Performance Analysis of Multiprocessor Schedules of Tasks with Stochastic Execution Times

Extended Abstract

Sorin Manolache, Petru Eles, Zebo Peng
Linköping University
{sorma, petel, zebpe}@ida.liu.se

1 Introduction

Systems controlled by dedicated electronic systems become indispensable in our lives and can be found in avionics, automotive industry, home appliances, medicine, telecommunication industry etc. [3]. Due to the application nature itself, as well as due to the high demands in terms of computation power and constraints such as cost and energy dissipation, these systems are mainly custom designed heterogeneous multiprocessor platforms. Their high complexity makes the design process a challenging activity. An accurate and efficient design process is hence the key to cope with the high time-to-market pressure. The enormous design space implies that accurate performance estimation tools in all design stages are of capital importance for guiding the designer and reducing the design cost and iterations.

The present work focuses on an analytic approach to performance estimation of multiprocessor real-time systems. We consider applications as sets of task graphs with the tasks statically mapped on a set of processors. Most of the work in this area carries out the analysis in the worst case of the task execution times (WCET) [4][13] and provides answers whether the tasks will meet or not all of their deadlines. Such an approach is well suited for critical applications. However, it leads to expensive and inefficient implementations in the case of other application classes like soft real-time systems and multimedia applications. The latter occupy an important market share. According to Tia [14], applications have been reported where the WCET is 145% of the average one and the worst case appears rarely.

Our work considers the case when the task execution times are of stochastic nature and their probabilities are distributed according to given arbitrary continuous time probability distribution functions. Because meeting a deadline in this case becomes also a stochastic event, our method outputs the probability of task deadline misses.

The variability of the task execution time may stem from several sources: input data characteristics (especially in differently coded video frames), hardware architecture hazards (caches and pipelines), environmental factors (network load), or insufficient knowledge regarding the design (task running on a not yet manufactured processor, for example).

Leung and Whitehead [9] showed that the schedulability analysis is an NP-complete problem in the case of fixed task execution times and more than two processors. Obviously, the problem is more challenging in the case of stochastic task execution times.

The sequel of the paper is organized as follows: The next section surveys some related approaches. Section 3 formalizes the problem. Section 4 presents the approach outline. Each of the points in the approach is detailed in the following sections. The last section draws the conclusions.

2 Related work

Lehoczky has pioneered the “heavy traffic” school of thought in the area of real-time queueing [7][8]. The theory was later extended by Harrison [5], Williams [15] and others. The application is modelled as a multiclass queueing network. This network behaves as a reflected Brownian motion with drift under heavy traffic conditions, i.e. when the processor utilizations approach 1, and therefore it has a simple solution. This approach, to our knowledge, fails yet to handle systems where a task has more than one immediate successor task. Moreover, the heavy traffic assumption implies an almost infinite queue in the case of input distributions with non-negligible variation. This leads to an unacceptably high deadline miss ratio and limits the applicability of such an approach in real-time systems.

Manolache et al. [10] considered monoprocessor systems. The analysis is based on solving a generalized semi-Markov process by means of the auxiliary variable

method. Although they concurrently construct and analyse the process, saving a significant amount of memory, the method is of limited applicability to multiprocessor systems due to the exploding complexity.

Shin et al. [6] also modelled the application as a queueing network, but restricted the task execution times to exponentially distributed ones for ease of analysis purposes. The tasks were considered to be scheduled according to a FIFO policy. The underlying mathematic model is then the appealing continuous time Markov chain (CTMC).

Our approach is also based on a CTMC in order to keep the appealing character of the solution procedure and to avoid the time and memory consuming convolutions implied by solving the otherwise generalized semi-Markov process. We consider the tasks to be scheduled according to a fixed priority non-preemptive policy. We overcome the limitation of the exponentially distributed execution time probabilities by approximating arbitrary real-world distributions by means of Coxian distributions, i.e. weighted sums of convoluted exponentials. The resulting CTMC is huge, but because of regularities in its construction, its infinitesimal generator needs not be stored explicitly, making the method applicable to real applications. The infinitesimal generator appears as a sum of Kronecker products of simple matrices.

