Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Performance of Memory Management Algorithms: A Systematic Review

Shujaat Ali Rathore*
Department of Computer Science & Information Technology, University of Kotli, Azad Jammu and Kashmir. Corresponding Author Email: shujaat.ali@uokajk.edu.pk

Muhammad Hammad u Salam
Department of Computer Science & Information Technology, University of Kotli, Azad Jammu and Kashmir.

Muhammad Irfan
Department of Computer Science, NCBA&E, Sub-Campus Multan, 60000, Pakistan

Sidra Zulfiqar
Department of Computer Science, Okara University, Okara, Punjab, Pakistan

Abstract
Modern computer systems must have effective memory management because it affects both user experience and overall performance. The need for memory-intensive applications is growing, making the choice of the best memory management method crucial. This paper provides a thorough examination of several memory management algorithms and assesses how well they work in terms of speed, effectiveness, and resource usage. Three well-known memory management algorithms—First-Fit, Best-Fit, and Next-Fit—are the subject of our study. By taking into account a variety of real-world workloads, we create a benchmark and methodically monitor important performance metrics including throughput, fragmentation, and allocation times.
Keywords: Memory Management, Memory Management Algorithm, Memory Fragmentation, Memory Management Utilization techniques.

1. Introduction

An essential component of computer systems is memory management, which guarantees effective use of the memory resources that are available. Memory management techniques are essential for arranging, assigning, and releasing memory to different processes, which affects a system's overall performance. Effective memory management is becoming more and more important as computer systems continue to advance and handle more complicated jobs. Thus, to maximize speed and resource consumption, researchers and developers have put a lot of work into creating and assessing memory management methods.

In computer systems, memory management is essential because it makes it easier to allocate and deallocate memory resources effectively to satisfy the needs of different activities. Effective memory management methods are becoming more and more necessary as current computer systems get bigger and more complicated. Memory management algorithms are in charge of maximizing system performance, guaranteeing equitable allocation, minimizing fragmentation, and optimizing memory use.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

DIALOGUE SOCIAL SCIENCE REVIEW

Numerous memory management algorithms have been proposed and implemented over the years, each with its own strengths and weaknesses. However, evaluating and comparing the performance of these algorithms is a complex task due to the diverse characteristics of computer systems, workloads, and memory requirements.

By investigating the performance of several memory management algorithms in various contexts, this systematic review seeks to provide a thorough evaluation of them. In order to help researchers and system designers choose the best memory management algorithm for a given use case, this review synthesizes the literature to determine the benefits, drawbacks, and trade-offs of various memory management algorithms.

Problem Statement

A crucial component of computer systems, memory management affects resource usage and overall performance. A number of memory management strategies have been developed and put into practice in order to maximize memory use. Nevertheless, little is known about these algorithms' efficacy and efficiency in many computing contexts. To determine memory management algorithms' advantages, disadvantages, and the best situations in which to use them, a thorough analysis of their performance is required.

Previous research has concentrated on assessing memory management algorithms separately or in particular settings, which has resulted to a fragmented body of knowledge and restricted applicability of the conclusions. To give a thorough grasp of the area, a systematic review that summarizes and evaluates the body of research on the performance evaluation of memory management algorithms is required. Thus, the issue that this systematic review attempts to solve is the absence of a thorough and organized evaluation of memory management algorithms' performance in a variety of computer contexts. This issue limits the capacity of academics, practitioners, and developers to optimize memory use and enhance system performance by preventing them from making well-informed decisions about the choice and application of memory management methods.

This work attempts to close the information gap and offer a thorough assessment of the performance of memory management algorithms by conducting a systematic review. The review's conclusions will aid in determining the best algorithms, as well as their advantages and disadvantages, and the right performance indicators. It will also highlight research trends and point out possible directions for future innovation and advancement in the evaluation and design of memory management algorithms.

Research Significance

The academic and corporate communities interested in computer systems and software development will find great value in the comprehensive review of memory management techniques' performance evaluation. The main facets of the research's importance are as follows.

