Performability Studies of Hypercube Architectures

Samir M. Koriem* **L. M. Patnaik****

** Microprocessor Applications Laboratory
* Department of Computer Science and Automation
Indian Institute of Science, Bangalore - 560 012, INDIA

Abstract

We propose a novel technique to study composite reliability and performance (Performability) measures of hypercube systems using Generalized Stochastic Petri Nets (*GSPNs*). This technique essentially consists of the following : (i) a *GSPN* reliability model; (ii) a *GSPN* performance model; and (iii) a way of combining the results from these two models. Models and performability results for an iPSC/2 hypercube system under the workload of concurrent matrix multiplication algorithm are presented.

1 Introduction

To study gracefully degradable systems, researchers have developed a conceptual framework of *performability* [12] which combines the analysis of performance and reliability/availability. The performability analysis based on *Queuing Network* (QN) is presented in [14]. Unfortunately, the use of QN requires a good background in applied probability theory. *Markov Chain* (MC) models [3,16] are similar to QN models. For practical problems, MC models suffer from an exponential state-space explosion.

In order to provide a simple and easily usable method, in this paper, we propose a technique to develop the *performability* of *parallel* and *distributed* architectures using the *Generalized Stochastic Petri Nets* (GSPNs) tool [11]. The derivation of the reachability graph of the GSPN model and the equivalent Markov chain, the computations of the steady state or transient state probabilities, can be automated and incorporated into *software packages* such as SPNP [4]. The GSPN is very effective for modeling the synchronization, contention and the parallelism of a system. However, GSPN analysis in terms of QN or MC tend to become very complex. Furthermore, incremental changes to a GSPN model can be done easily, while minor changes in QN or MC model, in most cases, require alterations to the entire model.

The available literature in the filed of performability analysis focusses mainly on the study of abstract parallel or distributed systems under very general workloads. To remove this lacuna and to obtain realistic performability measures, we focus our attention on the problem of modeling and evaluation of *iPSC/2 hypercube system* [2,6] under specific workloads, such as a *matrix multiplication algorithm* [5]. In the rest of the paper, we concentrate on the issues of the performance, reliability and performability modeling and analyzing the iPSC/2 hypercube system.

2 Strategy for Modeling Performability

The *strategy* for using the GSPN technique to design the *performability* model of the desired system consists of the following steps.

1. Choose the desired metrics that combine aspects of performance and reliability requirements.

2. Construct performance model for a system using the GSPN technique. This model represents the probabilistic nature of user demands (workload) and the variations in the internal states of the system.

3. Solve the performance model of the system for the steady state analysis to obtain the desired metrics.

4. Construct and analyze the GSPN reliability model of a system. This model represents a changes of the structure of the system due to occurrence of faults and repairs.

5. Formulate a performability model and evaluate the performability metrics of a system. This step is discussed in more detail in section 6.

3 A Commercial Hypercube

In this section, we study a brief summary of the structural and topological properties of the iPSC/2 hypercube [2,6] that are relevant to our analysis.

3.1 iPSC/2 Architecture

To reduce the software and hardware overheads of communications in an iPSC/2 hypercube [2], Hsu [6] designed two hardware devices: a *Message Passing Coprocessor* (MPC) and a *Virtual Channel Router* (VCR). The MPC and VCR are compatible with iPSC/2. The MPC is a microprogrammable processor unit with its own local memory. The PE issues commands to the MPC and reads values returned by it through I/O read/write. The MPC requests PE service using interrupts. The MPC carries out process scheduling and message buffer management for the PE, and performs fast buffer copying. Thus, the MPC saves some software processing time of the PE. The VCR is responsible for the transmission of messages in the hypercube system. From the point of view of the system bus, the VCR is an I/O device and also a bus master which performs message sending and receiving under the command of the MPC. When contention occurs, the VCR supports circuit-switched routing with blocking, which is also known as *Wormhole routing* [6].

0-8186-2672-0/92 $03.00 © 1992 IEEE

3.2 Communication Latency

The different communication latencies of iPSC/2 [2] and iPSC/2 with the MPC and VCR [6] for different message sizes can be analyzed as follows. Let $T_{comm(k)nn}$ be the time needed to transfer a message of size k *elements (words)* between two adjacent nodes, where each element (word) is equal to 4-byte.

The $T_{comm(k)nn}$ is defined as,

$$T_{comm(k)nn} = t_{setup_{nn}} + k * t_{send_{nn}} \quad (1)$$

Here, $t_{setup_{nn}}$ is the node-to-node communication startup delay time and $t_{send_{nn}}$ is the node-to-node transmission time, for a 4-byte element over the communication channel.

- For *short messages* $\leq$ 100 bytes,

In iPSC/2 with MPC and VCR :

$$T_{comm(k)nn} = 55.2\,\mu sec + k * 0.37\,\mu sec/word \quad (2)$$

In iPSC/2 :

$$T_{comm(k)nn} = 350\,\mu sec + k * 0.8\,\mu sec/word. \quad (3)$$

- For *long messages* $>$ 100 bytes,

In iPSC/2 with MPC and VCR :

$$T_{comm(k)nn} = 400\,\mu sec + k * 0.43\,\mu sec/word \quad (4)$$

In iPSC/2 :

$$T_{comm(k)nn} = 660\,\mu sec + k * 1.14\,\mu sec/word. \quad (5)$$

4 Performance Model

Our basic objective in this section is to develop a realistic performance metric appropriate for performability analysis. To obtain an accurate evaluation, we build our GSPN performance models based on the benchmarking results of the iPSC/2 hypercube presented in [2,6]. Also, for higher accuracy, we take into account the different communication methodologies used in hypercube.

