PERFORMANCE EVALUATION OF AN OPTIMAL PROCESSOR ASSIGNMENT

by

Soonhoi Ha and Edward A. Lee

Memorandum No. UCB/ERL M94/92

29 November 1994

PERFORMANCE EVALUATION OF AN OPTIMAL PROCESSOR ASSIGNMENT

by

Soonhoi Ha and Edward A. Lee

Memorandum No. UCB/ERL M94/92

29 November 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PERFORMANCE EVALUATION OF AN OPTIMAL PROCESSOR ASSIGNMENT

by

Soonhoi Ha and Edward A. Lee

Memorandum No. UCB/ERL M94/92

29 November 1994

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Performance Evaluation of an Optimal Processor Assignment

Soonhoi Ha * Edward A. Lee†

Abstract

Balancing the loads of processors and minimizing interprocessor communication (IPC) overhead are two conflicting goals of the scheduling of a parallel processing system. A parallel scheduler partitions the tasks into the same number of groups as the processors and assigns the task groups to the processors. Since the IPC overhead depends on the network topology, the scheduler may take into account the network topology either in the partitioning phase or in the assignment phase. This paper evaluates the performance of the optimal assignment in the latter approach. In the partitioning phase, the IPC overhead is ignored or simplified assuming that the distance between all processor pairs are the same. After the partitioning phase, the IPC requirements between the task groups are determined. In the assignment phase, pairs of groups with more IPC requirements are assigned to pairs of processors with lower distance to reduce the IPC overhead or the total traffic requirements on the network. A simpler scheduler may do without the assignment phase and assign the task groups randomly to the processors. The expected performance improvement of an optimal assignment over a random assignment is analytically investigated for a class of network topologies: hierarchical bus and binary hypercube. The analysis is based on an optimistic (but infeasible in most cases) assignment assumption. But its results are largely supported by simulation.

keywords: Multiprocessor scheduling, interprocessor communication overhead, processor assignment, task partitioning

1 Introduction

To exploit the parallelism of a program in a parallel processing system, various ways of representing a program as a set of tasks with partial ordering have been investigated. In a dataflow language [1], for example, each instruction comprises a task to fully express the fine grain parallelism of the program. In a hybrid representation of dataflow and sequential languages [2], a block of sequential instructions forms a coarse grain task.

Balancing the loads of processors and minimizing interprocessor communication (IPC) overhead are two conflicting goals of the scheduling of a parallel processing system. A parallel scheduler partitions and

*S. Ha is with the Department of Computer Engineering, Seoul National University, Seoul, 151-742, Korea; e-mail: sha@snucom.snu.ac.kr.

†E. Lee is with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720, USA; e-mail: eal@ohm.berkeley.edu.

This research is part of the Ptolemy project, which is supported by the Advanced Research Projects Agency and the U.S. Air Force (under the RASSP program, contract F33615-93-C-1317), Semiconductor Research Corporation (project 94-DC-008), National Science Foundation (MIP-9201605), Office of Naval Technology (via Naval Research Laboratories), the State of California MICRO program, and the following companies: Bell Northern Research, Cadence, Dolby, Hitachi, Mentor Graphics, Mitsubishi, NEC, Pacific Bell, Philips, Rockwell, Sony, and Synopsys.

assigns the tasks to the processors, determines the execution orders of the assigned tasks in each processor, and executes them. Depending on when these scheduling actions are performed, a parallel scheduler is classified into the following four classes: fully-dynamic, static-assignment, self-timed, and fully-static [3]. Fully-dynamic scheduling assigns a ready task to an idle (or least busy) processor to balance the loads. Run-time assignment, however, is often prohibitive since it incurs significant communication overhead for shipping tasks between processors or control overhead for monitoring and maintaining the execution status of the processors. The other scheduling techniques partition and assign tasks at compile-time. The execution order of the assigned tasks are determined at run-time in the static-assignment scheduling, and at compile-time in the self-timed and the fully-static scheduling. Fully-static scheduling guarantees the synchronization of tasks at compile-time to minimize the scheduling overhead at run-time. In this paper, we assume that the tasks are assigned at compile-time. Thus, the paper applies to three of the four classes of schedulers.

An unavoidable run-time overhead in parallel processing comes from interprocessor communication (IPC). If the IPC overhead between two tasks A and B is larger than the execution duration of either one task, assigning these tasks to different processors is not advantageous. The IPC overhead involves not only the communication delay required to transfer data between the source and the destination processors, but also the arbitration delay to acquire access to the interconnection network. Synchronization delay, for example checking semaphores, may also be involved. Since these delays depend on the run-time status of the system, they usually cannot be predicted at compile-time. A common hardware technique to compensate for this overhead is to use a split transaction scheme with dedicated hardware for network access. The overhead can be hidden effectively if the processors are kept busy with other program segments or with other tasks during this transaction.

A software technique is to assign the tasks with large IPC requirements to the same processor. Several clustering methods have been developed without [4] and with [5, 6] consideration of precedence relations between tasks. These methods cluster tasks subject to minimal interprocessor communication, and assign these clusters to processors to balance the loads [4] or minimize the bottleneck load [5]. In the clustering phase, the actual topology of the interconnection network is not taken into account. Since the unit of assignment is cluster which is larger than task, load balancing is likely to be sacrificed. Thus, clustering methods favor the minimization of the IPC overhead over the load balancing as the scheduling goal.