1. Advancing Knowledge

Through the synthesis and analysis of a wide range of studies on the performance evaluation of memory management algorithms, the review adds to the body of

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

DIALOGUE SOCIAL SCIENCE REVIEW

information already in existence. It gives academics and practitioners a thorough grasp of the advantages, disadvantages, and restrictions of various algorithms, enabling them to expand on current understanding and create more effective memory management techniques.

2. Improved Decision-Making

When choosing and implementing memory management algorithms, researchers, practitioners, and developers can make well-informed judgments thanks to the systematic review's conclusions. Decision-makers can optimize memory use and improve system performance in a variety of computing contexts by using the review's list of the best algorithms and their performance characteristics.

3. Optimization of Resource Utilization

Effective memory management is essential to computer systems' optimal resource use. Throughput, reaction time, memory utilization, page fault rate, and other metrics are just a few of the performance measures that the systematic examination sheds light on. With the use of this data, algorithms that optimize resource usage can be designed, improving system efficiency and economy.

4. Identification of Research Gaps and Future Directions

The study pinpoints opportunities for future investigation and research gaps by examining the state of the field at the moment. It draws attention to new developments in the field, including machine learning and adaptive techniques, and offers possible directions for future research and development in the creation and assessment of memory management algorithms.

Aim

The objective of this research is to examine and assess the efficacy of various memory management algorithms to determine their advantages, disadvantages, and overall influence on system performance and resource usage. The following goals are the focus of this research: Examine and contrast how well-known memory management algorithms—like First-Fit, Best-Fit, and Next-Fit—perform in terms of memory usage, speed, and efficiency.

Analyze and measure how these techniques affect system reaction times, throughput, and overall performance with different workload factors. Examine how memory size and system load affect memory management algorithms' performance, including how well they can handle requests for memory allocation and control memory fragmentation.

Examine the ways in which the performance of various memory management methods increases as system resources and workload demands increase. Determine the trade-offs between each memory management strategy, taking into account variables like fragmentation, memory overhead, and overall system responsiveness.

Research Questions

1. How can memory management algorithms be designed to prevent common security vulnerabilities such as buffer overflows or heap corruption?
2. How can memory management algorithms be designed to efficiently utilize memory resources in multicore or distributed systems?

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Research Objective

Efficiency and Performance: Determine and create memory management methods that reduce memory overhead and maximize resource allocation to enhance system performance overall. Look at ways to enhance memory usage and lessen fragmentation of memory. Examine adaptive memory management techniques that have the ability to dynamically modify resource allocation in response to workload trends.

Security & Protection: Create memory management methods that include safeguards against memory-related vulnerabilities including heap corruption and buffer overflows. Examine methods to make sure memory is protected and isolated between users or processes. Create algorithms that are capable of identifying and reducing memory-related vulnerabilities and threats.

Scalability and Multicore Systems

Create memory management methods that make effective use of distributed or multicore systems' memory resources. Look into algorithms that can manage the growing complexity and size of today's computing environments. Examine memory management strategies for big data workloads and large-scale data-intensive applications.

Energy Efficiency

Examine and enhance memory management methods to reduce the amount of energy used during memory operations. Create algorithms that minimize the need for data transfer and memory access caching overheads. For increased energy efficiency, modify memory management algorithms to take use of newly developed memory technologies, such as non-volatile memory.

3. Literature Review

Categorization of Memory Management Algorithms

Memory management algorithms have been grouped by several studies according to their underlying theories and methods. Fernandez et al. (2015), for example, divided algorithms into two groups: segmentation algorithms and paging methods. While segmentation algorithms assign processes to variable-sized memory segments, paging techniques partition memory into fixed-size frames. This classification offers a framework for contrasting and assessing the various algorithms' performances within each category.

Performance Metrics

Several performance criteria have been used in studies to assess memory management techniques. Metrics including memory usage, reaction time, fragmentation, throughput, and page fault rate are frequently employed. Memory utilization and page fault rate were used by Raju et al. (2017) as the main measures, emphasizing the significance of effective memory utilization and low page faults for best results. Researchers can evaluate the algorithm's efficacy and efficiency in managing memory resources by examining these metrics.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

DIALOGUE SOCIAL SCIENCE REVIEW

Benchmark Selection

Researchers frequently use benchmark programs that mimic real-world computing settings to assess memory management strategies. A popular collection of benchmarks for assessing memory management techniques in various computing contexts is the SPEC CPU benchmark suite. Additionally, memory behavior analysis and algorithm performance evaluation are common uses for simulation tools like as Simics, Pin, and Gem5. It is important to give considerable thought to the selection of simulation tools and benchmarks, as they might have a substantial impact on the evaluation results.

