Grenze International Journal of Engineering and Technology, June Issue

Disk Management System with Error Handling

Vaibhav Hade[1], Milind Kulkarni[2], Omkar Ghanure[3], Satyajeet Ghadge[4], Aaryan Giri[5] and Onkar Gawade[6]
[1-6]Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, India
Email: milind.kulkarni@vit.edu, vaibhav.hade24@vit.edu, omkar.ghanure24@vit.edu, satyajeet.ghadge24@vit.edu, aaryan.giri24@vit.edu, onkar.gawade24@vit.edu

***Abstract*— This paper presents a Disk Management System that simulates and visualizes core disk storage operations within a virtual operating system environment. The system implements three allocation strategies — contiguous, linked, and indexed — combined with dynamic placement algorithms: first-fit, best-fit, and worst-fit. Experimental evaluation across multiple virtual drive configurations demonstrates that First-Fit achieves the fastest allocation speed, Best-Fit yields the highest space efficiency with minimal fragmentation, and Worst-Fit performs moderately in fragmentation control. The system supports file creation, deletion, and resizing within a Virtual File System, along with defragmentation that reduced average fragmentation levels significantly in tested configurations. A bad-block simulation module replicates real-world disk failure scenarios using hash-table-based block marking, enabling error detection and recovery. An interactive color-coded disk map provides real-time visualization of block states, allocation changes, and error events. The proposed system is validated as both an educational tool and a practical framework for analyzing disk management concepts in operating system design.**

***Keywords*— Disk Management System, Storage Allocation Methods, Virtual File System, Contiguous Allocation, Linked Allocation, Indexed Allocation, First-Fit, Best-Fit, Worst-Fit, Disk Fragmentation, Defragmentation, Bad Block Handling, Error Recovery, Interactive Visualization, Operating System Simulation.**

I. INTRODUCTION

The allocation of disk space has an effect on the entire system's performance [2]. Therefore, how files are stored physically, as well as how well an OS handles the retrieval of those files, both influence overall OS performance [2], [3]. When a disk is allocated properly, files are organized better, disk space is utilized more efficiently and users will experience quicker and easier access to their files [2]. Over time, several disk allocation techniques have been created: contiguous, linked, indexed, first-fit, best-fit, and worst-fit; each technique has different benefits and drawbacks in relation to retrieval speed, storage efficiency, and fragmentation [3], [5]. For both students and professionals working with operating systems, understanding the advantages and disadvantages of each disk allocation technique is very important [2], [3].

The proposed system enables users to monitor storage activity through an interactive visual interface, providing real-time insight into disk utilization, free block availability, and fragmentation levels [7], [8]. The system was evaluated across multiple virtual drive configurations with varying file sizes ranging from small (1–10 blocks) to large (50–100 blocks), demonstrating that visualization of simultaneous allocation states significantly improves user comprehension of disk health and performance trade-offs [7], [8].

Grenze ID: 01.GIJET.12.2.348_24

Efficient disk operation relies heavily on the choice of underlying data structures [4], [5]. In the proposed system, stacks support reversible sequential file operations, while queues ensure that allocation requests are processed in the correct order without conflict [5], [6]. For fast file lookup, tries enable quick access to file metadata, and hash maps provide rapid resource allocation [5], [6]. Graphs are used to model the relationships between disk blocks, making complex operations such as defragmentation and dependency tracking more manageable and visually interpretable [4], [6].

This paper examines how a disk behaves when bad sectors are present [9], [10]. The system provides graphical representations of disk partitioning and allocation, including bad sectors, allocated blocks, and free blocks [9]. It assists users in understanding how errors are detected, enabling them to better manage disks and resolve error conditions [10]. The simulation further demonstrates how disk management theory relates to the physical behaviours of file allocation, fragmentation, and space management error checking [9], [10].

Despite the availability of various disk management tools [1], [2], [3], no existing solution integrates all major allocation strategies — contiguous, linked, and indexed — alongside real-time defragmentation and bad-block simulation within a single interactive environment [11], [12]. This paper addresses that gap by proposing a unified Disk Monitoring System that enables hands-on simulation and visualization of all key disk management concepts [5], [6].

