Review Article **Impact Factor: 0.621** **ISSN: 2319-507X**
Shrikant Chavhan, IJPRET, 2014; Volume 2 (8): 880-893 **IJPRET**

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

AN OVERVIEW OF SEEK BASED TECHNIQUES FOR REAL TIME DATABASE

Mr. SHRIKANT B. CHAVHAN[1], PROF. S T KHANDARE[2]

1. M.E 1st Year, Department of Computer Science and Engineering, B. N. C. O. E., Pusad, India.
2. Assistant Professor, Department of Computer Science and Engineering, B. N. C. O. E., Pusad, India.

Accepted Date: 15/02/2014 ; Published Date: 01/04/2014

Abstract: Real-time disk scheduling (RTDS) plays an important role in time-critical applications. Due to rigorous timing requirements for error free output, data must be accessed under real-time constraints. Therefore how to maximize data throughput under real-time constraints poses a big challenge in the design of real-time disk scheduling algorithms. Numbers of algorithms are proposed to schedule real time transactions in order to increase the overall performance. Currently Earliest-Deadline-First (EDF) is a basic algorithm which meets the real time constraints, but it gives poor disk throughput. While SCAN gives maximum disk throughput but violates the real time constraints. Thus, various algorithms proposed later combined EDF and SCAN to improve the real-time performance. In previously proposed algorithms (e.g. SCAN-EDF, DM-SCAN & RG-SCAN) the transactions are scheduled by forming groups, based on certain conditions. But these approaches are locally seek optimizing Schemes i.e. transactions can only be rescheduled by SCAN within a local group. Once a transaction belongs to a certain group, it cannot be rescheduled to a different group, even though such a rescheduling gives better performance. Therefore, to improve the performance a new algorithm is proposed based on globally seek optimizing scheduling approach: GSR (Globally Seek-optimizing Rescheduling) algorithm. In this algorithm, relation between SCAN and EDF is established by using EDF-to-SCAN mapping (ESM). Then on the basis of ESM groups are formed and transactions are rescheduled globally within that groups using GSR algorithm. Different from previous approaches, a transaction 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.

Keywords: Real-Time Disk Scheduling, Seek Based

PAPER-QR CODE

Corresponding Author: Mr. SHRIKANT B. CHAVHAN

Access Online On:

www.ijpret.com

How to Cite This Article:

Shrikant Chavhan, IJPRET, 2014; Volume 2 (8): 880-893

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. 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 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. Different from the conventional disk-scheduling problem, timing constraints on accessing real-time data is crucial for supporting timing critical applications.

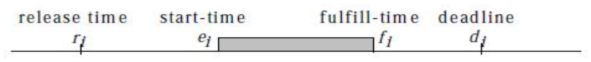


Fig. A simple example to demonstrate the terminologies (release time, start-time, fulfill-time and deadline) used to represent a real-time task.

SCAN is the well-known 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, its output result may not meet timing constraint on scheduling real-time disk transactions. Therefore, real-time data retrieved by SCAN may be meaningless and even harmful to systems. **Earliest-Deadline-First (EDF)**, which serves transactions in deadline order, is one of the best-known schemes for scheduling real-time transactions. EDF incurs excessive seek-time costs and results in poor disk throughput in overloaded condition.

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 transactions by the EDF order and, then reschedules transactions with the same deadlines by SCAN. Experiments show that their obtained results depend highly on the probability of transactions that have the same deadlines. To increase the probability of employing SCAN to reschedule transactions, **DM-SCAN** (deadline-modification-SCAN) and **RG-SCAN** (reschedulable-group-SCAN) are proposed to select automatically contiguous transactions that can be rescheduled by SCAN. In other words, these contiguous transactions can be viewed as having the ‘‘same deadline’’ in SCAN-EDF.

