Future Generation Computer Systems 59 (2016) 125–135

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

LBBA: An efficient online benefit-aware multiprocessor scheduling for QoS via online choice of approximation algorithms

Behnaz Sanati *, Albert M.K. Cheng

Real-Time Systems Laboratory, Computer Science Department, University of Houston, TX, USA

HIGHLIGHTS

- We propose LBBA, an online benefit-aware scheduling method for aperiodic tasks.
- Efficient strategy for better QoS in overloaded soft real-time multiprocessor systems.
- LBBA maximizes total gained benefit and minimizes the makespan at the same time.
- It uses online choice of two approximation algorithms, Greedy and Load-balancing.
- It improves QoS by maximizing the total gained benefit and reducing missed job ratio.

ARTICLE INFO

Article history:
Received 16 January 2015
Received in revised form
9 September 2015
Accepted 31 October 2015
Available online 2 December 2015

Keywords:
Real-time scheduling
Multiprocessor
Embedded system
Partitioning
Load-Balancing

ABSTRACT

Maximizing the benefit gained by soft real-time jobs in many applications and embedded systems is highly needed to provide an acceptable QoS (Quality of Service). This paper considers a benefit model for on-line preemptive multiprocessor scheduling. The goal is to maximize the total benefit gained by the jobs that meet their deadlines. This method prioritizes the jobs using their benefit density functions and schedules them in a real-time basis. We propose an online choice of two approximation algorithms in order to partition the jobs among identical processors at the time of their arrival without using any statistics. Our analysis and experiments show that we are able to maximize the gained benefit and decrease the computational complexity (compared to existing algorithms) while minimizing makespan (response time, also referred to as cost), with fewer missed deadlines and more balanced usage of processors. Our solution is applicable to a wide variety of soft real-time applications and embedded systems such as, but not limited to multimedia applications, medical monitoring systems or those with higher utilization such as bursty hosting servers.[1]

1. Introduction

Multiprocessor platforms are widely adopted for many different applications in embedded systems and server farms. They are becoming even more popular since many chip makers including Intel and AMD are releasing multi-core chips. Adopting multiprocessor platforms can enhance the system performance, but scheduling jobs optimally on a multiprocessor system is an NP-hard problem [1,2].

There are two major models for this scheduling problem. The first is the cost model; its goal is to minimize the *cost*, which is the *overall flow time*, also referred to as *response time* or *makespan*. The second model is the benefit model which aims to maximize the benefit of jobs that meet their deadlines. The latter model is used for soft real-time applications in which deadline misses are sometimes tolerable. Examples of such applications using multi-core platforms are multi-purpose home appliances such as HDTV streaming and interactive video games. More motivating examples from the domains of multimedia, air defense and enterprise-level, asynchronous, cooperating real-time computer systems are given by Welch and Brandt [3]. Other examples of such applications and embedded systems which use multi-core platforms are multimedia applications as explained in [4], image and speech processing [5–8], time-dependent planning [9], robot control/navigation systems [10,11], medical decision making [12], information gathering [13], real-time heuristic search [14], database query processing [15], and Internet of Things (IoT) [16].

In our research, we mainly focus on the benefit model to maximize the benefit of online preemptive scheduling of

* Corresponding author.
E-mail addresses: bsanati@cs.uh.edu (B. Sanati), cheng@cs.uh.edu (A.M.K. Cheng).

[1] This research is done by Behnaz Sanati and supervised by Professor Albert Cheng.

http://dx.doi.org/10.1016/j.future.2015.10.024
0167-739X/

soft real-time jobs on multiprocessor systems having identical processors. In the meanwhile, we propose an online choice of two approximation algorithms, Greedy and Load-Balancing, to reduce the cost or makespan. More balanced distribution of the jobs between processors in our novel approach results in a lower overall flow time, less total idle time, fewer missed deadlines and more efficient usage of CPU cycles. Also, in our model, the NP hard problem of multiprocessor scheduling is reduced into uniprocessor scheduling problem by partitioning the tasks at their arrival time.

In the following subsections, we provide an overview of previous works on approximation algorithms and maximizing benefit on-line for multiprocessors. Section 2 presents the details of our new approach and an example to illustrate its differences from the previous methods. The analysis of the new method is provided in Section 3. Section 4 contains the performance metrics and our experimental settings and results. Finally the conclusions of this research are presented in Section 5.

1.1. Related work on approximation algorithms

Approximation algorithms are often used to attack difficult optimization problems, such as job scheduling on multiprocessor systems which is an NP-hard problem [1,17,18]. An approximation algorithm settles for non-optimal solutions found in polynomial time, when it is very unlikely to find an efficient exact algorithm to solve NP-hard problems, or the sizes of the data sets are so large that they make the polynomial exact algorithms too expensive.