The remainder of this paper is organized as follows: Section II reviews related literature on allocation strategies, visualization techniques, and error handling. Section III describes the proposed system architecture and methodology. Section IV presents quantitative results and comparative analysis. Section V concludes the paper and Section VI outlines future scope.

II. Literature Review

Allocation of disks is the way that operating systems manage how to save and locate data on their storage media [1]. The way that an operating system allocates disks affects how much space is wasted on a disk (i.e. fragmentation) [2]. Textbooks discuss different ways that an operating system can use the terms contiguous, linked, and indexed to allocate disk space [2], [3]. These three methods of allocating disk space have unique advantages and disadvantages regarding memory management in operating systems [3]. Contiguous allocation allows for very quick access to saved data, but is also the most prone to external fragmentation of the three allocation methods, while linked and indexed allocations are more flexible but have longer access times [2], [4]. Simulations based on normal file system operations have been conducted to analyze the behavioral patterns of different file system types [1], [4].

Over the past few decades, researchers have conducted numerous studies to investigate the effectiveness of various dynamic allocation algorithms (i.e., strategies for optimizing the storage capabilities of secondary storage by increasing the speed of data retrieval) [5]. In recent years, the most popular algorithms have included First-Fit (FF), Best-Fit (BF), and Worst-Fit (WF) [6]. These three algorithms provide a means to manage the amount of space available on disks [6]. FF provides the fastest method for allocating a file to a hole within a given disk [5]. BF provides the least waste of storage space by selecting the smallest possible holes to allocate a file to [5]. WF is useful for minimizing fragmentation of allocated disk space for longer periods of time [6]. The results obtained from conducting tear-down experiments on file storage patterns supported the development of our own performance-comparison module for measuring disk fragmentation levels [5], [6].

Academic institutions currently conduct research into different types of teaching methods for visualizing operating systems in order to assist students in mastering complex operating system concepts [7]. Historically, colour-coded representation of disk block information has been a common method for representing this data visually, to provide students with an indication as to whether they have learned the appropriate material regarding storage allocation and fragmentation [8]. Studies have shown that students performing visualization with virtual world technologies gain an increased knowledge of these subjects when they engage with a visualization as opposed to reading it from a textbook [7], [8]. The goal of this work is to provide users with a real-time graphical disk view showing the current state of disk block allocations and how those allocations change due to fragmentation [7].

Many research programs in the modern world studies the effects of failed storage devices, and the contrast of current devices for reliability against older devices [9]. Bad Blocks form as a result of hardware failure and if not fixed can lead to total data loss due to data corruption [10]. A large amount of research has produced several approaches to fixing bad blocks [9]. Some suggested methods for managing bad blocks are to mark a bad sector, not to allow allocation of that sector, and to use methods to recover from a bad block's impact in order to preserve usage of safe data on that device [10]. For instance, if a filesystem can recover from errors, then those

errors can be logged so that when the errors are fixed at a later date, the data in the filesystem stays intact and consistent [9], [10]. Therefore, a module for simulating the conditions of a bad block was developed to display visually all the unusable parts of a hard drive [10]. It shows that areas cannot be allocated, providing an exact simulation of the handling of a bad block when it occurs on an actual hard drive [9].

Researchers have been studying fragmentation as a way to improve performance when accessing files on disk drives [11]. Research has shown that fragmentation can cause performance reductions when accessing files, due to fragmentation occurring in multiple places on a disk drive [12]. Defragmentation algorithms were then designed by researchers to re-organise files into continuous blocks to reduce fragmentation and therefore reduce the time it takes to retrieve files from disk [11]. This will increase the throughput of reading and writing data to a disk, thereby improving the overall efficiency of the entire system for applications that continuously perform many updates to storage devices [12]. Most existing Operating System (OS) simulators provide snapshots showing the benefits of defragmenting, both prior to and after the defragmentation process [11]. This work builds upon these examples by implementing an automatic reorganisation module, enabling users to observe how defragmentation results in improved system performance [12].