4.1 Analysis and Study of Hypercube Communication Methods

In building the GSPN performance models of iPSC/2 hypercube, we are concerned with the communication characteristics given below.

1. The time to transmit a message between two *neighboring nodes* is as presented in equations 2 and 4.

2. From the algorithm designer's point of view, in our GSPN models of iPSC/2 with VCR and MPC, we will study the *host-to-node* (or *node-to-host*) *communications* using the following approach.

In this approach, we consider that the host (in a d-cube of size $N = 2^d$ nodes) communicates with node zero and node zero communicate with all nodes of the hypercube system. The host in this configuration partitions a given problem into N *subproblems (tasks)* and broadcasts them to node zero. Then, node zero broadcasts these subproblems to all other nodes. We have calculated these broadcast processes based on the algorithms proposed in [7].

The communication time from the host to all the nodes is calculated using the two steps given below.

Step 1. Let $T_{comm(k)hn0}$ be the communication time required to send N subproblems from the host to node zero. This communication time is given by

$$T_{comm(k)hn0} = t_{setuphn} + (k * t_{sendhn}) * N \quad (6)$$

Here, $t_{setuphn}$ is a host-to-node (or node-to-host) communication startup delay time and t_{sendhn} is a host-to-node (or node-to-host) transmission time. Typical values of these parameters are $t_{setuphn} = 1700\,\mu sec$ and $t_{sendhn} = 2.83\,\mu sec/word$ [2].

Step 2. Let $T_{comm(k)n0n}$ be the communication time required to broadcast N subproblems from node zero to N nodes. This communication time is given by

$$T_{comm(k)n0n} = d * t_{setupnn} + ((N-1)/d) * k * t_{sendnn} \quad (7)$$

The values of $t_{setupnn}$ and t_{sendnn} are the same as those in the equation of step1.

The total communication time of the above two steps is given by

$$T_{comm(k)hn} = T_{comm(k)hn0} + T_{comm(k)n0n} \quad (8)$$

Once the nodes finish their calculations, they send their results to node zero. Node zero in turn sends all the results to the host. These two steps of communication take time $T_{comm(k)nh}\,\mu sec$. The $T_{comm(k)nh}$ is calculated in the same way as given in equation 8.

4.2 Modeling Matrix Multiplication Algorithm

4.2.1 Matrix Multiplication Algorithm

The mesh topology is sufficient [5] for an efficient implementation of matrix multiplication. However, we are interested in embedding an $N_1 \times N_2$ (or $2^{b1} \times 2^{b2}$) two dimensional mesh on a d-cube of dimension $d = log_2 N_1 + log_2 N_2$ (or $d = b_1 + b_2$) [8]. We found that [8] gives powerful homomorphic embedding into a hypercube when $b_1 = b_2 = b$. In our analysis of this matrix algorithm, we map a 4 x 4 (or $2^2 \times 2^2$) mesh into a 4-cube. Also, a 6-cube is arranged as an 8 x 8 (or $2^3 \times 2^3$) mesh. We are interested in performing the multiplication $C = A * B$ where C, A, and B are full $M \times M$ matrices.

In this algorithm, the matrices are distributed to a $\sqrt{N} \times \sqrt{N}$ mesh which is embedded in a d-cube. The matrix C is divided into square submatrices C_{ij} of size $m \times m$, where $m = M/\sqrt{N}$. Each submatrix is assigned to one of the nodes. Matrices A and B are split up in the same way. In order to compute C, the following calculations are performed.

$$C_{ij} = \sum_x A_{ix} * B_{xj} \qquad ; i \geq 0,\ j < \sqrt{N}$$

The algorithm for this computation is outlined below.

1. The host computer sends to each node of the d-cube a distinct square submatrix of $m \times m$ elements. For a message passing process of this algorithm, a message length k equals to m^2 elements.
   $Cost : T_{comm(k)hn}$

2. For $Z = 1$ to $\sqrt{N}$ do steps 3, 4, 5.

3. If $Z = 1$, then broadcast the diagonal submatrices of A in a horizontal direction.
   Otherwise, do a horizontal broadcast of the diagonal+1 submatrices of A. As we have explained earlier, the broadcast algorithm takes b steps to send the data to the desired nodes. These broadcast steps are completed in $T_{broadcast(k)}\,\mu sec$.
   $Cost : T_{broadcast(k)} = b * T_{comm(k)nn}$

4. Multiply the copied A submatrices by the B submatrices currently residing in each node. This submatrix product (C_{ij}) is performed in $T_{comput}\,\mu sec$ using a sequential algorithm.
   $Cost : T_{comput} = m^3\,ta + m^3\,tm$
   where, ta: addition time; tm: multiplication time; and m^3: number of addition or multiplication steps. Typical

values of these parameters, for integer elements, are $ta = 1.37\ \mu sec$ and $tm = 1.69\ \mu sec$ [2].

5. If $Z = \sqrt{N}$ do step 6
   Else, roll the B submatrices vertically.
   $Cost : T_{rollB(k)} = T_{comm(k)nn}$
6. Each d-cube node sends its subsequence results to the host.
   $Cost : T_{comm(k)nh}$

The broadcast-multiply cycle of the algorithm is repeated $\sqrt{N}$ times and the roll cycle $(\sqrt{N}-1)$ times. The computation time of the inner matrix multiplication algorithm (without $T_{comm(k)nh}$ and $T_{comm(k)hn}$) is given by :

$$T_{inner} = \sqrt{N}\,(T_{broadcast(k)} + T_{comput}) + (\sqrt{N}-1)\,T_{rollB(k)}$$

We wish to point out that the calculation of this time is different from the time calculated in [5].
The total computation time of the matrix multiplication algorithm is given by

$$T_{matix} = T_{comm(k)hn} + T_{inner} + T_{comm(k)nh} \qquad (9)$$

Note that $T_{comm(k)hn}$ and $T_{comm(k)nh}$ are not considered in [5].
Equation 9 represents the mathematical model of this algorithm. We use this equation to verify the correctness of the GSPN matrix multiplication model.

