Performance Model of Interference Avoidance Policy Using Stochastic Petri Nets

Takashi KOBAYASHI Kenzo KURIHARA
Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji Asao-ku, Kawasaki-shi 215, JAPAN

Abstract

A new method is described for analyzing scheduling policy for interference avoidance. The traditional queuing theory assuming a first-come-first-served policy is not suitable for analyzing scheduling policy. Stochastic Petri Nets (SPN) is suitable for analyzing scheduling policy, because with it job arrival states, resource assignment states, and job flow due to these states can be modeled. However, if the scheduling logic is described in detail with SPN, a combinatorial explosion is soon encountered. Therefore, a new method is proposed to cut down the size of SPN model and to analyze the control policy of a magnetic tape library.

1. Introduction

In many parallel processing systems, interference among jobs, which comes from resource contention, causes performance degradation. In order to solve the problem of resource contention, an important question is: "How shall we optimize job scheduling policy ?" Traditional queuing network models, in many cases, assume the first-come-first-served policy, and are not suitable for evaluating other policies. On the other hand, though computer simulation is almighty, this method requires a vast expenditures for program development.

Stochastic Petri Nets (SPN), a recent analysis method, has the potential for application to scheduling policy analysis, because with it we can model job arrival states, resource allocation states and job flow based on these states. The modeling power of Petri Nets is equal to that of a Turing Machine, so we can model almost all kinds of scheduling policy. However, if we model scheduling logic in detail with Petri Nets, that model becomes huge with a great number of places and transitions, so analyzing of the model becomes difficult.

In this paper, we consider a system in which resource contention causes job interference, and propose a method for cutting down the size of the performance model of the interference avoidance policy. We apply the proposed method to analyzing performance of the control policy of a magnetic tape library, and show the efficiency and limit of this method.

2. Analysis of scheduling policy

2.1 Problem

We consider a system, having the same kinds of main resources used for executing jobs in parallel and some different kinds of sub-resources used according to the job to be executed. We will call this system "parallel processing system (See Fig. 1)." For example, in a multiprocessor system, main resources are CPUs and sub-resources are common memories. Interference among jobs is defined as follows. When plural jobs access the same sub-resource, only one of these jobs can continue its process with this sub-resource and others must wait until this sub-resource becomes free.

We assume that the main resource will not become free from the start to the end of a job. Therefore, the

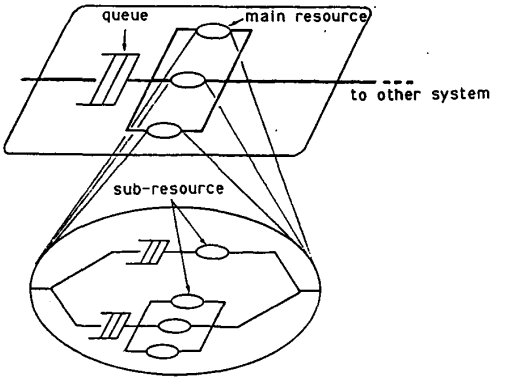

Fig. 1. Parallel processing system

TH0386-3/91/0000/0125/$01.00 © 1991 IEEE

occupied time for executing a job is the unity of the processing time in one of the main resources, the processing time in sub-resources, and the waiting time for sub-resources. Scheduling policy for interference avoidance, aims to avoid the contention of the sub-resources and to minimize the occupied time of the main resources. To accomplish this aim, we assign the most suitable job to each of the main resources from the queuing jobs.

We define as follow, jobs, main resources, sub-resources, and scheduling policies for interference avoidance.

(1) Jobs

a. The open system is assumed, namely each job enters the system and leaves the system after execution.

b. Jobs are classified into groups according to the types and numbers of the sub-resources used in executing each job.

c. The job arrival pattern is assumed to be a Poisson process, and the number of jobs in each job-group can be known beforehand.

(2) Main resources

a. There are plural main resources in the system, and with these main resources jobs can be executed in parallel. All jobs can be executed with any main resource.

(3) Sub-resources

a. Sub-resources are classified into types. Each job uses a type according to the details of the job.

b. Each sub-resource can be used by a single job at the same time. Therefore, in the case of plural jobs demanding the same sub-resource, only one job can acquire the right of use.