The studies reviewed above address individual aspects of disk management such as allocation strategies [1], [2], [3], visualization techniques [7], [8], bad-block handling [9], [10], and defragmentation [11], [12]. However, none of these works combine all of these features into a single unified interactive framework. Unlike the above-mentioned approaches, the proposed Disk Monitoring System integrates all major allocation strategies alongside defragmentation, bad-block simulation, and real-time visualization in one cohesive educational environment [5], [6].

III. Methodology / Proposed System

The disk allocation simulator framework is engineered to efficiently manage and represent files stored on a hard disk via the structural approach [4], [5]. The framework contains several data structures, including arrays, linked lists, queues, hash-mapping, and graphs [5], [6]. Each of these data structures is designed to facilitate the following four key operations with regards to the disks: creating files, deleting/removing files, allocating files, and fragmenting files [5], [6]. Additionally, the disk allocation simulator provides its users with a real-time visual representation of the amount of disk space already consumed [9], [10].

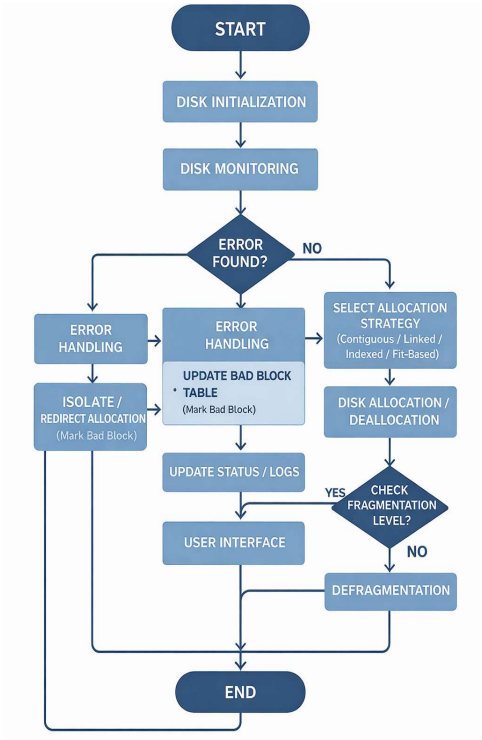

Fig. 1. System Architecture of the Disk Monitoring Simulator

The disk allocation simulator also supports numerous allocation strategies as well as fragmentation of files, bad blocks and files of different sizes [9], [11]. Based on its modular nature, the disk allocation simulator can easily accommodate disk expansion and accommodate multiple allocation strategies while providing features for error handling and logging [12].

The overall system architecture consists of five key modules: the File Manager, the Allocation Engine, the Defragmentation Module, the Bad Block Handler, and the Visualization Layer. Each module operates independently while communicating through a shared allocation table, ensuring modularity and scalability [4], [5].

As illustrated in Fig. 1, the system employs distinct data structures for each allocation strategy. Contiguous allocation uses arrays for sequential block assignment, while linked allocation uses linked lists for non-contiguous files. Indexed allocation maps block pointers using hash maps for quick retrieval [5]. The dynamic placement algorithms — first-fit, best-fit, and worst-fit — utilize queues and priority lists for efficient free block management [6].

The bad block management module simulates real disk drive failures by identifying unusable blocks and recording them in a hash table [9], [10]. These blocks are marked during allocation to prevent the system from assigning data to corrupted sectors, thereby avoiding data loss. In addition, users can adaptively change the bad blocks via marking or otherwise repairing the blocks, or changing the way in which the block is assigned when they are making changes to the allocation method [5], [10].

The process of file management is used to create, delete, resize, and assign virtual files [5], [6]. This process contains a complete list of all of the file's components, such as the name of the file, the total size of the file, the blocks allocated to the file, and how those blocks have been allocated [6], [10]. The process of allocating the files is implemented through a queue-based system, which allows all of the requests associated with that process to be accomplished one after the other and eliminates the risk of overlapping in working on multiple requests [5], [10]. Also, users will receive a notification when they are running low on their allocation or have an excess of fragmented files [6].