Plateau and Fourneau [11] address the problem of constructing the global infinitesimal generator of a CTMC modelling a concurrent system from the local infinitesimal generators of the system components. The execution times are still considered exponential. Although we address a different problem, our approach is related to theirs by using a generator given as a sum of Kronecker products of matrices.

3 Problem formulation

Informally, given a set of task graphs where the task execution time probabilities are distributed according to given arbitrary continuous distributions, the analysis outputs the task deadline miss ratios. Formally, the input to the analysis procedure consists of:

- a set TG of task graphs TG = $\{g_1, ..., g_G\}$, formed of the tasks in the set T = $\{t_1, ..., t_N\}$
- a set of processors P = $\{p_1, p_2, ..., p_s\}$
- a surjective mapping Map : T $\rightarrow$ P
- a set of continuous execution time probability distribution functions (ETPDF) E = $\{e_1, ..., e_N\}$, e_i : $[0{:}\infty) \rightarrow \Re$, statistically independent
- a set of periods, A = $\{a_1, a_2..., a_G\}$
- a set of fixed deadlines, deadline equals period
- a fixed priority non-preemptive scheduling policy, Prior : T $\rightarrow$ $\aleph$
- the number of late task graph instantiations allowed in the system, B = $\{b_1, ..., b_G\}$. If $b_i = 0$ then the late instantiations of task graph i are discarded

The analysis outputs the deadline miss ratios of the task graphs, F = $\{f_1, f_2, ..., f_G\}$

4 Approach outline

The underlying mathematical model of the application to be analysed is the stochastic process. The process has to be constructed and analysed in order to extract the desired performance metrics. The task execution times correspond to holding times in the states of the process. When considering arbitrary ETPDFs, the resulting process is a generalized semi-Markov process, making its analysis demanding in terms of memory and time. If the execution time probabilities were exponentially distributed, as assumed for instance by Shin, the process would be a CTMC.

As a first step in our approach, we approximate the arbitrary ETPDFs with Coxian distributions, i.e. weighted sums of convoluted exponentials. This ensures that we still deal with a CTMC. This step is detailed in the next section.

We imagine an application that differs from the one to be analysed by having exponentially distributed execution time probabilities instead of the real ones. The next step is to construct the underlying Markov chain C of the imagined application and to obtain its infinitesimal generator matrix Q. This is done by modelling the application as a Generalized Stochastic Petri Net and constructing its underlying CTMC by using Balbo's algorithm [1].

The third step is to get the generator matrix M of the CTMC resulting from C when considering the Coxian distributions obtained in the first step instead of the imagined exponential ones. The construction is detailed in Section 7.

As a last step, the chain with matrix M is solved with one of the available numerical iterative methods and the performance metrics extracted.

5 Coxian distribution approximation

Coxian distributions were introduced by Cox [2] in the context of queueing theory and we will use its specific terminology. A graphical representation of a Coxian distribution with four stages appears in Figure 1. Imagine a "customer" approaching the Coxian "service station" from the left. The ovals stand for stages with exponentially distributed "service time" probabilities. The service rate (inverse of the average service time) of stage i, $1 \le i \le r$, is μ_i. Upon leaving the stage i, the customer leaves the entire Coxian station with probability α_i or proceeds to the next stage with probability $1-\alpha_i$. The stages are invisible from outside the Coxian station, i.e. no other customer may enter the first stage if another customer has not yet left any

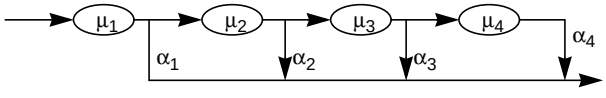

Figure 1 Coxian distribution

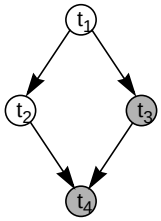

Figure 2 Task graph

of the stages.

