International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

Apriori-Based Prefetching Files for Caching

Prajwal Said[1], Ketaki Naik[2], Nupur Agrawal[3], Srushti Bhoite[4], Sayali Shelar[5]
[1,2,3,4]UG – Information Technology Engineering, Bharati Vidyapeeth's College of Engineering for Women, Dhankawadi, Pune, Maharashtra, India
[5]Associate Professor – Information Technology Engineering, Bharati Vidyapeeth's College of Engineering for Women, Dhankawadi, Pune, Maharashtra, India
Emails: *saidprajwal@gmail.com[1], ketaki.naik@bharatividyapeeth.edu[2], nupuragrawal810@gmail.com[3], srushti.bhoite12@gmail.com[4], shelarsayali990@gmail.com[5]*

Abstract
The project proposes an innovative solution aimed at optimizing file system performance through predictive caching techniques integrated with a Graphical User Interface (GUI). The GUI facilitates user interaction by offering functionalities such as browsing files and displaying performance metrics via graphical representations of bandwidth and Input/Output Operations Per Second (IOPS). The functionality revolves around dynamically determining file placement on Solid State Drives (SSDs) or Hard Disk Drives (HDDs). The system employs predictive caching to identify frequently accessed files, ensuring faster retrieval by storing them on SSDs. Conversely, less frequently accessed files are allocated to HDDs. The project utilizes the Flexible I/O (fio) tool to measure the performance of files accessed on both SSDs and HDDs. Furthermore, to obtain insights and relationships between different files within the dataset, the project incorporates the Apriori algorithm. By analyzing structured relationships, the algorithm provides valuable intelligence for optimizing file placement decisions, enhancing overall system efficiency and adjust caching strategies to adapt to changing access patterns. By dynamically adapting file placement strategies based on access patterns and leveraging advanced algorithms for intelligent decision-making, the system endeavors to enhance user experience and system efficiency in managing file operations.
Keywords: *Apriori; Caching; File access patterns; File system; Least recently used (LRU).*

1. Introduction
In modern computing environments, the storage subsystem plays a pivotal role in determining overall system performance and user satisfaction. Conventional storage technologies, such as hard disk drives (HDDs), offer ample storage capacity but suffer from inherent limitations in terms of read and write speeds. On the other hand, solid-state drives (SSDs) leverage advanced flash memory technology to deliver significantly faster access times, making them well-suited for applications demanding high I/O performance. Caching serves as a foundational technique aimed at bolstering storage system efficiency by temporarily storing frequently accessed data in faster storage media. By maintaining this cached data close to the CPU, caching mechanisms effectively mitigate latency issues and enhance the overall responsiveness of the system. Predictive the caching represents an evolutionary leap in caching methodologies, leveraging the power of machine learning (ML) algorithms to anticipate future data access patterns. By analyzing historical access data and discerning trends and patterns, predictive caching systems intelligently predict which data is likely to be accessed soon. Armed with these insights, the system proactively caches relevant data pre-emptively, optimizing storage resource utilization and further reducing access latency. Predictive caching represents a promising frontier in storage system optimization, leveraging the capabilities of machine learning to anticipate and proactively address future data access needs. By harnessing the power of predictive analytics, organizations can unlock significant performance improvements, optimize resource utilization, and deliver enhanced user

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

experiences in today's data-driven computing landscape.

