ORDONNANCEMENT HORS LIGNE D’APPLICATIONS TEMPS REEL COMPORANT DES TACHES À DUREES VARIABLES

30/09/2017
Auteurs : Stéphane Pailler, Annie Choquet-Geniet
Publication e-STA N° e-STA 2004-3
DOI : 

Télécharger
Le téléchargement implique l’acceptation de nos conditions d’utilisation

e-STA 2004-3

- Abstract
- Authors
- Metrics
- License

Abstract

Authors

Stéphane Pailler

ORDONNANCEMENT HORS LIGNE D’APPLICATIONS TEMPS REEL COMPORANT DES TACHES À DUREES VARIABLES

Annie Choquet-Geniet

ORDONNANCEMENT HORS LIGNE D’APPLICATIONS TEMPS REEL COMPORANT DES TACHES À DUREES VARIABLES

Metrics

Downloaded: 17
Viewed: 4
Size: 250.73 KB
La SEE (Société de l'Electricité, de l'Electronique et des Technologies de l'Information et de la Communication – Association reconnue d'utilité publique, régie par la loi du 1er juillet 1901) met à la disposition de ses adhérents et des abonnés à ses publications, un ensemble de documents numériques accessibles à partir de son portail des publications. Ces documents incluent notamment les articles des revues REE, 3 EI et e-STA disponibles sous forme numérique ainsi que des publications additionnelles regroupées dans l'espace eREE. Les présentes conditions précisent les conditions de diffusion et d'utilisation de ces documents et des informations qu'ils contiennent. L'accès à ces documents, qu'il se fasse de façon gratuite ou dans le cadre d'abonnements ou d'achats faits à titre onéreux, implique l'acceptation sans restriction de ces dispositions.

Droits de propriété et de diffusion des contenus téléchargés sur le portail des publications

Les contenus rendus accessibles sur le portail des publications sont, en règle générale, protégés par le droit d'auteur. En tant que producteur, et le cas échéant d'auteur, des informations rassemblées dans les contenus accessibles par ce portail, SEE se réserve l'exclusivité des droits de copie et de diffusion de tout ou partie de ces contenus.

Les contenus sont rendus accessibles à titre individuel, pour les besoins de la personne en détente des droits d'accès en cours de validité. Aussi, la modification, la reproduction et/ou la diffusion via Internet ou le Web, intranet, extranet ou toute autre forme numérique ou imprimée, de tout ou partie des contenus téléchargés sont interdites. Une tolérance est consentie quant à la reproduction d'extraits limités de ces contenus, dans le cadre de travaux ou d'activités auxquels ils sont utiles, à la condition que l'origine de ces reproductions partielles soit mentionnée de façon lisible et sans ambiguïté. Figurent en particulier : la REE (ou toute autre revue accessible sur le portail) en tant que la source, la référence de la publication et le nom de l'auteur (s'il figure dans la revue).

Ces dispositions s'appliquent également aux figures, illustrations, logos ou images.

Publication externe des contenus du portail des publications

Tout extrait des contenus du portail destiné à être utilisé dans des publicités, des communiqués de presse ou du matériel de promotion nécessite un accord préalable écrit de la SEE. Une version préliminaire du document proposé contenant ces extraits doit accompagner chacune de ces demandes. SEE se réserve le droit de refuser un tel usage externe pour quelque raison que ce soit.

Responsabilités

La SEE apporte tout le soin possible à la préparation des informations délivrées dans les contenus produits. Cependant elle ne peut être tenue pour responsable d'aucune perte ou frais qui pourrait
résulter d'imprécisions, d'inexactitudes, d'erreurs ou de possibles omissions portant sur des informations publiées, ni des résultats obtenus par l'utilisation et la pratique des informations délivrées.