(4) Scheduling policies for the interference avoidance

The job scheduling policy for interference avoidance is as follows.

a. The goal is to reduce the sojourn time in the parallel processing system and to improve the system throughput. In order to achieve this goal, we must avoid the interference caused by the resource contention.

b. The restrictions are as follow.

i. job-group of each job waiting in the queue of the main resources, and job-group of each job processed in the main resource.

ii. resource-type of each sub-resource in use.

c. The variable to determine is the job execution sequence in each main resource.

In this paper, we consider a method for analyzing variations of scheduling policies.

2.2 Traditional performance models

Traditionally, the performance evaluation of the system, which is composed of plural resources and into which plural jobs arrive at random, is usually executed with queuing network models. Many kinds of queuing network models mainly for computer systems, have been proposed by F. Baskett, K. M. Chandy, P. J. Burke, J. P. Buzen, and others [5]. Analytical methods, however, have not been developed in many practical cases. For example, they have not been developed for adopting various scheduling policies in each node of the network, nor for varying the job arriving route according to the system state. Moreover, queuing network models in themselves do not have functions for describing synchronized job control, nor exclusive resource assignment. The applications of these models are thus limited to a comparatively small region. With queuing network models, we cannot analyze the scheduling policy as it is described in this paper.

The Stochastic Petri Nets (SPN) was developed by M. K. Molloy to improve the weak points of queuing network models [2]. SPN is a model having not only immediate transitions in original Petri Nets, but also transitions firing with exponentially distributed delay. The modeling power of SPN is equivalent to the power of Turing machines [1]. Moreover, if we model the system with SPN, we can make a Markov model from the SPN model automatically. The advantages of SPN have been published, for example, the evaluation of the contention of shared memories and shared data buses in multi-processor systems, and the evaluation of the protocol overhead in communication systems [3] [4].

With SPN, it is theoretically possible to model scheduling policies such as we consider in this paper. That is to say, by describing conditions with places of SPN and events with transitions of SPN, we can construct a primitive logic, "if some condition holds, then some event takes place." By combining such a primitive logic, we can construct more abstract logic such as the scheduling policy. For example, we consider a simple system with two job-groups and two resource-types. Each arriving job belongs to either group-A or group-B with probability 0.5. Each job is executed with one of four main resources. Jobs of A-group use sub-resources belonging to a-type, and jobs of B-group use

sub-resources belonging to b-type. The number of sub-resources of each type is two respectively. In order to avoid the contention of sub-resources, more than three jobs belonging to the same group should not be assigned to main resources simultaneously. Therefore, we adopt the next scheduling policy.

i. If there is a free sub-resource of a-type and there is a job of A-group in the queue, we assign the job of A-group to a main resource, and execute the job using the sub-resource of a-type. We assign a job of B-group with the same rule.

ii. If there is a free sub-resource of b-type and no free sub-resource of a-type, and there is a job of A-group and no job of B-group in the queue, we assign the job of A-group to a main resource, and execute the job using the sub-resource of b-type. We assign a job of B-group with the same rule.

The SPN model of this simple system is depicted in Fig. 2. In this figure, places p_{01} and p_{02} represent the queuing jobs belonging to A-group and B-group, respectively. Place p_{11} represents the job execution of A-group using the sub-resource of a-type, and place p_{12} represents the job execution of A-group using the sub-resource of b-type. Place p_{13} represents the job execution of B-group using the sub-resource of a-type, and place p_{14} represents the job execution of B-group using the sub-resource of b-type. The firing conditions of transitions t_{11}, t_{12}, t_{13} and t_{14} represent the job scheduling logic.

