA and SCS. He is a Captain is the US Army and a registered professional engineer.

Department of Electrical Engineering
AFIT/ENG
WPAFB, OH 45433
(513) 255-3576

TIMOTHY S. MOORE is a Masters degree candidate in Information and Computer Systems at the Air Force Institute of Technology. He holds a B.S. in Computer Science from the University of Alabama and is a Captain in the US Air Force. Captain Moore previously served as a Warning Systems Analyst for the Command Center Processing and Display System at Offutt AFB in Nebraska.

Bldg 640, Box 4079
Air Force Institute of Technonolgy
WPAFB, OH 45433
(513) 255-3576