2. Literature Survey

The paper [1] presents a systematic survey of intelligent data caching approaches in wireless networks, leveraging artificial intelligence (AI) techniques to optimize caching strategies. It highlights the escalating wireless data traffic and the potential of AI-based caching to mitigate issues like duplicate data transmission and access delays. The survey starts with a review of conventional caching methods and their drawbacks, then moves on to explore various AI techniques. There are important research works utilizing AI for effective data placement and optimization of network performance metrics like cache hit rate, offloading, and throughput. The paper identifies existing challenges, including limited cache resources, fluctuating data popularity profiles, and patterns of user mobility, and suggests future research directions. While AI-based caching shows promise in enhancing network performance and user experience, the paper acknowledges the need for further research to address practical implementation challenges and optimize caching mechanisms for future wireless networks. The paper [2] investigates how the Apriori algorithm can be used to analyze web log data to find frequently accessed links. Web usage mining, a subset of web mining, focuses on understanding user behavior through data from web log files. The process includes three main phases: data preprocessing (cleaning data, identifying users, and defining sessions), pattern discovery (using the Apriori algorithm to find navigational patterns), and pattern analysis (analyzing and visualizing the discovered patterns). The Apriori algorithm, known for mining frequent item sets and generating association rules, is applied to web log data from an educational institute. After data cleaning and session identification, the algorithm calculates support and confidence levels for link combinations to identify the most frequently accessed links. Implemented in R, the study shows that this approach can effectively reveal user behavior patterns, providing insights to enhance website structure and content delivery based on user interactions. The paper concludes that the Apriori algorithm is a powerful tool for web usage mining.The paper [3] evaluates the performance of various cache replacement algorithms (CRAs), specifically First In First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), and The File Length Algorithm (LEN), in a distributed filesystem scenario. The study involves simulations of these algorithms under a setup with six interconnected cluster servers and 100 active files. The results show that LRU generally provides stable and predictable performance. The primary drawback of LFU and LEN is their tendency to cache large files for too long, which can lead to inefficient use of cache space and decreased performance in some scenarios. FIFO, on the other hand, demonstrates the worst performance due to its susceptibility to Belady's anomaly. LRU's simplicity and consistency make it a preferable choice for flexible filesystems. The paper [4] introduces the Automatic Prefetching and Caching System (APACS), designed to address limitations in existing prefetching solutions for improving I/O performance in computer systems. APACS utilizes dynamic adjustment of cache partitions, pipelining multiple stages of the prefetch process hence overlapping them, and strategically managing the prefetch buffer to optimize cache hit ratios and accelerate application execution speeds. Experimental results demonstrate that APACS surpasses the other prefetching algorithms which also includes the LRU cache management policy by over 50% on mean in trace-driven simulations. The study highlights the effectiveness of predictive prefetching in reducing the delays in disk I/O and dynamically allocating buffer/cache sizes based on global system performance. However, challenges remain in dynamically optimizing lookahead windows and integrating the algorithms of machine learning domain with proactive prefetching. Overall, APACS presents a promising solution to the performance challenge in the I/O subunit of modern computers, but more research is needed to be undertaken to provide a solution to these challenges, further validating its effectiveness in existent storage systems. This paper [5] introduces AutoCache, a novel schema for automated management of cache resource in Distributed File Systems (DFS) like

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

Hadoop and Spark. AutoCache employs ML models, specifically gradient boosted trees, to predict insightful patterns in file I/O activity and dynamically decide the files to cache to memory and evict from memory. The assessment using existent workload records demonstrates significant improvements in workload performance and cluster efficiency compared to existing cache management policies. However, this approach has some limitations. Firstly, this evaluation aims at a specific batch of workloads and may not generalize well to all scenarios. Additionally, the effectiveness of AutoCache may depend on the characteristics of the workload and the underlying system configuration. Future research could explore the scalability and robustness of AutoCache across various DFS environments and workload types, as well as investigate potential extensions to handle dynamic changes in system resources and workload patterns. The paper [6] introduces a predictive file caching approach aimed at reducing file system latency by transforming the file cache into a staging area for data about to be accessed, rather than merely storing recently accessed data. The system employs heuristic-based algorithms to predict and prefetch data without user intervention, thereby improving cache utilization and reducing read times. The approach is evaluated through simulations and a prototype implementation on SunOS, demonstrating significant improvements in cache miss rates and read times. However, limitations include the reliance on heuristic-based predictions, which may not always accurately anticipate future file access patterns, and the need for further research to optimize prefetching strategies for different system configurations and workloads. Future work could explore more sophisticated prediction algorithms and adaptive prefetching techniques to enhance performance across diverse computing environments and usage scenarios.