Another approach is list scheduling. In list scheduling, tasks are assigned priorities. During the scheduling process, the unscheduled task of the highest priority is assigned to the first available processor. By incorporating the IPC requirements of each task in the computation of its priority, we may achieve the effect of clustering two tasks with large IPC requirements [7].

If the interconnection topology is fully-connected or shared-bus, the distances between all processor pairs (or processor-memory pairs) are the same. The *interprocessor distance* is defined as the number of communication links on the shortest path between a pair of processors. In most scalable topologies, however, the interprocessor distances and so the IPC overheads depend on the position of processors. Some scheduling techniques take into account the IPC overhead specific for the physical interconnection network topology in the partitioning phase by scheduling the communication resources at the same time [7]. Another technique, called two-phase scheduling, divides partitioning and assigment into two separate phases. The primary objective of the partitioning phase is to balance the loads and minimize the IPC requirements assuming constant interprocessor distance. In this phase, the tasks can be thought to be assigned to the virtual processors of fully-connected network topology. At the end of partitioning, the IPC requirements between virtual processors are determined. The optimal assignment phase maps the virtual processors of larger IPC requirements to the physical processors as close as possible to minimize the total IPC overhead. In this paper, we examine how much gain we may obtain from an optimal assignment compared with random assignment in such a two-phase strategy by analysis and simulation. If distances

between processors are all the same, random assigment will be as good as any. Hence, network topologies of interest in this paper do not have a constant distance between processors.

1.1 Problem Definition

Once the partitioning phase is done, the IPC requirements for each pair of virtual processors are determined. The objective of the assignment phase is to minimize the total communication cost, or the total message traffic over the communication network. A unit of message traffic on a communication link is defined as the volume of data exchanged through the link. The total message traffic is the sum of message traffic on all links. Define v_{ij} and d_{ij} as follows.

v_{ij} is the volume of data exchanged between processors i and j.

d_{ij} is the interprocessor distance between processor i and j.

Then, the total message traffic C_T becomes

$$C_T = \sum_{i,j} v_{ij}\, d_{ij} \tag{1}$$

The units and absolute scaling are not important because such costs will only be compared against similar costs. The problem of minimizing the total message traffic by an optimal assignment is known as the quadratic assignment problem [8] and has NP complexity. Hanan et. al. reviewed three techniques for the placement of blocks in a layout design with the same form of the object function as equation (1): constructive-initial-placement, iterative-improvement, and branch-and-bound. The iterative-improvement technique chooses an initial assignment and adjusts the assignment, for example by pairwise interchange of processors, in order to optimize the objective function. The other two techniques assign a block to the best place once and for all and do not change the assignment later. They did not consider the effect of network congestion. Lee and Aggarwal [9], on the other hand, formulated a set of new objective functions quantifying the effect of congestion based on deterministic information about when each communication occurs and with how much volume. They used an iterative-improvement technique with pairwise interchange. The problem of minimizing network congestion with a given assignment may be attacked separately as the traffic scheduling problem [10], which schedules the communication route and time to minimize the link congestion.

In the next section, we analytically investigate the expected reduction of the total message traffic from an optimal assignment compared to a random assignment. If the number of processors is N, the number of processor pairs M, called *virtual links*, is $M = N(N-1)/2$. Each virtual link may be mapped to a physical link in some network topologies such as fully-connected graph and shared bus. In other networks, however, the number of physical links is much smaller than the number of virtual links and a virtual link may be mapped to a path of physical links. For example, the total number of physical links is $N log\, N/2$ in a binary hypercube network and $N-1$ in a ring network. We define the cost of a virtual link as the interprocessor distance on the physical network and classify the virtual links according to their costs.

The Encore Ultramax [11] multiprocessor system has a two-level hierarchical bus as the interprocessor network topology. In this topology, the cost of a virtual link between processors on the same local bus is 1 while the cost of a virtual link between processors on the different local busses is 3. On the other hand, many networks such as hypercube and ring have more than two virtual link costs. Our analysis will be applied to various kinds of topology.

In section 3, we simulate processor assignment two common network topologies, and investigate the effectiveness of an optimal assignment. We developed a heuristic suboptimal algorithm based on both the constructive-initial-placement technique and the iterative-improvement technique. In the concluding section, we discuss the simulation results and compare them with the analysis results.

2 Analysis

The communication requirements between virtual processors are determined in the partitioning phase. To minimize the total message traffic, pairs of virtual processors with large communication requirements should be assigned to closely spaced processors. We model the volume of data on each virtual link as a random variable and its distribution, called *routing distribution*, is assumed known for mathematical simplicity. We also assume an optimistic, oversimplified strategy that admits infeasible assignments. The strategy is a greedy one, in which each virtual link is assigned to a physical link in order; the next virtual link with the highest load is assigned to the next physical link with the lowest cost. The reason that this might result in an infeasible assignment is that the virtual links cannot, in practice, be assigned independently to physical links. For example, if A, B, C, and D are virtual processors on a ring network, and link A¡-¿B and A¡-¿D have been assigned to physical links, then link B¡-¿D has been assigned as a side effect. Thus, link B¡-¿C is assigned to a physical link with lower cost than link B¡-¿D regardless of its communication load. Our simplifying assumption is that the links can be independently assigned. This results in an assignment at least as good as the optimal assignment, and hence provides a lower bound on the optimal assignment.