However, previous approaches are *locally seek-optimizing schemes*; i.e., transactions can only be rescheduled by SCAN within a local group. Note that, each group is a set of consecutive transactions that can be rescheduled by SCAN without missing their respective timing constraints. For example, in SCAN-EDF, a group is made up of transactions having the same deadline. Similarly, given a set of EDF transactions, DM-SCAN automatically selects groups of consecutive transactions 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 transaction 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 next section.

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 transactions. Given a set of real-time disk transactions, schedule results of EDF and SCAN just denote two permutations of input transactions. By representing each transaction as a vertex and connecting each transaction in the EDF schedule to the same transaction 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 transactions 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 transactions 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 transactions, a naive algorithm will take O(n2) time to decide the best reschedule result for the selected transactions. 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 transactions can be decided in O (n) time.

2. Problem descriptions and related work

2.1. Real-time disk-scheduling problem

The problem input considered in this paper is a set of real-time disk transactions. T = {T0, T1, T3, ..., Tn} where n is the number of transactions. The transaction is represented by Ti = (ri, di, ai, li, bi) where ri is its ready time, di is its deadline, ai is its track location, li is its sector number and bi is its data size. While serving disk task Ti, the disk head needs to be moved from the current trackto the target track ai by a seek time cost. Then, a rotational latency is presented for the desired sector li rotated under the disk head. Finally, data under disk head are retrieved with size bi by a transfer time. The first task T0 is assigned as a special task to represent the initial location of disk head. Without loss of generality, it can be assumed to be at the outermost track (track 0). Assume that the schedule sequence is TjTi (Ti is served after Tj). The service time of task Ti is calculated as,

Cj, i = Seek_time (abs (ai - aj)) + Rotational_latency (li) + Transfer_time (bi)

Since Ti is a real-time task, the ready time ri and deadline di are used to characterize its timing constraint. Because disk service is non-preemptive, the related start-time and fulfill-time are ei = max {ri, fj} and fi = ei + cj, i.

A schedule result of real-time disk transactions T = {T0, T1, . . . , Tn} is called feasible if all input transactions Ti, for i = 0 to n, satisfy real-time requirements **ri <= ei and fi <= di**. To measure the efficiency of a real-time disk scheduling algorithm, given a set of real-time disk transactions, the applied disk scheduling algorithm should serve as many transactions as possible under transactions' timing constraints. If the same number of transactions is feasibly served, the applied disk scheduling algorithm needs to maximize data throughput.

To determine the data throughput improvement, we define schedule fulfill-time as the finish time it takes to serve all input transactions according to their respective timing constraints. Clearly, this is the finish time of the latest task f(n). Since the disk throughput is related to the inverse of schedule fulfill-time, thus, the problem objective to maximize disk throughput can be redefined as to minimize schedule fulfill-time. In real-time disk scheduling, the system time required for serving each task is determined by its schedule sequence (Peterson and Silberschatz, 1985). However, the schedule sequence of a task depends on its service time required. Thus, it is hard to decide the optimal schedule result that maximizes the disk throughput without violating real-time requirements.

2.2. Related work

Previous real-time disk scheduling algorithms thus apply heuristically the seek-optimizing SCAN scheme to an EDF schedule for reducing the disk service time. For example, the well-known SCAN-EDF scheme reschedules transactions having the same deadline in an EDF order to reduce service times of transactions. However, since only transactions having the same deadline are seek-optimized, the reduction in schedule fulfill-time compared with EDF is not significant.

To increase the probability of employing the SCAN scheme to reschedule input transactions, **DM-SCAN (deadline-modification-SCAN)** proposed the concept of maximum-scannable-group (MSG) .Given an EDF schedule, consecutive transactions that can be rescheduled by SCAN without missing their respective timing constraints can be directly derived by the concept of MSG. Given a set of real-time disk transactions with EDF-ordered T = T1T2. . .Tn, the MSG Gi starting from Ti is defined as the sequential transactions Gi = TiTi+1Ti+2. . .Ti+m with each task Tk for k = i to i + m satisfies fk <= di and rk <= si. However, DM-SCAN requires that the input transactions must be EDF-ordered. Therefore, they proposed a deadline modification scheme that transfers a non-EDF schedule into an EDF order by modifying transactions' deadlines. Unfortunately, in order to guarantee real-time constraints, the modified deadlines are earlier than the original ones. As a result, the deadline modification scheme causes a negative impact on the number of supported transactions by DM-SCAN.