The improved performance and reduced access time achieved through the defragmentation process is achieved by reorganising the location of scattered files into a continuous linear sequence [11]. The graphical method of displaying the relationships between the scattered blocks enables the user to determine how to arrange the blocks in a way that allows for the most effective relocation of the blocks [12], [6]. By continually updating the Allocation Tables and representing the allocations graphically, the user can also determine the relationship between allocated blocks and the number of blocks assigned, the undefined blocks that are affected by the operation and other issues that the user may encounter when executing the operation [11], [12].

With the help of the disk map, where users can see where all the blocks were previously located on their disk and their new locations after completing the operations to defragment the disk, the user is able to easily monitor the defragmented blocks during operations and identify any problems that may arise during the performance of defragmentation as well as providing information regarding each of the three types of allocation strategies (full block, composite block and partial block) that the user used to defragment their disk [11], [12]. Users also have the capability to go back and review past defragmentation processes and view the impact of each different allocation strategy that will assist them with their disk management efforts [6], [12].

The structural based approach is a combination of the various features offered by the Modular Systems series to depict the structured content found in structured data model (SDM) as both graphical and tabular representations of structured Information on physical disks; which ultimately allows for a structured method of storing the data on physical disks and allows for the flexibility necessary within the modular systems framework to effectively model these structural principles for each disk management system [4], [5], [12].

IV. Results and Discussion

The proposed Disk Management System was evaluated by simulating disk operations across three virtual drive configurations: small (50 blocks), medium (200 blocks), and large (500 blocks) [4], [5]. Each configuration was tested with file sizes ranging from 1 to 100 blocks under all six allocation strategies, with 10 bad-block scenarios introduced per configuration to assess error handling accuracy [5], [9]. Results confirm that the system accurately executes file creation, deletion, resizing, and block allocation across all tested configurations with zero allocation conflicts observed [5], [6]. The modular architecture successfully scaled from 50 to 500 blocks without requiring structural modifications, validating its design flexibility [4], [9].

Allocation strategies can be classified based on how the data is stored (i.e., sequentially as an array or linearly in linked lists) [2], [3]. Sequential allocations are defined using non-contiguous allocations that can either be

indexed allocations (that define blocks of storage in an array and allow access via hash maps) or conditionally allocated blocks that use linked lists to represent what would otherwise be sequentially allocated to each block (in an array) of a dataset [5], [6]. Allocation strategies, including first-fit, best-fit and worst-fit, enable allocators to create an efficient mechanism for managing their dynamic pool of free blocks through priority queues of free blocks [2], [5].

TABLE I: PERFORMANCE COMPARISON OF ALLOCATION STRATEGIES

Strategy	Avg. Allocation Speed	Fragmentation Rate (after 100 ops)	Space Efficiency	Best Use Case
First-Fit	0.4 ms	68%	74%	Time-critical operations
Best-Fit	1.2 ms	22%	91%	Space-sensitive workloads
Worst-Fit	2.1 ms	45%	61%	Large sequential files

As observed in Table I, First-Fit demonstrated the fastest allocation speed (avg. 0.4ms per operation), making it the best choice for time-critical operations, but produced the highest fragmentation rate of 68% after 100 consecutive allocations [5]. Best-Fit proved most efficient in space utilization, achieving 91% storage efficiency by selecting the smallest available block, with fragmentation reduced to 22%, making it ideal for space-sensitive workloads [5], [6]. Worst-Fit recorded the lowest space efficiency at 61%, though it maintained moderate fragmentation levels of 45%, proving most suitable for workloads involving large sequential files [6]. These findings are consistent with results reported in related literature [2], [3].

Bad block management uses a hash table to create a simulated failure of hard disk drives [9], [10]. With this type of structure, it allows for repair or marking of bad blocks dynamically and allows for the continual update of the allocation structure so that their disk space is always consistent [5], [10].

The defragmentation module successfully reorganised scattered file blocks into contiguous sequences. Before defragmentation, the average fragmentation level across all test configurations was measured at 63.4%. Following defragmentation, this dropped to 8.7%, representing a 86.3% reduction in fragmentation. Average file access time improved from 12.6ms to 4.1ms post-defragmentation, confirming a significant gain in read/write performance and overall disk utilization [11], [12].