The Laplace transform X(s) of the probability density function of a Coxian distribution with r stages is given by the following formula:

$$X(s) = \sum_{i=1}^{r} \alpha_i \cdot \prod_{k=1}^{i-1} (1-\alpha_k) \cdot \prod_{k=1}^{i} \frac{\mu_k}{s+\mu_k}$$

X(s) is a strictly proper rational transform, implying that the Coxian distribution may approximate a fairly large class of arbitrary distributions with an arbitrary accuracy provided a sufficiently large r. The solution to the approximation problem means finding μ_i, $i=\overline{1,r}$, and α_i, $i=\overline{1,r-1}$ ($\alpha_r=1$) such that an error function is minimized. This is done in the complex space by minimizing the distance between the Fourier transform X(jω) of the Coxian distribution and the computed Fourier transform of the distribution to be approximated. The minimization is a typical interpolation problem and can be solved by various numerical methods [12].

6 Application Modelling

The direct algorithmic generation of the underlying stochastic process, though possible, does not give too much insight in the properties of the stochastic process that might be exploited in the analysis. We consider a visual formalism like the Generalized Stochastic Petri Nets (GSPN) an appropriate intermediate representation in the analysis process. Its tangible reachability graph corresponds to the underlying stochastic process.

We illustrate the construction of the GSPN based on an example. Let us consider the task graph in Figure 2. Tasks t_1 and t_2 are mapped on processor p_1 and tasks t_3 and t_4 on processor p_2. The mutual exclusion of the execution of tasks mapped on the same processor is modelled by means

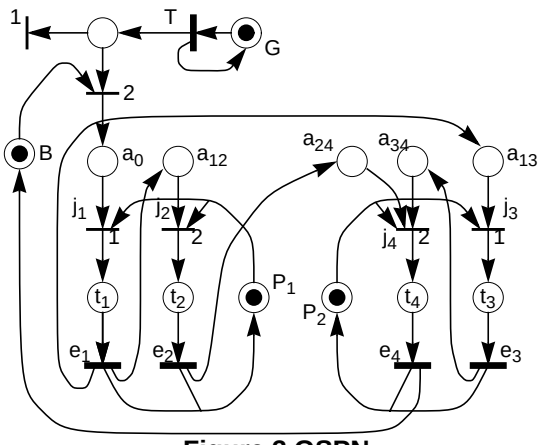

Figure 3 GSPN

of the places P_1 and P_2. The task execution is modelled by means of the timed transitions e_1, ..., e_4. The task priorities are modelled by prioritizing the immediate transitions j_1, ..., j_4. This results in having a Petri Net where the firing of a timed transition in a given marking leads to exactly one next marking, which can be unambiguously determined. In the underlying stochastic process, there will be at most one transition labelled with a task in a given process state. Another important property that is easily detected in the Petri Net is that the net does not contain competitively enabled transitions. Consider the states S_1 and S_2 in the underlying stochastic process and the corresponding sets of tasks, W_1 and W_2 that run in the two states. If, by firing a transition e_k (executing a job of task t_k), the state changes from S_1 to S_2, then $W_1 \setminus \{t_k\} \subseteq W_2$.

The continuous new generation of jobs is modelled by means of place G and transition T, a timed transition with the firing delay equal to the greatest common divisor of task graph periods. The initial marking of place B indicates the tolerated maximum number of late jobs in the system and is controllable by the designer.

7 Analysis

This section details how to construct the infinitesimal generator M of the CTMC with the Coxian distributions starting from the infinitesimal generator Q of the CTMC C where all the execution times probabilities are assumed to be purely exponentially distributed.

The events that may trigger a state transition in C are the arrivals and departures of task instantiations (jobs). We group the states of C in clusters. Two states belong to the same cluster if the same set of events may trigger transitions out of the states. Let R_i denote the set of tasks mapped on processor i, $R_i = \{t : Map(t) = p_i\}$, $i = \overline{1,s}$. The maximum number of clusters is then,

$$\prod_{i=1}^{s} (|R_i| + 1)$$

where $|R_i|$ is the cardinality of R_i. In the worst case, the number of clusters is

$$\left(\frac{N}{s} + 1\right)^s$$

but in reality it is smaller because some tasks may not run concurrently due to data dependencies even if they are mapped on different processors.

