Deadline Miss Rate Analysis of Applications with Stochastic Task Execution Times

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

Abstract

The expected fraction of missed deadlines is an important performance indicator of applications with stochastic task execution times. Obtaining this indicator is a challenging endeavour especially for multiprocessor applications. In this paper, we propose two analysis approaches that trade the analysis accuracy for analysis speed and memory in a designer-controlled way. The more accurate approach is applicable to the one-time design validation phase, while the faster approach can be plugged into an optimisation loop which explores several design alternatives during system synthesis. Experiments demonstrate the applicability and efficiency of the proposed approximate analysis methods.

1 Introduction

Reactive real-time systems [2], composed of possibly communicating tasks that respond to stimuli, have to deliver their response typically within a prescribed time interval from the stimulus arrival time. The validation of this timeliness property is done by schedulability analysis [1] among other methods, such as simulation and formal verification. Typical worst case analysis validates or invalidates the system while considering that all tasks execute a fixed, worst case execution time. In the more realistic case however, when task execution times are variable and their probability distributions given, the analysis computes the expected rate at which the timeliness property is not satisfied or equivalently the fraction of missed deadlines. In either case, deterministic or stochastic task execution times, the validation analysis is very important as deadline misses could have catastrophic consequences [12] or could significantly degrade the system quality.

Besides system validation, analysis plays an important role during design space exploration. Design transformations, such as decisions about task assignment to processors or priority assignment to tasks are driven by the analysis that assesses their impact on performance indicators such as deadline miss rates.

In previous work, we proposed an exact[1] schedulability analysis approach that can be efficiently applied to monoprocessor systems [7]. As exact analysis approaches are prohibitive in terms of consumed resources (analysis time and memory) in the case of multiprocessor systems, we present two approximate analysis strategies that trade accuracy for speed and reduced memory in a designer-controlled way. The first proposed approach is rather accurate and applicable to the one-time validation phase at the end of the high-level design. The second approach is less accurate but much faster and it is intended to be plugged into a design space exploration loop.

The two approaches are presented in dedicated sections, following Section 2 which gives the common problem formulation. The last section draws the conclusions.

2 Application modelling

The hardware architecture is modelled as a set of processors P_1, P_2, …, P_M, a set of buses, and the corresponding interconnection topology.

The application is modelled as a set of tasks τ_1, τ_2, …, τ_N. Each task τ_i, $1 \le i \le N$, is characterised by its period π_i, its deadline δ_i, its priority, its mapping $m(\tau_i)$, i.e. the processor on which every job of task τ_i executes, and its execution time probability density function (ETPDF) ε_i. The execution times of any two jobs (of the same or of different tasks) are assumed statistically independent. Jobs are dispatched for execution by a runtime scheduler according to the static priority of the task to whom the job belongs. The execution of jobs is assumed non-preemptive.

There may exist data dependencies among the tasks. For all pairs of tasks $\tau_j \rightarrow \tau_i$, where τ_i is data dependent of τ_j we assume that π_j divides π_i and that the k^{th} job of task τ_i may execute only after the jobs $\pi_i/\pi_j \cdot (k-1), \pi_i/\pi_j \cdot (k-1) + 1, \ldots, \pi_i/\pi_j \cdot k-1$ of task τ_j have completed their execution. The symmetric and transitive closure of the dependence relation between tasks partitions the set of tasks into task graphs.

The analysis solves the following problem: Given a hardware architecture and an application under the assumptions listed above, find the expected fraction of missed deadlines, $\lim_{t\to\infty} \pi_i \cdot M_i(t)/t$, for each task τ_i, where $M_i(t)$ is the number of missed task deadlines during the interval $[0, t)$.

1.In the sense that the obtained performance indicator (e.g. deadline miss probability) is exact and not an approximation

3 Analysis based on Coxian approximation

Because the task execution times are stochastic variables, the behaviour of the entire system is random and can be characterised by the stochastic process underlying the system.

The process has to be constructed and analysed in order to extract the desired performance metrics. When considering arbitrary execution time probability distribution functions (ETPDFs), the resulting process is a generalized semi-Markov process (GSMP), making the analysis extremely demanding in terms of memory and time. If the execution time probabilities were exponentially distributed, the process would be a continuous-time Markov chain (CTMC) which is easier to solve.

As a first step, we generate a model of the application as a Generalized Stochastic Petri Net (GSPN). We use this term in a broader sense than the one defined by Balbo [3], allowing arbitrary probability distributions for the firing delays of the timed transitions. More details on this step are found in our previous work [9]. The translation task graph→GSPN is automatically made in O(N) time, where N is the number of tasks. The tangible reachability graph (TRG) [3] of the GSPN is isomorphic to the generalised semi-Markov process underlying the application.

