DOI:10.5392/IJoC.2010.6.3.019

Analysis Task Scheduling Models based on Hierarchical Timed Marked Graph

Cheul Woo Ro, Yang Cao
Dep. of Computer & Info. Eng.
Silla University, Busan, KOREA

ABSTRACT

Task scheduling is an integrated component of computing with the emergence of grid computing. In this paper, we address two different task scheduling models, which are static Round-Robin (RR) and dynamic Fastest Site First (FSF) task scheduling method, using extended timed marked graphs, which is a special case of Stochastic Petri Nets (SPN). Stochastic reward nets (SRN) is an extension of SPN and provides compact modeling facilities for system analysis. We build hierarchical SRN models to compare two task scheduling methods. The upper level model simulates task scheduling and the lower level model implements task serving process for different sites with multiple servers. We compare these two models and analyze their performances by giving reward measures in SRN.

Keywords*: Task Scheduling, SRN, Petri nets, Hierarchical Model, timed marked graph*

1. INTRODUCTION

Task scheduling is an integral part of parallel and distributed computing. With the emergence of grid and ubiquitous computing, new challenges appear in task scheduling based on properties such as security, quality of service, and lack of central control within distributed administrative domains [1]. Task scheduling is the key technology in grid resource allocation. How to effectively match grid tasks with available grid resources is a challenge for a grid computing system because of the dynamic, heterogeneous and autonomous nature of the grid [2]. The design of a scheduling mechanism that can adaptive to different types of tasks and adjust the behavior and response of a system to meet certain performance requirements is a tedious and challenging problem.

Marked graphs are special cases of Petri nets for modeling asynchronous concurrent systems [3]. The model of a timed marked graph (TMG) is obtained from a marked graph by adding durations to the events in the system [4]. In complex man-made environments discrete event dynamic systems are frequently encountered, and a timed marked graph is widely accepted as a convenient tool to describe systems of this kind [5].

In this paper, we consider two task scheduling models using static Round-Robin (RR) and dynamic Fastest Site First (FSF) scheduling method respectively. And we use extended TMG to simulate task serving processes. TMG can be a special case of SRN [6]. We study the performance of task scheduling using a formalism called SRN to model its behavior. To classify task scheduling and task serving process, the hierarchical SRN modeling techniques are adopted. The upper level model simulates task scheduling and the lower level model implements task serving process. And we analyze these two models by giving reward measures.

* *Corresponding author E-mail : cwro@silla.ac.kr*
Manuscript received May. 24, 2010 ; accepted Sep.20, 2010

2. TMG AND HIERARCHICAL SPN

2.1 Timed Petri Nets

A Petri net (PN) is a bipartite directed graph with two disjoint sets called places and transitions [3]. The state or condition of the system is associated with the presence or absence of tokens in various places in the net. The condition of the net may enable some transitions to fire. This firing of a transition is the removal of tokens from one or more places in the net and/or the arrival of tokens in one or more places in the net. The tokens are removed from places connected to the transition by an input arc; the tokens arrive in places connected to the transition by an output arc [3]. A place/transition net N is a triple $N = (P, T, A)$ where [7]:

2.2 TMG

The timed marked graphs are special cases of SPN. The definition is given by $\boldsymbol{TMG}=(\boldsymbol{P},\boldsymbol{T},\boldsymbol{F},\boldsymbol{W},\boldsymbol{M_0},\boldsymbol{\pi},\boldsymbol{\tau})$ where $\boldsymbol{P}$ is a set of places, $\boldsymbol{T}$ is a set of transitions, $\boldsymbol{F} \subseteq (\boldsymbol{P}\times \boldsymbol{T}) \cup (\boldsymbol{T}\times \boldsymbol{P})$ is a flow relation, $\boldsymbol{W}:\boldsymbol{F}\rightarrow \{1, 2, \ldots\}$ is a weight function, $\boldsymbol{M_0}$ is an initial marking, and $\boldsymbol{\pi}$ is the place delay function $:\boldsymbol{P}\rightarrow\boldsymbol{R}^+$ (the set of non-negative real numbers), $\boldsymbol{\tau}$ is the transition firing time function $\boldsymbol{\tau}$: $\boldsymbol{T}\rightarrow\boldsymbol{R}^+$ [8]. The dynamic operation of the TMG is as follows. When a transition receives a token from its in-arcs, it performs some internal computation and then sends one token along each of its out-arc. It takes time that a token travel along a link.