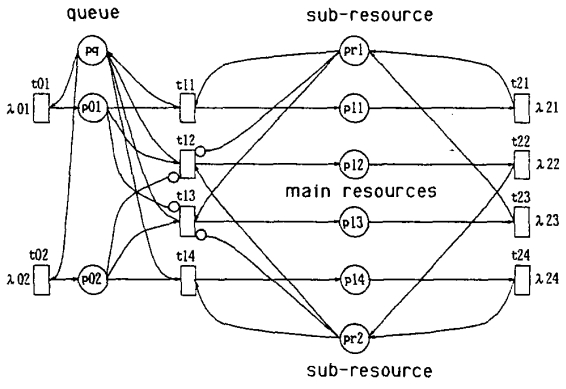

Fig. 2. Traditional SPN model of the simple system

In applying this modeling method, it is the complexity problem that becomes an obstacle. In order to simulate the scheduling logic strictly, as described above, we must make the model in which job-groups or resource-types are distinguished. As we can see easily, the number of coverable markings, namely system states, increases exponentially according to the number of job-groups or resource-types. In a case of the example system, the number of states is about 500. If the number of job-groups becomes three and the number of resource-types is becomes three, the number of states becomes about 250,000 (= 500^2). When we apply SPN to the analysis of the scheduling policy, the most important question will become: "How can we reduce the size of the model ?"

3. Performance function model

3.1 Concept

In analyzing a scheduling policy, it is important to find the differences in performance between variations of the scheduling policy. We need not make such a model as strictly simulates the scheduling logic. Therefore, we introduce a *performance function* which expresses the system performance in case of adopting each scheduling policy, and with it, remove the scheduling logic from the model. As described above, the performance degradation comes from interference. Therefore, the more frequently interference occurs, the smaller the value of this function should be. Because interference occurrence depends on the system states (the number of jobs in a queue, the utilization of resources, and so on), a performance function becomes the function of these states. Moreover, in order to decrease the number of states, we ignore job-groups and resource-types, and consider only the number of jobs in the queue and main resources. We predict interference occurrence stochasticly from the number of jobs.

By incorporating this performance function as the firing rate of the transition, we can make the size of the model small enough to be analyzed within practical periods of time. We automatically convert the SPN model into a Markov model, and analyze the model with the analysis methods of Markov models.

The model mentioned above is explained in detail as follows.

3.2 Analysis model

(1) Performance function

As described in the premise of the problem, it is determined by three kinds of states whether the resource contention will take place. These states are SQ, SM and SR, as follow. And, the total state of the system, S, can be described by the direct product of these three states SQ×SM×SR.

$SQ = \{Q_1, Q_2, \ldots, Q_x\}$: the arrival state in the queue of the main resources,

where Q_j is the number of jobs in job-group j (j = 1, 2, ..., x) in the queue of the main resources.

$SM = \{M_1, M_2, \ldots, M_x\}$: the assign state of the main resources,

where M_j is the number of jobs in job-group j (j = 1, 2, ..., x) processed in the main resources.

$SR = \{R_1, R_2, \ldots, R_y\}$: the assignment state of the sub-resources,

where R_k is the number of sub-resources in resource-type k (k = 1, 2, ..., y) used in executing a job.

If the resource contention takes place in a state $s_i \in S$, the occupied time of the main resource for processing a job increases by the waiting time for sub-resources. The occupied time of the main resource in the state $s_i \in S$, is obtained as the following equation.

$$TP_i = TM + TR + TW_i \qquad (1)$$

where TM is the average processing time by the main resource only, TR is the average processing time in using the sub-resources, and TW_i is the waiting time for the sub-resources. Here, TW_i can stochasticly be evaluated from each system states $s_i \in S$ and the scheduling policy adopted in assigning jobs to main resources. Therefore, TP_i can be expressed as the function of SQ, SM, and SR.

As described above, if we define the states by distinguishing job-groups or resource-types, the number of states will increase exponentially according to the increase in the number of job-groups and resource-types. Therefore, we make the model by considering only the number of jobs and resources. That is, we describe the system states with following three elements.

Q: the number of jobs in the queue of the main resources

M: the number of jobs processed in the main resources

R: the number of sub-resources in use

From the premise of the problem, the ratio of the number of jobs in each job-group is given, so the probability that a job will belong to job-group j (j = 1, 2, ..., x) is known in advance. Similarly, the resource-types and the number of the sub-resources to use, are given for each job-group, so the probability that a sub-resource in use will belong to resource-type k (k = 1, 2, ..., y) is also known in advance. Therefore, we can stochasticly estimate SQ, SM and SR mentioned above from Q, M and R. For the same reason, we can also stochasticly estimate R (the number of sub-resources in use) from M (the number of jobs processed in the main resources). Finally, the system state $s_i \in S$ can be described with two elements Q and M. Equation (1) can be transformed to the function of Q and M, as follows.

$$TP_i = f(Q, M) \qquad (2)$$

When the scheduling policy is given, the performance function (the number of jobs to be processed in a unit time) is given as $(TP_i)^{-1}$ for the policy.

