An Improved Group-EDF: A Real-Time Disk Scheduling Algorithm

S. Y. Amdani and M. S. Ali

Abstract—**Real-time disk scheduling (RTDS) plays an important role in time-critical applications. The main idea in real time database system is that the correctness of the system depends not only on the logical results of the computations but also on the time at which the results are produced. 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. Scan-EDF work only for those transactions which are having same deadline. In 2006 g-EDF algorithm has been proposed which works after making groups for transaction having close deadlines. In groups it apply SJF algorithm. We also propose a new algorithm "FEASIBLE GROUP-EDF" that works both in under load and overload conditions as well as show better throughput than earliest mentioned algorithms. It also makes groups and applies SSTF algorithm as well as check feasibility of transaction.**

Index Terms—**EDF, SCAN-EDF, G-EDF, SJF, SSTF, real-time, overloaded.**

I. Introduction

Traditionally, real-time system manages their data in application dependent structures. As real-time systems evolve, their applications become more complex and require accessing more data. It thus becomes necessary to manage the data in more systematic and organized manner. Database management system provides tools for such organization, so in recent year there has been interest in "merging" database and real-time system. The resulting integrated system which provides the database operations with real-time constraint is called as real-time database system (RTDBS) [1], [2].

A Real-Time Database System (RTDBS) is a transaction processing system that is designed to handle transactions with the timing constraint. Several previous RTDBS as in studies had been done to address the issue of scheduling transactions with the objective of minimizing the number of miss transaction. A common observation of these studies has been that, if we assigning priorities to transactions according to an Earliest.

Deadline policy minimizes the number of miss transactions in systems operating under low or moderate levels of workload condition. This is due to earliest Deadline giving the highest priority to transactions that have the least remaining time in which to complete.

These studies have also observed that the performance of Earliest Deadline steeply degrades in an overloaded system. This is because, under heavy loaded workload condition transactions gain high priority only when they are close to their deadlines.

While investigating scheduling algorithms, we have analyzed a variation of EDF that can improve success ratios, particularly in overloaded conditions. The algorithm can also decrease the average response time for tasks that is group-EDF, or *g*EDF, where the tasks with "similar" deadlines are grouped together (i.e., deadlines that are very close to one another), and the Shortest Job First (SJF) algorithm is used for scheduling tasks within a group. It should be noted that our approach is different from adaptive schemes that switch between different scheduling strategies based on system load; F*g*-EDF is used in overloaded as well as underloaded conditions.

II. Background

A. EDF Algorithm

The idea of EDF was published in 1973, in an article of Liu and Layland [3]. The Earliest Deadline First (EDF) algorithm is an analog of FCFS. Requests are ordered according to deadline and the request with the earliest deadline is serviced first. Assigning priorities to transactions an Earliest Deadline policy minimizes the number of late transactions in systems operating under low or moderate levels of resource and data contention. This is due to the highest priority given to the transactions that have the least remaining time in which to complete. However, the performance of Earliest Deadline steeply degrades in an overloaded system. Gaining high priority at this late stage may not leave sufficient time for transactions to complete before their deadlines. Under heavy loads, then, a fundamental weakness of the Earliest Deadline priority policy is that it assigns the highest priority to transactions that are close to missing their deadlines, thus delaying other transactions that might still be able to meet their deadlines.

B. SCAN-EDF Algorithm

It is the combination of two algorithms SCAN and EDF. When deadlines of two transactions are same then we applies this algorithm [4]. According to it, transactions with the same deadline are arranged using scan algorithms. Those who had shortest distance from disk head will get the

Manuscript received February 14, 2013; revised April 14, 2013.

S. Y. Amdani is with Department of CSE, BNCOE, Pusad, India (e-mail: salimamdani@yahoo.com).

M. S. Ali is with P. R. M. C. E. & M, Badnera, India (e-mail: softalis@gmail.com).

chance first to process. In scan algorithm, once scan direction is taken it will go in that direction only until all jobs are scheduled in that direction. So its limitation is that it cannot be applied on transactions having different deadline.

C. SJF Algorithm

It is one of the conventional algorithms. It is implemented by scheduling shortest job first i.e. transaction having smallest execution time. Average execution time depends upon transaction size. The demerits of this algorithm are that in that timing constraint i.e. deadline has been given no consideration. So there will be misutilization of resources.

D. Group-Earliest Deadline First Algorithm (g-EDF)

It is based on dynamic grouping of transactions with deadlines that are very close to each other and using SHORTEST JOB FIRST technique to schedule tasks within the group [5]. It is used in overload as well as under load conditions. It is particularly useful for real time systems as well as applications known as "approximate algorithms" and "anytime algorithms" where applications generate more account results or rewards with increased execution times. A transaction *ti* in a real-time system is defined as *ti* = (*ri*, *ei*, *Di*, *Pi*); where *ri* is its release time (or its arrival time); *ei* is either its predicted worst-case or average execution time; *Di* is its deadline and *Pi* is periodicity of transaction. We generated a fixed number (*N*) of jobs with varying arrivals, execution times and deadlines. We assume that the jobs are mutually independent.