A Greedy 2-approximation algorithm is used in [19] for fault tolerance and in [20] for benefit maximization in identical multiprocessor systems. Even though Greedy approximation can be a good solution in many cases, a load balancing approximation can result in a shorter flow time for a set of jobs when we have to distribute several jobs among multiple processors at the same time [21,22]. Piel et al. [23] have proposed a load balancing technique based on statistics for real-time scheduling on asymmetric multiprocessors. They apply partitioning for high priority jobs and migration for jobs with low priority.

1.2. Related work on benefit-aware real-time computing

The gained benefit can vary when using different benefit functions. Researchers have investigated applying benefit functions for allocating resources in limited, soft real time systems [24–26]. Andrews et al. [27] provided a framework to formalize the use of benefit functions in complex real time systems.

Buttazzo et al. [28] provided the results of studying jobs that are characterized by an importance value. The performance of the scheduling algorithm was then evaluated by computing the cumulative value (or benefit) gained on a job set. However, the target of their research was uniprocessor scheduling. Welch et al. [3] discussed how benefit is used in a variety of real-time paradigms and in example applications. Awerbuch et al. presented a constant competitive ratio algorithm for a benefit model of on-line preemptive scheduling [29]. This method can be used on both uniprocessor and multiprocessor systems. Aydin et al. [30] proposed a reward-based scheduling method for periodic real-time tasks and [31] presented on-line scheduling policies for a class of IRIS (Increasing Reward with Increasing Service) real-time tasks.

2. Our contribution: online choice of approximation algorithms

The algorithm proposed in [29] only focuses on maximizing the total benefit gained without being concerned with minimizing the overall flow time of a job set (*response time* or *makespan*). In the meanwhile, in that method, the benefit gained by each job that completes its execution is calculated using the benefit density function of its flow time. This function is a non-increasing, non-negative function of time, by definition [29]. It means the more the flow time, the less the benefit gained.

Therefore, we proposed, simulated and analyzed an efficient online benefit-aware technique with choices of approximation algorithms including Greedy and load balancing to partition jobs among multiple processors at the time of release. This method prioritizes the jobs using their benefit density functions and schedules them in a real-time basis in order to reduce the makespan (overall flow time) of the jobs and total idle time of the processors while maximizing the total gained benefit.

We also used our online choice of two approximation algorithms (Greedy and Load-Balancing) as a solution for special cases that were not considered in the existing benefit-aware multiprocessor scheduling algorithms such as the *Benefit-Based Algorithm* proposed in [29] which we refer to as BBA in the rest of this paper. Examples of those cases are when there are several high priority jobs which can preempt a running job or when a high priority job can preempt more than one running job.

In order to be able to balance the workload among the processors by partitioning the jobs as soon as they are released, we consider a separate pool of the waiting jobs for each processor. Our method is referred to as *Load-Balancing/Greedy Benefit-Aware* algorithm (or LBBA) throughout the paper. This load balancing technique is different from what Piel et al. [23] have used, since we do not use statistics for distributing the jobs. Instead, we make decisions online by using the actual (worst case) execution times of ready jobs and the remaining workload of the processors for partitioning on a real-time system with identical processors. Migration is not allowed in our real-time system model.

LBBA is superior to BBA in principle, since:

- **LBBA is a novel hybrid model** of soft real-time multiprocessor scheduling. Despite of BBA, which only follows benefit model, LBBA is a combination of benefit model and cost model. That is, it aims to **minimize makespan** in order to achieve the *maximum benefit at the lowest cost*.
- **Reducing multi-processor scheduling into uniprocessor scheduling:** BBA uses one pool shared by all the processors to keep the ready jobs while they are not scheduled. Hence, partitioning of each job is done when it is scheduled, and starts running on the assigned processor. Considering a separate pool for each processor, enables the LBBA scheduler to partition the jobs at their arrival time. The advantage of this method of early partitioning is that the NP-Hard problem of multiprocessor real-time scheduling will be reduced to a series of uniprocessor real-time scheduling.
- **LBBA facilitates load-balanced partitioning** of waiting jobs, while this case is not considered in BBA. For example, in case the waiting (or ready) jobs arrive *asynchronously*, then LBBA adapts the "Greedy Approximation" to assign a job to the pool of the processor with the least remaining workload. If jobs are *synchronous*, i.e. arrive at the same time, those that cannot start running and have to wait in a pool, will be partitioned among the processors, using our "Load-Balancing" technique.
- **LBBA optimizes the CPU usage and minimizes the total idle time** of the processors by balancing the workload among them.
- **It improves Quality-of-Service (QoS) by reducing missed deadline ratio:** As shown by an example in 2.4, LBBA reduces the possibility of starvation for low priority jobs, comparing to BBA. It also has **Minimal Response time,** including both scheduling and execution time, for a job set (up to 300% faster response time than BBA in our experiments shown in 4.1.2.).
- **LBBA is computationally less expensive** than BBA, as we prove in Section 3.2.