3. Method

3.1 Apriori Algorithm

The process begins by converting categorical data to numerical using one-hot encoding and then standardizes the data using StandardScaler. Apriori Algorithm [2] is used to obtain insight between different files and their relationships involved in the dataset. It begins by identifying all single-item itemsets that meet a specified minimum support threshold. Next, it methodically generates candidate itemsets of increasing lengths by combining the frequent itemsets identified in the preceding step. These candidate itemsets are pruned by removing those that contain infrequent subsets. The process continues iteratively, calculating support for the remaining candidates. Once no more frequent itemsets can be generated, association rules are derived from these frequent itemsets based on a minimum confidence threshold.

3.2 Symbolic Links

The symlink function is used to create a special file that acts as a pointer to another file in the file system. In this proposed work, symbolic links are shown in HDD that point to files in SSD. The frequently accessed files are moved to SSD along with creating symbolic links for HDD files. Symbolic links are removed when files are moved back to HDD from SSD.

3.3 LRU

LRU algorithm [3] is a cache replacement strategy that evicts the least recently accessed item from the cache when space is needed for a new item. Files that are identified as infrequent using LRU policy, are removed from the SSD. This process helps optimize storage space on the SSD by keeping only frequently accessed files. This approach is implemented by calculating the total cache size and maximum allowable cache size. Then while iterating through sorted files located in SSD, it is checked if removing the file keeps cache size within the limit. The file is removed if necessary and unlinked from the HDD.

4. Results And Discussion

4.1 Results

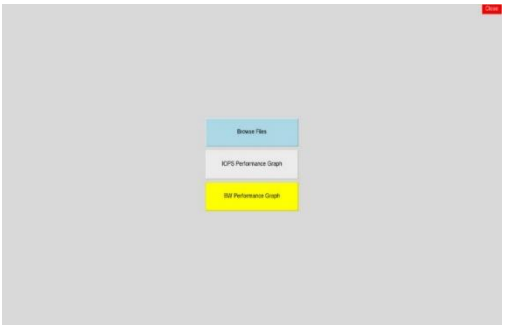

Figure 1 User Interface Home Page

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

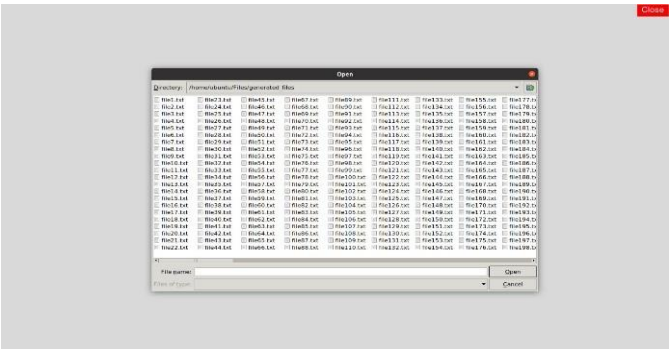

Figure 2 User Interface Browse Files

The Figure 1 and Figure 2 shows User Interface Home Page and User Interface Browse Files respectively. This is where users interact with the system. It includes the browse button for selecting files and directories. It consists of button to see the performance for the files accessed through HDD and SSD.

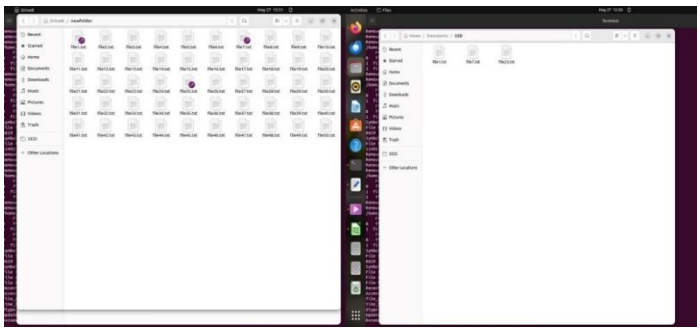

Figure 3 HDD and SSD folder

The symlink function is used to create a special file that acts as a pointer to another file in the file system. In Figure 3, symbolic links are shown in HDD that point to files in SSD. Symbolic links are removed when files are moved back to HDD from SSD.