Experimental Methodologies

Memory management methods have been tested using a variety of experimental techniques. A number of studies have made use of simulation-based methodologies, which enable researchers to manipulate and reproduce various settings. Some have employed the technique of trace-based evaluation, examining memory traces derived from actual workloads. Furthermore, tests using real hardware configurations have been carried out by researchers on testbeds to evaluate the algorithm's performance in practical scenarios. Diverse approaches provide distinct perspectives on the behavior and performance attributes of the algorithm. Thought about.

Comparison of Algorithms

Various memory management methods have been examined in studies to determine their advantages and disadvantages. Liu et al. (2019), for instance, contrasted and assessed the performance of segmentation and paging algorithms using reaction time, memory use, and page fault rate. These comparative evaluations help determine which strategy is best suited for a given computer environment by offering insightful information about the trade-offs between various algorithms.

4. Methodology

The process of assessing memory management algorithms' performance usually combines simulation, actual measurements, and theoretical analysis. An overview of the procedures involved in assessing memory management algorithms' performance is provided below: Explain Metrics: Choose the metrics that will be used to assess how well memory management methods are performing. Memory usage, throughput, reaction time, efficiency, and fairness are examples of common measurements.

The memory management methods should be the subject of a theoretical examination. To comprehend their anticipated performance characteristics, this may entail complexity analysis, algorithmic analysis, and mathematical modeling. A baseline understanding of how the algorithms should operate in various contexts is established with the use of theoretical analysis. Create a simulation environment that closely resembles how a memory management system might operate.

Through the creation of artificial workloads and the measurement of their effects on memory consumption, fragmentation, and other pertinent metrics, simulations can be used to assess the algorithms' performance. Simulations make it possible to test algorithms in a variety of circumstances and conduct

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

controlled trials. Empirical Measurements: To collect empirical data, do tests on actual systems or testbeds. This may entail measuring the relevant performance metrics while executing memory-intensive programs or benchmarks. Empirical measurements reveal how the algorithms function in real-world situations and can reveal problems or optimizations that are not always seen in simulations.

Comparative Analysis: Using the information gathered, evaluate how well various memory management strategies perform. Determine which algorithms perform best in particular settings or workloads, as well as their advantages and disadvantages. With the aid of comparative analysis, one may choose the best algorithm for a certain system or application and comprehend trade-offs. Sensitivity Analysis: To determine how memory management algorithms react to changes, perform sensitivity analysis by changing various parameters and workload characteristics. This aids in comprehending how resilient and flexible the algorithms are in various scenarios. Verification & Validation.

Verify the outcomes of empirical measurements and simulations by contrasting them with theoretical analysis, previously published work, or real-world implementations. This stage guarantees the evaluation's dependability and accuracy. Talk and Analysis: To learn more about the behavior and performance traits of the memory management algorithms, analyze and understand the data. Talk about the study's conclusions, restrictions, and ramifications. It's crucial to remember that the precise approach may change based on the evaluation's goals, scope, and resources. To improve the assessment of memory management algorithms, researchers may also use other methods like machine learning, trace-driven analysis, or mathematical modeling.