**Utilisation des informations recueillies lors du téléchargement de contenu**

Le portail des publications est susceptible d'utiliser des « cookies » afin notamment de permettre l'utilisation de paniers d'achat et de personnaliser les parcours sur le site. SEE se réserve la possibilité d'utiliser les informations recueillies lors des téléchargements pour ses besoins internes et notamment pour l'amélioration de ses services, sans qu'elles puissent être cédées à des partenaires commerciaux. Conformément à la loi "informatique et libertés" du 6 janvier 1978, chaque utilisateur du portail dispose d'un droit d'accès et de rectification aux informations qui le concernent. Pour exercer ce droit, les utilisateurs doivent s'adresser à SEE – 17 rue de l'amiral Hamelin – 75783 Paris Cedex 16, par simple lettre ou en utilisant le formulaire de contact disponible sur son site.

Paris, le 28 avril 2013

.  

**ORDONNANCEMENT HORS LIGNE D'APPLICATIONS TEMPS REEL COMPORTANT DES TACHES A DUREES VARIABLES. OFF-LINE SCHEDULING OF REAL TIME APPLICATIONS WITH VARIABLE DURATION TASKS.** Stéphane Pailler 1 Annie Choquet-Geniet 2 LACS(LISI) - ENSMA Téléport 2 - 1 Avenue Clément Ader BP 40109 86960 FUTUROSCOPE cedex, FRANCE Résumé: Nous proposons une méthode d’analyse d’ordonnancement pour des applications temps réel fortement couplées. Nous montrons que les variations de durée d’exécution des tâches peuvent mettre en péril la stabilité du système contrôlé. Pour réduire les risques d’erreurs et donc augmenter la fiabilité, nous prenons en compte explicitement les instructions conditionnelles présentes dans le code des tâches. Dans un premier temps, nous adaptons le modèle temporel de tâches pour ce contexte, puis nous modélisons ces applications à l’aide de réseaux de Petri autonomes fonctionnant sous la règle de tir maximal et munis d’ensembles terminaux. Nous définissons deux concepts d’ordonnancabilité : l’ordonnancabilité locale et l’ordonnancabilité globale et nous définissons le concept de graphe d’ordonnancement. Enfin, nous montrons comment obtenir un graphe d’ordonnancement à partir du graphe d’ordonnancabilité globale. Mots-clés: Ordonnancement hors-ligne, Rseau de Petri, instructions condition-elles, graphe d’ordonnancement. Abstract: We propose a model oriented scheduling methodology for highly coupled real time applications. We show that variations in the computation times of tasks may hazard the safeness of the controlled process. For the sake of reliability we take conditional instructions of tasks’ code explicitly into account, in order to reduce the potential failures. We first adapt the task’s temporal model to this context, then we model applications using autonomous Petri nets which run under the earliest firing rule with terminal marking set. We define two concepts of schedulability: the local schedulability and the global one and we define the concept of scheduling graph. Finally, we show how to obtain a scheduling graph from the graph of global schedulability. Keywords: off-line scheduling, Petri Nets, conditional instructions, scheduling graph. 1. INTRODUCTION A constantly increasing number of either au- tonomous or assisted processes (cars, nuclear power station, plane, space probe) are controlled by real time systems interacting with their own environment. Yet these systems are not safe from failures that may put human beings’ lives in dan- ger or jeopardize substantial economic value. That is why, it is necessary to develop methods to minimize those risks by validating the real time applications that control the processes. In most cases, the failures come from disregarded tempo- ral constraints. Indeed, a real time application is defined as a multitask application in which each task is subjected to the inherent temporal constraints of the controlled processes. Thus, the main problem will be to choose a scheduling policy which would distribute each active job of task on the processor(s) so that the temporal constraints are respected. Two techniques can achieve that. The first one consists in executing a scheduling algorithm, which aim is to define at every moment the task to execute: it is the online scheduling. The second one consists in a previous analysis of the application and of the computation of valid schedules of the task executions within a temporal window of suitable size. Whatever the method, the schedulability analysis relies on a temporal model of the task (Liu and Layland, 1973; Stankovic et al., 1998) allowing to represent each task by its temporal constraints: Ri, the date of the first ac- tivation, Di the relative deadline of the execution of the task, Ti the period and Ci the computation time of the task calculated on the target hardware. Many works have made validation tools achievable by relying on heterogeneous scheduling contexts: hard and soft temporal constraints, the presence of real time primitives (resources and synchro- nizations which make the scheduling problem NP- Hard (Dertouzos and Mok, Dec. 1989), the reduc- tion of response time, of jitter etc). Yet, a very few tools are actually used or usable in the industrial world. Indeed, the inaccuracy of execution time due to both the task’s code itself and the processor that executes it, questions the relevance of the classic tools of schedulability analysis. It is the reason why we propose an off-line schedulability analysis methodology that takes the fluctuations of duration of tasks into account by relying on works
that deal with the determination of the worst case of execution times. For that purpose, we have to extend the temporal model of Liu Layland in order to take the variations coming from the conditional instructions of the tasks (an important cause of variation) into account. We used the modeling of real time application by Petri nets (Choquet-Geniet et al., 1996; Grolleau and Geniet, 2000) to analyze the feasibility of highly coupled applications (i.e. applications with a lot of resources and synchronizations), then we show how to describe valid behaviors of the application, by means of scheduling graphs and finally, we explain how to get those graphs from the analysis of the Petri Net. In a first part, we look into the difficulties of computation time estimation and the scheduling problems that it generates. Then in a second part, we introduce a new temporal model of task which takes the conditional instructions of the code into account and we reformulate the scheduling problem according to this new context. In a third part, we present the modeling of the application by autonomous colored Petri nets under the earliest firing rule and we show how to extract a scheduling graph describing a feasible behavior of the application.