To relieve from such a constraint, **RG-SCAN (reschedulable-group-SCAN)** is proposed

with the concept of R-Group (reschedulable group). Given a set of real-time disk transactions T ={ T1,T2,. . .Tn} the R-Group Gi starting from task Ti is defined as the maximum number of consecutive transactions Gi = TiTi+1 . . . Ti+m with each task Tk for k = i to i + m satisfies following criteria:

$$f_{i+m} \leqslant \min_{k=i}^{i+m}\{d_k\} \quad \text{and} \quad \max_{k=i}^{i+m}\{r_k\} \leqslant s_i.$$

RG-SCAN also shows that after seek-optimizing transactions within an R-Group, it can obtain more data throughput while guaranteeing real-time requirements.

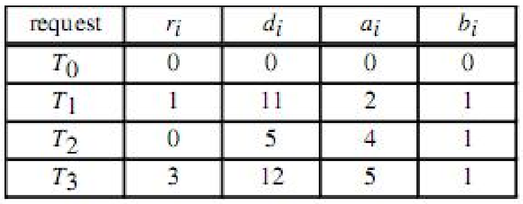
However, previous approaches are locally seek-optimizing algorithms. SCAN scheme is only applied to a set of consecutive transactions. Thus, a task would only be rescheduled by SCAN within its own group. In other words, a task belonging to a group i, say R-Group, cannot be rescheduled to another group j, since this would violate the constraints of Eq. (3) even if such a rescheduling would derive a higher disk throughput. Therefore, in this paper, we propose a globally seek-optimizing real-time disk-scheduling algorithm, GSR, to overcome the limitations of previous approaches. In GSR, a task belonging to a group i would be ‘‘globally” rescheduled to another group j as long as the new rescheduled result obtains a higher data throughput while guaranteeing real-time constraints

3. GSR: Globally seek-optimizing rescheduling scheme

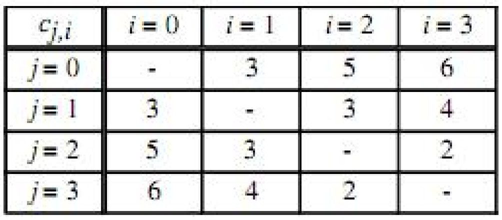
The design of our GSR real-time disk-scheduling algorithm is as follows:

First the construction scheme of EDF-to-SCAN mapping (ESM) graphs to relate an EDF schedule to a SCAN schedule. Then, we present the idea of scan-groups, which is derived from an ESM graph. On the basis of scan-groups, our proposed GSR algorithm is described in next section.

PARAMETER-TABLE



request	r_i	d_i	a_i	b_i
T_0	0	0	0	0
T_1	1	11	2	1
T_2	0	5	4	1
T_3	3	12	5	1



$c_{j,i}$	$i = 0$	$i = 1$	$i = 2$	$i = 3$
$j = 0$	-	3	5	6
$j = 1$	3	-	3	4
$j = 2$	5	3	-	2
$j = 3$	6	4	2	-

Ti- Transaction request	ai- track location
ri- release time	bi- transfer time
di- deadline	

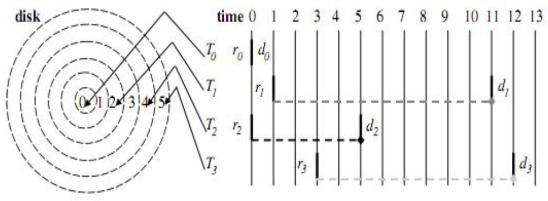


Fig.1

3.1. EDF-to-SCAN mapping

As described earliar, although SCAN can maximize data throughput, its schedule result does not meet the timing constraints of real-time transactions. In contrast, the EDF schedule is good for real-time requirements. However, its disk throughput is low. Fig. 2 demonstrates the SCAN schedule and EDF schedule of the example shown in Fig. 1.