The rows and columns of the Q matrix are then shuffled such that the states in the same cluster are consecutively numbered.

Consider an application with four independent tasks, each mapped on a different processor. Figure 4 depicts the matrix Q in such case. The rows and columns in the figure do not correspond to individual rows and columns in Q, but to *clusters* of states. The row labelled with the binary number 1101, for example, indicates that the tasks 1, 3, and 4 are running in the states belonging to the cluster. The shaded cells indicate those submatrices that may contain

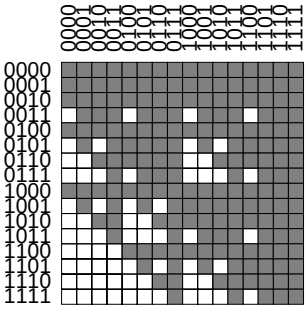

Figure 4 The Q matrix

non-zero elements. The blank ones are null submatrices. One such null submatrix appears for example at the intersection of row 1101 and column 1000. Due to the non-preemption assumption, a task arrival or departure event may not stop the running of another task. If the submatrix (1101, 1000) had non-zero elements it would indicate that an event in a state where the tasks 1, 3, and 4 are running triggers a transition to a state where only the task 4 is running and *two* of the previously running tasks are not running anymore. This is not possible in the case of non-preemption. In the case of k tasks, there are at most $3^{k-1}(k+3)$ non-zero submatrices out of 4^k.

When using the Coxian approximation, a set of new states is introduced for each state in C. We will illustrate the construction of a cell in M from a cell in Q based on an example. We consider a cell on the main diagonal as it is the most complex case.

Figure 5 depicts three states in C. Two tasks, u and v, are running in the states X and Y. These two states belong to the same cluster, 11. Only task v is running in state Z. State Z belongs to cluster 10. If task v finishes running in state X, a transition to state Y occurs in C. When task u finishes running in state X, a transition to state Z occurs in C. Consider that the probability distribution of the execution time of task v is approximated with a three stage Coxian distribution and that of u with a two stage Coxian distribution. The two approximations are depicted in Figure 5 and their arcs were labelled. The resulting stochastic process is depicted in Figure 6. The labels on the transitions in Figure 6 indicate which transition inside the Coxian distribution triggers the particular transition in the expanded Markov chain. Each state in C is replaced by a set of states in the expanded chain and appears in a shaded background in Figure 6. The number of states in such a set is given by the expression

$$\prod_{i \in E} r_i$$

where E is the set of tasks running in the state to be expanded, and r_i indicates the number of stages in the corresponding Coxian distribution. We construct the cell

Figure 5 Part of C

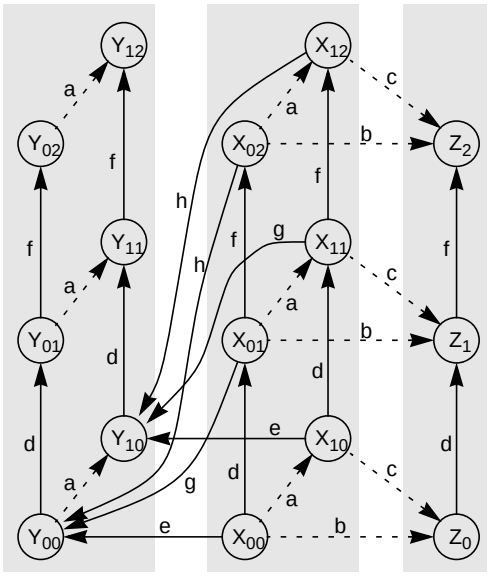

Figure 6 Expanded Markov chain

on the main diagonal, on the row and column of corresponding to the cluster 11, composed of the states X and Y. The observed regularity in the interconnection structure of the set of states is reflected in the expression of the incidence matrix of the cluster 11 as shown in Figure 8. This matrix can be compactly written as