3. SCHEDULING INSTABILITY PHENOMENA

In most real time applications, to find out a scheduling policy respecting constraints of the system proves to be a major issue. As an answer to this problem, we generally rely on a temporal model (Liu and Layland, 1973) that describes the temporal constraints of each task of the application in order to find and then validate a scheduling policy. But among the parameters that model a task, the execution time is probably the most complex to estimate. Those durations enable us to quantify the time allotted to the CPU to process each job of the tasks. It is thus an upper bound which is reckoned up from the task’s program code and the features of the hardware. Each tool of scheduling analysis being based on a pattern of the tasks, that upper bound (Worst Case Execution Time) represents then an input data whose accuracy of estimation will determine the quality of the scheduling policy’s validation. Therefore, researches on the WCET must take the two following preconditions into account (Puschner and Koza, 1989). First, the reckoned WCET must be safe, i.e. it has to be an upper bound whatever the execution context may be. Second, the estimation of WCET has to be as accurate as possible or instability phenomena may occur as shown in figure 1: the reduction of the duration of one of the job of the task can cause a temporal fault in spite of the validation of the scheduling policy (Deadline Monotonic 3). This kind of instability phenomena had been highlighted by (Graham, 1969). 3 DM (Deadline Monotonic) refers to the priority allocation algorithm: the task with the shortest deadline (the smallest value of D) is assigned the highest priority. 5 6 Deadline Monotonic algorithm task1<0.6,16,16> task2<0.2,6,8> task3<0.6,15,16> Using resource task1 task2 task3 0 0 Temporal fault Fig. 1. An example of a scheduling failure due to a computation time reduction. If task 3 has an effective computation time equal to 5 instead of 6, task 1 can get the resource at time 7 before task 2 is released, causing task 2 to miss its deadline. However the current methods of calculation of WCET do not all take the importance of real time primitives into account. In a context of highly coupled application, the traditional estimation of the WCET does not always correspond to the paths of execution where all the real time primitives are activated. However, as far as the analysis of schedulability is concerned, it is necessary to consider all the real time primitives that can occur. Consequently it is essential to consider the estimation of the WCET, to which should be added every real time primitive that can be activated in the various paths of the execution of the task. It seems then obvious that the resulting temporal model will be strongly over constrained what reduces the number of potential scheduling and complicates validation. To overcome that problem we suggest a method that allows to take the various paths of execution of a same task into account in order to isolate each real time primitive. Thus a task will be modeled for the analysis of schedulability by a set of execution paths. The existence of these paths is partly due to the different forms of conditional instructions present in the source code. That is why we propose to extend the temporal model of Liu Layland in order to explicitly consider the IF THEN ELSE relevant instructions that is to say those which make it possible to isolate temporal primitives or sufficiently different execution times. Thereafter we will more specifically study the influence of the conditional tests on the schedulability analysis while considering that the methods of WCET analysis allow to obtain our model of conditional task.