In Fig. 2(a), the SCAN schedule derives a shorter schedule fulfill-time but is not feasible. In contrast, the EDF schedule shown in Fig. 2(b) owns a feasible result but results in a longer schedule-fulfill time. There is a tradeoff relation between these two extreme schedule cases. It motivates us to construct a graph of EDF-to-SCAN mapping (ESM) to develop a new scheme for real-time disk scheduling.

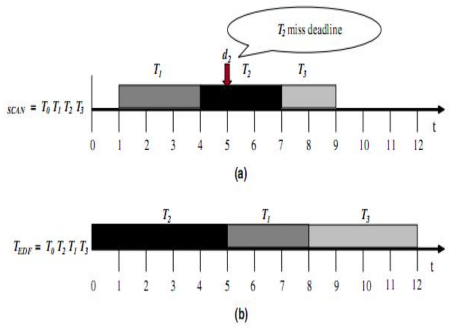
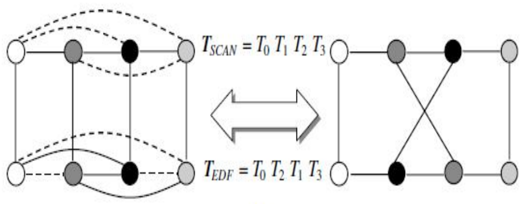


Fig.2 For the example presented in fig.1 **(a)** schedule obtained by applying SCAN algorithm **(b)** schedule obtained by applying EDF algorithm are demonstrated. Note that schedule result obtained by SCAN is not feasible.

Given a set of real-time disk transactions T = {T0,T1, . . .,Tn}, their SCAN schedule is TSCAN = TS(0) TS(1). . TS(n) with data locations aS(0) < aS(1) < ….. < aS(n) where S(i) for i = 0 to n, is a permutation of indexes {0, 1, . . .,n}. We can represent TSCAN by an one-dimensional graph GSCAN = (VSCAN, ESCAN) where the vertex set VSCAN = T and the edge set ESCAN = {(TS(i _ 1), TS(i)) j for i = 1 to n}. The same idea can be applied to transactions' EDF schedule TEDF = TE(0)

TE(1). . TE(n) with deadlines dE(0) < dE(1) <...... < dE(n) where E(i), for i = 0 to n, is also a permutation of indexes {0, 1, . . .,n}. The related graph is GEDF = (VEDF, EEDF), where the vertex set VEDF = T and the edge set EEDF = {(TE(i _ 1), TE(i)) | for i = 1 to n}. Since VSCAN = VEDF = T, there is a bipartite mapping between GSCAN and GEDF.


$T_{SCAN} = T_0\, T_1\, T_2\, T_3$
$T_{EDF} = T_0\, T_2\, T_1\, T_3$

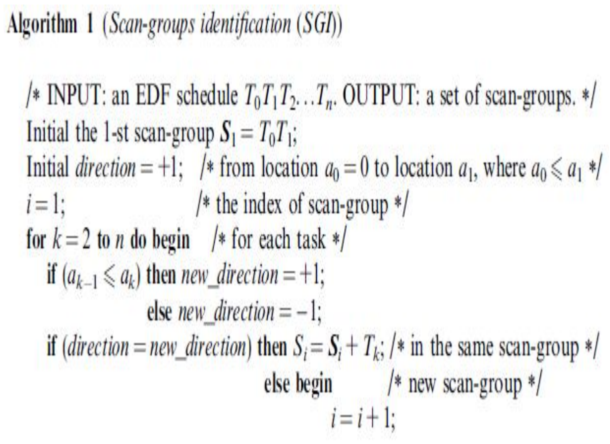
(a)

EDF-to-SCAN mapping is a bipartite mapping obtained by connecting each node in EDF schedule to corresponding node in SCAN schedule with an edge.