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Research Process

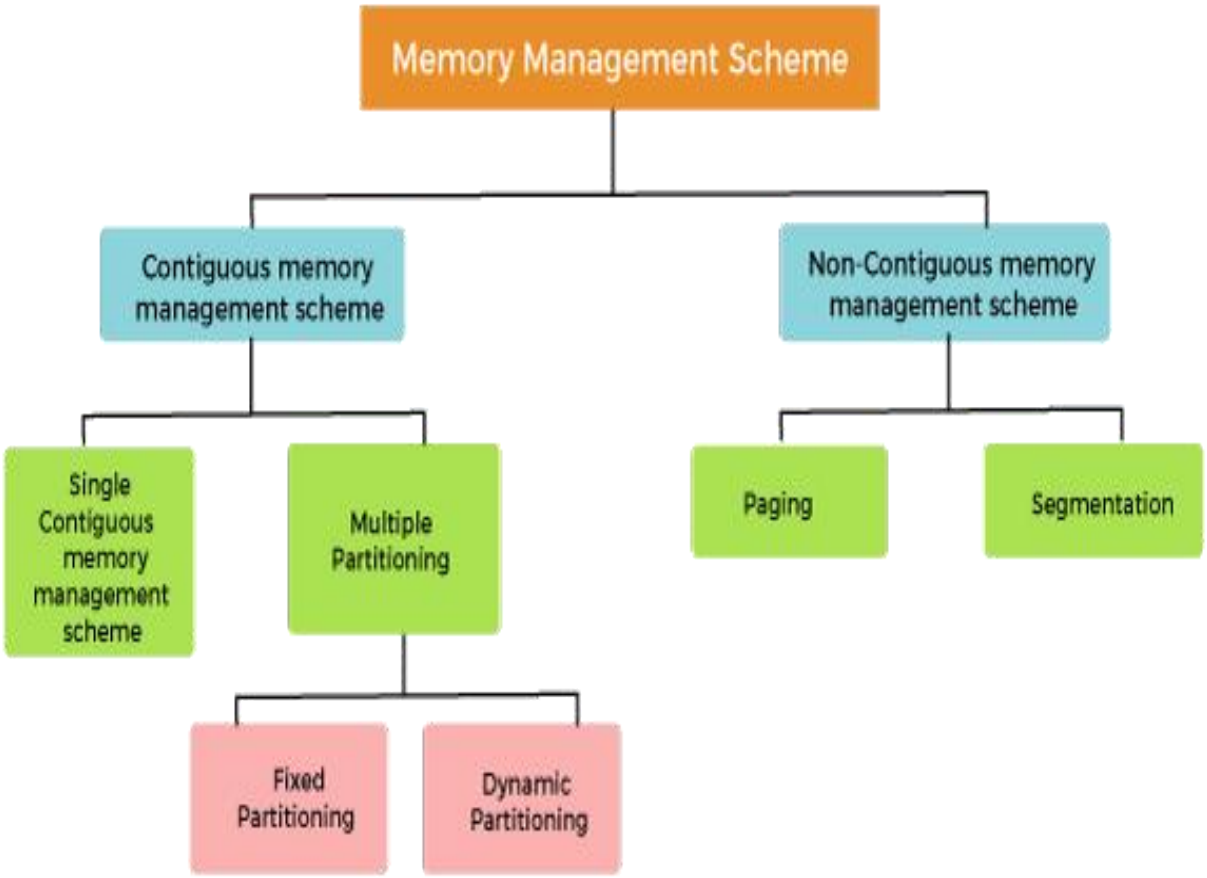

Figure 1. Classification of Memory Management

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Prisma Diagram

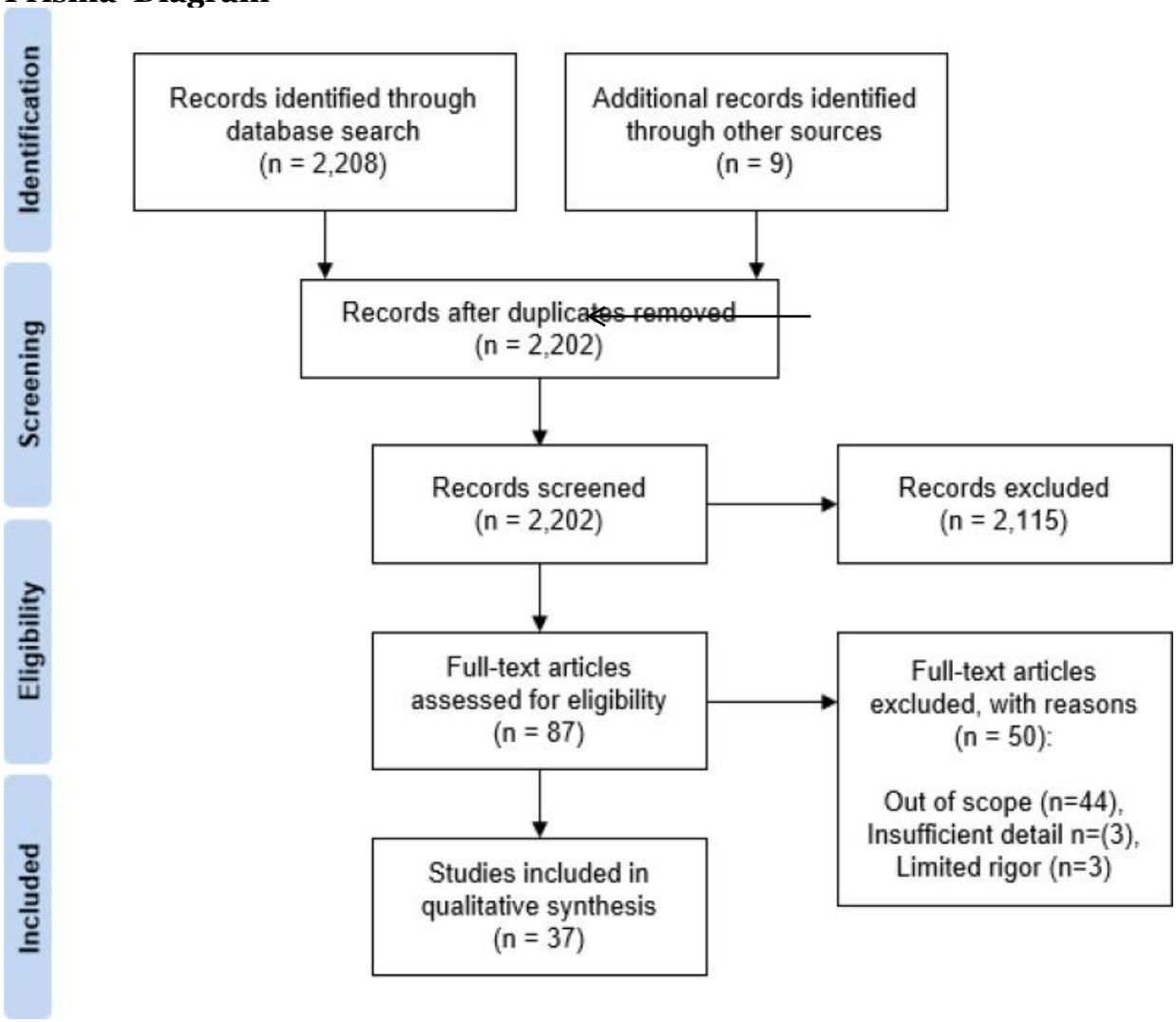

Figure 2. PRISMA Diagram Flowchart

Search Strategy

Table 1: Search String

Data Base	Search String	Year	Number of Articles Searched
Taylor & Francis Online	("Memory Management")	2022-2024	10,863
ACM Digital Library	("Memory Management") AND ("Algorithm")	2022-2024	*34,329*
Scopus	("Memory Management in OS")	2022-2024	21,216
Springer link	("Algorithms in OS")	2024-2024	32,600
IEEE Explorer	("Memory Management OS") ("Algorithm) OR ("OS")	2022-2024	20,292

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Selection Strategy

Table 2: Selection and Screening Process

Phase	Process	Selection Criteria	Taylor & Francis Online	ACM Digital Library	Science Direct	Springer link	IEEE Explorer	Total
1	Searching	Keywords	25	29	35	42	60	191
2	Screening	Title	12	7	25	35	40	119
3	Further Screening	Abstract	5	4	4	3	5	21
4	Further screening	Introduction, and Conclusion	1	1	1	1	1	5
5	Evaluation	Complete Article	1	1	1	1	1	5

Table 3: Quality Assessment

Medium	Year	Quality Assessment (a)	(b)	(c)	(d)	Score
Automatica	20201	1	1	1	4	
ACM Computing Surveys	2019 1	1	1	0	3	
Journal of Systems andSoftware	2018 1	1	1	0	3	
Journal of Intelligent &Robotic Systems	2021 1	0	1	1	3	
IEEE Access	20201	1	1	1	4	

5. Results

How can memory management algorithms be designed to prevent common security vulnerabilities such as buffer overflows or heap corruption?