A group in the gEDF algorithm depends on a group range parameter *Gr*. *tj* belongs to the same group as *ti* if $di < dj < (di + Gr*(di - t))$, where t is the current time, $1 < i, j < N$. In other words, we group jobs with very close deadlines together. We schedule groups based on EDF (all jobs in a group with an earlier deadline will be considered for scheduling before jobs in a group with later deadlines), but schedule jobs within a group using shortest job first (SJF) approach as shown in Fig. 1. Since SJF results in more (albeit shorter) jobs completing, intuitively *g*EDF should lead to a higher success rate than pure EDF.

Q*g*EDF is a queue for *g*EDF scheduling. The current time is represented by t. |Q*g*EDF| represents the length of the queue. We define a group in our gEDF algorithm as

g-EDF group={ T*k* |T*k* € QgEDF,$dk\text{-}d1 \le D1*Gr, 1 \le k \le m$ where $m \le$|Q*g*EDF|}

*D*1=deadline of first transaction i.e. smallest one

Gr=group range factor=0.4(assumed)

E. Algorithm

```
Enqueue (QgEDF , T)
    If (Ti's deadline $d>t$) then
    Insert transaction T into QgEDF by EDF i.e. $di \le di+1 \le di+2$
where Ti, Ti+1, Ti+2 € QgEDF, $1 \le i \le$|Qgedf|-2;
    End

Dequeue (QgEDF)
    Ek=average execution time=1.5*block size
    If QgEDF ≠ ø then
    Find a transaction Tmin with emin=min{ek|Tk € gEDF
    Run it & delete Tmin from QgEDF
    End
```

Enqueue is invoked on job arrivals and Dequeue is called when the disk becomes idle. The algorithm needs to sort the jobs in each group, which could incur more overhead during execution than EDF. However, in most practical systems, the number of jobs in a group is small and the added runtime overhead will be negligible.

F. Flow of Algorithm

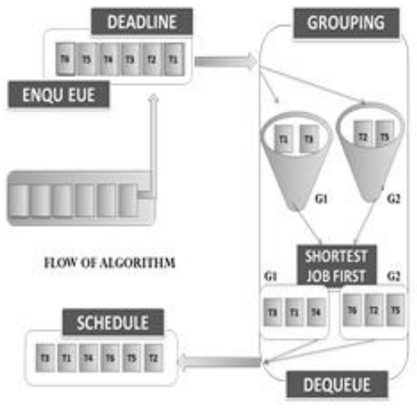

Fig. 1. Flow of *g*-EDF.

G. Drawback of Algorithm

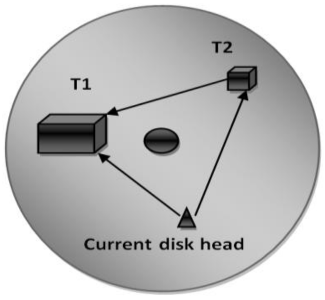

Fig. 2. Drawback of *g*-EDF.

In Fig. 2 we can see that transaction T1 is of larger size and transaction T2 is of smaller size. Also we can observe that distance of T2 transaction from current head position is greater than the distance of T1 transaction from current head position. When we use *g*-EDF algorithm here, it is observed that T2 is scheduled first and T1 follows it. But if deadline of T2 is smaller than it's seek time then it is get missed and when we get to T1 then its deadline is also over so both transaction are missed. So there is no point going like this in such manner i.e. why to go for non-feasible transaction?

So to remove such drawback we had modified this algorithm and proposed a new algorithm which consider deadline, seek time as well as feasibility factor all together. We had discussed in next section.

III. Proposed Approach

A. Fg-EDF Algorithm

We had seen in g-EDF algorithm that once a group is made then within that group we apply shortest job first technique. But what if a transaction is having small size but it is having longest seek time which is greater than its deadline. So when we apply *g*-EDF algorithm we missed it as it is not feasible as well as also those transaction which are very close to disk head, but of large transaction size. So to remove such constraint we propose a new algorithm by modifying earlier one and named it **FEASIBLE GROUP EDF ALGORITHM**. In this algorithm after making groups using group range *Gr* we first check the feasibility of each transaction within the group and if all the transaction is feasible then we apply shortest seek time first algorithm within the group and make the schedule.

How To Check Feasibility…?