5. Discussion

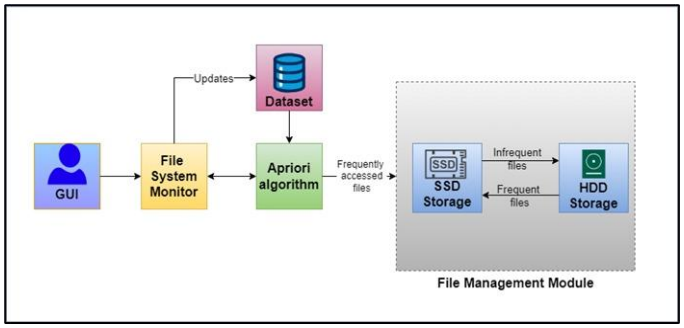

Figure 4 System Architecture

The Figure 4 denotes the system architecture. The GUI is the home page through which the user interacts with the system. The user browses the file through GUI by clicking on the browse file button. The dataset is updated by updating the recent access of the browsed file. This triggers the apriori algorithm which identifies the frequently accessed files and generates association rules based on predefined threshold values for support, confidence and lift. The frequently accessed files are moved to SSD creating symbolic links for HDD files. To optimize the storage space on SSD, LRU approach is used that monitors the available space on the SSD. It removes infrequently accessed files from SSD and unlinks it, to make space for moving frequently accessed files from HDD to SSD. Performance metrics using IOPS and Bandwidth of file is stored in CSV for further analysis and visualized through graphs getting higher performance of files in SSD than HDD. This is done by monitoring and comparing the IOPS and BW parameters of the files when it is present in HDD and after copying them to SSD, using FIO tool. The graphs can be visualized through the GUI.

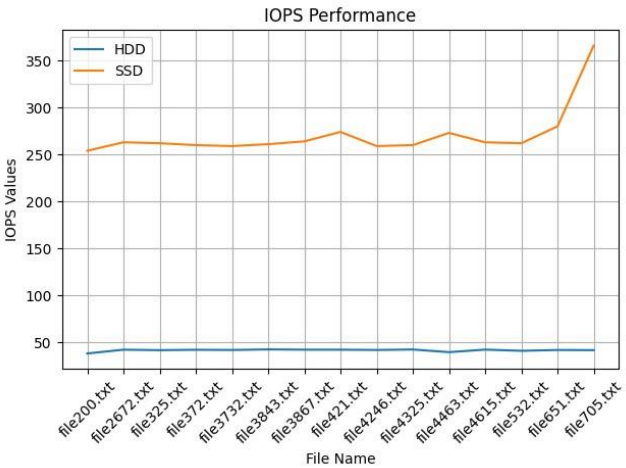

Figure 5 Performance Graph

The Figure 5 depicts performance of the file access through HDD and SSD. The experimental results are graphically represented to visualize the performance metrics, including I/O operations per second (IOPS) and bandwidth (BW) performance for file access through HDD and SSD. These graphs illustrate the

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

impact of our proposed predictive caching system on improving file access performance compared to the baseline measurement. The experiments performance is measured using the Flexible I/O tester (fio) tool, a widely adopted benchmarking tool for measuring I/O performance under various workloads and configurations. The significant improvements in IOPS and bandwidth when accessing files on the SSD highlight the system's capability to enhance performance by intelligently managing file placement. This optimization leads to faster data access and improved overall system efficiency.