At first, we consider the network topologies with bi-valued interprocessor distances such as hierarchical bus topology. Virtual links can be classifies into high-cost links and low-cost links after assignment. Let the number of the high-cost links H and the cost c_h. The number of the low-cost links is $M-H$. We define the cost of low-cost links as c_l. In a two-level hierarchical bus topology, c_h becomes 3 and c_l becomes 1. An optimistic optimal assignment assigns $M-H$ virtual links with large communication requirements to the low-cost links and remaining virtual links to the high-cost links. To determine whether the assignment is realizable or not, we construct a graph with virtual processors and the $M-H$ virtual links of low-cost. If the graph can be embedded with unit dilation in the physical network topology, the assignment is realizable. Otherwise, it is not. Nonetheless, our analysis gives a lower bound of the total message traffic for the given communication requirements. In our analysis, we ignore the effect of traffic congestion.

2.1 Uniform Distribution Assumption

Let's assume that the normalized communication requirement on a virtual link is approximated as a random variable X_i of uniform distribution on interval [0,1]. Suppose we have a set of M random samples $\{X_i\}$ that are independent and identically distributed. We rearrange them into an ordered set $\{X_{i|M}\}$, in which $X_{i|M}$ is a random variable representing the i-th smallest sample among M samples in $\{X_i\}$. Clearly, the difference between any outcome of the set $\{X_i\}$ and the corresponding outcome of the set $\{X_{i|M}\}$ is the order. Then, the expected total message traffic $E[C_T]$ becomes

$$\begin{aligned} E[C_T] &= E[c_h \sum_{i=1}^{H} X_{i|M} + c_l \sum_{i=H+1}^{M} X_{i|M}] \\ &= E[c_l \sum_{i=1}^{M} X_{i|M} + (c_h - c_l) \sum_{i=1}^{H} X_{i|M}] \\ &= c_l E[\sum_{i=1}^{M} X_{i|M}] + (c_h - c_l) E[\sum_{i=1}^{H} X_{i|M}] \end{aligned} \tag{2}$$

where $E[]$ means the expected value of the expression in the bracket. The first term in the above equation is the sum of all M random samples in the ordered set, which is the same as the sum of all samples in $\{X_i\}$. That is,

$$E[\sum_{i=1}^{M} X_{i|M}] = E[\sum_{i=1}^{M} X_i] = M\, E[X_i]. \tag{3}$$

Since the expected value of X_i is 1/2, the first term of equation (2) becomes $c_l\, M/2$. Now, we have to compute the expected value of $X_{i|M}$. If the probability density function of $X_{i|M}$ is $f_i(x)$,

$$E[X_{i|M}] = \int_0^1 x\, f_i(x)\, dx. \tag{4}$$

Since the probability distribution function $P(x)$ of a uniform random variable on $[0,1]$ is x and the probability density function $p(x)$ is 1, the probability density function $f_i(x)$ becomes, from equation (23) in the appendix,

$$f_i(x) = \frac{x^{i-1}(1-x)^{M-i}}{B(i, M-i+1)}, \tag{5}$$

where $B(a,b)$ is a beta function defined in 22. Using this equation, we reduce equation (4) as follows:

$$E[X_{i|M}] = \int_0^1 \frac{x^i(1-x)^{M-i}}{B(i, M-i+1)} dx = \frac{B(i+1, M-i+1)}{B(i, M-i+1)} = \frac{i}{M+1}. \tag{6}$$

Therefore, the expected total traffic in equation (3) becomes

$$\begin{aligned} E[C_T] &= c_l \frac{M}{2} + (c_h - c_l) \sum_{i=1}^{H} \frac{i}{M+1} \\ &= c_l \frac{M}{2} + (c_h - c_l) \frac{H(H+1)}{2(M+1)}. \end{aligned} \tag{7}$$

Now, suppose that we assign processors randomly. Since the expected cost of a virtual link, c_{avg}, is $\frac{H}{M} c_h + \frac{M-H}{M} c_l$, the expected total message traffic E_{random} becomes,

$$E_{random} = M\, E[X_i]\, c_{avg} = c_l \frac{M}{2} + (c_h - c_l) \frac{H}{2}. \tag{8}$$

From equations (7) and (8), the difference of the expected total message traffic between an optimistic optimal assignment and a random assignment, δE, is obtained:

$$\delta E = E_{random} - E[C_T] = H(c_h - c_l) \frac{M-H}{2(M+1)}. \tag{9}$$

The performance improvement, or the reduction ratio of the total message traffic, by adopting an optimal assignment over a random assignment is defined as the ratio of δE and E_{random}. Define the *cost ratio* r_c as c_h/c_l, and the *link ratio* r_l as H/M. Then, the expected performance improvement $P_U(r_c, r_l)$ can be represented in terms of r_c and r_l as follows (we approximate the term $M+1$ in the denominator of equation (9) as M)

$$P_U(r_c, r_l) = \frac{\delta E}{E_{random}} = \frac{r_l(1-r_l)(r_c-1)}{1 + r_l(r_c-1)}. \tag{10}$$

We display the expected performance improvement in figure 1, varying r_l and r_c.