- Access(*n*)= Total Transaction Time
- Then check this condition

if ((current time + Access(*n*))<= deadline)
then feasible
else it is not feasible

1) Algorithm

```
Enqueue (QgEDF , T)
    If (Ti's deadline d>t) then
    Insert job T into QgEDF by EDF i.e. di≤di+1≤di+2
where Ti, Ti+1, Ti+2 € QgEDF, 1≤i≤|Qgedf|-2;
    End
 Dequeue (QgEDF)
```

- Select transaction from first group having minimum seek time(Tmin) from current head
  Check
  If (Deadline of *T*min >= (Current Time + Access(*T*min))
  Then *T*min is Feasible. Run it &delete Tmin from Q*g*EDF. Set Current head= End block of Tmin
  Else Delete Tmin From Q*g*EDF
- **If** there are more transactions in first group. Then select next transaction from that group having next minimum seek time and **goto Check**
- Repeat same procedure for all the groups

2) Example

TABLE I: PARAMETER CALCULATIONS

\ T-Id	RI	Block Location	Block Size	SI	EI	AET	DI	TT
T0	1	12	3	12	14	4.5	10	1.8
T1	0	10	2	10	11	3	6	1.2
\ T2	0	5	5	5	9	7.5	15	3
T3	4	19	6	19	24	9	23	3.6
T4	4	15	4	15	18	6	17	2.4
T5	5	11	5	11	15	7.5	19	3
T6	6	13	6	13	18	9	24	3.6
T7	6	7	5	7	11	7.5	21	3

Service Table (CURRENT HEAD POSITION =4)

TABLE II: SERVICE TABLE

Cji	0	1	2	3	4	5	6	7
0	-	2.4	5.7	5.1	2.7	3.9	3.9	5.1
1	2.1	-	4.8	6	3.6	3	4.2	4.2
2	2.7	1.5	-	6.6	4.2	3.6	4.8	3.6
3	5.4	5.4	8.7	-	5.1	6.9	6.9	8.1
4	3.6	3.6	6.9	3.9	-	5.1	5.1	6.3
5	2.7	2.7	6	4.8	2.4	-	4.2	5.4
6	3.6	3.6	6.9	3.9	3.3	5.1	-	6.3
7	2.1	1.5	4.5	6	3.6	5	4.2	-

a) EDF Schedule (Fig. 3)

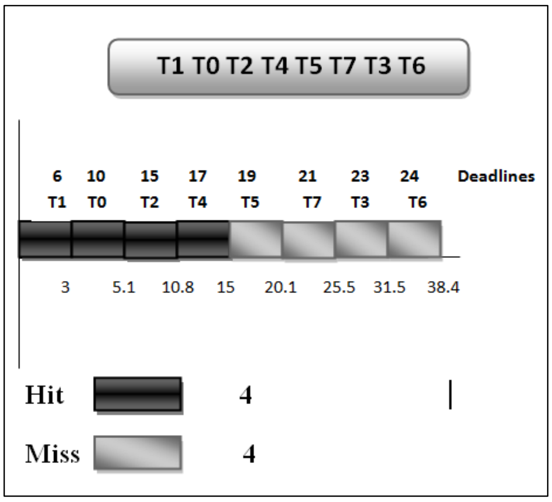

Fig. 3. Timing diagram for EDF.

b) g-EDF Schedule

Enqueue

- Insert the transaction T0, T1, . . ., T7
  in a queue in EDF order as (T1, T0, T2, T4, T5, T7, T3, T6)
- Making Groups
  (Group range factor Gr=0.4)

D1* 0.4=6*0.5=3
D0-D1<=3 false
Therefore G1={ T1 }

D0 * 0.4=10*0.4=4
D2-D0<=4 false
Therefore G2={ T0 }

D2*0.4=15*0.4=6
D4-D2<=6 true
D5-D2<=6 true
D7-D2<=6 true
D3-D2<=6 false

Therefore G3={ T2 T4 T5 T7 }

D3*0.4=23*0.4=9.2

D6-D3<=9.2 true

Therefore G4={ T3 T6 }

After applying SJF in each group the Final g-EDF Schedule is shown in Fig. 4.

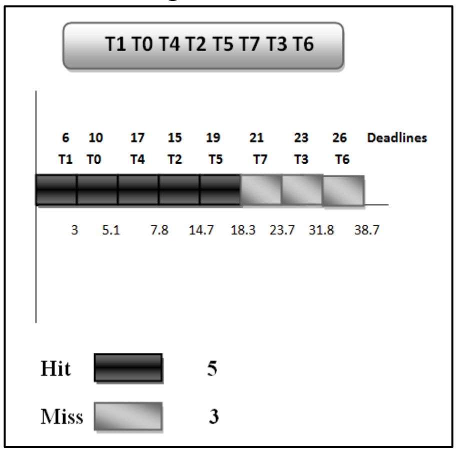

Fig. 4. Timing diagram for g-EDF.