4.2.2 GSPN Matrix Multiplication Model

Using the matrix multiplication algorithm described in the previous section, the behavior of the iPSC/2 hypercube that embeds 4 X 4 or 8 X 8 mesh can be represented by the GSPN model shown in Figure 1. The initial marking of the model contains a number of tokens in places p_2 and p_4. The number of tokens in p_2 is equal to N which represents the number of submatrices (i.e. subproblems or tasks) that the host has allocated to the nodes. The number of hypercube nodes, $N = 2^d$, is represented by the tokens in p_4.

The random exponential distribution times of the timed transistions of the GSPN matrix model are calculated based on our following usage :
(i) the computation and communication times of matrix algorithm, such as
$t_2 = T_{comm(k)hn}$, $t_6 = m^3 ta$,
$t_7 = m^3 tm$ and $t_{22} = T_{comm(k)nh}$; and
(ii) the actual benchmarking results that represent the timing of short/long message transmission and the different commands in iPSC/2 incorporating MPC and VCR components [6]. These timings are represented by the transitions from t_{10} to t_{13} and from t_{15} to t_{20}. A detailed explanation of handling specific problem using the GSPN matrix multiplication model, the meanings of various places and transitions, as well as the detailed calculations of the times of the timed transitions can be found in [10].

4.2.3 Performance Evaluation

From the GSPN model of Figure 1, the equivalent MC is automatically generated and solved to yield the steady-state probabilities of the model. The following performance measures can be obtained from Figure 1.

1. *Mean time to Complete the Program* (T_{MP})

From Figure 1, T_{MP} is equal to the average time required by N tokens starting in places p_2 and p_4 to return to places p_2 and p_4. The value of T_{MP} can be calculated, based on the Renewal theory given in [15], as follows.

$$T_{MP} = \delta(M_0) + \tau(M_0) \qquad (10)$$

Here, $\delta(M_0)$ is the average return time to initial marking M_0 and $\tau(M_0)$ is the average sojourn time in M_0.
The $\tau(M_0)$ can be computed as follows

$$\tau(M_0) = 1/w_2 \qquad (11)$$

where w_2 is the firing rate of transition t_2.
We calculate $\delta(M_0)$ using the definition of T_{MP} as follows. The ratio of the average time that the tokens in both p_2 and p_4 spend to that in the rest of the places equals the ratio of the steady-state probality that each p_2 and p_4 has N tokens to the steady-state probality that each p_2 and p_4 has no tokens. Then,

$$\delta(M_0) = \alpha/\beta \qquad (12)$$

where,
$\alpha = 1 - (SSP_{rob}\{\,(m(p_2) = N) + (m(p_4) = N)\,\})$
$\beta = w_2 * SSP_{rob}\{\,(m(p_2) = N) + (m(p_4) = N)\,\}$
$SSP_{rob}\{\,m(p_i) = N\,\}$ is the steady-state probability that place p_i has N tokens.

2. *Speedup* (S_{up})

$$S_{up} = T_{MP}/T_{seql} \qquad (13)$$

Here, T_{seql} denotes the execution time of the sequential algorithm, i.e. quicksort algorithm.

3. *Effective Processor Node Utilization* (E_f)

$$E_f = S_{up}/N \qquad (14)$$

4.2.4 Performance Results

In Figure 2, We have plotted the T_{MP} versus matrix size M X M for the hypercube that is organised as square meshes of dimensions 4 X 4 and 8 X 8 respectively. As we expected, the T_{MP} for a 64 node hypercube is much smaller than that of a 16 node hypercube for large matrix sizes. In Figure 2, the solid plots are for the GSPN model results and the dotted plots are for the mathematical model results of equation 9. It can be observed that the GSPN model results are in close agreement with those from mathematical model for different matrix sizes. The other performance indices of this algorithm are easily derived from T_{MP}.

5 Reliability Model

Three types of hypercube reliability models have been reported in the literature [1,9,13]. The first model reported in [9] uses a combinatorial enumeration technique. The other models: disconnected reliability model [13] and subcube reliability model (*SRM*) [1], are developed based on the Markov Chain (MC) analysis. Our study is concerned only with the SRM.

The novelty of our approach is that we develop a simple GSPN reliability model which gives the same results of the SRM. The main advantage of our GSPN model over SRM is that there is no need to manually enumerate all the possible states since they are automatically generated from the GSPN model. Another advantage of our approach is that we do not require a closed-form solution unlike [1].

5.1 Subcube Reliability Approach

Abraham *et al.* [1] have proposed an analytical model, based on MCs, for the subcube reliability analysis associated with the hypercube architecture. We briefly review the states of the damaged cube under a node failure model. The MC of the node failure model is depicted in Figure 3. The states of this model can be defined as follows. The state S_* represents the initial fault-free state of the d-cube. The state S_i indicates the system state in which all node failures in the cube are contained in an i-subcube, but not in an (i-1)-subcube, for $0 \leq i < d$. In fact, state S_i can be characterized as embedding exactly d-1 disjoint subcubes of order d-1, d-2,..., i in the system. The state S_d denotes that there are no embedding of a (d-1)-subcube is possible. All nodes are identical with exponential distribution of failure times. The failure rate is given by λ. The state transition rate from state S_i to state S_{i+j} $(0 < j \leq d-i)$ is given by the following formula [1]:

$$\lambda 2^i \binom{d-i}{j} \quad ;0 < j \leq d-i \;, 0 \leq i < d \qquad (15)$$

The analytic solution to study the reliability model of node failures of Figure 3 is presented in [1] as a closed-form expression.