Above shows an example of ESM for the input transactions in Fig. 1. Using this mapping, we can investigate the possible transformations from the real-time EDF schedule to the seek-optimized SCAN schedule to find a good real-time disk schedule.

To mimic the behavior of the SCAN schedule, we first track the scan directions of transactions in the input schedule (EDF, usually but not necessary) to decompose the input schedule into a sequence of scan-groups where each scan-group contains the maximum number of contiguous transactions with the same SCAN direction. Therefore, the input schedule can be represented by a piecewise-SCAN schedule. Given an input schedule T0T1T2. . .Tn, an O(n) algorithm to identify all these scan-groups is shown as follows:

SCAN-GROUP-IDENTIFICATION ALGORITHM

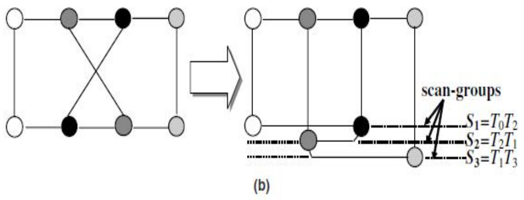


Algorithm 1 (*Scan-groups identification* (*SGI*))

```
/* INPUT: an EDF schedule $T_0T_1T_2 \dots T_n$. OUTPUT: a set of scan-groups. */
Initial the 1-st scan-group $S_1 = T_0T_1$;
Initial $direction = +1$;  /* from location $a_0 = 0$ to location $a_1$, where $a_0 \leqslant a_1$ */
$i = 1$;                   /* the index of scan-group */
for $k = 2$ to $n$ do begin   /* for each task */
  if ($a_{k-1} \leqslant a_k$) then $new\_direction = +1$;
                      else $new\_direction = -1$;
  if ($direction = new\_direction$) then $S_i = S_i + T_k$; /* in the same scan-group */
                               else begin        /* new scan-group */
                                    $i = i + 1$;
```

```
                Initial the ith scan-group $S_i = T_{k-1}T_k$;
                Initial $direction = new\_direction$;
            end /* else */
    end /* for */
```

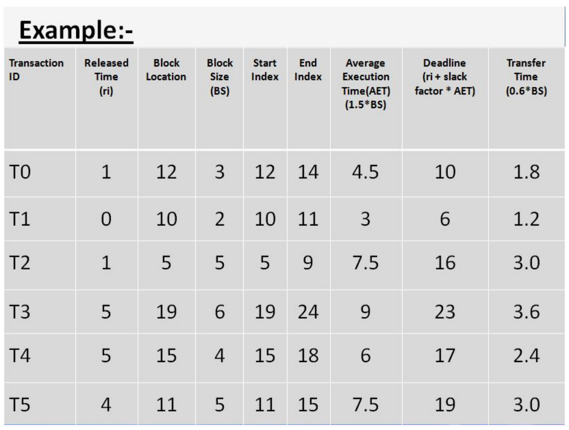
By applying the above algorithm to the EDF-to-SCAN mapping, we can get the following result.



As the example shown in above fig, by mapping the input EDF schedule T0T2T1T3 to the SCAN schedule T0T1T2T3, we obtain a piecewise-SCAN schedule with three scan-groups S1 = T0T2, S2 = T2T1 and S3 = T1T3. It introduces a new point of view for analyzing the relations between the input schedule and the SCAN schedule. Different heuristic methods can be developed for real-time disk scheduling. For example, we may try to minimize the number of scan-groups or to maximize the sizes of scan-groups under real-time requirements of such a piecewise-SCAN schedule. Since the more the transactions can be seek-optimized, the more the disk throughput is obtained. In this paper, on the basis of ESM, an effective and efficient real time disk scheduling method GSR is proposed. Note that although the same idea can be employed to represent a SCAN schedule by a piecewise-EDF schedule, its schedule result usually violates real-time requirements.

