Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 198–215
The Journal of Systems and Software
www.elsevier.com/locate/jss

GSR: A global seek-optimizing real-time disk-scheduling algorithm

Hsung-Pin Chang [a,*], Ray-I Chang [b], Wei-Kuan Shih [c], Ruei-Chuan Chang [d]

[a] Department of Computer Science, National Chung Hsing University, 250 Kuo Kuang Road, Taichung 402, Taiwan, ROC
[b] Institute of Engineering Science and Ocean Engineering, National Taiwan University, Taipei, Taiwan, ROC
[c] Department of Computer Science, National Tsing Hau University, Hsinchu, Taiwan, ROC
[d] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC

Received 22 August 2005; received in revised form 31 March 2006; accepted 31 March 2006
Available online 22 May 2006

Abstract

Earliest-deadline-first (EDF) is good for scheduling real-time tasks in order to meet timing constraint. However, it is not good enough for scheduling real-time disk tasks to achieve high disk throughput. In contrast, although SCAN can maximize disk throughput, its schedule results may violate real-time requirements. Thus, during the past few years, various approaches were proposed to combine EDF and SCAN (e.g., SCAN-EDF and RG-SCAN) to resolve the real-time disk-scheduling problem. However, in previous schemes, real-time tasks can only be rescheduled by SCAN within a local group. Such restriction limited the obtained data throughput. In this paper, we proposed a new *globally* rescheduling scheme for real-time disk scheduling. First, we formulate the relations between the EDF schedule and the SCAN schedule of input tasks as *EDF-to-SCAN mapping* (ESM). Then, on the basis of ESM, we propose a new real-time disk-scheduling algorithm: globally seek-optimizing rescheduling (GSR) scheme. Different from previous approaches, a task in GSR may be rescheduled to anywhere in the input schedule to optimize data throughput. Owing to such a globally rescheduling characteristic, GSR obtains a higher disk throughput than previous approaches. Furthermore, we also extend the GSR to serve fairly non-real-time tasks. Experiments show that given 15 real-time tasks, our data throughput is 1.1 times that of RG-SCAN. In addition, in a mixed workload, compared with RG-SCAN, our GSR achieves over 7% improvement in data throughput and 33% improvement in average response time.

Keywords: Real-time disk scheduling; Disk scheduling; Operating systems

1. Introduction

Recent advancement in hardware technology and network communications has increased the popularity of data services. In some applications, the data services must be provided with timing characteristics. For example, media data must be accessed under real-time constraints to guarantee jitter-free playback. Furthermore, media data are often in large volume and consume significant disk bandwidth. As a result, performances of multimedia applications depend heavily on the real-time disk-scheduling algorithm applied (Lougher and Shepherd, 1993; Steinmetz, 1995). A well-behaved real-time disk scheduling should maximize data throughput while guaranteeing real-time constraints.

The earliest time at which a disk task can start is defined as its *ready time* (or *release time*). The latest time at which a disk task must be completed is its *deadline*. The actual times at which a disk task is started and completed are its *start-time* and *fulfill-time*, respectively. To meet timing constraints for real-time data services, each disk task must guarantee

* Corresponding author. Tel.: +886 4 22852106; fax: +886 4 22853869.
E-mail address: hpchang@cs.nchu.edu.tw (H.-P. Chang).

0164-1212/$ - see front matter
doi:10.1016/j.jss.2006.03.045

that its start-time scheduled is not earlier than its ready time achieved. Moreover, its fulfill-time scheduled is not later than its deadline set (Stankovic and Buttazzo, 1995). Different from the conventional disk-scheduling problem, timing constraints on accessing real-time data is crucial for supporting timing critical applications (Anderson et al., 1991, 1992; Chang et al., 1997). Although the well-known SCAN algorithm, which scans disk surface back and forth to retrieve the data under disk head, has been proved as the best algorithm for maximizing disk throughput (Chen and Yang, 1992; Chen et al., 1992), its output result may not meet timing constraint on scheduling real-time disk tasks. Therefore, real-time data retrieved by SCAN may be meaningless and even harmful to systems (Gemmell et al., 1995; Gemmell and Christodoulakis, 1992).