The expected performance improvement in equation (10) increases as the cost ratio r_c increases for a fixed link ratio. On the other hand, if we fix the cost ratio, the expected performance improvement is maximized when the link raio r_l is $\frac{1}{\sqrt{r_c}+1}$ and the value is $\frac{\sqrt{r_c}-1}{\sqrt{r_c}+1}$. When the cost ratio is greater than 1, the link ratio that gives the maximum performance improvement is smaller than 0.5. Particularly in a two-level hierarchical bus topology where the cost ratio is 3, the link ratio should be 0.37 for maximum performance improvement, which is 27%. As the cost ratio increases, the maximum performance improvement approaches 1 or 100% and the optimal link ratio, not surprisingly, approaches zero.

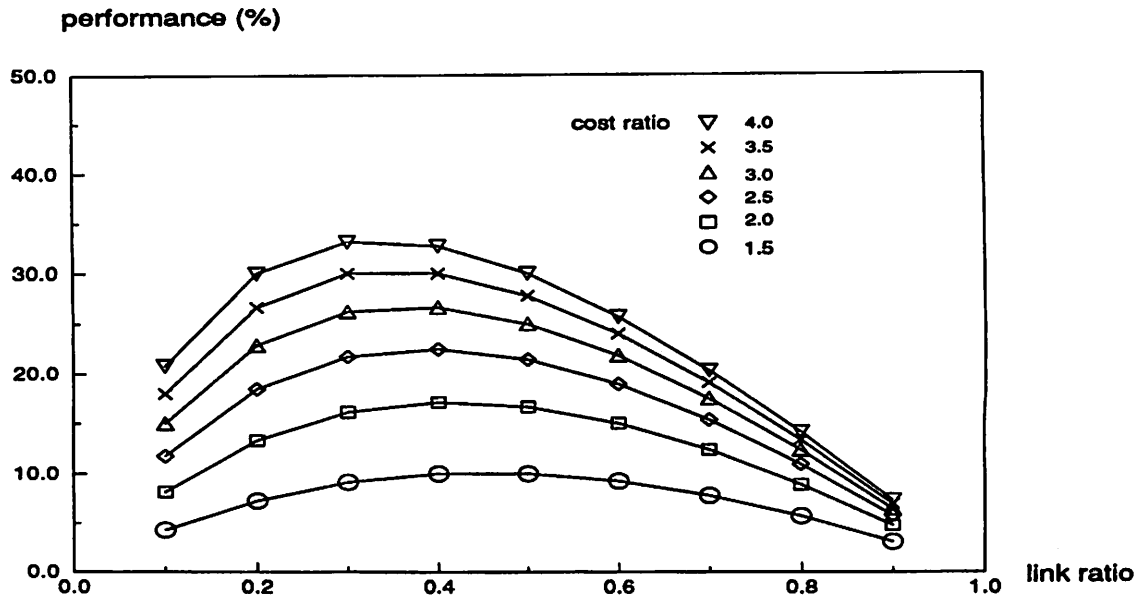

Figure 1: Analytical message traffic reduction with varying the cost ratio r_c and the link ratio r_l from the optimal assignment under the assumption of uniform distribution of communication loads.

For a given network topology, however, the cost ratio and the link ratio may not be changed. When the network is a fully-connected network or a bus network, both r_l and r_c are equal to 1, and $E[C_T]$ becomes $M/2$ as expected. Since a random assignment results in the same expected total message traffic, no assignment phase is necessary. Consider a hierarchical bus network, in which a local bus connects 4 processors and a global bus connects 2 local buses. The cost ratio is 3 and the link ratio is $16/28 = 0.57$. Hence, the expected performance improvement becomes 23% from equation (10).

2.2 Geometric Distribution Assumption

In the previous subsection, we assume that the communication loads between processors are uniformly distributed. In reality, there may be situations when the communication loads between processors are not uniform. For example, a program may be partitioned to a subset of the processors either because of lack of parallelism or because of partitioning constraints. Moreover, a partitioning strategy may result in a skewed communication loads between processors.

As a model of a localized distribution of the communication requirements between processors we use a geometric distribution with parameter q. The probability that random variable X_i is greater than x, $P[X_i \geq x]$ is q^x and the probability density function $p(x)$ of X_i is $q^x(1-q)$. Therefore, the expected value of X_i becomes $\frac{q}{1-q}$. The expected value of the ordered variable $X_{i|M}$ can be obtained from equation (24) in the appendix:

$$E[X_{i|M}] = \sum_{x=0}^{\infty} x \{(1 - F_i(x-1)) - (1 - F_i(x))\} = \sum_{x=0}^{\infty} (1 - F_i(x)), \tag{11}$$

where $F_i(x)$ is given by equation (20). Since

$$1 - F_i(x) = 1 - \sum_{k=i}^{M} \binom{M}{k} P^k(x)(1-P(x))^{M-k} = \sum_{k=0}^{i-1} \binom{M}{k} P^k(x)(1-P(x))^{M-k}, \tag{12}$$

equation (11) can be rewritten as follows:

$$E[X_{i|M}] = \sum_{x=0}^{\infty} \sum_{k=0}^{i-1} \binom{M}{k} P^k(x)(1-P(x))^{M-k}. \tag{13}$$

Equation (2) for the expected total message traffic is valid regardless of the distribution. Therefore, we obtain the following formula for the expected total message traffic under the assumption of geometrically distributed communication requirements.

$$E[C_T] = c_l M \frac{q}{1-q} + (c_h - c_l) \sum_{i=1}^{H} \sum_{x=0}^{\infty} \sum_{k=0}^{i-1} \binom{M}{k} (1-q^x)^k (q^x)^{M-k}. \tag{14}$$

After performing the summation on the x variable in the second term, the above equation is reduced to

$$E[C_T] = c_l M \frac{q}{1-q} + (c_h - c_l) \sum_{i=1}^{H} \sum_{k=0}^{i-1} \sum_{n=0}^{k} \binom{M}{k} \binom{k}{n} (-1)^n \frac{q^{M-k+n}}{1-q^{M-k+n}}, \tag{15}$$

Unfortunately, the equation for the total message traffic can not be reduced to a closed form, unlike the uniform distribution case.