(2) Making the SPN model

We incorporate the performance function into the SPN model, as follows. The obtained SPN model of the said parallel processing system is depicted in Fig. 3. In this model, place p_0 represents the jobs waiting in the queue, and place p_1 represents the jobs processed in the main resources. Place p_2 represents the complementary place to limit place p_0's capacity, and place p_3 represents the complementary place of place p_1. Transition t_0 represents the job arrival into the system, and its firing rate λ_0 represents the arrival rate. Transition t_2 repre-

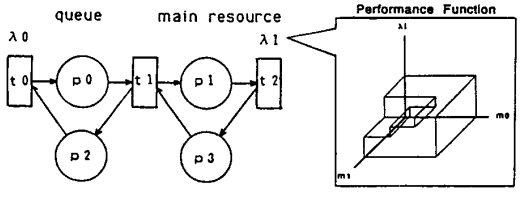

Fig. 3. SPN model of the parallel processing system

sents the end of the job execution, and its firing rate λ_1 represents the total service rate of the main resources, that is, the performance function $(TP_i)^{-1}$. Transition t_1 represents the job assignment to the main resources.

We analyze the SPN model made by procedures (1) and (2). First, we make the state change diagram of the parallel processing system from the SPN model. We can automatically make this diagram using the coverability tree algorithm in the theory of Petri Nets. Next, we will get the steady-state probability of each state from this state change diagram using the analysis method in the Markov model. Last, we easily get the sojourn time in and the throughput of the system, using Little's formula [5].

4. Application to a magnetic tape library

In this chapter, we apply the analysis method proposed in the previous chapter to a magnetic tape library, and examine the validity and limits of this method.

4.1 Control Policy of the magnetic tape library

A magnetic tape library (MT library) is developed to manage and operate a large number of magnetic tapes automatically. The components of this system are:

about 6500 cells for storing the same number of magnetic tapes,

32 magnetic tape drivers set behind the cells,

a pair of accessors for transporting tapes one by one between cells and MT drivers,

as depicted in Fig. 4.

In this system, the job is to transport a tape between a cell and a MT driver. Two accessors run on the common rail in executing each job, so the interference between jobs takes place from the contention of the same rail. The purpose of the control of the MT library is to avoid the interference between jobs, and to realize a high response and throughput. In order to achieve this purpose, we establish a territory for each accessor, which is not invaded by any other accessor. We assign the optimum MT driver and accessor to a job, and make each accessor execute a job in its own territory. We execute the next two-step control.

Step 1: Optimum selection of MT drivers. For the tape demanded by the host computer, we select one of the MT drivers existing in the same territory as the demanded tape. Only in a case of all the MT drivers in the same territory being busy, do we select an MT driver in another territory.

Step 2: Optimum job assignments to accessors.

For each accessor, we assign a job which can be executed in its own territory. Only in a case that there is no job in the queue to be executed in its own territory, do we assign a job to be executed in another territory.

4.2 Analysis of the control policy

We apply the proposed analysis method considering two accessors as main resources, and two territories on the common rail as sub-resources. The job arriving at the system is executed by one of the two accessors. In executing the job, right territory or left territory or both of these territories are occupied according to the starting and terminal points of the transporting job. The said three elements of the system