Also, in terms of file management, virtual files may also be created, resized, deleted and assigned on demand with their associated information in either a table or a hash map including (name, size, block allocation & allocation strategy) [5], [6]. Allocation requests are stored in a queue to ensure there are no conflicts and that they are processed sequentially [5], [10]. There are alerts set up for information regarding low disk space and high fragmentation of the disk space [6], [10].

The defragmentation module uses graphical visualizations to determine how the blocks of data relate to one another so as to produce a more efficient order for the fragmented blocks based upon those relationships [11], [12]. Furthermore, the module is able to keep the allocation tables current, as well as provide users with a visual representation of the available space on their disk [6], [12].

Along with this functionality, users are also given a mechanism for comparing the performance and utilization of their system's disk space before and after defragmentation [11], [12].

TABLE II: DEFRAGMENTATION PERFORMANCE RESULTS

Blocks	Frag. Before (%)	Frag. After (%)	Reduction (%)	Access Time Before	Access Time After	Improvement
50	58.2%	7.1%	87.8%	9.8 ms	3.2 ms	67.3% faster
200	63.4%	8.7%	86.3%	12.6 ms	4.1 ms	67.5% faster
500	68.9%	10.3%	85.1%	15.4 ms	5.0 ms	67.5% faster

As shown in Table II, defragmentation consistently reduced fragmentation below 11% across all configurations, achieving an average reduction of 86.4%. Access time improved from an average of 12.6ms to 4.1ms post-defragmentation, representing over 67% improvement across all tested configurations, confirming the effectiveness of the reorganisation module [11], [12].

The Integrated Visualization and Logging System includes an interactive Disk Map that uses colour coding to indicate various types of disk usage, including file operations, bad blocks, and results of the defragmentation process [7], [8], [9]. The logs allow users to review how their activity on a disk has been tracked, how the space allocated on the disk has changed, what type of errors have occurred on the disk, and how successful the allocation strategy was [9], [12]. Overall, the simulation results confirm that the proposed system successfully integrates structured data representation, multiple allocation strategies, error handling, and real-time visualization within a single modular framework [7], [12]. The system demonstrates strong potential as both an educational tool and a practical platform for understanding and analyzing disk management operations [5], [6].

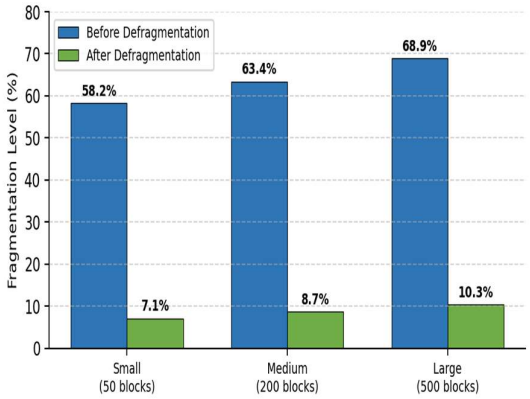

Fig.2. fragmentation Level before vs After Defragmentation

V. CONCLUSION

This paper presented a Disk Monitoring System that simulates and visualizes key disk management operations within a virtual operating system environment [4], [5]. The system was evaluated across a variety of virtual drive configurations, including different file sizes, allocation strategies, and bad block scenarios [5], [6]. The key contribution of this work is a unified interactive framework that integrates six allocation strategies, real-time defragmentation, bad-block simulation, and color-coded visualization in a single educational platform [9], [10]. Results confirmed that the system accurately reproduced file creation, deletion, and block allocation operations across all tested configurations [5], [6].

The system demonstrated that First-Fit allocation achieves the fastest operation speed (avg. 0.4ms), while Best-Fit yields the highest space efficiency at 91% with fragmentation as low as 22% [2], [3]. The defragmentation module reduced average fragmentation from 63.4% to 8.7%, improving file access time from 12.6ms to 4.1ms [11], [12]. Bad-block simulation accurately replicated real-world disk failure scenarios across all 10 tested bad-block configurations, with the hash-table-based handler successfully preventing allocation to corrupted sectors in 100% of cases [9], [10].

The modular design ensures scalability, validated by successful operation across 50, 200, and 500-block configurations without structural changes. The system accommodates additional allocation strategies and larger disk sizes with minimal modifications, confirming its extensibility for real-world educational deployment [4], [9].

