Off line Scheduling of Applications with Variable Duration Tasks

Stéphane PAILLER and Annie CHOQUET-GENIET

LISI, ENSMA
Téléport 2 1 Avenue Clément Ader
B.P. 40109 F 86961 Futuroscope Cedex France
{pailler,ageniet}@ensma.fr

SUPERVISOR: Annie CHOQUET-GENIET.

KEYWORDS: variable duration, conditional instructions, local scheduling, global scheduling, scheduling graph, Petri Nets, off-line scheduling.

Abstract. We propose an off-line scheduling methodology for real time applications including variable duration tasks, the variations coming from the presence of conditional instructions in the body of the tasks. After adapting the task temporal model to this context, we model these applications using autonomous Petri nets functioning under the earliest firing rule with terminal set marking. We define two concepts of schedulability: the local schedulability and the global ones and we define the concept of schedulability graph. Then, we show how to obtain a scheduling graph starting from the graph of global schedulability.

1 Research area

We are concerned with the scheduling for real time applications including tasks with variable durations. These durations being may depend on the state of the process controlled by the application, of the position of a parameter compared to a threshold, inducing different processing according to whether it's located above or below the threshold... Different kinds of imprecise duration tasks were already studied, quote the multi representation model [CHB79][Che91], which associates to a task two implementations, one corresponding to an optimal service quality, but with a dubious duration, and the other, with a known duration, but corresponding to a service of less quality, and the incremental model [CL88], where a task is broken up into two parts, an essential part which must absolutely be carried out, and a secondary part, which aim is to refine the produced results, which will be computed if the serviceable time is sufficient. The approach which we adopt is completely different, since we consider duration fluctuations related to the presence of junctions in the task code. They are thus structural variations, in opposition to conceptual variations mentioned above.

We consider applications composed of hard periodic tasks. Each task is characterized by four temporal parameters [LL73][SSR98]: its first release time r_i,

its period T_i, its relative deadline D_i(which is the size of the temporal window inside which each instance of the task must be computed) and its computation time. This duration is generally supposed to be known and determinist, this determinism being coming from the consideration of the worst case: for each task, we consider the processor maximum execution time. However, during the application life, some task instances may have shorter durations. A first problem arising from this worst case consideration is the instability of the online classical scheduling algorithms (Earliest Deadline (ED) [LL73], Rate Monotonic (RM) [LL73][LSD89], Least Laxity (LL) [MD78]): this means that if critical resources are used, the worst case in term of schedulability does not correspond to maximum effective durations. The decreasing of the duration of a task can lead to it lateness temporal faults. Furthermore, this approach can correspond to a non-realistic vision of the application. Indeed, a cause of fluctuation of the execution times is the presence within the task body, of conditional instructions which have distinct duration branches. The optics of the worst case transforms, from the temporal point of view, the conditional blocks into fixed duration blocks, which duration is equal to the longest duration path [Bab96] which moreover include all real time primitives present in one or the other of the branches. The specificity of the conditional instructions is thus erased, and the conditional management of the resources and synchronization is obliterated.

Even if we direct towards off-line scheduling method, which can be preferable when we consider highly coupled applications, because they are more powerful [GCG02], the sight of the worst case can lead to nonrealistic results: produced sequences do not correspond to effective executions, and the constraints taken into account are much stronger than the effective ones.

Our objective is to propose a more realistic vision of such applications, while considering in an explicit way the conditional blocks. We base our methodology on a Petri nets modelling enlarging the method proposed in [CGC96][GCG00a] for applications constituted of fixed duration periodic tasks. This methodology deals with synchronous (release time are all equal) or asynchronous tasks (the first release times are not all the same) using critical criticize mono or multi instance resources, possibly in reader/writer mode, and communicating.

2 Directions of the work

2.1 The temporal model

We consider real time applications compounded of synchronous periodic tasks. They can use non-preemptive resources and communicate. These tasks consist of functional blocks written in a high-level language and of real time primitives (lock and unlock resources, emission/reception of messages). When task duration are known and fixed (or if we adopt the worst case point of view), the temporal model consists of four classical temporal parameters r_i,C_i,D_i,T_i [LL73].

Our objective is to refine the functional block description in order to take conditional instructions into account, which requires to modify the temporal

model of tasks.We extend the Liu-Layland model in order to describe all the durations of the conditional branches: a task is represented by three deterministic parameters (the date of the first release, the relative deadline and the period) and by a **multi-set** D of durations, each one corresponding to a possible behavior of the task. Let us note that if there is no conditional instruction, the multi-set D contains a single duration, which corresponds to the usual model.

