ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st& 22nd March Organized by

Recovery of Disk Failure in RAID-5 Using Disk Replacement Algorithm

M.P.Ramkumar Dr.N.Balaji , G.Rajeswari

Dept of CSE, Thiagarajar College of Engineering , Madurai, India.

Professor & Head , Dept of IT, KLN College of Engineering, , India.

Dept of CSE, Thiagarajar College of Engineering, Madurai, India.

Abstract— In (Redundant Arrays of Independent Disk) RAID, failure of single disk is tolerated up to Level 5. The performance of disk drives may degrade due to I/O requests directed towards failed disk. RAID controller supports data reconstruction in the event of a disk failure, to recover the data from failed disk. In degraded mode, the recovery of data from failed disk would cause additional workflow in all operational disks. Another mode of recovery called Hot-spare, where disks that are in the end of their lifetime are replaced with spare disks without requirement for additional I/O operations and parity calculations. In this work an existing disk replacement algorithm known as M-SSTF (Modified SSTF) is used for early recovery of failed disk. Early recovery of single disk failure might help tolerating another subsequent disk failure. Hence it is important to rebuild a failed disk as early as possible.

Index Terms— RAID, Disk Failure, Parity, Disk Reconstruction, Disk Replacement algorithm**.**

I. INTRODUCTION

Many businesses, including financial institutions, pharmaceutical companies, and trading companies, must retain data for several years to meet legal regulatory requirements. Emerging applications require large, high-performance, and reliable systems with high data throughput and short response times for requests. The performance of storage subsystem during its recovery from a disk is critical to applications that need both high I/O performance and high data reliability [1].Disk arrays, in particular, RAID-5 has become an accepted way for designing highly available and reliable disk subsystems.

When disk failure occurs, it has an enormous impact on the reliability and data availability of large scale storage systems. If the second disk fails before reconstruction of first failed disk, then data is lost [2]. Losing data is even worse than failing to provide access to it, when it is needed.

This paper describes and evaluates mechanism by which the disk array failure-recovery performance can be improved. The recovery process will be able to restore data and fault-free, without affecting system performance [1]. To improve the performance of the storage system that operates at a high reliable level, a dependable approach is required to recover a failed disk as quickly as possible [3].

There are three modes of operation for a disk subsystem in disk arrays [4], [5]:

A. Normal mode – where all the disks are operational.
B. Degraded mode – where one (or more) disk have failed. If one disk fails, accessing blocks in that makes significant increase in system load to recreate lost data blocks by parity calculations.
C. Rebuild mode - where disks are still down, but process of reconstructing missing information on spare disks is still in progress. The rebuild process should be started quickly after a disk failure occurs, since the pseudo-normal operating mode provides a mean response time close to normal mode.

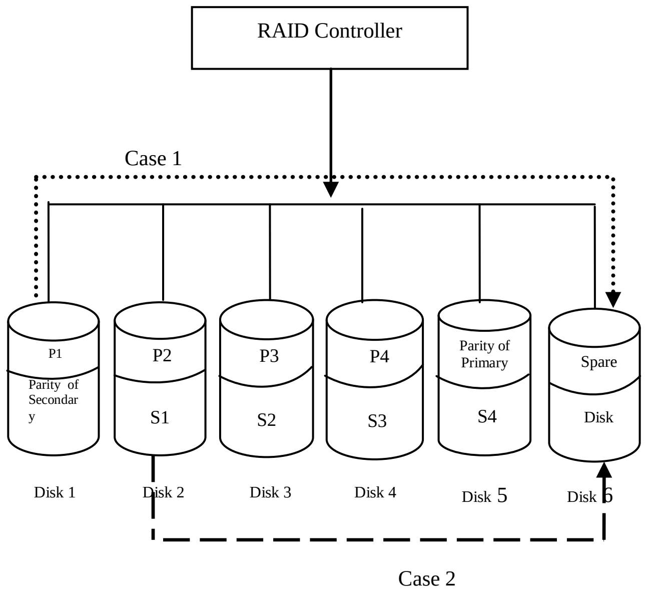

Fig. 1. Disk Recovery in RAID-5

P1, P2, P3, P4- Primary disks
S1, S2, S3, S4-Secondary disks

Fig.1 shows the RAID5 array with primary and secondary data and its parity. Replacement of disk takes place in two cases:

Case 1: Disk which is about to fail is considered based on threshold value of lifetime [6] and replaced using a spare disk. Failed disk is mirrored in a spare disk without affecting normal I/O requests of other disks.