On the other hand, the expected message traffic in case of a random assignment becomes, from equation (8)

$$E_{random} = c_l M \frac{q}{1-q} + (c_h - c_l) H \frac{q}{1-q}. \tag{16}$$

From equations (15) and (16), the expected performance improvement from an optimal assignment over a random assignment can be represented as follows:

$$P_U(r_c, r_l) = \frac{(r_c - 1) r_l (1 - \frac{1-q}{Hq} \sum_{i=1}^{H} \sum_{k=0}^{i-1} \sum_{n=0}^{k} \binom{M}{k} \binom{k}{n} (-1)^n \frac{q^{M-k+n}}{1-q^{M-k+n}})}{1 + r_l (r_c - 1)}. \tag{17}$$

For large M, care should be taken during computation of equation (17) because of the risk of overflow or large rounding error. One method of overcoming this difficulty is to obtain the ensemble average $E[X_{i|M}]$ from the experiments and use the experimental result as the approximation of $E[X_{i|M}]$ in equation (11). It is well-known that the ensemble average obtained from the large number of experiments approaches to the exact expected value. We repeated the experiments 1000 times and used the ensemble average as the expected value.

In the next section, we will investigate the performance improvement through simulation for two-level hierarchical bus networks with 8 processors and with 16 processors assuming that the local bus connects 4 processors. In the network of 8 processors, the values of M and H are 28 and 16 respectively. In the network of 16 processors, they are 120 and 96 respectively. Using these values of M and H, we compute the expected performance improvement of equation (17) by varying the geometric parameter q from 0.1 to 0.9 as shown in table 1.

2.3 Hypercube Networks

Up to now, we consider the networks that have two different interprocessor distances. In this subsection, we extend out analysis to the networks with more than two interprocessor distances. In a static interconnection network, the interprocessor distance ranges from 1 to the diameter of the network. The diameter of a binary hypercube network of N processors is $\log N$.

Table 1: Analysis of the expected performance improvement from the optimal assignment under the assumption of geometrically distributed communication requirements in the two-level hierarchical bus networks

parameter q	0.1	0.3	0.5	0.7	0.9
8 processors	0.53	0.52	0.45	0.39	0.35
16 processors	0.62	0.43	0.34	0.29	0.26

We represent the set of interprocessor distances as $\{c_i\}$, where $c_i = i$. Define the number of virtual links whose cost is not greater than c_i as N_i. Then, the expected total message traffic can be written as follows (we define N_0 as 0).

$$E[C_T] = E[\sum_{j=1}^{d} c_j \sum_{i=N_{j-1}+1}^{N_j} X_{M+1-i|M}] = \sum_{j=1}^{d} c_j \sum_{i=N_{j-1}+1}^{N_j} E[X_{M+1-i|M}]. \quad (18)$$

In the previous subsections, we obtain the equations of $E[X_{i|M}]$ in case of uniformly distributed and geometrically distributed communication requirements. We can substitute them in the above equation to compute the expected total message traffic. On the other hand, the total traffic expected from a random assignment is the product of the expected cost of a virtual link and the total number of the virtual links, which becomes

$$E_{random} = E[X_i] \sum_{j=1}^{d} c_j (N_j - N_{j-1}). \quad (19)$$

For any static network topology, we can compute the expected performance improvement by comparing equations (18) and (19). For example, we first consider two binary hypercube networks with 8 and 16 processors each. Values of c_i and $N_i - N_{i-1}$ are displayed in table 2.

Table 2: The number of virtual links $N_i - N_{i-1}$ of costs c_i in the binary hypercube networks

c_i	1	2	3	4
$N_i - N_{i-1}$ (8 proc)	12	12	4	
$N_i - N_{i-1}$ (16 proc)	32	48	32	8

The analysis on the expected performance improvement for binary hypercube networks is summarized in table 3.

Table 3: Analysis of the expected performance improvement from the optimal assignment under the assumption of geometrically and uniformly distributed communication requirements in the binary hypercube networks

parameter q	0.1	0.3	0.5	0.7	0.9	uniform
8 processors	0.40	0.40	0.36	0.32	0.29	0.21
16 processors	0.53	0.48	0.42	0.37	0.34	0.22

3 Simulation

In this section, we evaluate by simulation the expected performance improvement from an optimal processor assignment over a random assignment with two network topologies: two-level hierarchical bus networks

and binary hypercube networks. The problem of optimally assigning the processors to the nodes of these network topologies to minimize the total message traffic is NP hard. Hence, we develop a heuristic with a polynomial time bound for each network topology.