2.2 Scheduling graph

Considering on line scheduling strategies, the presence of conditional instructions matters only for the application validation, and so can be used directly. On the contrary, off line scheduling strategies must be adapted. If we use the deterministic model, the objective of an off line scheduling strategy is to build one or more valid schedules. This cannot be used in our case, because the various choices in conditional instructions will induce different behaviors of the application, which could not be described in a single schedule. In order to describe the behavior of a conditional application, we define the concept of scheduling graph: it is a graph where each branch corresponds to a schedule of the application obtained by considering for each task only one of their paths. We call **split sub-application** each of these applications. We introduce two concepts of schedulability: An application is said to be locally schedulable if each one of its split sub-application is schedulable, i.e. there is for each one of them a valid schedule, An application is said to be globally schedulable if there is at least one valid scheduling graph, i.e. all deadlines are respected whatever the conditional choices.

An globally schedulable application is locally schedulable. This comes from the fact that each branch of a scheduling graph is a valid schedule for one split sub-application.

2.3 Petri Nets modelling

The schedulability analysis methodology which we proposed relies on [CGC96] [GCG00a]. It consists in modelling the application by a constrained marking colored Petri nets, under the earliest firing rule. The feasible schedules are then obtained through the construction of the state graph. The model includes two parts: the task system which is obtained through a classical modelling of the functional description of the application, and a clock system which models time (see figure 1). We have adopted a discrete modelling of time [Kop92][Foh94]:An external clock (RTC) counts the time in each place $Time_i$, which acts as a local clock used to release periodically the related tasks. Let us note that each transition corresponds to an action of duration one time unit, and that all transitions of the task system are in competition for obtaining the processor. It follows that, according to the earliest firing rule, at each time, one single valid transition is fired. It results, that only conservative schedules can be produced. But for the state of scheduling power, we need also to consider non work conserving schedules. For that purpose, we introduce in the task system a further task. This is

called idle task which models the inactivity of the processor. When transitions of this task fire the processor remains idle, non work conserving are then also obtained.

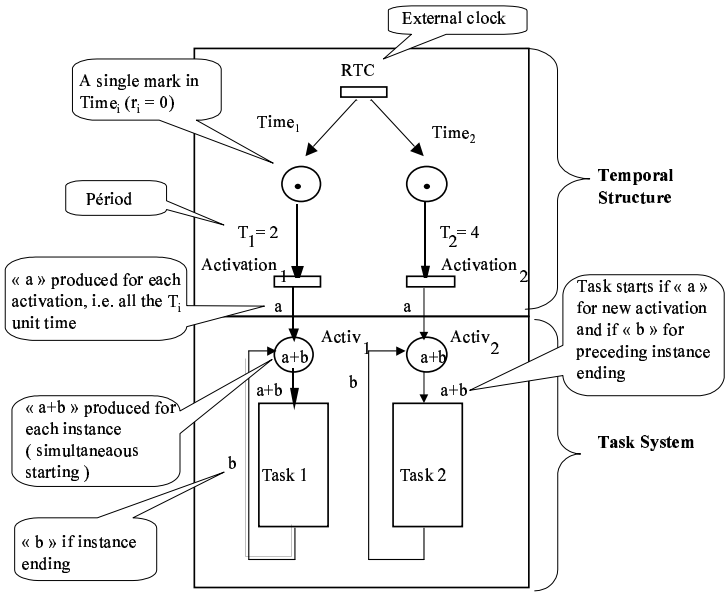

Fig. 1. Petri Nets model for an application of 2 tasks with respective periods $T_1 = 2$ and $T_2 = 4$.

The computation time of the idle task is P(1-U) (where U is the processor utilization factor of the application and P is the LCM of the periods of the tasks). Since, when conditional tasks are involved, the utilization factor U is no more deterministic, the duration of the idle task becomes also variable. In order to match the previously used rule (a transition of the task system fires each unit time), we have adapted our model so that the duration of the idle task could be modified during the simulation of the Petri Nets (roughly speaking, it's equal to its least value in the initial marking and it is incremented each time a choice of a shorter path is made within a task).

2.4 Off-line analysis

Once the model constructed, we are are interested in its exploitation in order to obtain valid scheduling graphs. From this Petri net, we build a P depth final marking graph. We first remove from it all states without successor, of depth less than P. They correspond to states of the application where fatal scheduling

decisions have been made: temporal faults can no more be avoided. We call accessibility graph the reduced graph which contains only valid states.

Besides, this graph contains all the valid scheduling graph. In order to extract them, we first need to purge the marking graph: the basic idea is that each time, a choice is made of a path in a task, the choice of the other path corresponding to the other alternative in the conditional instruction, must be possible. Else, the first path must be removed.