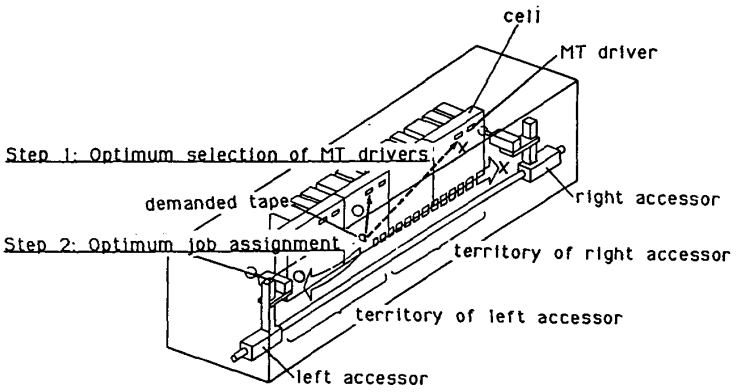

Fig. 4. Magnetic tape library and its control method

state, Q, M and R, correspond to the number of jobs in the queue, the number of jobs processed by the accessors, and the number of territories occupied, respectively. Because, which territory to be occupied by the arriving job is decided according to the way of selecting an MT driver, the probability that an arriving job belongs to each job-group, is the function of the number of MT drivers in use. The concrete analysis method is as follows.

(1) Performance function

We define the three job-groups below, and calculate the probability that an arriving job belongs to one of the job-groups.

L-group: the group of jobs to be executed only in the left territory

R-group: the group of jobs to be executed only in the right territory

W-group: the group of jobs to be executed in both territories

First, we calculate the ratio of the number of jobs belonging to each job-group in the case of the optimum MT driver selection. We consider the case of using m_2 drivers of the existing Nd drivers. In this case, we need to select an MT driver from unused $(N_d - m_2)$ MT drivers when the demand for a tape takes place. In a case of no unused drivers existing in the same territory as the demanded tape, the job to be executed will belong to W-group. In another case, the job will belong to L-group or R-group. If we can assume with 0.5 probability that an unused MT driver exists in either the left or the right territory, the probability that each job will belong L-group or R-group, PD, is as described in the next equation. The probability that each job will belong W-group, is described as (1 - PD).

$$PD = \begin{cases} 1 - 0.5^{(Nd - m2)} & (m_2/N_d > 0.5) \\ 1 & (m_2/N_d \le 0.5) \end{cases} \quad (3)$$

On the other hand, in random MT driver selection, the probability PD is constantly 0.5 without depending on the number of MT drivers in use.

On the the assumption explained above, we can obtain a performance function, denoting a job assignment policy to accessors. First, we consider the case of the optimum job assignment. It is when there is at least one job belonging to L-group or R-group in the queue, and the territory to use in executing this job is not occupied by another accessor, that we can avoid the interference between jobs. In a case of both accessors being not busy, the condition to be able to avoid interference is that there is at least one job belonging to L-group or R-group. On the other hand, when either accessor is busy, it is necessary that working accessor executes the job of its own territory, and there is at least one job to be executed in the territory of another accessor in the queue. If we assume that there are m_0 jobs in the queue, we can calculate, as follows, the probability that at least one job in the queue can be executed in the territory of the accessor to be assigned.

$$PT = \Sigma^{m0-1}_{i=0} (1 - 0.5 \times PD)^i \times (0.5 \times PD) \quad (4)$$

The probability that the working accessor executes the job of its own territory is the same as PT. Therefore, the probability that there is no interference between the two accessors, PA, is described as the following equation, where m_1 is the number of working accessors.

$$PA = \begin{cases} PT^2 & (m_1 > 1) \\ PD & (m_1 \le 1) \end{cases} \quad (5)$$

On the other hand, in assigning in first-come-first-served policy, the value of PT becomes $(0.5 \times PD)$ independently of states of the queue.

Now, when we describe the transporting time T_0 in the case of no interference and $(\alpha \times T_0)$ in the case of interference, the average transporting time, TP, is described as follows.

$$TP = T_0 \times PA + \alpha \times T_0 \times (1 - PA) \quad (6)$$

Here, α is the parameter to be decided according to the problem. In this application, we let α be 2, assuming the worst case that one accessor must wait from the start to the end of another accessor's execution. The performance function is a reciprocal of TP in equation (6).

As we have seen, the performance function of accessors is described with the number of jobs in queue m_0, the number of accessors in working m_1, and the number of MT drivers in processing m_2.