5.1.1 GSPN Subcube Reliability Model

In this section, we use GSPN technique to analyze the reliability aspects of the subcube reliability approach which was discussed in the previous section. This approach is explained in detail through the GSPN model shown in Figure 4. The places and transitions of Figure 4 are described below for 4-cube, 5-cube, and 6-cube topologies:

• *Places*

p_1: number of hypercube nodes; p_2: number of subcubes; p_3, p_4: 0 - subcube fails; p_6: 1 - subcube fails; p_7: 2 - subcube fails; p_8: 3 - subcube fails; p_9: 4 - subcube fails; p_{10}: 5 - subcube fails; p_5: whole systems fails.

• *Transitions*

t_{22}, t_{29}, t_{33}, t_{34}, t_{35}, t_{37}, t_{38}, t_{39}: repair of node(s).

t_7 - t_{15}: If there is no way to embed a (d-1)-subcube, the whole system fails.

The other transitions: one of the subcubes in the system has failed.

• The interpretation of the weight on the arcs are as follows.

$N-1=n_1$, $N-2=n_2$, $N-3=n_3$, $N-4=n_4$, $N-5=n_5$, $N-6=n_6$. $d-1=d_1$, $d-2=d_2$, $d-3=d_3$, $d-4=d_4$, and $d-5=d_5$.

The firing rates of the timed transitions that represent hypercube node failures are calculated as follows.

Equation 15 is used to calculate the firing rate for each transition. In our GSPN model, the values for the parameters i and j of this equation are obtained from the marking of the places attached with each transition. In this model, incorporation of repairs (by using repair transitions) is easy unlike [1]. We consider on-line repair of nodes with exponential distribution of repair times given by $1/\mu$.

The detailed explanation [10] of the reliability model of Figure 4 is omitted for conciseness. The interested readers can obtain the details illustrated with examples from authors.

As shown in Figure 4, the GSPN subcube reliability model is complicated but it does provide all the details that explain subcube failures in 4-cube, 5-cube, and 6-cube topologies. To extend this GSPN model for higher dimensional cubes, we add a few more places and transitions to the model.

The GSPN model of Figure 4 is refined to obtain the GSPN subcube reliability model of Figure 5. The model of Figure 5 has the following important advantages over the model of Figure 4:

(i) simple : The model has three places and two transitions; and

(ii) easy to use : Obtains the reliability results for d-cube without adding places or transitions to the model.

The GSPN model of Figure 5 is described below.

• *Places*

p_1: number of nodes

p_2: number of subcubes in the d-cube

p_3: number of node(s) awaiting repair

p_4: node(s) whose failure causes the whole system to fail

• *Transitions*

t_1: one subcube in the hypercube has failed

t_2: if there is no way to embed a (d-1)-subcube, the whole hypercube fails

t_3: repair of node(s).

Note that all the arcs attached to t_2 are *variable arcs* [4].

In the GSPN model of Figure 5, the system is in a fault-free state when place p_1 holds N tokens and p_2 holds d tokens. The firing rates of transitions t_1 and t_2 are obtained using the equation 15 as follows.

In equation 15 presence of 2^i indicates that a node has failed in the i-subcube and that the number of hypercube partitions is 2^i. Each hypercube partition is a $(d-i)$-cube and there are $\binom{d-i}{j}$ number of such node failures in each partition [1]. Based on this model, we consider the firing rates of transitions $t_1 = \lambda\, 2^i\, (d-1)\, C_0$ and $t_2 = \lambda\, 2^i\, (1-C_0)$.

Here, C_0 is the fault coverage factor [16] for the recovery scheme. Coverage is included in our analysis to characterize the ability of the hypercube system to detect, reconfigure and successfuly recover from faults occurring during normal system operations.

It is interesting to note that the formula of the firing rate of t_1 gives exactly the same result as that obtained from equation 15 for any state transition rate from state S_i to state S_d. Also, from the formula of t_2 , one can obtain the same result using equation 15 for any state transition rate from S_i to $S_i + 1$.

5.1.2 Reliability Results

The comparison between reliability results of SRM_1 for Figure 4 and SRM_2 for Figure 5 are shown in Table 1. The results of Table 1 is based on allowing repair for covered failures and no repair for uncovered failures.

It is interesting to observe that the results of SRM_2 are approximatly the same as those of SRM_1, which establishes the correctness of the refined model. Also, our GSPN reliability models allow us to include coverage and repair strategies in the analysis; these are not considered in [1]. Furthermore, from the GSPN reliability studies, one can obtain the subcube reliability results without any need to get closed formula unlike [1].

6 Performability Model

The reliability model discussed in section 5 is not sufficient to evaluate the hypercube system, since such model gives only the probability that the system is operational at time t but do not reveal the performance degradation due to failures occurring in $(0,t)$. Therefore the performability model is studied in this section to model performance degradation occurring due to node failures over the interval $(0,t)$.

The *performability model* of the *hypercube system* is constructed as follows.

• The changes in the structural state of the hypercube system caused by different events (failures/repairs) are described by the GSPN subcube reliability model of Figure 5.

The results of this GSPN model are represented by a discrete state, continuous time Markov chain (CTMC), { Z(t), t $\geq$ 0 }, with state space $\Omega = \{1, 2, ..., n\}$ (see Figure 3).