The heuristic consists of two stages: select an initial assignment and anneal the assignment by pairwise exchange of processors. This is an example of an iterative-improvement technique even though the first stage alone may work as the constructive-initial-place technique. In the first stage, the virtual links are ranked in the order of increasing communication requirements and the unassigned link of the highest rank is first assigned to an available physical link until all virtual processors are mapped to the physical processors. Once an initial assignment is selected, the algorithm converges to a local minimum by pairwise exchange of processors and stops when no more improvement can be made. To increase the performance, we repeated the pairwise exchange stages with different orders and chose the best among several local minima. The quality of the algorithm is determined by the proximity of the local minimum to the global minimum. By developing a heuristic to select a good initial assignment, we aim at both speeding up the computation and pulling the local minimum toward the global minimum. In fact, our test shows that the heuristic gives only 1% performance difference compared with the optimal algorithm when the number of processors used is 8 and the number of virtual links is 28. When the number of processors is 16, the number of virtual links becomes 120 that is too large for a naive optimal algorithm. Therefore, we approximate the optimal performance with our heuristic within a few percent of errors.

The communication requirements on the virtual links are generated based on the same distributions as we considered in the previous section. The first model M1 uses a uniform distribution on interval [0,1]. The second model M2 uses a geometric distribution with parameter q to approximate the localized communication requirements. The smaller q, the more the communication is localized. The simulations on the hierarchical bus networks and the binary hypercube networks are based on these two models.

With each model of communication requirements, we obtain simulation results on the performance improvement from an optimal assignment by the heuristic. For comparison purpose, we developed another heuristic for a sub-worst assignment, where pairs of processors with larger communication requirements are placed far apart in the physical network. Since a random assignment may result in the worst assignment, the performance comparison between an optimal assignment and a sub-worst assignment tells the most performance improvement we may expect from the optimal assignment. We repeat the simulation five times for each model and average the simulation results to reduce the standard deviation.

3.1 Hierachical Bus Network

In a hierarchical bus network, we assume that each local bus connects 4 processors, and the global bus connects either 2 or 4 local buses depending on the total number of processors in the system. The communication cost between processors on the same local bus is 1 while between processors on two different local buses, the communication cost is 3. We summarize the analysis result performed in the previous section in table 4.

Table 4: The analysis on the performance improvement from an optimal assignment in the hierarchical bus networks

No. processors	link ratio	cost ratio	uniform	geometric($q = 0.5$)
8	0.57	3	22%	45%
16	0.8	3	12%	34%

The simulation results on the performance improvement is displayed in figure 2.

In figure 2, we omit the simulation results for the uniform communication model (model M1), which

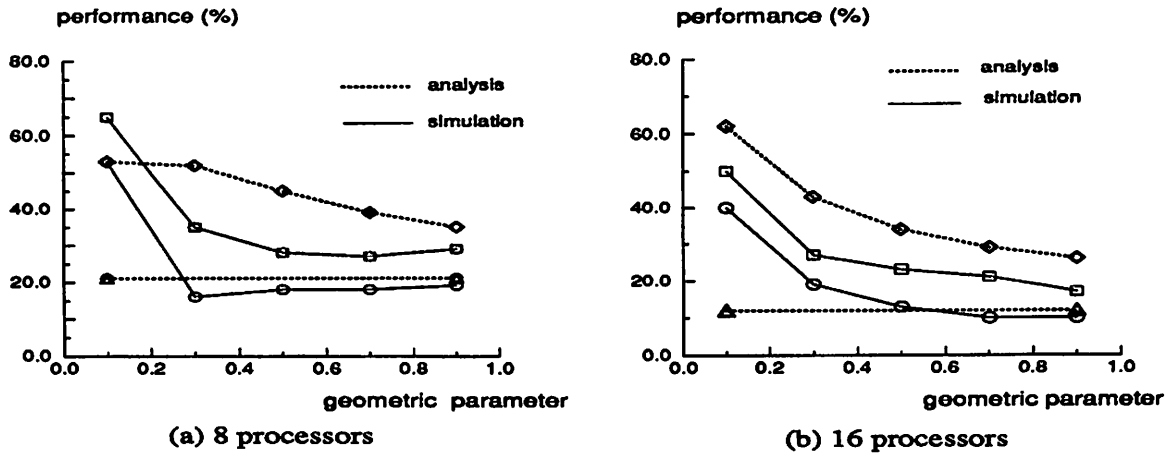

Figure 2: Simulation results on the performance improvements from an optimal assignment over a random assignment (circle line) and the sub-worst assignment (square line) in case of geometric distribution in hierarchical bus networks with (a) 8 and (b) 16 processors. For comparison, analytical results are also displayed for geometric distribution (diamond dotted line) and for uniform distribution (triangle dotted line).

proved to be about 10% improvement from an optimal assignment over a random assignment and about 20% improvement over a sub-worst assignment for 8 processor system. For 16 processor system, the results were about 10% and 15% respectively. The performance difference between the analysis results and the simulation results was 5-10%. The performance difference is also noticeable for the geometric communication model (model M2) as shown in figure 2. It is because the analysis assumes the optimistic but non-realistic assignment while the simulation is based on a sub-optimal assignment. However, it is worth noting that the 16 processor system has smaller performance improvement than the 8 processor system as expected from the analysis. The 16 processor system has a large link ratio and the performance improvement is a decreasing function on the link ratio over a certain limit as discussed in section 2.