Conclusion

The proposed predictive caching system offers a promising solution for optimizing file system performance through proactive caching based on machine learning predictions. By leveraging Apriori algorithm to identify frequently accessed files and dynamically adjusting the cache contents by symbolic link and accessing frequently files with less access time, the system demonstrates significant improvements in file access times and overall system responsiveness. The frequently accessed files are accessed effectively with less access time. Furthermore, the least frequently accessed files from SSD are removed using LRU approach. This approach enhances the efficiency of file access and provides a scalable and adaptive solution to the challenges of managing large volumes of data.

Acknowledgments

We are sincerely grateful to our encouraging project guide Dr. K. A. Malgi, our project coordinator Dr. K. A. Malgi and our motivating panel member Prof. K. V. Patil who have extended valuable guidelines, aid and unwavering encouragement throughout the various stages of the project. We also express our gratitude to Prof. Dr. D. A. Godse for her insightful directions. We take pleasure in offering big thanks to our honourable Principal Prof. Dr. P. V. Jadhav.

References

[1]. M. Sheraz, M. Ahmed, X. Hou, and D. Jin (2021), “Artificial Intelligence for Wireless Caching: Schemes, Performance, and Challenges,” IEEE Communications Surveys & Tutorials, vol. 23, issue. 1, pp. 31-32. doi: 10.1109/COMST.2020.3008362.

[2]. Dr. P. IsakkiDevi, and M. Sathya (2017), “Apriori Algorithm on Web Logs for Mining Frequent Link,” IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS).

[3]. S. Maffeis (1993), “Cache Management Algorithms for Flexible Filesystems,” Association for Computing Machinery.

[4]. J. Lewis, M. Alghamdi, M. A. Assaf, X. Ruan, Z. Ding, and X. Qin (2011), “An automatic prefetching and caching system,” IEEE, pp. 7-8.

[5]. H. Herodotou (2019), “AutoCache: Employing Machine Learning to Automate Caching in Distributed File Systems,” 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW). doi: 10.1109/ICDEW.2019.00-21.

[6]. J. Grioen, and R. Appleton (1996), “The Design, Implementation, and Evaluation of a Predictive Caching File System,” Citeseer.

[7]. P. Shah, J. Francois Paris, A. Amer, D. D. E. Long (2004), “Identifying Stable File Access Patterns,” 21st IEEE Conference on Mass Storage Systems and Technologies.

[8]. B. Daniel, “Understanding the Linux Kernel.”

[9]. L. Robert, “Linux Kernel Development.”

[10]. Y. Fu, H. H. Yang, K. Nguyen Doan, C. Liu, X. Wang and T. Q. S. Quek (2020), “Effective Cache-Enabled Wireless Networks: An Artificial Intelligence- and Recommendation Oriented Framework,” IEEE Vehicular Technology Magazine, vol. 16, issue. 1, pp. 20-28. doi: 10.1109/MVT.2020.3033934.

[11]. M. Al-Maolegi, B. Arkok (2014), "An Improved Apriori Algorithm for Association Rules," International Journal on Natural Language Computing (IJNLC), vol. 3, no. 1. doi: 10.5121/ijnlc.2014.3103.

[12]. C. Borgelt (2003), “Efficient Implementations of Apriori and Eclat,” Department of Knowledge Processing and Language Engineering School of Computer Science, Otto-von-Guericke-University of Magdeburg.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2024.0339
e ISSN: 2584-2854
Volume: 02
Issue: 07 July 2024
Page No: 2348-2353

[13]. Butt, A.R., Gniady, C., Hu, Y.C. (2007), "The Performance Impact of Kernel Prefetching on Buffer Cache Replacement Algorithms," IEEE Transactions on Computers, vol. 56, no. 7., pp. 889-908.
[14]. J. Griffioen, R. Appleton (1994), "Reducing File System Latency using a Predictive Approach," USTC'94: Proceedings of the USENIX Summer 1994 Technical Conference- Volume 1.
[15]. K. Korner (1990), Dept. of Computer Sci., Univ. of Southern California, Los Angeles, "Intelligent caching for remote file service," IEEE Computer Society Digital Library.