• Associated with each operational state of the CTMC of the GSPN subcube model is a *performance level*, r_i, (*reward rate*). The reward (benifit) rate, r_i, [12] indicates how much work the system completes per unit time when the system is in state i. The performance levels add a reward structure to the configuration state process $Z(t)$. The reward rate we consider here is the *Mean Time to Complete the Program*, T_{MP}, of matrix multiplication algorithm.

The CTMC of the reliability model and the reward rate, r_i, of the performance model are combined to form a Markov Reward Model (MRM). The MRM is the base model for the performability model.

6.1 Performability Measures

In this section, based on the MRM, we define the following combined measures [3,16] of performance and reliability.

The *Reward Rate*, $X(t)$, of the system at time t can be express as

$$X(t) = r_{Z(t)} \tag{16}$$

The *Expected Reward Rate*, $E[X(t)]$, at time t is given by

$$E[X(t)] = \sum_i r_i p_i(t) \tag{17}$$

where $p_i(t) = P_{rob}[Z(t) = i]$ is the probability that the system is in state i at time t. The $E[X(t)]$ measure answers the question: what is the expected performance of the system at time t ?.

The *Accumulated Reward*, $Y(t)$, is the amount of work completed by a system during the interval $(0, t)$. Also, $Y(t)$, is defined as the system *performability* at time t which is given by

$$Y(t) = \int_0^t X(u)du \tag{18}$$

The *Expected Accumulated Reward*, $E[Y(t)]$, in the interval $(0, t)$ is given by

$$E[Y(t)] = E[\int_0^t X(u)du] = \sum_i r_i \int_0^t p_i(u)du \tag{19}$$

The *Expected Time Averaged Accumulated Reward* , $E[W(t)]$, can be express as

$$E[W(t)] = E[Y(t)/t] = \frac{1}{t}\sum_i r_i \int_0^t p_i(u)du \tag{20}$$

The $E[W(t)]$ accumulated measure answers the question: what is the time averaged performance of the system over the interval $(0, t)$?.

The *Expected Uptime*, $E[U(t)]$, is the mean time spent by the system in operational states in the interval $(0, t)$.

$$E[U(t)] = E[Y(t)]$$

with $r_i = 1$, $i \in \Omega_0$ and $r_i = 0$, $i \in \Omega_f$. Subsequently,

$$E[U(t)] = \sum_{i \in \Omega_0} \int_0^t p_i(u)du \tag{21}$$

The *Expected Interval Availability*, $E[A_I(t)]$, is a cumulative measure that depends on the cumulative amount of time spent by the system in operational states during the interval $(0, t)$. The $E[A_I(t)]$ can be computed as

$$E[A_I(t)] = E[U(t)/t] = \frac{1}{t}\sum_{i \in \Omega_0} \int_0^t p_i(u)du \tag{22}$$

Also, $E[A_I(t)]$ can be obtained by another way as follows. $E[A_I(t)] = E[W(t)]$ when $r_i = 1$, $i \in \Omega_0$ (operational states) and $r_i = 0$, $i \in \Omega_f$ (failed states).

6.2 Performability Results

To obtain the hypercube performability measures, we use the performability metrics that were explained in the previous section. The effect of modeling failure and repair on $E[X(t)]$ is explained in Figure 6 and the effect on $E[W(t)]$ is illustrated in Figure 7. In Figures 6 (the expected amount of computation avaliable on the hypercube system at time t) and 7 (the average performance of the system over the interval $(0, t)$), we observe the following. Plot I represents the hypercube system without failure and repair (pure performance). The results of this modeling assumption are that no degradation of performance takes place so the hypercube performance (pure) is independent of time. In plot VIII the system has no repair but plot VII indicates that the system has no repair and $C_0 = 0.9$. These plots reflect the effect of coverage factor on $E[X(t)]$ and $E[W(t)]$. The expected performance level (or the expected time-averaged accumulated reward) of plot VIII deteriorates more rapidly than that of plot VII.

From plot VII, we observed that the expected performance level (reward rate) gets halved at time $t = 2400$. At time 2400 in Figure 7, the expected time-average accumulated reward decreases by only one quarter because $E[W(t)]$ is the time average of $E[X(t)]$ over $(0, t)$. Plots VI and IV ($\mu = 0.1$ and 2.0 respectively) show that the higher repair plots dominate the lower repair plots illustrating the effect of repair strategy on $E[X(t)]$ and $E[W(t)]$. The expected performance level (or the expected time-averaged accumulated reward) of plot III ($C_0 = 0.9$ and $\mu = 2.0$) shows greater improvement compared to plot V ($C_0 = 0.9$ and $\mu = 0.1$). Plot II reflects the increase of coverage value to 0.999999 (with $\mu = 0.1$) on the $E[X(t)]$ and $E[W(t)]$. This plot clearly show that $X(t)$ (or $W(t)$) increases with the increase in the value of C_0. However, it can be observed that both plots I and II are almost identical. Actually, C_0 reflects how well the redundancy schemes and recovery strategies have been implemented by the system designer.

In Figure 8, the behavior of $E[A_I(t)]$ is described for different C_0 and μ values. This cumulative measure indicates the expected amount of time spent by the system in the operational states during the interval $(0, t)$. Figure 8 shows the following plots : I($C_0 = 0.999999$ and $\mu = 0.1$), II($\mu = 2.0$), III($\mu = 0.1$), IV($C_0 = 0.9$), and V(no μ, or C_0). As expected, the results obtained from Figure 8 are similar to the results depicted by the plots of $E[W(t)]$, because $A_I(t)$ is a special case of $W(t)$ as mentioned in the previous section.