The following methods can be taken into consideration when creating memory management algorithms that can stop common security flaws like buffer overflows or heap corruption: Verifying Memory Boundaries: Put in place bounds checking procedures to make sure that memory operations stay inside the allotted memory limits. In order to ensure that the data being written or accessed stays within the allotted memory blocks, this may entail runtime checks. Randomization of Address Space Layout (ASLR):

Dialogue Social Science Review (DSSR)
www.thedssr.com

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

DIALOGUE SOCIAL SCIENCE REVIEW

By using ASLR techniques, you can randomly arrange a process's memory to make it more difficult for attackers to forecast memory addresses and take advantage of vulnerabilities. Stack Defense Systems: Make use of stack protection techniques like as stack canaries or stack cookies, which arrange the return address on the stack after a known sentinel value or a random value. By determining whether the value has changed, this aids in the detection of buffer overflows. Checks for Heap Integrity:

To stop heap corruption, apply integrity tests for heap data structures. Heap metadata and data can be monitored for unwanted alterations using methods such as guard bands or heap cookies. It is recommended to promote the utilization of safe memory functions, such as those that carry out automatic bounds checking or offer secure substitutes for susceptible operations like sprintf or strcpy. Alternatively, you can use functions like snprintf or strncpy to guarantee correct memory handling. Static Code Analysis: During the development phase, use static code analysis tools to find possible heap corruption or buffer overflow vulnerabilities.

How can memory management algorithms be designed to efficiently utilize memory resources in multicore or distributed systems?

The following strategies can be taken into consideration when designing memory management algorithms to effectively use memory resources in multicore or distributed systems: Memory Management with NUMA Awareness: In distributed or multicore systems, non-uniform memory access (NUMA) architectures are frequently used. Consider the NUMA topology while creating memory management algorithms, and place memory closer to processor cores that will use it frequently. Performance is enhanced and memory access latency is decreased as a result. Recall Affinity: Adopt memory affinity policies to minimize remote memory access and maximize cache utilization by keeping data and processes closer to one another. This may entail strategies like memory allocation on the same NUMA node as the thread or process that is executing, such as thread or process placement regulations.

6. Conclusion

The literature analysis concludes by emphasizing the significance of methodically assessing memory management algorithms' performance in order to improve system effectiveness and resource utilization. Important conclusions, approaches, and developments in the discipline have been determined by the review. When assessing algorithm performance, a number of factors are important to consider, including the choice of assessment metrics, algorithm classification, benchmark selection, experimental approaches, and comparative studies. Furthermore, new developments in machine learning and adaptive algorithms provide new ways to improve the effectiveness of memory management. The results of this evaluation of the literature are an invaluable tool for practitioners and scholars studying memory management, helping to

References

1. Krauss, R., Goli, M., & Drechsler, R. (2023, January). EDDY: A multi-core BDD package with dynamic memory management and reduced