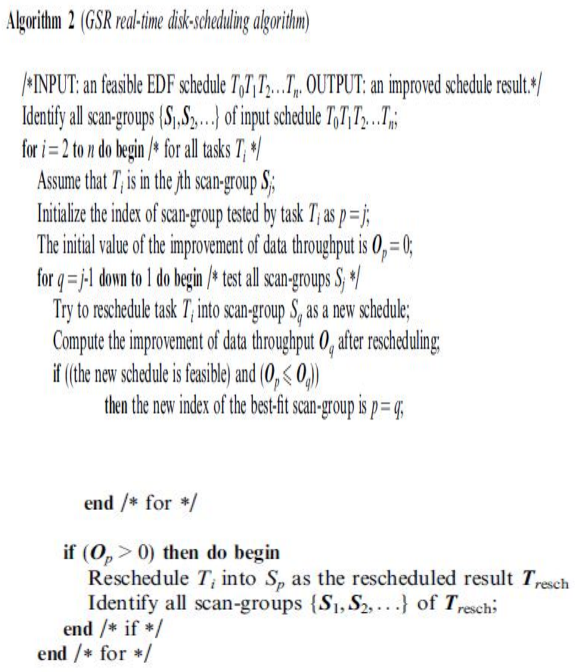
3.2. GSR algorithm

In this subsection, we describe our proposed GSR algorithm. Without loss of generality, we assume that the input is a feasible TEDF schedule to meet basic timing constraints. Our algorithm then selects an input task according to the first-in first-serve (FIFS) order and tries to reschedule the selected task into the best-fit scan-group to maximize disk throughput under real-time requirements. For example, given the input transactions shown in Fig. 1, we can improve disk throughput by rescheduling task T3 into scan-group T0T2. The new scan-groups are S1 = T0T2T3 and S2 = T3T1, as shown in Fig. 4. Note that, it reduces the number of scan-groups by 1 (S3 is removed) and increases the size of scan-group S1 from 2 to 3. In this paper, a rescheduled result is accepted only when it is feasible and the disk throughput is improved. A detailed description of the proposed GSR algorithm is illustrated as follows.



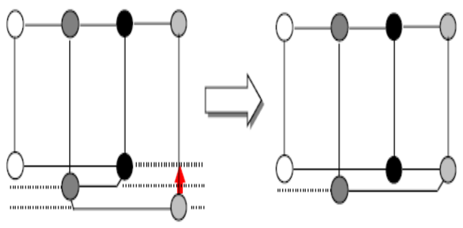
Example:-

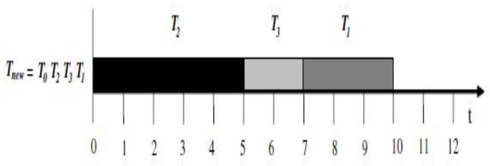
Transaction ID	Released Time (ri)	Block Location	Block Size (BS)	Start Index	End Index	Average Execution Time(AET) (1.5*BS)	Deadline (ri + slack factor * AET)	Transfer Time (0.6*BS)
T0	1	12	3	12	14	4.5	10	1.8
T1	0	10	2	10	11	3	6	1.2
T2	1	5	5	5	9	7.5	16	3.0
T3	5	19	6	19	24	9	23	3.6
T4	5	15	4	15	18	6	17	2.4
T5	4	11	5	11	15	7.5	19	3.0



Algorithm 2 (*GSR real-time disk-scheduling algorithm*)