The performability measures $E[Y(t)]$ and $E[U(t)]$ are compared for different values of the parameters C_0 and μ and are shown in Figure 9. The plots of Figure 9 show the plots I(no μ, or C_0), II($C_0 = 0.9$), III($C_0 = 0.9$ and $\mu = 0.1$), IV($\mu = 2.0$), and V($C_0 = 0.9$ and $\mu = 2.0$). These plots clearly demonstrate that $E[Y(t)]$ is always greater than $E[U(t)]$. Actually, $E[Y(t)]$ represents the total work done by the system. Further, the operational states of the system are assumed to produce work represented by a reward rate greater than unity. Note

that $E[U(t)]$ is a special case of $E[Y(t)]$ as pointed out in the previous section.

7 Sensitivity Analysis

The results of the performability measures we have already obtained from a GSPN model of the hypercube system, are sensitive to many factors (e.g. failures rates, repair rates, and coverage probabilities). In this section, we concentrate our attention on *sensitivity analysis* [3] that allows us to compute the effect of changes in the various rate constants of a Markov model (created from GSPN hypercube model) on the measures of interest. Mathematical details of sensitivity analysis using uniformization technique are given in [3].

The sensitivity results of $R(t)$, $E[X(t)]$, $E[Y(t)]$ with respect to λ, μ, C_0, parameters of the hypercube system are given in Table 2. The results of Table 2 can be summarized as follows. The negative (*positive*) values of this table indicate that a fractional increase in the parameter value(s) (μ and C_0) λ would decrease (*increase*) the $R(t)$, $E[X(t)]$ and $E[Y(t)]$ of the hypercube system. However, to more accurately model the reliability or performability of the hypercube system, the failure rate, repair rate and coverage factor should be determined with greater accuracy.

From the above analysis it is observed that the parametric sensitivity analysis provides guidance for refining the model and also for improving the accuracy of the model.

8 Conclusions

We have proposed a novel approach based on GSPN technique to model and analyze the reliability and its effects on the performance of hypercube systems. In the GSPN performance model, evaluation of important aspects of hypercube such as concurrency, message passing mechanism between the host and the nodes as well as among the nodes, have been modeled in a natural way. The capability to model such diverse aspects of system behavior is illustrated through a realistic parallel algorithm running on the GSPN model of iPSC/2 hypercube incorporating MPC and VCR components. The representative algorithm for this study is chosen to be matrix multiplication. We have built mathematical model for this algorithm to verify the correctness of the results with those obtained from the GSPN performance models. To obtain an accurate performance evaluation, we have built our GSPN performance model based on the actual benchmark parameters of iPSC/2 hypercube.

In the reliability analysis, we have developed a simple GSPN model giving the same feature of subcube reliability approach in an easy way. To obtain more realistic reliability results, we include in the GSPN reliability models a coverage and a repair strategies, which are not considered in the subcube reliability approach.

In order to properly assess the effectiveness of hypercube systems, performability model that combine reliability model and performance metric is studied. The performability model is built by evaluating the performance metric, such as the mean time to complete the program, for each state of reliability model. The performability model is solved for various instantaneous and cumulative measures. Furthermore, our study indicates how changes in the failure/repair behavior of the hypercube system such as the node failure rate, repair strategy, and coverage probability can affect the reliability and cumulative measures. These sensitivity analyses are useful to refine the portions of the reliability and performability model that are particularly sensitive to error.

References

[1] S. Abraham and K. Padmanabhan, "Reliability of the Hypercube", Int. Conf. on Parallel Processing, Vol.1, Aug.1988, pp.90-94.

[2] L.Bomans and D.Roose, "Benchmarking the iPSC/2 Hypercube Multiprocessor", Concurrency: Practice and Experience, Vol.1, No.1, pp.3-18, Sept.1989.

[3] J. T. Blake, A. L. Reibman and K. S. Trivedi, "Sensitivity Analysis of Reliability And Performability Measures For Multiprocessor Systems", ACM/SIGMETRICS, Conf. on Measurement and Modeling of Computer Systems, 1988.

[4] G.Ciardo, J.Muppala and K.Trivedi, "SPNP: Stochastic Petri Net Package", International Conference on Petri Nets and Performance Models, Kyoto, Japan, Dec. 1989.

[5] G.Fox and S.Otto, "Matrix Algorithms on a Hypercube I : Matrix Multiplication", Parallel Computing, Vol.4, pp.17-31, 1987.

[6] Jiun-Ming Hsu "Performance Measurement and Hardware Support for Message Passing in Distributed Memory Multicomputers", Ph.D. Thesis, UILU-ENG-91-2209,CRHC-91-5, Univ. of Illinois at Urbana - Champaign, 1991.

[7] S.Johnson and C-T Ho "Optimum Broadcasting and Personalized Communication in Hypercubes" IEEE Trans. on Computers, Vol.38, No.9, pp.1249-1268, Sept. 1989.

[8] S.L.Johnson, "Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures", Journal of Parallel and Distributed Computing, Vol.4, pp.133-172, 1987.

[9] J.Kim, C.Das, W.Lin, and T.Feng, " Reliability Evaluation of Hypercube Multicomputers", IEEE Trans. on Reliability, Vol.38, No.1, pp.121-129, April 1989.

[10] S. M. Koriem, "Reliability and Performability Studies of Distributed Memory Architectures", Ph.D. Thesis, CSA Dept., Bangalore, India. To be published.

[11] M.A.Marson, G.Balbo and G.Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", ACM Trans. on Computer Systems, Vol.2, No.2, pp.93- 122, May 1984.

[12] J.F.Meyer, "Closed-Form Solutions of Performability", IEEE Trans. on Computers, Vol.31, No.7, pp.648-657, July 1982.

