# Real-time Disk Scheduling for Mirrored Disk Systems

**QIN XIAO*, PANG LI-PING*, HAN ZONG-FEN*, GUI CHAO*, and LIU JIE***

*Huazhong University of Sci. & Tech., Department of Computer Science, Wuhan Hubei, 430074, P.R.China

**Abstract.** This paper presents a real-time model of the mirrored disk system. Earliest Deadline First(EDF) and Feasible Earliest Deadline First(F-EDF) are real-time disk scheduling algorithms for mirrored disk systems. Comparing real-time and non-real-time scheduling algorithms, we find that F-EDF is the best one for real-time mirrored disk systems.

**Key words.** Mirrored disk system; Real-time disk scheduling; Simulation; Performance evaluations

## 1 INTRODUCTION

In order to support real-time database and real-time information systems, the algorithms for scheduling I/O requests should also have some real-time properties. Robert et al.(1990) studies the algorithms that schedule I/O requests with deadlines. Mirrored disk system stores the same data on each disk. When reading a data from the system, the choice of the disk to be accessed is made by applying the "minimum distance" policy. When writing new data into the disk system, all the disks must be written the same data. Bitton et al. (1989) present a scheduling for the mirrored disk system. The performance of the mirrored disk system is much better than single disk system, especially when read request load is higher than that of write request. Real-time I/O requests have deadlines, so real-time disk scheduling should be carefully designed in order to meet the deadlines of the I/O requests.

## 2 BASIC MODEL

**Definition 1:** Every disk is modeled as a tuple, DISK = (id, st, hd, rq), where id is the identity of the disk, st denotes the status of the disk; hd represents the location of the disk head and rq is a request which is being served by DISK; st = BUSY | FREE. hd is a integer and hd $\in$ [1, MC], where MC is the maximum cylinder of the disk.

**Definition 2:** The mirrored disk system is defined as a set of disks, assume the system has k disks, MD = $\{d_1, d_2, \ldots, d_K\}$; di is a DISK, where i = 1, 2, ... , k;

MD = FS $\cup$ BS, and FS $\cap$ BS = $\varnothing$, where FS is a set for free disks, and BS is a set for busy disks, FS = { d | d.st = FREE, d $\in$ MD }, BS = { d | d.st = BUSY, d $\in$ MD }.

**Definition 3:** I/O request is a tuple: IORQ = (rw, c, dk, at, dl), where rw is the type of the request, rw = READ|WRITE; the data which is accessed by the IORQ is on the cylinder c; IORQ must be served by disk dk, dk$\in$[1, k]|ANY-DISK, when dk = ANY-DISK, this request can be served by any disk; at is the arrival time of the I/O request; dl denotes the deadline of the request.

**Definition 4:** RS is a request set in MD, RS = WS $\cup$ SS, WS is a waiting set which contains all the requests which are waiting to be served; SS is a serving set, the requests in SS are being served. WS has k+1 subsets, defined as follows: WS = RDS $\cup$ $WTS_1$ $\cup$ $WTS_2$ $\cup \ldots \cup$ $WTS_K$ , RDS = {iorq | iorq.rw = READ, iorq $\in$ RS}, WTSi = {iorq | iorq.rw = WRITE, rq.dk = i, iorq $\in$ RS}, i = 1, 2, ..., k.

**Definition 5:** A deadline d is called feasible if the deadline can be met. When d $\geq$ t + Access-time(n), deadline d is feasible, where Access-time(n) is the time for the I/O request being served by a disk and t is the current time.

**Definition 6:** NMS( di ) denotes a set of requests, it contains requests with feasible deadlines feasible, NMS(di) = {rq | rq.dl is feasible, rq$\in$ RDS$\cup$WTSi };

## 3 DISK SCHEDULING

In user's level, we model I/O request by a pair URQ = (rw, c, dl), where rw indicates the type of request , c is the cylinder of the data which is accessed by the request, and dl is the deadline of URQ. Disk scheduling has two parts: insert and dispose. The function of the insert algorithm is translating URQ into IORQ, and moving IORQ into WS, then if there are some idle disks in the system, algorithm will select a proper disk to serve IORQ.