(2) Making SPN model

The SPN model for evaluating the control policy of the MT library is depicted in Fig. 5. In this figure, the area enclosed by a broken line shows the parallel processing and interference between accessors. Place p_0

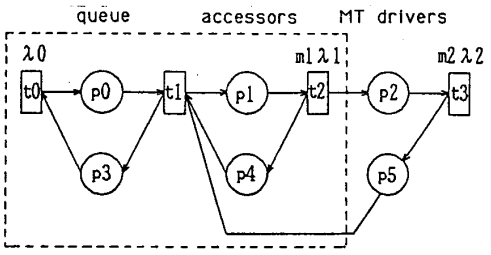

Fig. 5. SPN model of the magnetic tape library

represents waiting for accessors or MT drivers, place p_1 represents transporting accessors, and place p_2 represents processing MT drivers. Places p_3, p_4, and p_5 represents situations without jobs in the queue, without jobs to be executed by accessors, and without jobs to be executed by MT drivers, respectively.

Transition t_0 represents job arrival at the system, and transition t_1 represents job assignment to the accessor in a case of at least one accessor and one MT driver being free. Transition t_2 represents the end of a transporting job and the start of MT driver processing. Transition t_3 represents the end of MT driver processing. While transition t_1 fires immediately, transitions t_0, t_2, and t_3 fire with some delay. We assume that each delay time is random and exponentially distributed. Moreover, λ_0 represents the average arrival rate of demands, $m_1 \times \lambda_1$ represents the total service rate of accessors, and $m_2 \times \lambda_2$ represents the total service rate of MT drivers. λ_1 represents the service rate of one accessor and is given by the performance function $(TP)^{-1}$, and m_1 represents the number of accessors in working. λ_2 represents the service rate of one MT driver, and m_2 represents the number of MT drivers in processing.

We make a coverability graph by firing the SPN model depicted in Fig. 5. The result is shown in Fig. 6. Here, because the numbers of tokens in places p_3, p_4, and p_5 are decided according to the numbers of tokens

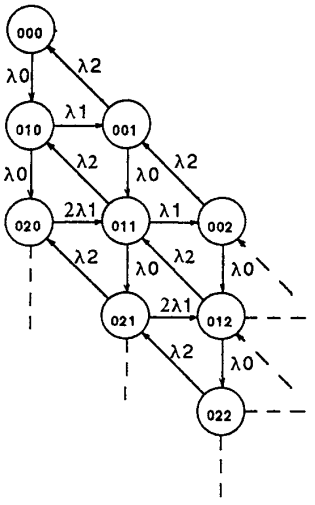

Fig. 6. State change diagram of the SPN model

in places p_0, p_1, and p_2 respectively, only the numbers of tokens in places p_0, p_1, and p_2 are depicted in Fig. 6. This coverability graph corresponds to the state change diagram of a Markov model. As described in the previous chapter, we can easily obtain the system throughput and the average turnaround time from the state change diagram.

4.3 Results of analysis

As we have described before, the control policy of an MT library consists of the two-step procedure, the optimum selection of MT drivers and the optimum job assignment to accessors. We compare three policies in order to analyze the effect of each procedure.

a. Optimum policy: optimum selection of MT drivers and optimum job assignment to accessors.

b. Intermediate policy: random selection of MT drivers and optimum job assignment to accessors.

c. Traditional policy: random selection of MT drivers and first-come-first-served job assignment to accessors.

The results of throughput are depicted in Fig. 7. In this figure, a horizontal axis denotes the arrival rate of I/O demands and a vertical axis denotes the throughput. The results of the proposed analysis method are depicted in thick lines, and the results of the computer simulation are depicted in thin lines. In the case of the Optimum policy and the Intermediate policy, the result

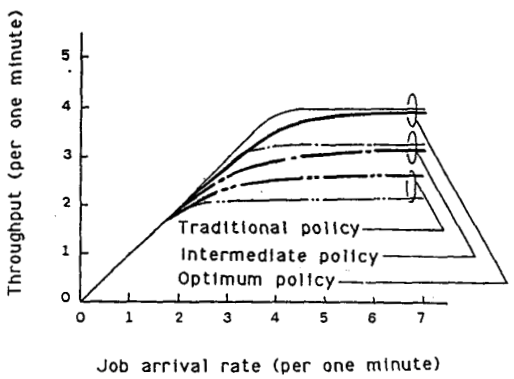

Fig. 7. Result of turnaround analysis

obtained by our proposed analysis method is nearly equal to the result obtained by computer simulation. Therefore, we can estimate our method to be proper. On the other hand, in the case of the Traditional policy, the result of our method is better than the result of simulation. This result comes from not considering the overhead accompanied by the start and stop operations in the case of interference. It is necessary to consider this overhead in cases where there is little difference between the results of the Traditional policy and the results of other control policies.