As we expected, figure 2 shows that the more communication is localized, the more the total message traffic is reduced by an optimal assignment. In particular, the performance improvement varies rapidly in the region where the communication is highly localized within 3 processors (geometric parameter is smaller than 0.5). This is an intuitive result because a processor connects to only three processors on the same local bus. The performance difference between the analysis results and the simulation results is more distinguishable in this network than in the other network to be discussed in the following subsection. Our conjecture is that the performance difference may increase as the effect of the high cost links on the total message traffic increases.

3.2 Binary Hypercube Network

We summarize the analysis result on the 8 and 16 processor binary hypercube system performed in the previous section in table 5, and the simulation results in figure 3.

In figure 3, we omit the simulation results for the uniform communication model (model M1), which proved to be about 10% improvement from an optimal assignment over a random assignment, independently of the number of processors. As we expected from the analysis, the performance improvement on the 16 processor system is greater than than on the 8 processor system, which is contrary to the hierarchical bus

Table 5: The analysis on the performance improvement from an optimal assignment in the binary hypercube networks

No. processors	uniform	geometric($q = 0.5$)
8	21%	36%
16	22%	42%

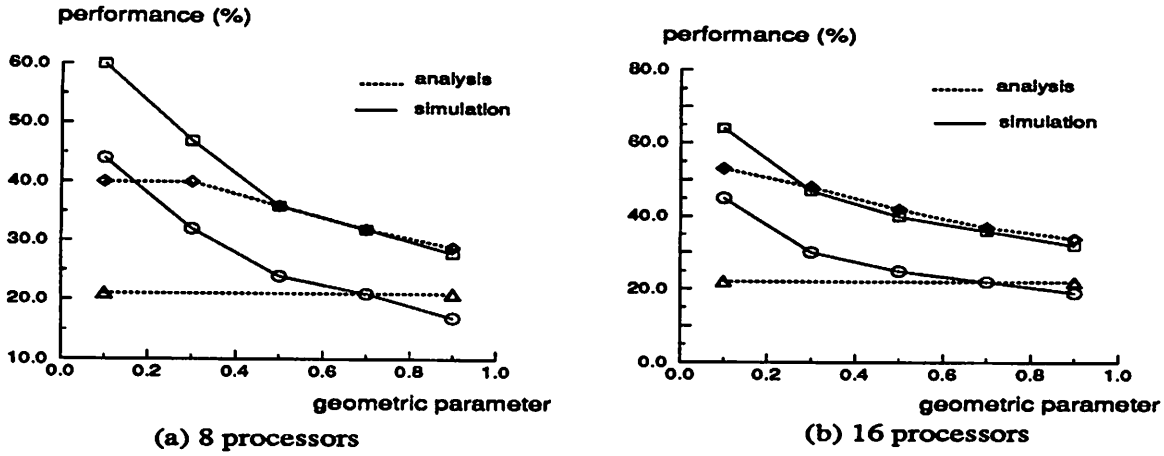

Figure 3: Simulation results on the performance improvements from an optimal assignment over a random assignment (circle line) and the sub-worst assignmen (square line) in case of geometric distribution in binary hypercube networks with (a) 8 and (b) 16 processors. For comparison, analytical results are also displayed for geometric distribution (diamond dotted line) and for uniform distribution (triangle dotted line).

networks discussed in the previous subsection.

In the localized communication model, the performance improvement from an optimal assignment is inversely proportional to the geometric parameter, ranging from 20-45% over a random assignment and from 30-60% over a sub-worst assignment. Since the performance improvement is large, we expect that the optimal assignment should be an important scheduling task in the binary hypercube networks. It is observed that simulations results based on the localized communication model lie above the analysis result based on the uniform communication model, Thus, the effect of localizing the communication requirements on the reduction of the total message traffic is greater than that of an optimal assignment. The analysis results on the localized communication model largely coincide with the simulation results of the performance improvement over a sub-worst assignment. Therefore, we roughly expect that the analysis results give the upper bound of the expected performance improvement from an optimal assignment.

4 Summary

We investigated the effectiveness of the optimal processor assignment to reduce the total message traffic in a two phase scheduling strategy. Assuming that the communication requirements between processors are uniformly distributed or geometrically distributed, we obtained analytical formulas for the optimistic performance improvement expected from an optimal assignment over a random assignment. The analysis is applicable to networks of general topology while we performed simulations with two common networks:

hierarchical bus networks and binary hypercube networks. Since the analysis allows a non-feasible assignment, it expects 10-20% more improvement than the simulation reveals. When the communication requirements are uniformly distributed, the simulation shows that the expected performance improvement from an optimal assignment over a random assignment proves to be about 10%, which is largely insensitive to the network topology. But, under the localized communication assumption, the optimal assignment gives more than 20% reduction on the total message traffic. The more communication is localized, the greater the performance improvement an optimal assignment results in. Therefore, localizing the communication requirements is an important task of the scheduling when an optimal assignment is performed.