**Insert Algorithm:**

(1) if (urq.rw = READ) then do (2); else do(9);
(2) Produce a iorq = (READ, urq.c, ANY-DISK, Current-Time, urq.dl);
(3) Insert iorq into RDS;
(4) if ( FS $\neq \varnothing$ ) then do next step; else goto (15)

(5) Find a disk d∈ FS where
$\forall d` \in FS$ (|d.hd-ioreq.c| ≤ |d`.hd-iorq.c|);
(6) Allocate d to iorq; Move d from FS into BS;
(7) Move iorq from RDS into SS;
(8) Disk d start positioning; goto (15)
(9) for i = 1 to k do Produce a iorq = (WRITE, urq.c, i, Current-Time, urq.dl); Insert iorq into WTSi;
(10) if ( FS ≠ ∅ ) do next step; else goto(15);
(11) Allocate $d_i \in$ FS to iorq which in WTSi;
(12)Move $d_i$ from FS into BS;
(13)Move iorq from WTSi into SS;
(14)Disk $d_i$ start positioning; goto step (10)
(15)End of Insert;

When a disk in the system has finished a I/O request, dispose is evoked, it is applied as follows:
**Dispose( di )**
(1) Delete di.rq from SS;
(2) if ( RDS ∪ WTSi = ∅ ) then do (3); else do (4);
(3) di.st = FREE; Move di from BS into FS;goto(7);
(4) iorq = Schedule(RDS∪WTSi);
(5) Allocate di to iorq;
(6) Move iorq from RDS∪WTSi into SS;
(7) End of Dispose;

Dispose evokes scheduling algorithm represented by Schedule(RDS∪WTSi). FCFS and Short Seek-Time First(SSTF) are non real-time disk scheduling algorithms studied in many papers (Hofri, 1980). EDF and F-EDF are real-time disk scheduling algorithms. EDF is a algorithm which schedules the request according to their deadlines only, the I/O request with the earliest deadline will be scheduled first. EDF does not care whether the chosen I/O request has missed its deadline or not. In F-EDF algorithm, only the request whose deadline is feasible will be chosen, it may happen that all requests in the WS are missing their deadlines, so the order of service is no longer important, SSTF will be used to serve the requests as soon as possible.
**Earliest Deadline First( EDF ):**
(1) Find iorq ∈ RDS ∪ WTSi where
$\forall iorq' \in$ RDS∪WTSi( iorq.dl ≤ iorq′.dl );
(2) Return(iorq);
(3) End of EDF;
**Feasible Earliest Deadline First( F-EDF ):**
(1) if ( NMS ≠ ∅ ) then do (2); else do (3);
(2) Find iorq ∈ NMS where ∀iorq′ ∈ NMS( iorq.dl < iorq′.dl ); do (4);
(3) Find iorq ∈ RDS ∪ WTSi where ∀iorq′ ∈ RDS ∪ WTSi( | di.hd - iorq.c | ≤ | di.hd - iorq′.c | );
(4) Return(iorq);
(5) End of FEDF;

## 4 SIMULATION RESULTS AND DISCUSSION

Assume that I/O requests arrive according to Poisson Processes with λ, so the mean inter-arrival time is 1/λ. There are 100,000 I/O requests in our experiment. I/O requests are modeled by a sequence of independent random variables uniformly distributed over [1, 1000]. Access - time(n) = DiskFactor*sqrt(n) + DiskConstant, where DiskFactor is the seek time scaling factor and DiskConstant represents the latency time and data transfer time respectively. DiskFactor is 0.88 and DiskConstant is 9.66. The deadline of the request is determined as the following equation: Deadline = Arrival-time + Access-time(n) + Slack-time, Slack time is modeled by a random variable uniformly distributed over [10, 45]. Percentage of missed deadlines is given as: PMD = (Missed read requests + Missed write requests) * 100 / Total requests.

Because of the space limitation, we only illustrate the PMD of the disk scheduling algorithms. Assume there are 4 disks in system, and percentage of read requests is 100%. Comparing FCFS, SSTF, EDF and F-EDF, we find that the performance of FCFS and EDF are nearly the same, and F-EDF has the best performance across all load settings. At the low settings(λ <70 No./Sec.), PMD of all the algorithms have no significant difference. At the high load(λ=150 No./Sec.), F-EDF misses 88.3% fewer deadlines than that of FCFS and 33.3% fewer deadlines than that of SSTF. SSTF is the next best algorithm and FCFS is the worst one.

## 5 CONCLUSION

A model of the mirrored disk system is presented formally and two real-time disk scheduling algorithms are presented. Among disk scheduling algorithms for mirrored disk systems, F-EDF has the best performance and SSTF is the second best algorithms.

## 6 EFERENCES


Abbott.Robert and Hector Garcia-Molina (1990) 'Scheduling I/O Request with Deadlines: a per-formance Evaluation,' IEEE Real-time Systems Symposium, pp.113-124.

D.Bitton and J.Gray (1989) 'Arm scheduling in shadowed disks,' Proc. of IEEE COMPCON 89 conference, pp.132-136.

M.Hofri (1980) ‘Disk Scheduling: FCFS vs. SSTF Revisited,’ Communications of the ACM, Vol 23, pp.645-653, Nov.

**Acknowledgments:** This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense, P.R.China under contracts Nos. 15.3.2.