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

fragmentation. In Proceedings of the 28th Asia and South Pacific Design Automation Conference (pp. 423-428)
2. Jagadeeswari, N., Mohanraj, V., Suresh, Y., & Senthilkumar, J. (2023). Optimization of virtual machines performance using fuzzy hashing and genetic algorithm-based memory deduplication of static pages. Automatika, 64(4), 868-877.
3. Parida, T., Nath, K., & Das, S. (2022). SAM: a mechanism to facilitate smear-aware forensic analysis of volatile system memory. Journal of Applied Security Research, 1-30.
4. Rai, S., & Talawar, B. (2023). Challenges in Design, Data Placement, Migration and Power-Performance Trade-offs in DRAM-NVM-based Hybrid Memory Systems. IETE Technical Review, 40(4), 498-520.
5. Yi, Q., Puxiang, X., & Zhu, T. (2020). The Design and Implementation of the RT-thread Operating System. Auerbach Publications.
6. Abbas, T., Fatima, A., Shahzad, T., Alissa, K., Ghazal, T. M., & Al-Sakhnini, M. M. & Ahmed, A.(2023). Secure IoMT for disease prediction empowered with transfer learning in healthcare 5.0, the concept and case study. IEEE Access.
7. Winter, M., Mlakar, D., Parger, M., & Steinberger, M. (2020, June). Ouroboros: virtualized queues for dynamic memory management on GPUs. In Proceedings of the 34th ACM International Conference on Supercomputing (pp. 1-12).
8. Sareen, K., & Blackburn, S. M. (2022, September). Better Understanding the Costs and Benefits of Automatic Memory Management. In Proceedings of the 19th International Conference on Managed Programming Languages and Runtimes (pp. 29-44).
9. Janjua, J. I., Kousar, S., Khan, A., Ihsan, A., Abbas, T., & Saeed, A. Q. (2024, December). Enhancing Scalability in Reinforcement Learning for Open Spaces. In 2024 International Conference on Decision Aid Sciences and Applications (DASA) (pp. 1-8). IEEE.
10. de Moura, R. C., de Souza Oliveira, L., Schneider, G. B., Pilla, M. L., Yamin, A. C., & Reiser, R. H. S. (2021). Intf-HybridMem: Page migration in hybrid memories considering cost efficiency. Sustainable Computing: Informatics and Systems, 29, 100466.
11. Wynn, A., & Wu, J. (2000). The effect of compression on performance in a demand paging operating system. Journal of Systems and Software, 50(2), 151-170.
12. Xu, L., Chen, G., Li, D., & Luo, H. (2023, October). PM-Migration: A Page Placement Mechanism for Real-Time Systems with Hybrid Memory Architecture. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 313-324). Singapore: Springer Nature Singapore.
13. Aiza Shabir, Aiza Shabir, Khawaja Tehseen Ahmed, Khawaja Tehseen Ahmed, Khadija Kanwal, Afshan Almas, Sehrish Raza, Meherwar Fatima, & Tahir Abbas. (2024). A Systematic Review of Attention Models in Natural Language Processing. STATISTICS, COMPUTING AND INTERDISCIPLINARY RESEARCH, 6(1), 33-56. https://doi.org/10.52700/scir.v6i1.157
14. Yang, S., Li, X., Dou, X., Gong, X., Liu, H., Chen, L., & Liu, L. (2020, September). Monitoring Memory Behaviors and Mitigating NUMA Drawbacks on Tiered NVM Systems. In IFIP International Conference on

ISSN Online: 3007-3154
ISSN Print: 3007-3146
https://doi.org/10.5281/zenodo.17064792
Vol. 3 No. 10 (October) (2025)

Network and Parallel Computing (pp. 386-391). Cham: Springer International Publishing.

15. Zhao, B., Liu, C., Dong, J., Cao, Z., Nie, W., & Wu, W. (2023, May). Enabling switch memory management for distributed training with in-network aggregation. In IEEE INFOCOM 2023-IEEE Conference on Computer Communications (pp. 1-10). IEEE.
16. Li, R., Yadav, S., Wu, Q., Kavi, K., Mehta, G., Yadwadkar, N. J., & John, L. K. (2023, October). Performance Implications of Async Memcpy and UVM: A Tale of Two Data Transfer Modes. In 2023 IEEE International Symposium on Workload Characterization (IISWC) (pp. 115-127). IEEE.
17. Chakraborty, P., Ezzati-Jivan, N., Azhari, V., & Tetreault, F. (2023, December). AltOOM: A Data-driven Out of Memory Root Cause Identification Strategy. In 2023 IEEE International Conference on Big Data (BigData) (pp. 1637-1646). IEEE.
18. Ali, M., Arshad, M., Uddin, I., Ali, G., Asim, M., & ElAffendi, M. (2024). A Resource Aware Memory Requirement Calculation Model for Memory Constrained Context-Aware Systems. IEEE Access.
19. Blazevic, B., Peter, M., Hamad, M., & Steinhorst, S. (2023, August). TEEVseL4: Trusted Execution Environment for Virtualized seL4-based Systems. In 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) (pp. 67-76). IEEE.
20. de Moura, R. C., de Souza Oliveira, L., Schneider, G. B., Pilla, M. L., Yamin, A. C., & Reiser, R. H. S. (2021). Intf-HybridMem: Page migration in hybrid memories considering cost efficiency. Sustainable Computing: Informatics and Systems, 29, 100466.