Normally, the TMG have all arc weights equal to one and each place has one token. In this paper, we use an extended TMG that some special places can have multiple tokens.

2.3 Hierarchical SPN

In the use of SPN to solve real world problems, we often generate a system model that is too big to be solved by SPN. A solution is dividing the model into several SPNs and solving it by iteratively execution of these SPNs.

The formulism of Hierarchical SPN is given by a tuple $H\_SPN=(P,T,D,D^0,g,m^0,>,d,w)$ where

- P is the set of places. Each place may contain tokens. The marking of the SPN is then defined by the number of tokens in each place. Let M be the set of markings.
- T is the set of transitions.
- D: $(((P\times T)\cup(T\times P))\times M) \rightarrow IN$ is the set of input arcs and output arcs. Each arc is given by its (marking-dependent) multiplicity.
- D^0: $(P\times T\times M) \rightarrow IN$ is the set of inhibitor arcs, from a place to a transition, with its associate multiplicity.
- $m^0 \in INp$ is the initial marking.
- $>$: $T \rightarrow IN$ is the explicit priority to transitions.
- d: $T \rightarrow$ {distributions} defines the ring time distribution of each transition (note that it includes immediate transitions).
- w: $(T\times M) \rightarrow IR^+$ is a weight function used to choose the one which will fire if several have the same ring time.
- g: $(T\times M) \rightarrow \{0,1\}$ is the guard function for each transition. It is a generalization of inhibitor arcs.

Marking-dependent enabling function (also called a guard) with each transition can be used to specify the firing rate of a timed transition or the firing probability of an immediate transition. The rates of transition are computed at lower level model and fed into the system level model.

Compared to the brutal force approach, this hierarchical approach largely reduces the number of states at the system level. This approach is an exact technique (not an approximation) for steady state measurement.

2.4 SRN

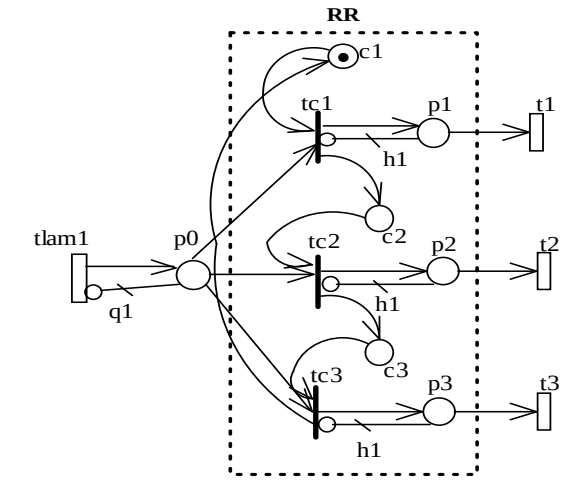
SRN is an extension of stochastic Petri nets (SPN) and is SPN augmented with the ability to specify output measures as reward-based functions, for complex systems performance evaluation. SRN has the ability to allow extensive marking dependency. It also has one important feature of expressing complex enabling/disabling conditions through guard functions. This can greatly simplify the graphical representations of complex systems. For an SRN, all the output measures are expressed in terms of the expected values of the reward rate functions. To get the performance and reliability/availability measures of a system, appropriate reward rates associated with the markings are assigned to its SRN. As SRN is automatically transformed into a Markov Reward Model (MRM) [6], steady state and/or transient analysis of the MRM produces the required measures of the original SRN [6,9,10].

