New Hybrid Genetic Based Approach for Real-Time Scheduling of Reconfigurable Embedded Systems

Ibrahim Gharbi, ENSI, Manouba University, Manouba, Tunisia
Hamza Gharsellaoui, National Engineering School of Carthage (ENIC), Carthage University, Tunis, Tunisia
Sadok Bouamama, FCTT, University of Jeddah, Jeddah, Saudi Arabia

ABSTRACT

This journal article deals with the problem of real-time scheduling of operating systems (OS) tasks by a hybrid genetic-based scheduling algorithm. Indeed, most of real-time systems are framed with aid of priority-based scheduling algorithms. Nevertheless, when such a scenario is applied to save the system at the occurrence of hardware-software faults, or to improve its performance, some real-time properties can be violated at run-time. In contrast, most of the applications of real-time systems are based on timing constraints, i.e. OS tasks should be scheduled properly to finish their execution within the time specified by the real-time systems. For this reason, the authors propose in their article, a hybrid genetic-based scheduling approach that automatically checks the systems feasibility after any reconfiguration scenario was applied to an embedded system. A benchmark example is given, and the experimental results demonstrate the effectiveness of the originally proposed genetic-based scheduling approach over other such classical genetic algorithmic approaches.

KEYWORDS

Hybrid Genetic Based Approach, Operating Systems Tasks, Real-Time Scheduling, Reconfigurable Embedded Systems

1. INTRODUCTION

Today, real-time embedded systems are one of the major parts in our industry and are found in many diverse application areas including: automotive electronics, avionics, telecommunications, space systems, medical imaging, and consumer electronics. in industry, the development of safe distributed embedded systems is not a trivial activity because a failure can be critical for the safety of human beings (e.g., air and railway traffic control, nuclear plant control, and aircraft and car control). They have classically to satisfy functional and temporal properties according to user requirements, but their time to market should be shorter than ever (Baruah, 2004).

In all of these areas, there is rapid technological progress. The software engineering principles for embedded system should address specific constraints such as hard timing constraints, limited memory and power use, predefined hardware platform technology, and hardware costs. Most of the applications of real-time systems are based on timing constraints. Real-time tasks should be scheduled properly using suitable scheduling algorithms to finish the execution of the tasks within the time specified by the real-time systems (Jinkyu, 2014). On the other

DOI: 10.4018/IJAPUC.2018010102

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
hand, the new generations of embedded control systems are addressing new criteria such as flexibility and agility (Gharsellaoui, 2012).

For these reasons, there is a need to develop tools, methodologies in embedded software engineering and dynamic reconfigurable embedded control systems as an independent discipline. The task scheduling problem has been extensively studied for multiprocessor, grid, and distributed systems and numerous heuristics have been proposed. These algorithms for traditional distributed and parallel systems, however, cannot work well in the new cloud computing environment. This makes the problem of task-scheduling in embedded systems a promising research direction and has attracted several researchers to develop efficient heuristics. Most of the work on task allocation and scheduling in the embedded systems environment is focused on the minimization of energy consumption, makespan, and economic cost of renting virtual machines and the provision of the desired level of quality of service (QoS) (Awad, 2015). However, there is not much work that explicitly deals with the allocation and scheduling of hard real-time tasks in a cloud environment to minimize execution and communication costs. Indeed, Scheduling is a process of assigning task to a processor for execution. The main objective of scheduling is to assign processor to a task while generating an efficient makespan (Total time to complete the task list) and valid schedules. The task scheduling can be further classified into static and dynamic scheduling (George, 2015).

1. **Static Scheduling**: The algorithm is also known as offline deterministic scheduler, the execution times of the task and data dependencies between them are known a priori. These types of scheduling are done during compile time therefore the run-time overhead of these schedulers are very low (George, 2015).

2. **Dynamic Scheduling**: The execution times of the task and data dependencies between them are not known a priori and thus scheduling decisions and processors are allocated on run time. Dynamic scheduling thereby furnishes a faster and flexible system (George, 2015).

Real-time tasks should be scheduled properly using suitable scheduling algorithms to finish the execution of the tasks within the time specified by the real-time systems (Jinkyu, 2014). Each real-time task has its own priority and deadline. If enough CPU time is available to execute all the tasks within their respective deadlines then those tasks are said to be schedulable tasks. The real-time scheduling algorithm can be characterized by its prioritization policies and preemption policies. The tasks' priorities can be determined by prioritization policies (Abhaya, 2014). The major classifications of priority scheduling are static priority scheduling and dynamic priority scheduling (Poongothai, 2014) such that static reconfigurations are applied off-line to apply changes before the system could start (Jinkyu, 2014), whereas dynamic reconfigurations are applied dynamically at run-time. The most common prioritization policies are EDF, FP (Fixed Priority) and LLF (Least Laxity First) (Krishna, 2010). The preemption policy determines whether a high priority task preempts the currently running low priority task or not. The preemption policies can be classified as (1) the non-preemptive policy (2) the fully preemptive policy (Abhaya, 2014). Among all prioritization policies, EDF is the most commonly used dynamic scheduling algorithm. EDF scheduling algorithm executes a job with the earliest deadline (ASK10). According to (Ballester, 2011), a hyper-period is defined as \( HP = [\xi, 2*LCM+\xi] \), where LCM is the well-known Least Common Multiple of the tasks periods and \( \xi \) is the largest task offset. This algorithm, in our original paper assumes that sporadic and aperiodic tasks span no more than one hyper-period of the periodic tasks \( HP \) and the minimization of the response time is evaluated for each reconfiguration scenario.

Each system is a subset of \( n \) tasks. Each task is characterized by its worst-case execution times (WCETs) \( C_i \), an offset (release time) \( a_i \), a period \( T_i \) and a deadline \( D_i \). The general goal of this paper is to be reassured that the system is feasible and meets real-time constraints even if we change its implementation and to correctly allow the minimization of the response time of this system after any reconfiguration scenario (Gharsellaoui, 2012). To obtain this optimization (minimization of
New Optimal Preemptively Scheduling for Real-Time Reconfigurable Sporadic Tasks Based on Earliest Deadline First Algorithm
www.igi-global.com/article/new-optimal-preemptively-scheduling-real/71886?camid=4v1a

An Adaptable Context Management Framework for Pervasive Computing
www.igi-global.com/chapter/adaptable-context-management-framework-pervasive/7118?camid=4v1a