Case 2: Failure of disk leads to operation of RAID under degraded mode. In this case, disk replacement is done from secondary copy in rebuild mode. Failed disk is identified using parity of primary data. The copy of primary that resides on secondary is also identified using relevant parity. Failed disk is recovered using disk replacement algorithm.

Performance of disk array in degraded mode is eradicated by fast recovery using disk replacement algorithm with reduced seek time.

II. RELATED WORK

Thomasian Alexander, and Mario Blaum [1], used a tradeoff between RAID-1 and RAID-5, named Track-Based Recovery (TBR) to balance the storage efficiency and the recovery performance. This algorithm provides a tradeoff between block-based recovery and cylinder-based recovery, and balances the user response time and the recovery duration. However, TBR requires much more buffer space compared to block-based recovery.

In another work, Xin, Qin, Ethan [2] analyzed the performance of RAID with respect to reconstruction algorithms. They describe and briefly evaluate two alternatives termed stripe-oriented reconstruction and disk-oriented reconstruction.

Feasible group-EDF algorithm proposed by S. Y. Amdani and M. S. Ali [7], works both in under load and overload conditions and produced better throughput compared to older algorithms. It also applies Shortest Seek Time First (SSTF) algorithm and check feasibility of transaction [7].

Hossein Rahmani, Mohammad Mehdi Faghih and Mohsen Ebrahimi Moghaddam [8], proposed a novel real-time disk-scheduling algorithm called WRR - SCAN (Weighted-Round- Robin-SCAN) to provide quality that guarantees for all in-service streams encoded at variable bit rates and bounded response times for periodic jobs.

Ajay Dholakia, Evangelos Eleftheriou, Xiao-Yu Hu, Ilias Iliadis, and Jai Menon [9], use a EVEN ODD technique, for tolerating up to two disk failures in RAID 6 architectures. EVEN ODD employs the addition of only two redundant disks and consists of simple exclusive-OR computations. This redundant storage is optimal, in the sense that two failed disks cannot be retrieved with less than two redundant disks.

Paolo Valente and Fabio Checconi [10], revealed that high throughput can be recovered by just idling the disk for a short time interval after the completion of each request. Budget Fair Queuing (BFQ) combined with proper back-shifting of request time stamps may allow a time-stamp-based disk scheduler to preserve and guarantees a high throughput.

III. PROPOSED WORK

The intention here is to identify whether the disk is in the end of its lifetime or has failed. Once the disk is identified attempt is made to recover it at the earliest, so that system runs on Normal mode. The identified disk is replaced abruptly using a disk replacement algorithm.

A. Identifying Disk Failure

When one disk fails, accessing I/O requests in that disk would require reconstruction of data on remaining disks in RAID set. Increase in I/O requests for data recovery from surviving disk affects the disk bandwidth [7]. Mean Time Between Failure (MTBF) measures (in hours) the average life expectancy of a disk. Today, data centers deploy thousands of disks in their storage infrastructures. The greater the number of disks in a storage array, the greater the probability of a disk failure in the array. For example, consider a storage array of 100 disks, each with an MTBF of 750,000 hours. The MTBF of this collection of disks in the array, therefore, is 750,000/100 or 7,500 hours. This means that a disk in this array is likely to fail at least once in 7,500 hours [11]. In a RAID group of 14, a 144GB disk on a Fibre Channel interface will require a minimum of 3 hours with no other I/O to reconstruct the failed disk. A 500GB will require 10.4 hours to read all other disks and reconstruct a failed disk [6].