The last step of the analysis consists then in the extraction of the scheduling graphs (possibly according to some further criteria as the minimization of the mean responsive time of a subset of tasks or of the jitter...).

3 Results

We have proposed a scheduling methodology for variable duration tasks with strict temporal constraints. These duration variations result from strict conditional applications, modelled by Petri nets for an off-line analysis. We have defined two concepts of schedulability, coming directly from the implications of conditional tasks: local and global schedulability. We have showed that there is not equivalence between both.

In addition, we have adopted the concept of idle task, because the duration of this task is closely related to the duration fluctuations of the application tasks. We have proposed two way of modelling for the idle task allowing on one hand a production of non work conserving scheduler and on the other an optimization in the construction of the marked graph (the idle task enables to reduce considerably the number of non valid states). One of the proposed modelling will suited to the study of the local schedulability, an other to the global one.

The next step of this work consists in enlarging our methodology in order to take into account asynchronous tasks. For that purpose, we will first concentrate on the number and the location of idle times, and then on the date of the beginning of the steady running, in the context of conditional tasks [GCG00b].

An other way is to extend our methodology to sporadic tasks, induced by conditional blocks within which new tasks are requested. Our methodology must be adapted in order to take into account this new kind of task model. Our goal is then, thanks to a efficient idle time managing, to define a mixed scheduler for both periodic and sporadic tasks.

References

[Bab96] J.P. BABAU, "Étude du comportement temporel des applications temps réel contraintes strictes basées sur une analyse d'ordonnançabilité", Thèse de Doctorat, Universit de Poitiers, July 1996

[CGC96] A.CHOQUET-GENIET, D.GENIET, F.COTTET "Exhaustive Computation of the scheduled Task Execution Sequences of a Hard Real-time Application", 4th symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, Uppsala, Sweden, LNCS n 1135, p. 246-262, Springer Verlag, September 1996

[CHB79] R.H. CAMPBELL - K.H. HURTON, G.G. BELFORD, "Simulations of fault-tolerant real-time deadline mechanisms ", Proceedings of FCTS'9, p. 95-101, Jan. 1979

[Che91] H. CHETTO, M. CHETTO, "An adaptative scheduling algorithm for fault-tolerant real-time systems", Software Engineering Journal, p. 179-186, May 1991

[CL88] J.Y. CHUNG, J.W.S LIU, " Algorithms for scheduling periodics jobs to minimize average error ", 9th IEEE RTSS, p.142-152, December 1988

[Foh94] G.FOHLER, "Flexibility in statically scheduled hard real-time systems", Thesis, university of Wien , autriche, April 1994

[GCG02] E.GROLLEAU, A.CHOQUET-GENIET, "Off-line Computation of Real-Time Schedules using Petri nets.", Discrete Event Dynamic Systems, DEDS, vol. 12(4), Kluwer Academic Publishers, pp. 311-333, October 2002

[GCG00a] E.GROLLEAU, A.CHOQUET-GENIET, " Off-line computation of real time schedule by means of Petri nets ", Discrete events systems, eds. R.Boel et G. Stremersch, p. 309-316, Kluwer academic Publishers, Wodes 2000, Gent, August 2000

[GCG00b] E.GROLLEAU, A.CHOQUET-GENIET, "Cyclicité des ordonnancements de systêmes de tâches périodiques différées", RTS, P.216-231, Paris, March 2000

[Gro99] E.GROLLEAU, "Ordonnancement temps rel hors-ligne optimal l'aide de rseaux de Petri en environnement monoprocesseur et multiprocesseur. ", Thse de Doctorat, Universit de Poitiers, September 1999

[Kop92] H. KOPETZ, "Sparse time versus dense time in distributed real-time systems", 12th Int. Conf. On Distributed Computing Systems, p. 460-467, Japan, June 1992

[LL73] C.L. LIU, J.W. LAYLAND, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the ACM, 20 (1), pp 46-61, 1973

[LSD89] J. LEHOCZKKY, L.SHA, Y. DING "The rate monotonic scheduling algorithm : exact characterization and average case behavior" Proceedings of the 10th IEEE real-time systems symposium, pp 166-171, 1989

[MD78] A.K. MOK, M.L. DERTOUZOS, "Multi processor scheduling in a hard real-time environment", 7th Texas conference on computer Systems, 1978

[SSR98] J.A. STANKOVIC, M. SPURI , K. RAMAMRITHAM, G. C. BUTTAZZO, "Deadline scheduling for real-time systems", Kluwer Academic Publishers, ISBN 0 7923 8269 2, 1998