In contrast, earliest-deadline-first (EDF), which serves tasks in deadline order, is one of the best-known schemes for scheduling real-time tasks (Liu and Layland, 1973; Lehoczky, 1990). However, EDF earns its optimization under the assumption that tasks are independent. Nevertheless, in real-time disk scheduling, tasks are non-preemptive and inter-dependent. Once a disk task is issued to a disk drive, its service cannot be interrupted. Moreover, for each disk task, its service time depends not only on the location of the data block retrieved, but also on the location of the current disk head. As a result, taking only deadlines into account without service time consideration, EDF incurs excessive seek-time costs and results in poor disk throughput (Yee and Varaiya, 1991; Reddy and Wyllie, 1994). Actually, owing to the non-preemptive property and the non-prespecified service time of disk tasks, it is very hard to optimize the schedule result of real-time disk tasks. This problem has been proved to be NP-complete (Wong, 1980).

Previous studies have examined heuristic methods for combining the features of SCAN type of seek-optimizing algorithms with EDF type of real-time scheduling algorithms. In 1993, Reddy and Wyllie proposed the *SCAN-EDF* method that first sorts input tasks by the EDF order and, then reschedules tasks with the same deadlines by SCAN. Experiments show that their obtained results depend highly on the probability of tasks that have the same deadlines. To increase the probability of employing SCAN to reschedule tasks, DM-SCAN (deadline-modification-SCAN) and RG-SCAN (reschedulable-group-SCAN) are proposed to select automatically contiguous tasks that can be rescheduled by SCAN (Chang et al., 1998, 2002). In other words, these contiguous tasks can be viewed as having the "same deadline" in SCAN-EDF.

However, previous approaches are *locally seek-optimizing* schemes; i.e., tasks can only be rescheduled by SCAN within a local group. Note that, each group is a set of *consecutive* tasks that can be rescheduled by SCAN without missing their respective timing constraints. For example, in SCAN-EDF, a group is made up of tasks having the same deadline. Similarly, given a set of EDF tasks, DM-SCAN automatically selects groups of consecutive tasks and these groups are named as MSGs (maximum-scannable-groups). RG-SCAN also has its own group definition and is called R-Group (reschedulable-group). However, no matter in SCAN-EDF, DM-SCAN, or RG-SCAN, once a task belongs to a certain group, it cannot be rescheduled to a different group; even though such a rescheduling derives a better performance. The detailed operations of DM-SCAN and RG-SCAN with their proposed MSG and R-Group concepts are introduced in Section 2.

To resolve the drawback of previous approaches, we propose herein a *globally seek-optimizing* scheduling approach: *GSR* (globally seek-optimizing rescheduling) scheme. First, a graph of *EDF-to-SCAN mapping* (ESM) is introduced to explore relations between the EDF schedule and the SCAN schedule of input tasks. Given a set of real-time disk tasks, schedule results of EDF and SCAN just denote two permutations of input tasks. By representing each task as a vertex and connecting each task in the EDF schedule to the same task in the SCAN schedule with an edge, there is a bipartite mapping; which is called ESM in this paper. On the basis of this ESM mapping, our algorithm then identifies *scan-groups* where each scan-group contains the maximum number of contiguous tasks that are in the same SCAN direction (left-to-right or right-to-left). Now, the input schedule can be viewed as a piecewise-SCAN schedule. After that, input tasks are tested for being rescheduled into suitable scan-groups to achieve the highest improvement of disk throughput while guaranteeing real-time requirements. Thus, our scheme provides a good combination of the EDF scheme and the SCAN scheme.

Note that, since there are at most n scan-groups where n is the number of input tasks, a naive algorithm will take $O(n^2)$ time to decide the best reschedule result for the selected task. To speed up its computation, we introduce a concept of the *schedulable-region* to each input task. With the help of the pre-computed schedulable-regions, the best-fit scan-group for rescheduling each input task can be decided in $O(n)$ time. In addition, we extend the GSR to serve mixed real-time/non-real-time disk tasks such that non-real-time tasks can be served to minimize response time while guaranteeing the timing constraints of real-time tasks. Compared with DM-SCAN, experiments show that our GSR algorithm can support over 11% data throughput improvement in a real-time system. Moreover, in a mixed workload, our GSR achieves over 7% improvement compared with RG-SCAN scheme in data throughput and offers 33% improvement compared with RG-SCAN in terms of average response time of non-real-time tasks.

The remainder of this paper is organized as follows. Section 2 gives mathematical definitions about real-time disk scheduling and shows some related work. The EDF-to-SCAN mapping and our proposed GSR algorithm are introduced in Section 3. In Section 4, we present the definition of reschedulable region and proposed a speed-up method for scheduling. Section 5 demonstrates how GSR is extended to efficiently serve mixed real-time/non-real-time disk tasks. Finally, Sections 6 and 7 show the experimental results and conclusion remarks, respectively.