Identifying the disk which is going to fail uses a threshold value of disk lifetime as 7500 hours for 100 disks in storage array. This paper considered RAID-5 which tolerates single disk failure can be recovered by parity based reconstruction in degraded mode. Failed disk can be identified and recovered using mirroring or parity calculations. Mirroring maintains the copy of primary disk in the secondary disk. The primary and secondary disks can satisfy read requests in parallel to enhance performance in terms of throughput and response time.

Parity methods tend to increase response time due to recalculation of parity and recovery of data.

B. Disk Replacement Algorithm

This work had deliberate to use existing disk scheduling algorithm known as M-SSTF (Modified – SSTF) [12], an evolution of basic Shortest Seek Time First (SSTF) algorithm [13]. The objective of this algorithm is:
1. Reducing seek time
2. Minimizing response time of requests

In this method, either the identified disk or the disk that has primary and secondary parts of failed disk can be divided into two areas as lower half and upper half area based on current position of read/write head [12].

Recalculation of parity is required if the failed disk has parity. Data recovery from failed disk takes much time and causes overall disk I/O requests to be delayed. So, early recovery of failed disks can be achieved by reducing disk seek time.

M-SSTF ALGORITHM

```
Procedure M-SSTF ( read requests )
    /* FD-Failed disk, SD-Secondary disk,
    LH- lower half area, UH- Upper Half area*/
BEGIN
    // Identify FD
    // Create Queue of read requests from FD/SD.
    /* Using current read/write head position in FD/SD
    consider the disk into two areas: LH and UH */.
    // Count number of requests on both areas.

IF (number of requests in LH > number of requests in UH)
        Serve LH requests
    ELSE  Serve UH requests
END IF
IF (number of requests in LH < number of requests in UH)
        Serve UH requests
    ELSE  Serve LH requests
END IF

IF (number of requests in LH = = number of requests in UH)
    IF (current read/write head position is in LH)
            Serve LH requests
        ELSE  Serve UH requests
    END IF
END IF

    // end Procedure
    END
```

C. Performance Evaluatuion

The key performance metrics in disk array are seek time, rotational latency, and data transfer rate [14], [2].

Seek time is defined as time taken to position the R/W heads across the platter with a radial movement (moving along the radius of the platter) [11].

Rotational Latency is defined as time taken by the platter to rotate and position the data under the R/W head [11].

Data transfer rate is the average amount of data per unit time that the drive can deliver to the HBA [11].

Considering these metrics, the simplest way to decrease rotational latency is to increase the disks rotational speed as it's depend on hardware components. Also data transfer rate is depending on the disk subsystem components and its interface bandwidth. Reducing average seek time only requires facts about the relative seek distance between the requested data [14]. As the storage capacity of disk array is growing at faster rate than disk I/O speed, the disk recovery process takes much longer time [15].

Since the seek time is responsible for the most time of disk access, most studies on the disk scheduling have focused on the reduction of the average seek distance or the number of cylinders from the current head position to the requested cylinder to improve the response time [16].

1) Reducing Seek Time Latency:

To maintain disk array in Normal mode, failed disk is replaced at the earliest by reducing seek time. Assume the following three different cases of I/O requests to failed disk.

Seek time is calculated using existing M-SSTF algorithm and the results are compared with existing disk replacement algorithms like FCFS (First Come First Serve), SSTF.

Case 1: Consider the disk with 6 I/O requests to blocks on cylinders 17, 2, 34, 25, 45, 20. M-SSTF algorithm count the number of requests on lower half area and upper half area with the assumption that current read/write head position is at cylinder 30.

As per the algorithm, number of requests on lower half area is higher, so it will be served first. From current head position serve the requests in lower half area first based on Shortest seek time. So here first move of read/write head is to cylinder 25, since it is the shortest seek distance in lower half area. Similarly all remaining requests in lower half area and then upper half area requests are served.

The total head movement is 67 cylinders .For the same request queue, the total head movement is 79 cylinders in SSTF and 114 cylinders in FCFS. Fig. 2 shows the read/write head movement of M-SSTF, SSTF and FCFS algorithms respectively.