$$(A_u \oplus A_v) \otimes I_{|11|} + I_{r_u} \otimes B_v \otimes e_{r_v} \otimes D_v$$

where

$$A_v = \begin{bmatrix} 0 & d & 0 \\ 0 & 0 & f \\ 0 & 0 & 0 \end{bmatrix}, B_v = \begin{bmatrix} e \\ g \\ h \end{bmatrix}, e_{r_v} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, D_v = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

and A_u and B_u are similarly defined, |11| denotes the size of the cluster 11 and $\oplus$ and $\otimes$ are the Kronecker sums and products for matrices. In order to obtain a cell in the generator, the labels in the matrices above have to be replaced with the corresponding transition rates. Thus

$$d \leftarrow (1-\alpha_{v1}) \cdot \mu_{v1},\ f \leftarrow (1-a_{v2}) \cdot \mu_{v2},$$
$$e \leftarrow \alpha_{v1} \cdot \mu_{v1},\ g \leftarrow \alpha_{v2} \cdot \mu_{v1},\ h \leftarrow \mu_{v3}$$

In general, a matrix $A_k=[a_{ij}]$ is a $r_k x r_k$ matrix, and is defined as follows:

$$a_{ij} = \begin{cases} (1-\alpha_{ki}) \cdot \mu_{ki} & j = i+1 \\ 0 & j \neq i+1 \end{cases}$$

A $B_k=[b_{ij}]$ matrix is a $r_k x 1$ matrix and $b_{i1}=\alpha_{ki}\mu_{ki}$. An

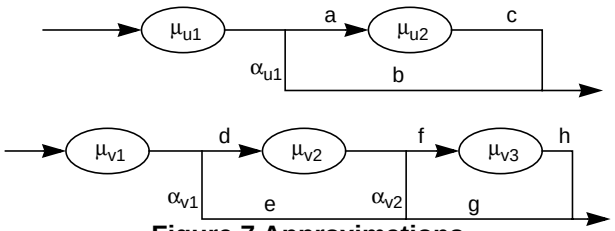

Figure 7 Approximations

	X_{00}	Y_{00}	X_{01}	Y_{01}	X_{02}	Y_{02}	X_{10}	Y_{10}	X_{11}	Y_{11}	X_{12}	Y_{12}
X_{00}		e	d				a					
Y_{00}				d				a				
X_{01}		g			f				a			
Y_{01}						f				a		
X_{02}		h									a	
Y_{02}												a
X_{10}								e	d			
Y_{10}										d		
X_{11}								g			f	
Y_{11}												f
X_{12}								h				
Y_{12}												

Figure 8 Incidence matrix for cluster 11

$e_{rk}=[e_{ij}]$ matrix is a $1xr_k$ matrix and $e_{11}=1$, $e_{1i}=0$, $1<i\le r_k$. A matrix $D_k=[d_{ij}]$ corresponding to a cluster U of size |U| is a |U|x|U| matrix defined as follows:

$$d_{ij} = \begin{cases} 1 & \text{an arc labelled with k leads from i to j in C} \\ 0 & \text{otherwise} \end{cases}$$

Because the cell belongs to the main diagonal, some correction elements have to be introduced such that the entire generator is truly a stochastic matrix (the sum of elements on the rows is 0). The formula for computing a cell on the main diagonal corresponding to a cluster U is given below:

$$(\otimes_{i\in U} I_{r_i}) \otimes (Q_U - \sum_{i\in U} \mu_i D_i) + \oplus_{i\in U} A_i' + \sum_{i\in U} (\otimes_{\substack{j\in U \\ j<i}} I_{r_j}) \otimes B_i \otimes e_{r_i} \otimes (\otimes_{\substack{j\in U \\ j>i}} I_{r_j}) \otimes D_i$$

where μ_i is the transition rate in the artificially constructed chain C and Q_U is the restriction of Q to the states in U.