3. HIERARCHICAL SYSTEM MODELING

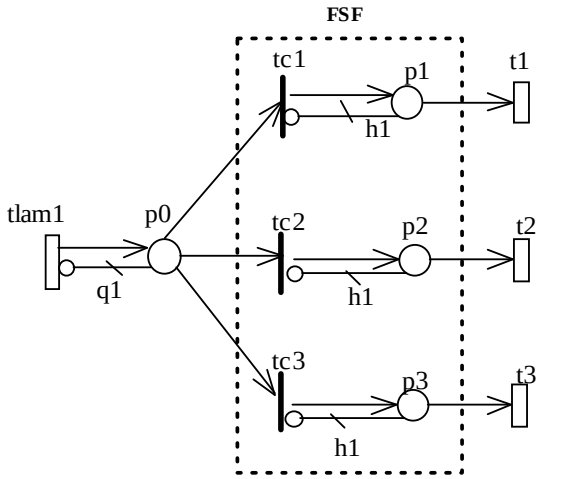
Integrating system availability and performance in a single model often causes the largeness and stiffness problem [11,12]. We built hierarchical model for system modeling. The higher-level represents the task scheduling process, and the lower-level simulate the actual task serving process. Hierarchical SRN model is the Markov reward model where the reward rates come from a sub-model [10].

3.1 Higher-level Modeling

For the higher-level, we built two kinds of task scheduling models. One uses static Round Robin (RR) task scheduling method and the other adopts dynamic Fastest Site First (FSF) task scheduling method. Figure 1(a) and 1(b) show the RR and dynamic task scheduler, respectively.



(a). RR task scheduler.



(b). Fastest Site First task scheduler.

Fig. 1. Higher-level Tas kScheduling Models.

The firing of transition ***tlam1*** means tasks arriving with rateλ, then tasks goes to place p_0 waiting for scheduling. For the static RR scheduling method, tasks are scheduled according to load balancing and assigned in circular order. To ensure this, we define three places c_1, c_2, and c_3. With initial tokens given to c_1, transition t_1 first get fired which move one token to place c_2, then transition t_2 get fired which move one token to place c_3 then transition t_3 get fired. So the scheduler assigns tasks to three sites with the cyclic sequence $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_1$.

For the dynamic task scheduling method, tasks are scheduled according to the processing ability of these three sites, i.e. depending on the sub-models return value of processing time of

these sites, the scheduler dynamically assigns tasks to the fastest site. We use ***guardfun***() [6] to represent the enable condition of transitions tc_1, tc_2, and tc_3, so when task arrives at place p_0 , it will compare and find which one has the smallest cycle time, then assign task to the corresponding place p_i, i=1,2,3.

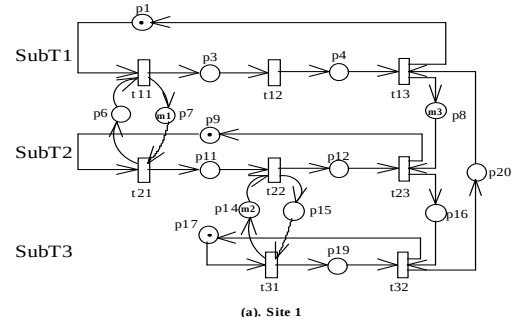
Transition t_1, t_2, and t_3 stand for three sites which are actually executed in the lower-level model. We use ***ratefun()*** [6] to represent that the firing rate of transition t_1, t_2, and t_3 are the return value from the lower-level model which is the smallest cycle time of lower-level model. For RR scheduling method, this return value is only used as the initial firing rate of transition t_1, t_2, and t_3. But for dynamic scheduling method, we also compare these three return values and take the fastest one to assign task to it. Place p_1, p_2, and p_3 represent number of tasks assigned to these three sites respectively. The different ***ratefun()*** we set for transitions t_i are as follows:.

ratefun() t_i { if t_i = min(SCT1, SCT2, SCT3) return (1) else return(0); }, i=1,2,3

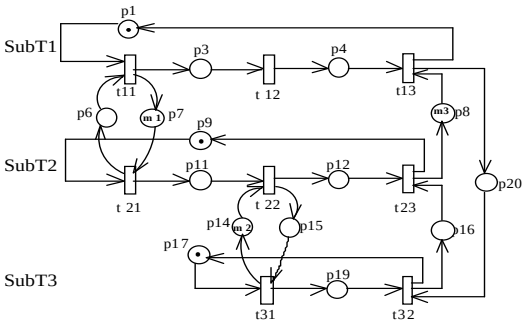
An inhibitor arc drawn from a place to a transition means that the transition cannot fire if the place contains at least as many tokens as the cardinality of the inhibitor arc [6]. The inhibitor arc from p_0 to tlam1 and three from p_1, p_2, p_3 to tc_1, tc_2, tc_3 are used to put constraints with their limited queue size.