Comparison of M-SSTF algorithm with SSTF and FCFS algorithms based on Total Head Movement and Average Seek Time is shown in Table 1. M-SSTF takes average seek time as 11.16ms for recovery while SSTF and FCFS takes average seek time as 13.16ms and 19ms respectively. From this analysis, M-SSTF algorithm has reduced average seek time and Total Head Movement.

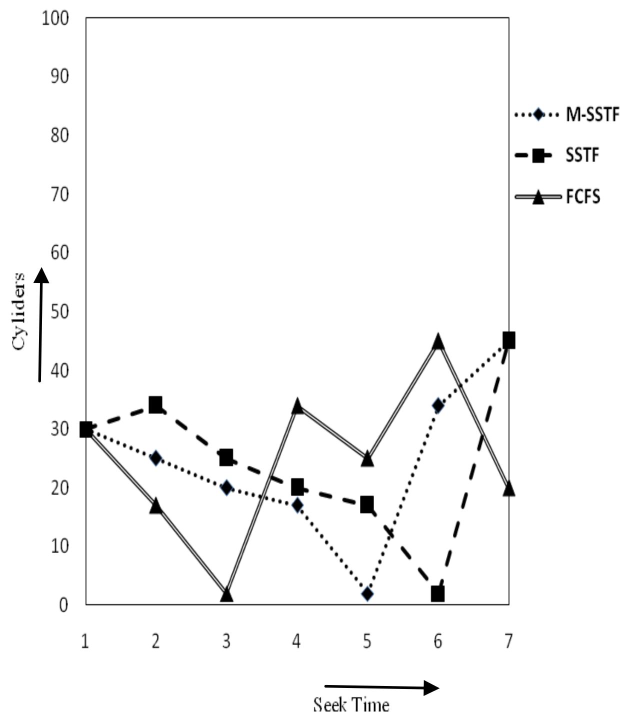

Fig. 2. Disk head movement for Case1

TABLE 1:

Algorithms	Total Head Movement (No. of Cylinders)	Average Seek Time(ms)
M-SSTF	67	11.16
SSTF	79	13.16
FCFS	114	19

Case 2: Consider disk array with 8 requests on cylinders 50, 58, 64, 12, 78, 60, 73, and 38. M-SSTF checks number of requests on lower half and upper half from the present read/write head position. If read/write head is presently at cylinder 55 then lower half has 3 and upper half has 5 requests. M-SSTF algorithm will serve upper half requests first and then lower half requests, since upper half requests are more than lower half requests. Read/write head is first moves to cylinder 58 since it is the nearest from the current read/write head position. All the requests in the upper half area have been served using shortest seek time, then read/write head is ready to serve lower half requests .

The total head movement is 89 cylinders. For the same request queue, the total head movement is 99 cylinders in SSTF and 203 cylinders in FCFS algorithms. Fig. 3 shows the read/write head movement of M-SSTF, SSTF and FCFS algorithms respectively.

M-SSTF algorithm has average seek time 11.25ms and Total Head Movement as 89. SSTF and FCFS algorithm has average seek time as 12.37ms, 25.37ms and Total Head Movement as 99, 203 respectively. M-SSTF has less average seek time and total head movement where compared with SSTF and FCFS algorithms as shown in Table2.

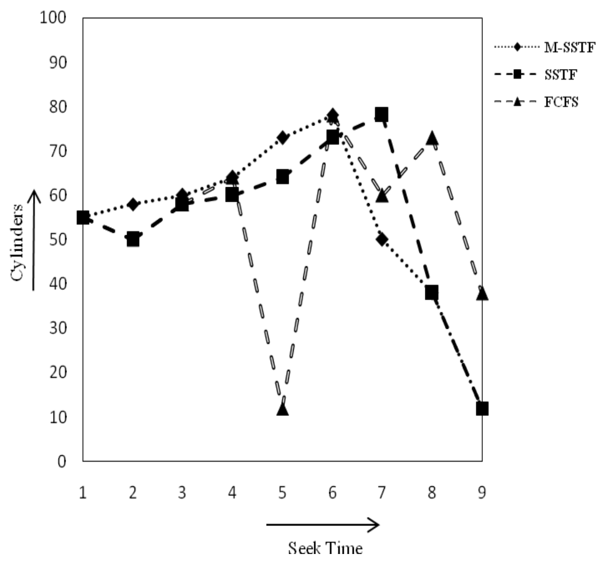

Fig. 3. Disk head movement for Case2

TABLE 2:

Algorithms	Total Head Movement (No. of cylinders)	Average Seek Time(ms)
M-SSTF	89	11.25
SSTF	99	12.37
FCFS	203	25.37

Case 3: Consider disk array with 10 requests on cylinders 22, 30, 6, 92, 81, 53, 90, 100, 27, and 14. M-SSTF checks number of requests on lower half and upper half based on current read/write head position. If read/write head is presently at cylinder 45, then lower half has 4 and upper half has 4 requests. As per the algorithm, since number of requests is same on both lower and upper half, now decision is based on location of current read/write head position. Since read/write head is currently in lower half, first it moves to cylinder 30 and serves rest of the lower half requests. Once all the requests in the lower half area have been served; now read/write head is ready to serve upper half requests.

From Table 3, the M-SSTF algorithm takes average seek time as 12.9ms which is less than other algorithms like SSTF takes 14.9ms, FCFS takes 31.3ms.

The total head movement is 129 cylinders. For the same request queue, the total head movement is 149 cylinders in SSTF and 313 cylinders in FCFS. Fig. 4 shows the read/write head movement for M-SSTF, SSTF and FCFS disk replacement algorithms.

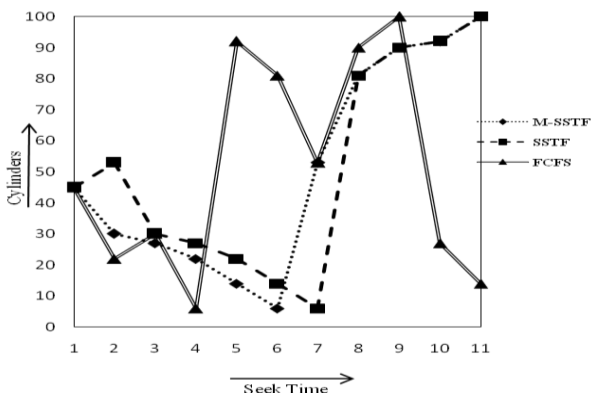

Fig. 4. Disk head movement for Case3

TABLE 3:

Algorithms	**Total Head Movement (No. of cylinders)**	**Average Seek Time(ms)**
M-SSTF	129	12.9
SSTF	149	14.9
FCFS	313	31.3

Hence from Table 1, 2, and 3, it is revealed that M-SSTF algorithm has taken less average seek time when compared with SSTF and FCFS disk replacement algorithms.

IV. CONCLUSION

The existing M-SSTF algorithm used for disk scheduling has been realized for disk replacement in this work. It has improved performance in the early recovery process of RAID-5. The basic idea is to treat the faulty disks more favorably, or give higher priority to the faulty disks. Identifying a disk which is about to fail or a failed disk is done at the earliest and reconstruction is completed into a spare disk, this will improve the performance of the system by avoiding it from entering into degraded mode of operation. Recovery from disk failure is done so early by reducing the seek time. The number of requests with minimal value has been considered. Experimental results show that M-SSTF disk replacement algorithm has given reduced seek time than SSTF and FCFS disk replacement algorithms. The results based on performance evaluation have demonstrated the effectiveness of M-SSTF in terms of reduced seek time that in turn results in faster recovery. When number of requests increases, definitely there will be a major performance variation in terms of average seek time between the M-SSTF, SSTF and FCFS.

V. FUTURE WORK

As a future work, even though the rotational speed of platters is under the control of spindle motors, finding the conceptual way to reduce rotational latency and probabilities of merging or optimizing the disk replacement algorithms in order to achieve minimized seek time.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers of the paper for their excellent feedback and comments, all of which helped to improve the quality of presentation.