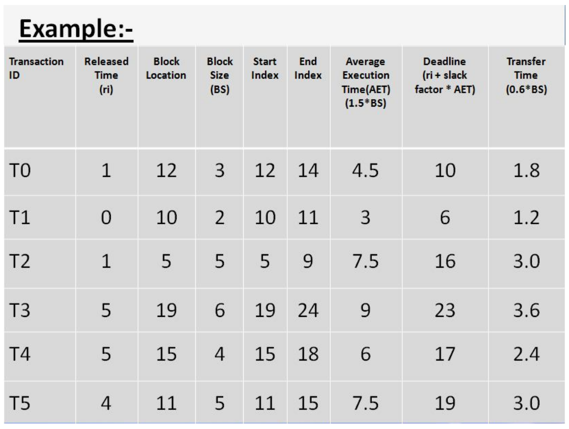
```
/*INPUT: an feasible EDF schedule $T_0T_1T_2 \ldots T_n$. OUTPUT: an improved schedule result.*/
Identify all scan-groups $\{S_1, S_2, \ldots\}$ of input schedule $T_0T_1T_2 \ldots T_n$;
for $i = 2$ to $n$ do begin /* for all tasks $T_i$ */
  Assume that $T_i$ is in the $j$th scan-group $S_j$;
  Initialize the index of scan-group tested by task $T_i$ as $p = j$;
  The initial value of the improvement of data throughput is $O_p = 0$;
  for $q = j$-1 down to 1 do begin /* test all scan-groups $S_j$ */
    Try to reschedule task $T_i$ into scan-group $S_q$ as a new schedule;
    Compute the improvement of data throughput $O_q$ after rescheduling;
    if ((the new schedule is feasible) and ($O_p \leqslant O_q$))
          then the new index of the best-fit scan-group is $p = q$;


      end /* for */

    if ($O_p > 0$) then do begin
      Reschedule $T_i$ into $S_p$ as the rescheduled result $T_{resch}$
      Identify all scan-groups $\{S_1, S_2, \ldots\}$ of $T_{resch}$;
    end /* if */
  end /* for */
```





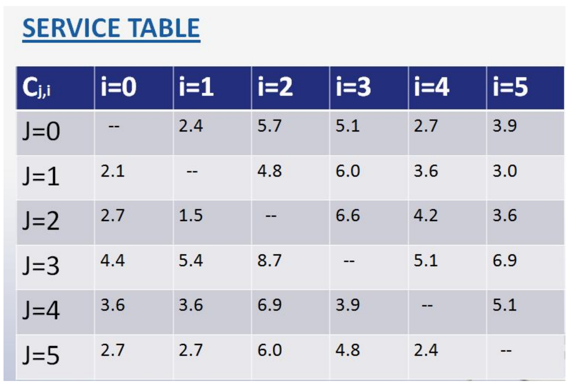
After identifying all scan-groups, GSR tries to reschedule each task into all scan-groups before its own scan-group and to compute the improvement of data throughput of each rescheduling result. The one with the largest throughput improvement while guaranteeing a feasible schedule is selected for rescheduling (the task is rescheduled from its own scan-group to the new one.) Notably, the above algorithm assumes that the input schedule is feasible. However, even an infeasible input schedule is given, GSR may still produce feasible output schedule, although it is not guaranteed.

EXAMPLE



Example:-

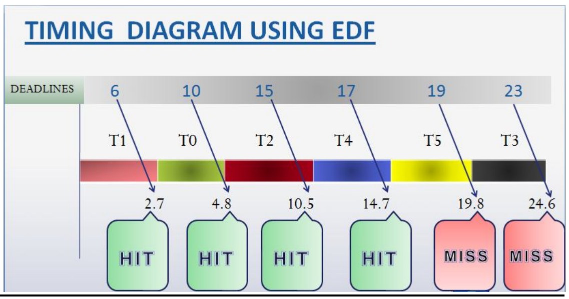
Transaction ID	Released Time (ri)	Block Location	Block Size (BS)	Start Index	End Index	Average Execution Time(AET) (1.5*BS)	Deadline (ri + slack factor * AET)	Transfer Time (0.6*BS)
T0	1	12	3	12	14	4.5	10	1.8
T1	0	10	2	10	11	3	6	1.2
T2	1	5	5	5	9	7.5	16	3.0
T3	5	19	6	19	24	9	23	3.6
T4	5	15	4	15	18	6	17	2.4
T5	4	11	5	11	15	7.5	19	3.0