4.4 Effect of state reduction

We examine the effect of state reduction when we apply the proposed analysis method. In a case of not using the proposed method, we must rewrite the SPN model of the area enclosed by a broken line in Fig. 5 into the model in Fig. 8. In this figure, places p_{01}, p_{02}, and p_{03} represent the queuing jobs belonging to L-group, W-group, and R-group, respectively. Places p_{11}, p_{12}, and p_{13} represent the job execution by the right accessor, and places p_{14}, p_{15}, and p_{16} represent the job execution by the left accessor. The firing conditions of transitions t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, and t_{16} represent the scheduling logic of accessors. The number of states in this model is about 95,000, whereas the number of states in the model using the proposed analysis method is 228. This difference comes from the fact that in the model in Fig. 8, the scheduling policy is strictly

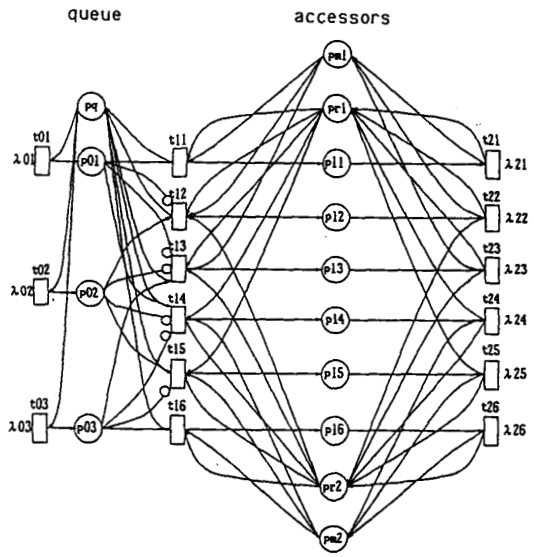

Fig. 8. SPN model without using the proposed analysis method

simulated, whereas in the proposed model, it is replaced by the performance function. Accordingly, we can reduce the number of states by 1/400 compared with the traditional SPN model, and analyze within practical periods of time.

5. Conclusion

In this paper, we have proposed a method of analyzing job scheduling policies for the system in which plural jobs are executed concurrently using plural common resources. This method is based on Stochastic Petri Nets, and has the advantages of decreasing the model size small enough to be analyzed within practical periods of time. Therefore, as we have denoted in this paper, this method can be applied to analyzing performance of actual systems.

Acknowledgement

Special thanks to Ai Satoyama in our research group for her contribution to evaluating our analyzing method. And we also thank Shingi Doumen, the chief of our laboratory, and Michio Miyazaki, the chief engineer of Odawara works for giving us the chance to do this research.

Reference

[1] T. Murata, "Petri Nets: Properties, Analysis and Applications," Proc. of IEEE, Vol. 77, No. 4, pp. 541-580, April 1989.

[2] M. K. Molloy, "Performance Analysis Using Stochastic Petri Nets," IEEE Trans. Comput., Vol. C-31, pp. 913-917, Sept., 1982.

[3] J. B. Dugan, "Extended Stochastic Petri Nets: Application and Analysis," Performance '84, North-Holland, pp. 507-519, 1984.

[4] M. A. Marsan, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," ACM Trans. Comput. Systems, Vol. 2, pp. 93-122, May, 1984.

[5] E. Gelenbe, I. Mitarani, "Analysis and Synthesis of Computer Systems," Academic Press Inc. (London) Ltd., (1980).

[6] M. A. Marsan, "Performance Models of Multiprocessor Systems," The MIT Press., Cambridge, (1986).

[7] W. Feller, "An Introduction to Probability Theory and Its Applications Vol. 1," Wiley, New York, (1950).