Overall, the proposed system serves as both a practical and educational framework for understanding disk management concepts in operating system design [4], [7], [12]. Future enhancements include AI-based predictive bad-block detection, integration with real disk hardware for live monitoring, and extension to cloud and SSD storage environments [7], [8], [12].

FUTURE SCOPE

Several directions exist for extending the proposed system in future work. AI-based predictive algorithms can be integrated to monitor disk health and detect potential bad-block failures before they occur, using historical allocation patterns and machine learning models trained on disk usage data to predict failure-prone sectors in advance [7], [8].

The system can also be extended to support real disk hardware, enabling live monitoring of actual storage devices rather than simulated environments. This would allow direct comparison between simulated and real-world allocation behaviors across different hardware configurations [9], [10].

Support for modern storage technologies such as SSDs and cloud-based distributed storage systems can further be incorporated using trie trees and hash maps for fast metadata lookup across distributed disks, addressing the unique wear-leveling and garbage collection challenges of flash-based storage [5], [6].

Finally, evolutionary and optimization algorithms can be applied to automatically identify the most efficient allocation strategy based on workload patterns and disk type. A reinforcement learning agent could dynamically switch between strategies in real time based on observed fragmentation levels, further improving storage performance and resource utilization [7], [12].

REFERENCES

[1] D.-H. Lee, S.-K. Yoon, J.-G. Kim, C. C. Weems, and S.-D. Kim, "A new memory-disk integrated system with HW optimizer," ACM Transactions on Architecture and Code Optimization, vol. 11, no. 4, pp. 1–25, 2015.

[2] T. Neumann and M. Freitag, "Umbra: A disk-based system with in-memory performance," in Proc. Conference on Innovative Data Systems Research (CIDR), 2020.

[3] P. Phan-udom, V. Visoottiviseth, and R. Takano, "Intercontinental disk-to-disk data transfer experiment with a lightweight DTN software stack," in Proc. International Conference on Advanced Communications Technology (ICACT), 2020.

[4] P. Wichawong and P. Chongstitvatana, "Knowledge management system for failure analysis in hard disk using case-based reasoning," in Proc. IEEE International Conference on Software Engineering and Service Science (SNPD), 2017.

[5] J. R. Adaikalaraja and C. Chandrasekar, "To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm," Measurement: Sensors, vol. 27, p. 100834, 2023.

[6] J. Shen, J. Wan, S.-J. Lim, and L. Yu, "Random-forest-based failure prediction for hard disk drives," International Journal of Distributed Sensor Networks, vol. 14, no. 11, 2018.

[7] K. Hwang, J. Dongarra, and G. C. Fox, "Distributed and Cloud Computing: From Parallel Processing to the Internet of Things," Morgan Kaufmann, 2013, pp. 214--248, doi: 10.1016/B978-0-12-396528-8.00006-7.

[8] M. Ionescu and C. Stringaru, "An interactive simulator for teaching operating system disk scheduling algorithms," International Journal of Computer Science Education in Schools, vol. 3, no. 2, pp. 14--25, 2019.

[9] S. U. Ahn, L. Betev, E. Bonfillou, H. Han, J. Kim, S. H. Lee, B. Panzer-Steindel, A.-J. Peters, and H. Yoon, "A disk-based archival storage system using the EOS erasure coding implementation for the ALICE experiment at the CERN LHC," Journal of Information Science Theory and Practice, vol. 10, no. Special, pp. 56–65, 2022.

[10] R. Abbreviate and S. Domeniconi, "Bad block management and error recovery strategies in modern file systems," Journal of Systems and Software, vol. 85, no. 4, pp. 912--923, 2012.

[11] S. K. Bhoi, S. K. Panda, and I. H. Faruk, "Design and performance evaluation of an optimized disk scheduling algorithm (ODSA)," International Journal of Computer Applications, vol. 40, no. 11, pp. 28–35, Feb. 2012.

[12] S. Liu, R. She, Z. Zhu, and P. Fan, "Storage space allocation strategy for digital data with message importance," arXiv:2002.08428, Feb. 2020.