3. CONDITIONAL TASKS AND SCHEDULING

3.1 The extended temporal model

We consider real time applications compounded of synchronous periodic tasks (ri = 0). However let’s point out that our method can take asynchronous periodic tasks into account, but it does at the cost of a combinatorial explosion as far as the accessibility graph construction and the scheduling graph extraction are concerned. Optimization criteria need to be found to solve the problem. 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 multiset E of durations, each one corresponding to a possible behavior of the task. This new pattern of tasks is very close to the multi-frame model of (Mok and Chen, 1996). However, unlike the later, we don’t need the knowledge of the series of execution time for each task since each job of task may be processed with whatever execution time of the multiset. Let us note that if there is no conditional instruction, the multiset E contains a single duration, which corresponds to the usual model. In this temporal model, we have left the association between the real time primitives and the execution times aside. However, this information is not lost since it will be entirely integrated in the modeling of the application by Petri nets. 3.2 Scheduling graph Considering on line scheduling strategies, the presence of conditional instructions matters only for the application validation, but most of these on line algorithms can be used without any adaptation even when conditional instructions are involved, 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 introduce scheduling graphs: a scheduling graph 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 then...
formulate the scheduling problem, and define two concepts of schedulability: • An application is said to be locally schedu- lable if each 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 schedu- lable if there is at least one valid scheduling graph, i.e. all deadlines are met whatever the conditional choices. A globally schedulable application is obviously locally schedulable (but the converse does not hold if the application uses real time primitives). This comes from the fact that each branch of a scheduling graph is a valid schedule for one split sub-application. 4. MODELLING BY MEANS OF PETRI NETS The schedulability analysis methodology which we proposed relies on (Choquet-Geniet et al., 1996; Grolleau and Geniet, 2000; Grolleau, 1999). It consists in modeling the application by a con- strained marking colored Petri nets, under the earliest firing rule with terminal marking set. 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 modeling of the functional description of the application, and a clock system which models time (see figure 2). We have adopted a discrete modeling of time (Kopetz, 1992; Fohler, 1994): an external clock (RTC) counts the time in each place Timei, 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 4 can be produced. But for the sake of scheduling power since there is no optimal conservative scheduling algorithm in a scheduling context with resource and synchronization, we need also to consider non work conserving schedules. For that purpose, we introduce in the task system a further task, called idle task, which models the inactivity of the processor. When transitions of this task fire, the processor remains idle what allows to produce non work conserving schedules too. The computation time of the idle task is P(1-U) 5 . Moreover, the knowledge of the number of idle times during the metaperiod P help us for the detection of temporal fault during marking graph construct. When conditional tasks are involved, the utilization factor U is no more deterministic and the duration 4 in a conservative (or work-conserving) schedule, a task never intentionally waits 5 where U is the processor utilization factor of the appli- cation and P is the LCM of the periods of the tasks. Idle task Construction of the accessibility graph Once the model constructed, we build a P depth final marking graph since the appli- cation is cyclic with a period of P (metaperiod). This means that, we have the same states at a t-depth and at t+k×P (k>0) in the marking graph. We then remove from the marking graph external clock Task starts if « a » for new activation and if « b » for preceding instance ending « b » if instance ending « a+b » produced for (8) Activation Activation Activation Activation Activation Activation Resource Task System Temporal Structure Processor P P P P P P T T T T T T T T Time P ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ⏰ ##
l'aide de re


https://www.see.asso.fr/en/node/20054/landing