B. *Fg-EDF Schedule*

The groups formed are

G1={ T1 }

G2={ T0 }

G3={ T2 T4 T5 T7 }

G4={ T3 T6 }

1) For group G1 ={ T1 }

Current Head = 4 Current Time = 0

Seek Time for T1=(4-10)*0.3=1.8

Here seek time of T1 is 1.8 and in group G1 there is only one transaction. Therefore we check feasibility of T1 using:

(Deadline of T1 >= (Current Time + Access (T1))

(6>= (0+1.8+1.2)) true

Therefore T1 is feasible.

2) For group G2={ T0 }

(Deadline of T0 >= (Current Time + Access (T0))

(10>= (3+2.1)) true

Therefore T0 is feasible.

3) For group G3={ T2 T5 T4 T7}

(Deadline of T4 >= (Current Time + Access (T4))

(17>= (5.1 + 2.7)) true

Therefore T4 is feasible.

(Deadline of T5 >= (Current Time + Access (T5))

(19>= (7.8 + 5.1)) true

Therefore T5 is feasible.

(Deadline of T7 >= (Current Time + Access (T7))

(21>= (12.9 + 5.4)) true

Therefore T7 is feasible.

(Deadline of T2 >= (Current Time + Access (T2))

(15>= (18.3 + 4.8)) false

Therefore T2 is not feasible

4) For group G4={ T3 T6 }

(Deadline of T6 >= (Current Time + Access (T6))

(24>= (18.3 + 4.2)) true

Therefore T6 is feasible.

(Deadline of T3 >= (Current Time + Access (T3))

(23>=(22.5+3.9)) false

Therefore T3 is not feasible

After applying SSTF in each group the Final Fg-EDF **Schedule is**

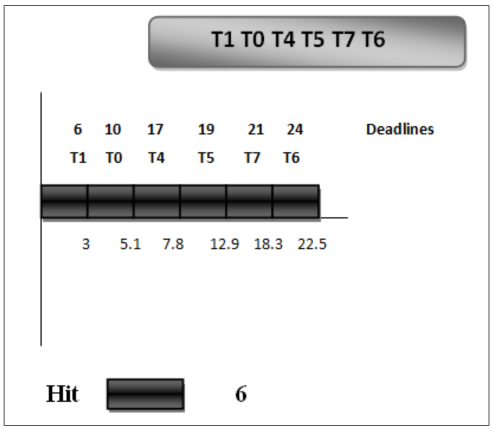

Fig. 5. Timing diagram for Fg-EDF.

IV. Performance Graph (Fig. 6)

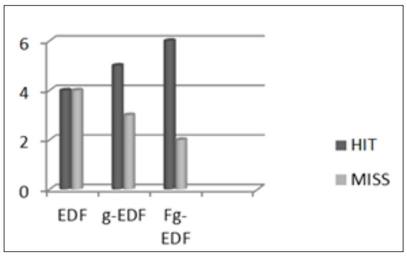

Fig. 6. Results Comparisons.

References

[1] B. Kao and Hector Garcia-Molina "An overview of real-time database systems," in *proc. NATO Advanced Study Institute on Real-Time Computing*, St. Maarten, Netherlands Antilles, Springer-Verlag, 1993.

[2] J. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, *Deadline Scheduling For Real-Time Systems: EDF and Related Algorithms*, Kluwer Academic Publishers, Boston, 1998.

[3] C. L. Liu and J. W. Layland., "Scheduling Algorithms for Multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, 1973.

[4] A. N. Reddy and J. Wyllie, "Disk scheduling in multimedia I/O system," in *Proc. ACM Multimedia'93, Anaheim, CA,* August 1993, pp. 225-234.

[5] W. M. Li, "Group-EDF-a new approach and an efficient non-preemptive algorithm for soft real-time systems," Doctor of Philosophy in Computer Science, August 2006.

S. Y. Amdani received his M.E. CSE degree from SGB Amravati university, Amravati in 2008, and research scholar from 2009. He is now working as an associate professor and Head in Deptt. Of CSE B.N.C.O.E., Pusad (India), and he is a life member of Indian Society for Technical Education New Delhi.

M. S. Ali is currently working as a Principal at Prof Ram Meghe College of Engineering & Management, Badnera-Amravati. He did his B.E (Electrical) from Government College of Engineering, Amravati in 1981, M. Tech. from IIT Bombay in 1984 and Ph.D. from S.G.B. Amravati University in 2006 in the faculty of Engineering & Technology in the area of e-Learning. He is a life member of ISTE, New Delhi, Fellow of IETE, New Delhi and Fellow of IE (India). He is the Chairman of IETE Amravati Local Cente.