REFERENCES

[1] Alexander Thomasian and Mario Blaum, “Higher reliability redundant disk arrays: Organization, operation, and coding,” *ACM Transactions on Storage (TOS)*, vol. 5, no. 3, pp.1-59, 2009.

[2] Xin, Qin, Ethan L. Miller, Thomas Schwarz, Darrell DE Long, Scott A. Brandt, and Witold Litwin, “Reliability mechanisms for very large storage systems,” *Proceedings on 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, (MSST 2003)*,pp. 146-156, 2003.

[3] Shenggang Wan, Xubin He, Jianzhong Huang, Qiang Cao, Shiyi Li, and Changsheng Xie, “An Efficient Penalty-Aware Cache to Improve The Performance of Parity-Based Disk Arrays under Faulty Conditions, ” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1500-1513, August 2013.

[4] Alexander Thomasian, Gang Fu, and Chunqi Han, “Performance of Two-Disk Failure-Tolerant Disk Arrays,” *IEEE Transactions on Computers*, vol. 56, no. 6, pp. 799-814, June 2007.

[5] Cheng-Han Tsai, Tai-Yi Huang, Edward T.H. Chu, Chun-Hang Wei, and Yu-Che Tsai, “An Efficient Real-Time Disk-Scheduling Framework with Adaptive Quality Guarantee, ” *IEEE Transactions on Computers*, vol. 57, no. 5, pp. 634-657, May 2008.

[6] Elerath, Jon G, and Michael Pecht, “Enhanced reliability modeling of RAID storage systems,” *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, (DSN'07)*, pp. 175-184, 2007.

[7] S.Y.Amdani and M.S.Ali, “An Improved Group-EDF: A Real-Time Disk scheduling algorithm, ” *International Journal of Computer Theory and Engineering*, vol. 5, no. 6, pp. 873-876, December 2013.

[8] Hossein Rahmani, Mohammad Mehdi Faghih, and Mohsen Ebrahimi Moghaddam, “A new real time disk-scheduling method based on GSR algorithm, ” *Elsevier Journal of Systems and Software*, vol. 83, pp. 2147–2164, November 2010.

[9] Ajay Dholakia, Evangelos Eleftheriou, Xiao-Yu Hu, Ilias Iliadis, and Jai Menon, “A new intra-disk redundancy scheme for high-reliability RAID storage systems in the presence of unrecoverable errors, ” *ACM Transactions on Storage (TOS)*, volume 4, May 2008. ISSN: 1553-3077.

[10] Paolo Valente and Fabio Checconi, “High Throughput Disk Scheduling with Fair Bandwidth,” *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1172-1186, September 2010.

[11] G. Somasundaram and A. Shrivastava, Information Storage and Management: Storing, Managing, and Protecting Digital Information. Wiley, 2009.

[12] Manish Kumar Mishra, “Major Half Served First(MHSF) Disk Scheduling Algorithm,” *International Journal of Computer Applications & Information Technology*, vol. 2, pp. 31-35, January 2013.

[13] Hyokyung Bahn, Soyoon Lee, and Sam H. Noh, “P/PA-SPTF: Parallelism-aware request scheduling algorithms for MEMS-based storage devices, ” *ACM Transactions on Storage (TOS)* , vol. 5, March 2009. ISSN: 1553-3077.

[14] Elizabeth Varki, Arif Merchant, Jianzhang Xu, and Xiaozhou Qiu, “Issues and Challenges in the Performance Analysis of Real Disk Arrays, ” *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 6, pp. 559-574, June 2004.

[15] Zhang, Jianyong, Anand Sivasubramaniam, Qian Wang, Alma Riska, and Erik Riede, “Storage performance virtualization via throughput and latency control, ” *ACM Transactions on Storage (TOS)*, vol.2, no. 3, pp.283-308, 2006.

[16] Yin-Fu Huang and Jiing-Maw Huang, “Disk Scheduling on Multimedia Storage Servers, ” *IEEE Transactions on Computers*, vol. 53, no. 1, pp. 77 – 82, January 2004.