3.2 Lower-level Modeling

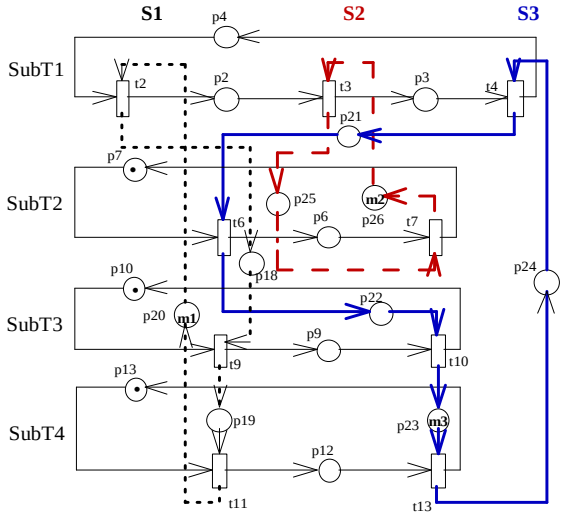
For the lower-level, we model three different sites with multiple servers. Figure 2 shows three extended timed marked graph represent three sites that consist of several servers. When a task is assigned to a site, it is divided into several subtasks according to the model of the site, and each subtask contains several operations. Then different severs in the model execute some operations of subtasks. The initial value of place p_1 means number of tasks which one site can execute.



(a). Site 1



(b). Site 2



(c). Site 3

Fig. 2. Lower-l evel TMG wi th multi pl eservers of three websites.

Figure 2(c) shows one site with three servers (S_1, S_2, S_3) and can process four subtasks ($SubT_1$, $SubT_2$, $SubT_3$, $SubT_4$).

SubTask i has number of operations (i=1~4): $SubT_1$ has three operations which are executed through transition t_2, t_3 and t_4 with server S_1, S_2 and S_3. For Subtask 2-3, the interpretations are the same. Server j includes several subtask operations (j=1~3): S_1 has process parts operation of $SubTask_1$, $SubTask_3$ and $SubTask_4$ which are executed through transition t_2, t_9 and t_{11}. For server 2-3, the interpretations are the same. We can depict the activities of Subtasks and servers as the cycles:

$SubT_1$: $p_4 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4$
$SubT_2$: $p_7 \rightarrow p_6 \rightarrow p_7$
$SubT_3$: $p_{10} \rightarrow p_9 \rightarrow p_{10}$
$SubT_4$: $p_{13} \rightarrow p_{12} \rightarrow p_{13}$
S_1: $p_{18} \rightarrow p_{19} \rightarrow p_{20} \rightarrow p_{18}$
S_2: $p_{25} \rightarrow p_{26} \rightarrow p_{25}$
S_3: $p_{21} \rightarrow p_{22} \rightarrow p_{23} \rightarrow p_{24} \rightarrow p_{21}$

We extended the TMG by putting multiple tokens of m1, m2, and m3 in p_{20}, p_{26}, and p_{23} respectively, which means that each server has the capability to handle multiple task operations simultaneously.

The main parts of Figure 2(a) and (b) are same except that task processing sequences are different with some control [13].

The subtasks executing sequence in Figure 2(a) is $SubT_1 \rightarrow SubT_2 \rightarrow SubT_3 \rightarrow SubT_1$.

The subtasks executing sequence in Figure 2(b) is $SubT_2 \rightarrow SubT_1 \rightarrow SubT_3 \rightarrow SubT_2$.

3.3 Hierarchical Models Executing Process

Hierarchical SRN models can be executed using a UNIX shell file and submodels can communicate information via files that can be declared and opened inside individual SPN submodel input files.

The interaction of higher level model and lower level model is done by system calls, which is executed in guard function of system level transition, and passing the output file of lower level model.

To build the hierarchical model we make two separate

models written in C-like programming language, which is used to describe SRN models. The one file represents higher level and others are low level models. The higher level file contains the code which calls low level models with some parameters at some step of execution.