The second step implies the approximation of the arbitrary real-world ETPDFs with Coxian distributions [4], i.e. weighted sums of convoluted exponentials. A Coxian distribution is depicted in Figure 1. The circles represent exponentially distributed execution times, while the dashed box represents the entire Coxian execution time. A task having the Coxian execution time shown in the figure, would execute an exponentially distributed time interval with an average rate of μ_1, and then it would finish its total execution with probability α_1. With probability $1–\alpha_1$, the task execution would enter a second stage, also an exponentially distributed stage with an average rate of μ_2, and it would finish its total execution with probability α_2. With probability $1–\alpha_2$ it would enter a final third stage, that is exponentially distributed with an average rate of μ_3.

The Laplace transform of the probability density of a Coxian distribution with r stages is given below:

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

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. Practically, the approximation problem can be formulated as follows: given an arbitrary

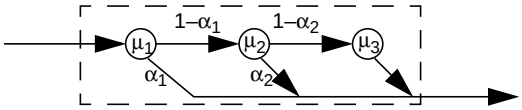

Figure 1. Coxian distribution

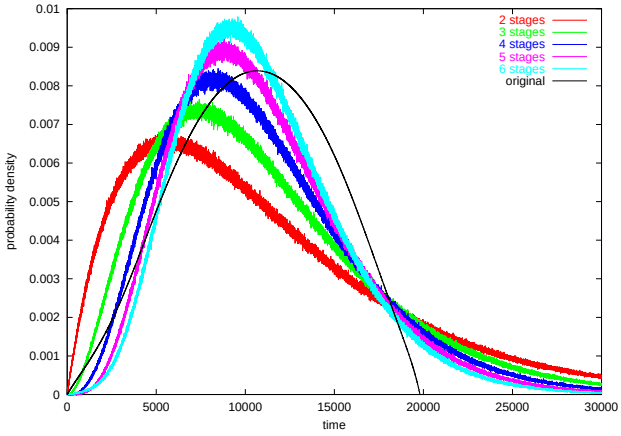

Figure 2. Coxian approximation

probability distribution, find μ_i, $i=\overline{1,r}$, and α_i, $i=\overline{1,r-1}$ ($\alpha_r=1$) such that the quality of approximation of the given distribution by the Coxian distribution with r stages is maximized. This is usually done in the complex space by minimizing the distance between the Fourier transform $X(j\omega)$ of the Coxian distribution and the computed Fourier transform of the distribution to be approximated. The minimisation is a typical interpolation problem and can be solved by various numerical methods [10]. We use a simulated annealing approach that minimizes the difference of only a few most significant harmonics of the Fourier transforms which is very fast, if provided with a good initial solution. We choose the initial solution in such way that the first moment of the real and approximated distribution coincide.

Figure 2 shows the Coxian approximation with two to six stages of a generalised ETPDF.

By replacing all generalized transitions of the GSPN with Coxian subnets containing only transitions with exponentially distributed firing delays, the GSMP underlying the Petri Net becomes a CTMC. It is obvious that the introduced additional places of the subnets trigger an explosion in the TRG and implicitly in the resulted CTMC. However, instead of storing large sets of samples of arbitrary distribution functions, only the average firing rates for each exponential transition needs to be stored and manipulated during the analysis process. Moreover, classic numerical techniques like the power method of the Jacobi method [11] can be used for solving the CTMC. However, the biggest advantage is that the newly introduced states of the CTMC form regular structures. Therefore, the elements of the infinitesimal generator of the CTMC do not have to be stored in memory but they are generated at analysis time. In this way, larger applications can be analysed.

We will illustrate the above mentioned property using an example. Let us consider three states in the GSMP as depicted in Figure 3. Two tasks, u and v, are running in the states X and Y. Only task v is running in state Z. If task v

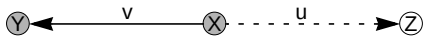

Figure 3. Part of a GSMP

finishes running in state X, a transition to state Y occurs in the GSMP. This corresponds to the situation when a new instantiation of v becomes active immediately after the completion of a previous one. When task u finishes running in state X, a transition to state Z occurs in the GSMP. This corresponds to the situation when a new instantiation of u is not immediately activated after the completion of a previous one. Consider that the probability distribution of the execution time of task v is approximated with the three stage Coxian distribution and that of u is approximated with the two stage Coxian distribution. The resulting CTMC corresponding to the GSMP in Figure 3 is depicted in Figure 4. The edges between the states are labelled with the average firing rates of the transitions of the Coxian distributions. As seen, there are many more states in the approximating CTMC than in the original GSMP. However, due to the regularity of the structure of the chain, its infinitesimal generator is expressed as a sum of Kronecker products of very small matrices as we have shown in our previous work [9]. These small matrices have a dimension equal to the number of stages of the approximating Coxian distributions, that is typically in the range 2 to 6. Only the very small matrices are stored in memory while the large infinitesimal generator is generated on-the-fly at analysis time, according to the Kronecker products.