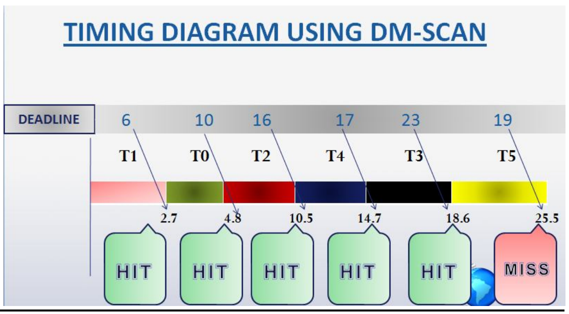
SERVICE TABLE

$C_{j,i}$	i=0	i=1	i=2	i=3	i=4	i=5
J=0	--	2.4	5.7	5.1	2.7	3.9
J=1	2.1	--	4.8	6.0	3.6	3.0
J=2	2.7	1.5	--	6.6	4.2	3.6
J=3	4.4	5.4	8.7	--	5.1	6.9
J=4	3.6	3.6	6.9	3.9	--	5.1
J=5	2.7	2.7	6.0	4.8	2.4	--

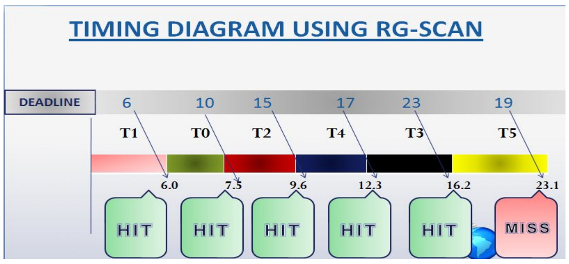
SCHEDULE USING EDF: T1 TO T2 T4 T5 T3



SCHEDULE USING DM-SCAN: T1 TO T2 T4 T3 T5



SCHEDULE USING RG-SCAN: T2 T1 T0 T4 T3 T5



4. ADVANTAGES

- Higher disk throughput than all previously proposed algorithms.
- It is best algorithm to serve the transactions if the input schedule is feasible.
- Average response time is reduced

5. DISADVANTAGES

- Complexity is more due to the number of internal calculations.
- Overhead is more.

6. CONCLUSION:

In order to improve data throughput, the seek-optimizing SCAN scheme should be employed to reschedule the input tasks as much as possible. However, previous approaches limit their flexibility and efficiency in that a task can only be seek-optimizing rescheduled with the tasks having the same deadline or within the same local group (a set of contiguous tasks). In order words, in conventional schemes, a task can only be rescheduled by SCAN with a locally best position. In this paper, we propose a globally seek-optimizing disk-scheduling scheme called GSR. In GSR, a task can be rescheduled to the globally best position, i.e., position with the maximal data throughput while guaranteeing a feasible schedule. The experimental results show that our proposed GSR scheme is better than the previous methods not only in improving the data throughput, but also in shortening response time.

REFERENCES

1. Hsung-Pin Chang , Ray-I Chang , Wei-Kuan Shih , Ruei-Chuan Chang, 2006. GSR: A global seek-optimizing real-time disk-scheduling algorithm. The Journal of Systems and Software 80 (2007) 198–215.

2. Chang, R.I., Shih, W.K., Chang, R.C., 1998. Deadline-modification-scan with maximum scannable-groups for multimedia real-time disk scheduling. In: Proceedings of the 19th IEEE Real-Time Systems Symposium, pp. 40–49.

3. Chang, H.P., Chang, R.I., Shih, W.K., Chang, R.C., 2002. Reschedulable-Group-SCAN Scheme for Mixed Real-Time/Non-Real-Time Disk Scheduling in a Multimedia System. J. Syst. Software 59 (2), 143–152.

4. S. Y. Amdani, H. R. Deshmukh, S. A. Bhura, 2009. A Novel disk scheduling algorithm.

5. A. L. N. Reddy and J. Wyllie, "Disk scheduling in a multimedia I/O system," *Proc. ACM Multimedia Conf.*, pp. 225-233, 1993