4. MODEL ANALYSIS

4.1 Measures of Interest

In order to obtain the interested measures numerically from the SRN model, underlying Continuous-Time Markov Chains (CTMC) is generated and solved through the use of the software package SPNP [13]. We assume that all transition firing rates in our SRN models are exponentially distributed, so we perform the steady-state analysis of the model we have constructed [10].

For the lower-level model, we calculate the cycle time of each subtask, choose the smallest one and calculate its processing rate, then return these value as the firing rate of t_1, t_2, and t_3 in the higher-level model.

- Smallest Cycle Time (SCT)

According to Little's Result: $\overline{N} = \lambda T$, we can calculate the average time spent in the system $T = \frac{\overline{N}}{\lambda}$. In SRN, we extend it to calculate the cycle time of a subtask. The formula is as following: where $\sum_i mark(P_i)$ represents the average queue length of subtask i, and $rate("t_i")$ is the ith subtask arriving rate. We then compare these cycle time and get the smallest one.

$$CT_i = \frac{\sum_i mark(P_i)}{rate("t_i")}, \quad SCT = \min(CT_i)$$

For the higher-level model, we calculate system throughput and system response time of these two different scheduling models.

- System Throughput (ST)

To calculate the total system throughput, we use the following formula:

$$ST = \sum_i rate(t_i)$$

- System Response Time (SRT)

To calculate the total system response time, we use the following formula:

$$SRT = \frac{\sum_i mark(P_i)}{rate("tlam1")}$$

- Grade of Service (GoS)

We then define a formula to represent grade of service, W is the weight to take use of SRT. Here we just let W=1. The formula is as the following:

$$GoS = ST + \frac{1*W}{SRT}$$

4.2 Numerical Results

We get the numerical results by giving some different input parameters $f(\lambda, q1, h1)$ to the higher-level model, $\lambda \in \{0.1, 0.2 \ldots, 1.4\}$; $q1=2$; $h1=6$. We give input parameters $f(njob, m1, m2, m3)$ to the lower-level model, where the value of njob is gotten from the higher-level model as parameter transferred which means the scheduler assigns number of tasks to the site. We give $m1=5$; $m2=7$; $m3=2$; which means the ability of each server to handle multiple task operations simultaneously. Table 1 shows the processing time of subtasks on each server as input data [5].

Table 1. Subtask processing time

SubTask	S_1	S_2	S_3
$SubT_1$	5.82	4.36	3.93
$SubT_2$	-	2.93	7.56
$SubT_3$	3.76	-	9.61
$SubT_4$	1.53	-	4.38

Table 2 shows ST, SRT and GoS of two models with different tasks arriving rateλ.

Table 2. ST and SRT

RR Scheduling			
lambda	ST	SRT	GoS
0.1	0.100	10.580	0.195
0.2	0.186	22.720	0.230
0.4	0.182	44.130	0.204
0.6	0.178	46.427	0.200
0.8	0.177	47.035	0.198
1.0	0.177	47.293	0.198
1.2	0.177	47.430	0.198
1.4	0.176	47.511	0.197
Dynamic Scheduling			
lambda	ST	SRT	GoS
0.1	0.100	15.391	0.165
0.2	0.197	29.046	0.231
0.4	0.265	39.329	0.291
0.6	0.263	39.609	0.288
0.8	0.254	39.567	0.279
1.0	0.244	39.643	0.269
1.2	0.234	39.816	0.260
1.4	0.226	40.049	0.251

Figure 3 shows the GoS comparison of these two models.

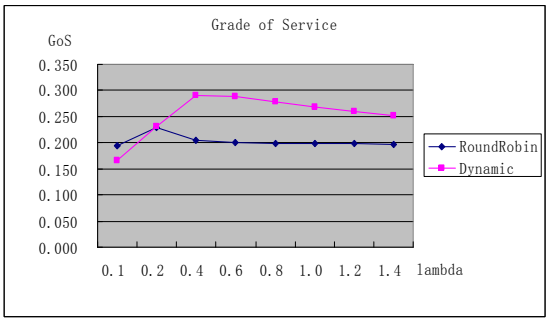


Fig. 3. GoS comparison.

We also compare the smallest cycle time in the lower-level models. Table 3 shows different SCT of each site according to number of tasks from higher level model.