[13] W.Najjar and J.Gaudiot, "Reliability and Performance Modeling of Hypercube-Based Multiprocessors", Proc. of the 2nd Int. MCPR Workshop, Rome, Italy, May 1987, pp.305-320.

[14] V.Nicola, V.Kulkarni and K.Trivedi, "Queuing Analysis of Fault-Tolerant Computer Systems", IEEE Trans. on Software Engineering, Vol.13, No.3, pp.363-375, March 1987.

[15] S.M.Ross, "Stochastic Processes", Wiley, New York, 1983.

[16] R.Smith, K.Trivedi, and A.Ramesh, "Performability Analysis : Measures, an Algorithm, and a Case Study", IEEE Trans. on Computer, Vol.37, No.4, pp.406-417, April 1988.

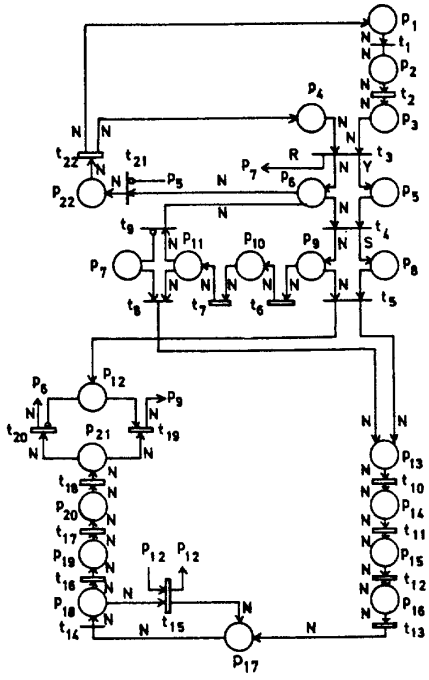

Figure 1. The GSPN Model for Concurrent Matrix Multiplication Algorithm.

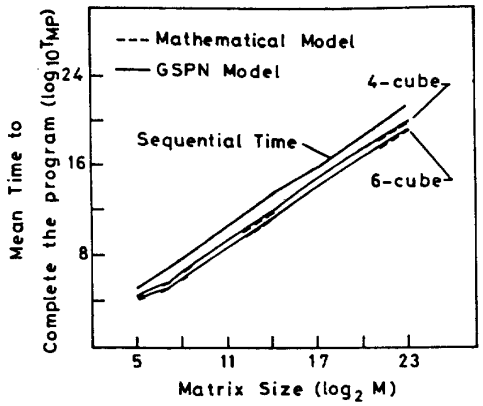

Figure 2. The T_{MP} of Matrix Multiplication Algorithm for 4-cube, 6-cube and Different Matrix Sizes.

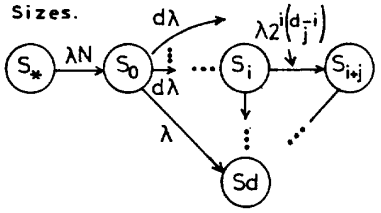

Figure 3. State Transition under Nude Failure Model.

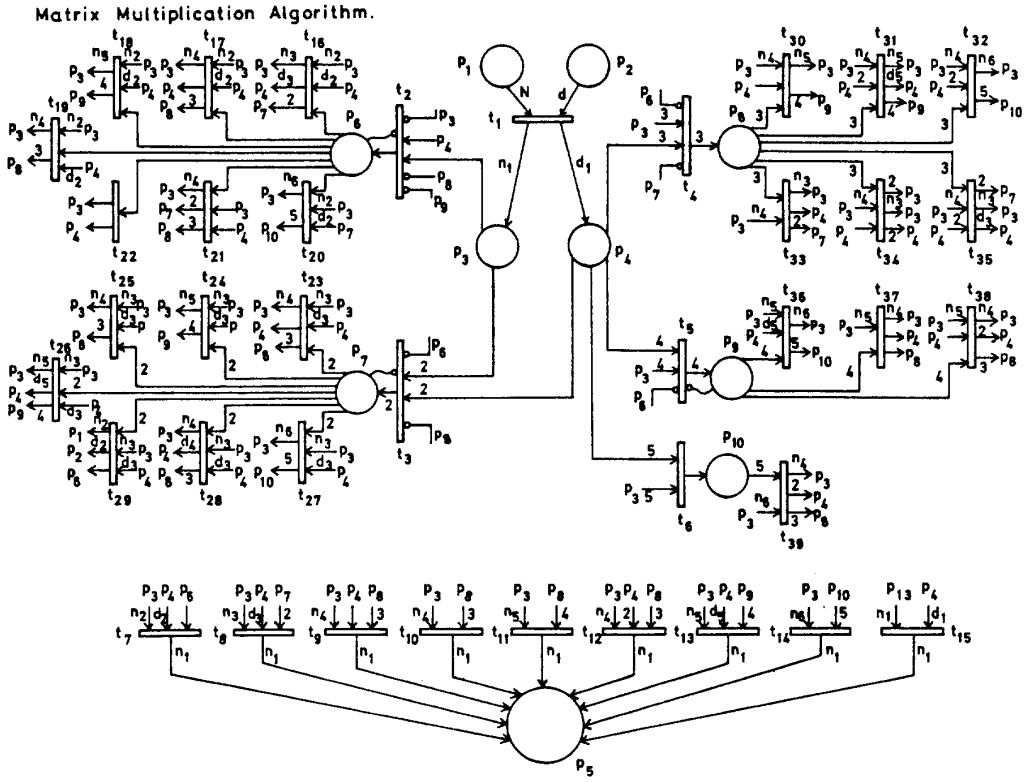

Figure 4. Detailed GSPN Model of Subcube Reliability Approach.