We developed a heuristic for an optimal assignment since this problem is NP hard. Since the number of virtual links is proportional to the square of the number of processors, the algorithm should have small time complexity to be applicable to a large system. Thus, developing a fast algorithm for an optimal assignment is still an open research area. Our heuristic carefully determines an initial assignment and interchanges processors pairwise to reduce the total message traffic iteratively. When the bandwidth of the network affords the total message traffic from a random assignment, we may do without the expensive optimal assignment to save a few ten percentages of performance improvement. Therefore, to determine the validity of an optimal assignment, we should consider the expected performance improvement based on the analytical formula derived in this paper, the algorithm complexity, and the bandwidth availability.

In this paper, we did not consider the effect of link conflict, in which one message requests a communication link already taken by another message. Resolving the link conflicts is related to another scheduling problem, called the traffic scheduling problem, which is to determine the routing path of each message transfer. We rougly assume that reducing the total message traffic reduces the possibility of link congestion.

References

[1] W.B.Ackerman, "Data Flow Languages", *Computer*, Vol. 15, No. 2, pp. 15-25, Feb. 1982.

[2] R.A.Iannucci, "Toward a Dataflow/von Neumann Hybird Architecture", *Proceedings of the 15th Annual Symposium on Computer Architecture*, IEEE Computer Society, 1988

[3] S.Ha and E.A.Lee, "Compile-Time Scheduling and Assignment of Dataflow Program Graphs with Data-Dependent Iteration", *IEEE Trans. Computers*, November, 1991.

[4] K.Efe, "Heuristic Models of Task Assignment Scheduling in Distributed Systems", *Computer*, pp. 50-56, June, 1982.

[5] S.J.Kim and J.C.Browne, "A General Approach to Mapping of Parallel Computations Upon Multiprocessor Architecture", *Int. Conf. on Distributed Computing Systems*, pp. 1-8, 1988.

[6] W.W.Chu and L.M.T.Lan, "Task Allocation and Precedence Relations for Distributed Real-Time Systems", *IEEE Trans. Computers*, C-36(6), pp. 667-679, June, 1987.

[7] G.C.Sih, "Multiprocessor Scheduling to Account for Interprocessor Communications", Ph.D. Thesis, University of California, Berkeley, April, 1991.

[8] M.Hanan and J.M.Kurtzberg, "A Review of the Placement and Quadratic Assignment Problems", *SIAM Review*, 14(2), pp. 324-242, April, 1979.

[9] S.Lee and J.K.Aggarwal, "A Mapping Strategy for Parallel Processing", *IEEE Trans. Computers*, C-36(4), pp. 433-442, April, 1987.

[10] R.P.Bianchini.JR. and J.P.Shen, "Interprocessor Traffic Scheduling Algorithm for Multiple-Processor Networks", *IEEE Trans. Computers*, C-36(4), pp. 396-409, April, 1987.

[11] I.R.Nassi, "A Preliminary Report on the Ultramax: A Massively Parallel Shared Memory Multiprocessor", Technical Report ETR 87-4, Encore Computer Corporation, Fort Lauderdale, FL. 1987.

[12] H.Burkhardt, Technical Summary of KSR-1, Kendall Square Research Corporation, 170 Tracer Lane, Waltham, MA 02154, 1992.

[13] D.Lenoski et al., "The Stanford Dash Multiprocessor", *IEEE Computer*, pp. 63-79, March, 1992.

[14] H.A.David, "Order Statistics", Wiley Press, 1981

APPENDIX

In this appendix, we summarize some statistical properties on the ordered random variables that are statistically independent and uniformly distributed with distribution function $P(x)$ and probability density function $p(x)$. The detailed discussion can be found in any reference book on order statistics [14]. Let's denote $X_{r|n}$ as the r-th smallest value among n random variables. The distribution function of $X_{r|n}$, $F_r(x)$ (for $1 \leq r \leq n$), is a function of $P(x)$ as follows.

$$\begin{aligned} F_n(x) &= Pr[X_{n|n} \leq x] = Pr[all\ X_i \leq x] = P^n(x) \\ F_1(x) &= Pr[X_{1|n} \leq x] = 1 - Pr[all\ X_i > x] = 1 - (1 - P(x))^n \\ F_r(x) &= Pr[X_{r|n} \leq x] = \sum_{i=r}^{n} \binom{n}{i} P^i(x)(1 - P(x))^{n-i} = I_{P(x)}(r, n - r + 1) \end{aligned} \tag{20}$$

where $I_p(a, b)$ is an incomplete beta function defined as

$$I_p(a, b) = \frac{\int_0^p t^{a-1}(1-t)^{b-1}\,dt}{B(a, b)} = \frac{\int_0^p t^{a-1}(1-t)^{b-1}\,dt}{\int_0^1 t^{a-1}(1-t)^{b-1}\,dt} \tag{21}$$

Here, $B(a, b)$ is a beta function defined as follows:

$$B(a, b) = \frac{(a-1)!(b-1)!}{(a+b-1)!} = \int_0^1 t^{a-1}(1-t)^{b-1}\,dt \tag{22}$$

If the variable $X_{r|n}$ is continuous, the probability density function $f_r(x)$ is the derivative of the distribution function.

$$f_r(x) = \frac{1}{B(r, n-r+1)} P^{r-1}(x)(1 - P(x))^{n-r} p(x) \tag{23}$$

On the other hand, if $X_{r|n}$ is a discrete random variable, the probability density function $f_r(x)$ is given as follows.

$$f_r(x) = F_r(x) - F_r(x-1) = (1 - F_r(x-1)) - (1 - F_r(x)) \tag{24}$$