Table 3. Lower-level SCT

Site	No. tasks	SCT
1	2	5.909
	4	5.865
	6	9.381
	8	8.814
	10	11.778
2	2	10.087
	4	10.087
	6	11.664
	8	11.664
	10	14.127
	12	16.468
3	2	10.672
	4	10.672
	6	10.672
	8	12.695
	10	12.695
	12	16.718

5. CONCLUSION

In this paper, two task scheduling models with different scheduling methods have been constructed to get performance analysis using extended timed marked graphs. Hierarchical models are built for system modeling to resolve state largeness problem. We compare these two models and analyze their performances such as system throughput and system response time by giving reward measures in SRN. We then create another measure , grade of service which take both ST and SRT into account. According to the final results we can declare that system using dynamic scheduling method is more efficient than that using Round Robin scheduling method.

REFERENCES

[1] Xiao Shan He, Xianhe Sun, Gregor von Laszewski, “QoS guided Min-Min heuristic for grid task scheduling,” Journal of Computer Science and Technology, vol. 18(4), 2003, pp. 442-451.1

[2] Peijie Huang, Hong Peng, Piyuan Lin, Xuezhen Li, "Static strategy and dynamic adjustment: An effective method for grid task scheduling", *Future Generation Computer Systems* 25, 2009, pp. 884-892

[3] W. M. Zuberek and W. Kubiak, "Timed Petri Nets in Modeling and Analysis of Simple Schedules for Manufacturing Cells", *Computers and Mathematics with Applications 37*, 1999, pp. 191-206.

[4] Yossi Malka,Sergio Rajsbaum, "Analysis of Distributed Algorithms based on Recurrence Relations (Preliminary Version)," Lecture Notes In Computer Science; Proceedings of the 5th International Workshop on Distributed Algorithms ,vol. 579,1991, pp. 242-253.

[5] Shimohata, H., S. Kataoka and T. Yamada, "A resource allocation problem on timed marked graphs: a decomposition approach," International Journal of Systems Science, vol. 27, 1996, pp. 405-411.

[6] G. Ciardo, A. Blakemore, P.F. Chimento, and K.S. Trivedi, "Automated Generation and Analysis of Markov Reward Models Using Stochastic Reward Nets", Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Math. and its Applications, A. Friedman and J.W. Miller, eds., vol. 48, pp. 145-191.Heidelberg, Germany: Springer-Verlag, 1993.

[7] Miguel Felder, Angelo Gargantinib, Angelo Morzentib, " A theory of implementation and refinement in timed Petri nets," *Theoretical Computer Science 202*, 1998, pp. 127-161.

[8] Branislav Hrúz,MengChu Zhou, “Modeling and Control of Discrete-event Dynamic Systems: with Petri Nets and Other Tools (Advanced Textbooks in Control and Signal Processing),” ISBN: 184628872X,2007,vol. 178.

[9] Jogesh Muppala, Gianfranco Ciardo, and K. S. Trivedi, "Stochastic Reward Nets for Reliability Prediction, Communications in Reliability, Maintainability and Serviceability", An International Journal published by SAE International, vol. 1, no. 2, July 1994.

[10] Cheul Woo Ro, Kyung Min Kim. " Stochastic Petri Nets Modeling Methods of Channel Allocation in Wireless Networks," IJOC(International Journal of Contents) vol. 4, no.3, 2008.9

[11] K. S. Trivedi, J. K. Muppala, S. P. Woolet, and B. R.Haverkort, "Composite performance and dependability analysis", Performance Evaluation, vol 14, no. 3, 1992, pp. 197–215.

[12] A. Bobbio, K.S. Trivedi, "An aggregation technique for the transient analysis of stiff Markov chains," IEEE Transactions on Computers 35(9), 1986, pp. 803-814.

[13] Cheul Woo Ro, Yang Cao, Yun Xiang Ye, Wei Xu, “Task Scheduling Modeling using Timed Marked Graph,” *Proceeding of KIMICS*, 2010.5.

Cheul Woo Ro
Refer to IJOC, Vol.4, No. 3, 2008.9

Yang Cao
She received the M.S in Belgium in 2003. She works at Eastern Liaoning University in China. She is currently a PhD candidate in the department of computer engineering at Silla University. Her main research interests include modeling and analysis of communication systems.