In order to assess the proposed analysis method, we performed a set of experiments. The results are presented in more detail in the referred paper [9]. We observed a linear increase of the analysis time with the number of tasks, an exponential increase of the analysis time with the number of processors and with the average number of stages of the Coxian distributions. For applications consisting of 60 tasks mapped on 2 processors, the analysis took 2300 sec. on average. Applications consisting of 10 tasks mapped on 6 processors were analysed in 5200 sec. on average. Table 1 shows the accuracy of our approach as a function of the number of stages of the approximating Coxian distributions. The exact numbers for the deadline miss rates were obtained from the exact analysis approach that we previously proposed [7] and that is efficient for monoprocessor systems. As seen, good accuracy levels can be obtained even for a small number of stages, but, of course, they depend on the shape of the original ETPDFs.

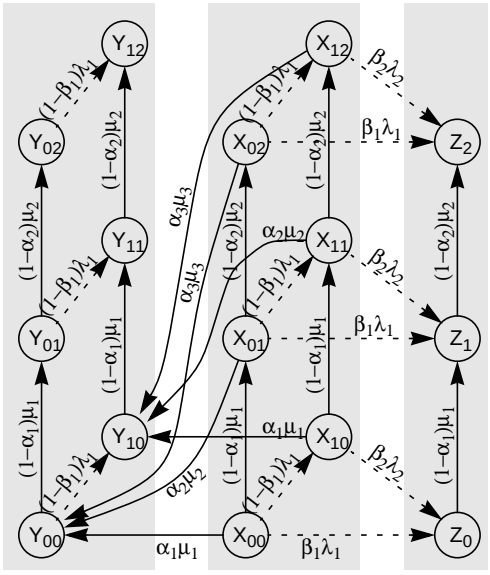

Figure 4. Expanded Markov chain

Table 1. Accuracy vs. no. of stages

	2 stages	3 stages	4 stages	5 stages
Relative error	8.467%	3.518%	1.071%	-0.4%

4 Fast approximate analysis

The analysis based on Coxian approximation, described in the previous section, is too slow to be plugged into an optimising loop driving a design space exploration process such as a task-to-processor mapping or a task priority assignment heuristic [8]. In this section, we propose a faster but less accurate deadline miss rate analysis method. The basic idea is to sweep over the time axis from 0 to the least common multiple (LCM) of task periods, and to approximate the state of the system at each time based on approximations at previous time points. In this context, the state of the system at a time t is given by a vector of probabilities, p_i, $1 \le i \le N$, where p_i is the probability that task τ_i is running at time moment t (the instantaneous processor load caused by task τ_i at time t). The probability that a task misses its deadline is given by the corresponding element of the systems state at the time of the deadline.

As a first approximation, only a discrete set of time moments $t_1, t_2, \ldots$ in the interval [0, LCM) are selected, and the density of these time moments is designer-specified depending on the desired accuracy.

A second approximation is used when computing the probability that a task with two or more predecessors is ready to execute prior to time t_n, denoted $P(A_i \le n)$. If all the finishing times of the predecessor tasks were statistically independent among themselves, we could write.

$$P(A_i \le n) = \prod_{\sigma \in Pred(\tau)} P(F_i \le n)$$

If any two predecessor tasks of task τ_i have a common predecessor task or if any of the ancestor tasks of task τ_i share the same processor, the independence assumption does not hold. However, as shown by Li [6] and Kleinrock

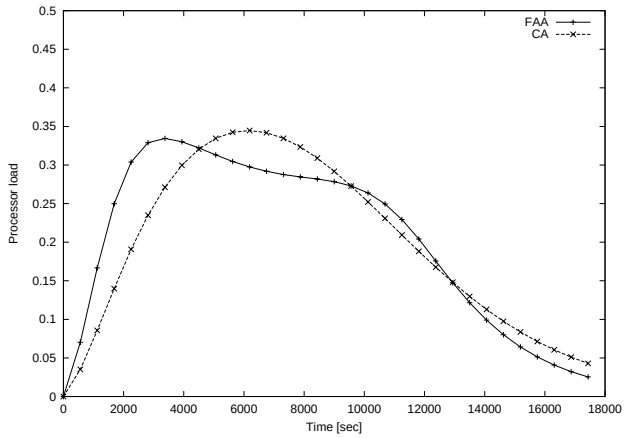

Figure 5. Approximation accuracy

[5], the dependence is weak enough to accept the equation as being a reasonable approximation.