The matrix $A_k'=[a_{ij}']$ is derived from $A_k=[a_{ij}]$ as follows:

$$a'_{ij} = \begin{cases} a_{ij} & i \neq j \\ \mu_k - \mu_{ki} & i = j \end{cases}$$

We will discuss the construction of the cell at the intersection of the row corresponding to cluster U with the column corresponding to cluster V, U being different from V (cell not on the main diagonal). In this case, the cell will be of the following form:

$$\sum_{i\in U} (\otimes_{j\in V\cup\{i\}} F_{ij}) \otimes D_i$$

The matrices F are given by the following expression:

$$F_{ij} = \begin{cases} e_{r_j} & j \notin U \\ I_{r_j} & j \in U \wedge j \neq i \\ B_i & j \notin V \wedge j = i \\ B_i \otimes e_{r_i} & j \in V \wedge j = i \end{cases}$$

Hence, we may express the generator M of the expanded Markov chain in terms of the generator Q of C, A_i, B_i, and D_i, $1\le i\le N+1$ (+1 for the transition T in Figure 3). All these matrices are both sparse and of small size. Having M as a sum of Kronecker products of matrices allows us not to store M explicitly in memory while performing the analysis.

8 Conclusions and future work

We presented an approach to performance analysis of multiprocessor schedules of tasks with variable execution times. The real-life probability distributions of the execution times are approximated with Coxian distributions, and the expanded underlying Markov chain is constructed in a memory efficient manner exploiting the structural regularities of the chain.

The future work attempts at finding an analytic expression for the number of non-null elements in M, as they determine the analysis speed. Experiments have to be run in order to assess the size of the problems that can be treated and the possible trade-offs between problem size, and approximation accuracy.

References

[1] G. Balbo, G. Chiola, G. Franceschinis, G. M. Roet “On the Efficient Construction of the Tangible Reachability Graph of Generalized Stochastic Petri Nets”, Proc 2nd Workshop on Petri Nets and Performance Models, pp. 85-92, 1987

[2] D.R. Cox “A Use of Complex Probabilities in the Theory of Stochastic Processes”, Proc. Cambridge Philosophical Society, pp. 313-319, 1955

[3] R. Ernst “Codesign of Embedded Systems: Status and Trends”, IEEE Design & Test of Computers, April-June, 1998, pp. 45-54

[4] C. Fidge “Real-Time Schedulability Tests for Pre-emptive Multitasking”, Real-Time Systems, 14, 61-93 (1998)

[5] J.M. Harrison, V. Nguyen “Brownian Models of Multiclass Queueing Networks: Current Status and Open Problems”, Queueing Systems 13 (1993) 5-40

[6] J. Kim, K.G. Shin, “Execution Time Analysis of Communicating Tasks in Distributed Systems”, IEEE Trans. on Computers, 45 No. 5, May 1996

[7] J.P. Lehoczky “Real-Time Queueing Theory”, Proc. of the 17th IEEE RTSS (1996)

[8] J.P. Lehoczky “Real-Time Queueing Network Theory”, Proc. of the 18th IEEE RTSS (1997)

[9] J.Y.-T. Leung, J. Whitehead “On the Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks”, Performance Evaluation, 2(4):237-250, Dec. 1982

[10] S. Manolache, P. Eles, Z. Peng “Memory and Time Efficient Schedulability Analysis of Task Sets with Stochastic Execution Time”, Euromicro Conf. on RTS (2001).

[11] B. Plateau, J-M. Fourneau “A Methodology for Solving Markov Models of Parallel Systems”, J. of Parallel and Distributed Computing, 12, 1991

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, “Numerical Recipes in C”, Cambridge Univ. Press, 1992

[13] J. A. Stankovic, M. Spuri, M. Di Natale, G. Butazzo, “Implications of Classical Scheduling Results for Real-Time Systems”, IEEE Computer, June 1995

[14] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, J.W.-S. Liu “Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times”, Proc. of IEEE Real-Time Techn. and Applic. Symp., 1995

[15] R.J. Williams “Diffusion Approximations for Open Multiclass Queueing Networks: Sufficient Conditions Involving State Space Collapse”, Queueing Systems 30 (1998) 27-88