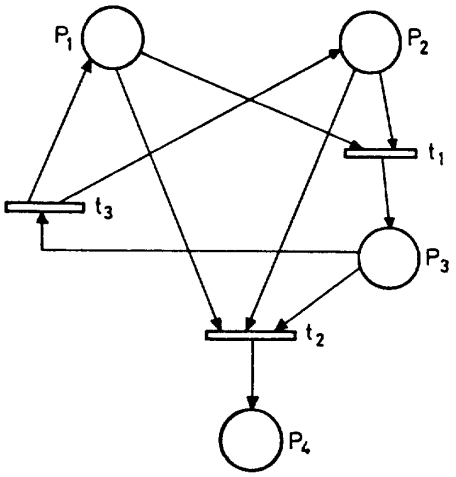

Figure 5. Refined GSPN Subcube Reliability Model of Figure 4.

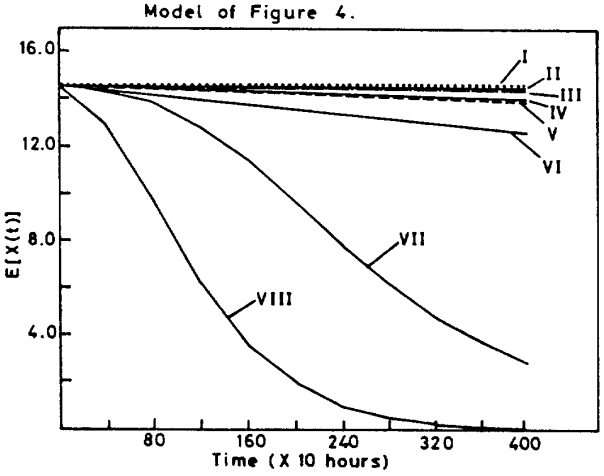

Figure 6. E[X(t)], Variation with Time for the Hypercube System.

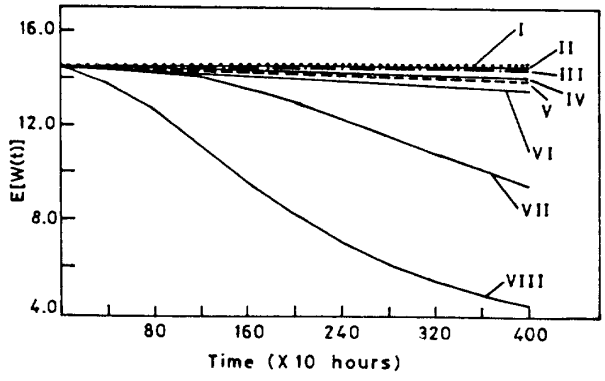

Figure 7. E[W(t)], Variation with Time for the Hypercube System.

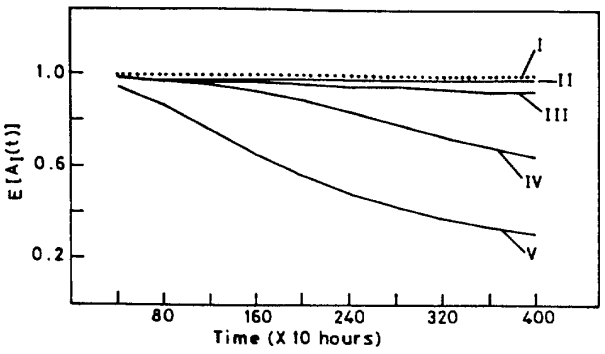

Figure 8. E[A_1(t)], Variation with Time for the Hypercube System.

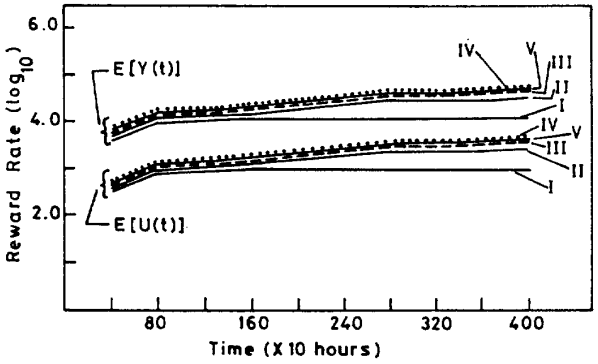

Figure 9. E[Y(t)] and E[U(t)], Curves for the Hypercube System.

Table 1. Reliability results of 6-cube ($\lambda = 1*10^{-6}$, $\mu = 0.1$, $C_o = 0.95$)

t	SRM_1	SRM_2
2000	0.99999379	0.99999993
4000	0.99997622	0.99999987
6000	0.99994862	0.99999981
8000	0.99991221	0.99999974
10000	0.99986805	0.99999968
12000	0.99981707	0.99999962
14000	0.99976010	0.99999956
16000	0.99969784	0.99999950
18000	0.99963095	0.99999944
20000	0.99955997	0.99999938

Table 2. Sensitivity analysis of the hypercube Model.

(t = 2000 hours, $\lambda = 0.00025$ per hour, $\mu = 0.2$ per hour, $C_o = 0.999999$)

Performability Measures	Sensitivity Parameter	Sensitivity
R(t)	λ	-0.427264
E[X(t)]	λ	-4.272641
E[Y(t)]	λ	-8545.60
R(t)	μ	+ 1.984182
E[X(t)]	μ	+ 19.84182
E[Y(t)]	μ	+ 3984.7058
R(t)	C_o	+ 0.999981
E[X(t)]	C_o	+ 9.999812
E[Y(t)]	C_o	+ 19999.8709