Last, we approximate the probability that task τ_i is running at time t_n knowing that task τ_j has arrived prior to time t_n with the probability that task τ_i is running at time t_n ($P(L_i(n) \mid A_j \le n) = P(L_i(n))$. More details and an illustrative example are found in previous work [8].

We use these three approximations to compute $P(A_i \le n)$, $P(S_i \le n)$, $P(F_i \le n)$, $P(L_i(n))$ at each considered discrete time moment, where $P(A_i \le n)$ is the probability that task τ_i has arrived prior to time moment t_n, $P(S_i \le n)$ is the probability that task τ_i has started prior to time moment t_n, $P(F_i \le n)$ is the probability that task τ_i has finished prior to time moment t_n, and $P(L_i(n))$ is the probability that task τ_i is running at time moment t_n. These probabilities are computed based on the following formul ae:

$$P(F_i = n \mid S_i = k) = P(E_i = n - k)$$

$$P(L_i(n) \mid S_i = k) = P(E_i > n - k)$$

$$P(S_i = n) = (P(A_i \le n) - P(S_i < n)) \cdot \left(1 - \sum_{\sigma \in MT} P(L_\sigma(n))\right)$$

where MT is the set of tasks mapped on the same processor with task τ_i, and E_i is the execution time of task τ_i.

In order to assess the accuracy of the proposed fast approximate analysis (FAA), we compared the processor load curves obtained by FAA with processor load curves obtained by the analysis method described in the previous section (CA). The benchmark application consists of 20 processing tasks mapped on 2 processors and 3 communication tasks mapped on a bus connecting the two processors. Figure 5 gives a qualitative measure of the approximation. It depicts the two processor load curves for a task in the benchmark application. A quantitative measure of the approximation is given in Table 2. We present only the extreme values for the average errors and standard deviations. Thus, row 1 in the table, shows the largest obtained average error, while row 2, shows the smallest obtained average error. Row 3, shows the worst obtained standard deviation, while row 4, shows the smallest obtained standard deviation. The average of standard deviations of errors over all tasks is around 0.065. Thus, we can say with 95% confidence that FAA approximates the processor load curves with an error of ±0.13. The analysis time grows linearly with the number of tasks. For a set of benchmark applications consisting of 40 tasks mapped on 3 to 8 processors, the average analysis time was 3ms.

Table 2: Approximation accuracy

Task	Average error	Standard deviation of errors
19	0.056351194	0.040168796
13	0.001688039	0.102346107
5	0.029250265	0.178292338
9	0.016695770	0.008793487

5 Conclusions

In this paper we have proposed two analysis methods for obtaining the deadline miss rate for real-time applications with stochastic task execution times. Both methods manage complexity by trading result accuracy for required analysis resources.

References

[1] N.C. Audsley, A. Burns, R.I. Davis, K.W. Tindell, A.J. Wellings, “Fixed priority pre-emptive scheduling: a historical perspective“, J. of Real-Time Systems, 8(2-3), 1995, pp. 173-198

[2] A. Burns, A.J. Wellings, “Real-time systems and programming languages”, Addison-Wesley, 2001

[3] 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

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

[5] L. Kleinrock, “Communication Nets: Stochastic Message Flow and Delay”, McGraw-Hill, 1964

[6] Y.A. Li, J.K. Antonio, “Estimating the execution time distribution for a task graph in a heterogeneous computing system”, 6th Heterogeneous Computing Workshop, HCW 97

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

[8] S. Manolache, P. Eles, Z. Peng, “Optimization of Soft Real-Time Systems with Deadline Miss Ratio Constraints”, RTAS04, pp. 562-570

[9] S. Manolache, P. Eles, Z. Peng, “Schedulability Analysis of Multiprocessor Real-Time Applications with Stochastic Task Execution Times”, ICCAD 2002, pp 699-706

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

[11] W.S. Stewart, “Introduction to the Numerical Solution of Markov Chains”, Princeton Univ. Press, 1994

[12] N. Storey, “Safety-Critical Computer Systems”, Addison-Wesley, 1996